USC-SIPI REPORT #460

SCALABLE SAMPLING AND RECONSTRUCTION FOR GRAPH SIGNALS

by

Ajinkya Jayawant

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(ELECTRICAL ENGINEERING)

August 2023

Acknowledgements

Advancing from coursework in Master's to research in Ph.D. requires a significant leap in maturity and vision, and a good mentor is indispensable to avoid pitfalls and provide momentum. My Ph.D. advisor, professor Antonio Ortega has mentored me regarding everything ranging from problem selection, critical thinking, to writing. I am thankful to him for showing me the ropes for the process of research. However, a Ph.D. is more than just a technical journey. In this journey, for more than being a first-rate advisor, I am grateful to professor Ortega for being a good person.

Before starting my Ph.D., I was worried about being homesick staying away from family and friends in India for an extended period of time, and about being able to manage living by myself on the other side of the globe. During my stay in Los Angeles, I was thankful to be close enough to the families of Aditya and Saurabh Jayawant who provided me with a homely atmosphere in a country I was living for the first time. Living by yourself forces you to navigate all sorts of scenarios ranging from housing, finance, to career, more or less on your own. From the first to the last year of my Ph.D., three friends - Saurav Prakash, Alexander Serrano, and Miguel Moscoso, have stuck with me through all ups and downs and helped me on various occasions. I owe it to them for making my life in US smooth sailing.

Once I had managed to settle in Los Angeles, research proved to be a challenge in the beginning. Collaborations and discussions with professor Salman Avestimehr, colleagues Basak Guler, and Eduardo Pavez, helped me get up to speed with the latest research. During a following summer internship, I got the opportunity to work with Wenqing Jiang.

I am especially thankful to him for providing an exciting and encouraging atmosphere for my internship and I am glad that I met him. For various Ph.D. milestones since then, Akshay Gadde provided me timely advice. On reaching the thesis proposal milestone, for my qualifying exam, the committee of professors Jay Kuo, Salman Avestimehr, Keith Jenkins, Ramesh Govindan, and my advisor gave many useful suggestions and comments. In addition, pursuing a doctoral degree requires dependable academic advisement, funding[1], and logistics to proceed smoothly. For that, I want to thank professor Richard Leahy, Tracy Charles, Andy, Gloria, Seth, and Diane.

Ph.D. students through most parts of the world suffer from anxiety and depression, but a good social circle is known to help maintain mental well-being. I was fortunate enough to have a healthy office atmosphere and social circle where I could learn new things and also to wind down outside of work. For that, I thank my colleagues and friends Aamir, Benjamin, Keng-Shih, Shashank, Johanna, Laura, Ecem, Samuel, Darukeesan, Carlos, Jitin, Nitin, Vivek, Rashmi, Sagar, Pratyusha, Chaitanya, Sarath, and Dhruva.

Now as I look forward towards the completion of Ph.D., I feel fortunate to have grown up in a family interested in science, which primarily led me to think about pursuing a Ph.D. When it came to pursuing a Ph.D., allowing their only child to travel overseas to study for a program that does not have a fixed time limit took strength on the part of my parents. From the beginning and throughout the process, I am grateful for the support of my parents - Kirti and Anand Jayawant.

This Ph.D. has been possible because of many people who have contributed along the way in their own unique way. To all these people who have accompanied me to where I am today, I thank you.

[1]This work is supported in part by NSF under grants CCF-1410009, CCF-1527874, and CCF-2009032 and by a gift from Tencent.

Table of Contents

List of Tables

List of Figures

Abstract

Processing data such as spatially scattered weather measurements or point clouds generated from 3D scans is a challenge due to the lack of an inherent structure to the data. Graphs are convenient tools to analyze and process such unstructured data with algorithms analogous to those in traditional signal processing, which treat the data as a graph signal. However, measuring the whole graph signal can be expensive. Observing a limited subset of graph nodes may be better in such cases, i.e., sampling the graph signal and inferring information at the remaining nodes using reconstruction algorithms.

Although graph signal sampling and reconstruction algorithms exist in the literature, making them practical enough to be used in real-life applications requires numerous theoretical and fundamental improvements. One such requirement is that the algorithm should not require substantially more execution time as the number of vertices and edges in the graph increases. Even if the algorithm execution time scales well with the graph size, some samples may get corrupted. Reconstruction of such data is challenging and requires knowledge of the signal model or its parameters, which are commonly assumed to be known in the literature but in practice have to be estimated.

In this thesis, we propose algorithms to minimize the reconstruction error when sampling in the presence of signal corruption, estimate reconstruction error as a function of the signal bandwidth and develop scalable graph sampling algorithms, in particular algorithms amenable to parallelization. This makes the graph signal reconstruction algorithms

more flexible and increases the capabilities of the graph sampling algorithms. Through these improvements, we push the limits of graph signal sampling and reconstruction even further.

Chapter 1

Introduction

Graphs are a convenient way to represent and analyze data having irregular relationships between data points [63] and can be useful in a variety of different scenarios, such as characterizing the Web [40], semi-supervised learning [73], community detection [23], or traffic analysis [16]. We call *graph signal* [48] the data associated with the nodes of a graph. Graph signals are useful in analyzing real-world systems, such as sensor networks, biological data, or machine learning systems, using tools from graph signal processing [63, 49, 26].

Due to the size of most real-world graphs, it is often unfeasible to observe all the data points on the graph. In such scenarios, one needs to select a small subset of nodes, observe the corresponding samples and make inferences about samples in the remaining nodes in the network using the data obtained from the selected nodes. Such setups are also related to the problem of active semi-supervised learning [24], where one chooses a small set of data points to label and learns the missing labels by utilizing the labeled data along with the graph topology. The question is then to choose the best data points to sample to reconstruct the underlying data structure as accurately as possible. This is known as the *graph signal sampling* problem [54].

Optimizing the choice of sampling set using concepts from *experiment design* [56] can help minimize the effect that noise in the input signal may have on the quality of signals reconstructed from observed samples. Many existing sampling set selection methods are computationally intensive because they require an eigendecomposition. For large graphs,

even existing eigendecomposition-free methods are still much slower than random sampling algorithms, which are the fastest available methods. In Chapter 2, through optimizing sampling sets towards the D-optimal objective from experiment design, we propose a sampling algorithm that has complexity comparable to that of random sampling algorithms while reaching accuracy similar to existing eigendecomposition-free methods for a broad range of graph types.

While the proposed graph signal sampling algorithm in Chapter 2 improves existing state-of-the-art algorithms, some practical scenarios often place additional demands on a sampling algorithm. For example, some algorithms may encounter a situation where some selected samples are lost or unavailable due to sensor failures or adversarial erasures. To address this setting, in Chapter 3, we formulate a robust graph signal sampling problem where only a subset of selected samples are received, and the goal is to maximize the worst-case performance. We propose a novel greedy robust sample selection algorithm and study its performance guarantees. Our numerical results demonstrate the performance improvement of the proposed algorithm over the existing schemes.

Another desirable characteristic of sampling algorithms is that they require minimal assumptions. Data on graphs is often modeled as bandlimited graph signals. Predicting or reconstructing the unknown signal values for such a model requires estimating the bandwidth. In Chapter 4, we propose a signal bandwidth estimation technique. In doing so, we design a cross-validation approach with a stable graph signal reconstruction and propose a method for estimating the reconstruction errors for different choices of signal bandwidth. Using this technique, we can estimate the reconstruction error on various real-world graphs.

Data such as point clouds often contain millions of data points and need to be downsized before processing. Downsampling algorithms for point cloud data typically demand high sampling rates and require fast processing for run-time applications. In Chapter 5, we propose parallelized algorithms for point clouds in the high sampling rate regime. We test these

algorithms on various point clouds and compare them to the algorithm in Chapter 2, observing only marginal loss in performance with an order of magnitude speedup in processing times.

1.1 Graph preliminaries and notation

A graph is defined as the pair $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes or vertices and $\mathcal{E}$ is the set of edges [10]. The set of edges $\mathcal{E}$ is a subset of the set of unordered pairs of elements of $\mathcal{V}$. A graph signal is a real-valued function defined on the vertices of the graph, $\mathbf{f} : \mathcal{V} \to \mathbb{R}$. We index the vertices $v \in \mathcal{V}$ with the set $\{1, \cdots, n\}$ and define w_{ij} as the weight of the edge between vertices i and j. The $(i,j)^{\text{th}}$ entry of the adjacency matrix of the graph $\mathbf{A}$ is w_{ij}, with $w_{ii} = 0$. $\mathbf{A}$ is $n \times n$, where n is the number of vertices in the graph, which we also call the graph size. The degree matrix $\mathbf{D}$ of a graph is an $n \times n$ diagonal matrix with diagonal entries $d_{ii} = \sum_j w_{ij}$. This thesis considers weighted undirected graphs without self-loops and with non-negative edge weights. Throughout the thesis, $\mathbf{I}$ is the $n \times n$ identity matrix.

The combinatorial Laplacian of a graph is given by $\mathbf{L} = \mathbf{D} - \mathbf{A}$, with its corresponding eigendecomposition defined as $\mathbf{L} = \mathbf{U}\Sigma\mathbf{U}^T$ since the Laplacian matrix is symmetric and positive semidefinite. The eigenvalues of the Laplacian matrix are $\mathbf{\Sigma} = \text{diag}(\lambda_1, \cdots, \lambda_n)$, with $0 = \lambda_1 \leq \cdots \leq \lambda_n$ representing the frequencies. The column vectors of $\mathbf{U}$ provide a frequency representation for graph signals so that the operator $\mathbf{U}^T$ is usually called the graph Fourier transform (GFT). The eigenvectors $\mathbf{u}_i$ of $\mathbf{L}$ associated with larger eigenvalues λ_i correspond to higher frequencies, and the ones associated with lower eigenvalues correspond to lower frequencies [63, 48]. The GFT of $\mathbf{x}$ is defined as $\tilde{\mathbf{x}} = \mathbf{U}^T\mathbf{x}$ [63].

In this thesis, we represent sets using calligraphic uppercase, e.g., $\mathcal{X}$, vectors using bold lowercase, $\mathbf{x}$, matrices using bold uppercase, $\mathbf{A}$, and scalars using plain uppercase or lowercase as x or X. Table 1.1 lists additional notations. We use $\text{tr}(\mathbf{A})$ to denote the trace of $\mathbf{A}$. $\text{diag}(x_1, \cdots, x_n)$ represents a square diagonal matrix with values $x_1, \cdots, x_n$ on the diagonal.

Table 1.1: Linear algebra notation in this thesis

Notation	Description
$\mathcal{X}_i$	$\mathcal{X}$ after iteration i
$\lvert\mathcal{X}\rvert$	Cardinality of set $\mathcal{X}$
$\mathbf{A}_{\mathcal{X}\mathcal{Y}}$ or $\mathbf{A}_{\mathcal{X},\mathcal{Y}}$	Submatrix of $\mathbf{A}$ indexed by sets $\mathcal{X}$ and $\mathcal{Y}$
$\mathbf{A}_{ij}$	$(i,j)^{\text{th}}$ element of $\mathbf{A}$
$\mathbf{A}_{\mathcal{X}}$	$\mathbf{A}_{:,\mathcal{X}}$, selection of the columns of $\mathbf{A}$
$\mathbf{A}_i$	$\mathbf{A}$ after iteration i
x_i or $\mathbf{x}(i)$	i^{th} element of the vector $\mathbf{x}$
$\mathbf{x}_{\mathcal{X}}$ or $\mathbf{x}(\mathcal{X})$	Subset of the vector $\mathbf{x}$ corresponding to indices $\mathcal{X}$
$\mathbf{x}_v$	Vector corresponding to a vertex v among a sequence of vectors indexed over the set of vertices $\mathcal{V}$
$\lVert.\rVert$	Two/Euclidean norm of matrix or vector
$\lvert x\rvert, \lvert\mathbf{x}\rvert$	Entry wise absolute value of scalar x or vector $\mathbf{x}$

1.2 Problem formulation

1.2.1 Sampling problem

We consider an n-dimensional scalar real-valued signal $\mathbf{x}$ on the vertex set $\mathcal{V}$. We assume that only a part of this signal is known, corresponding to a subset of vertices, $\mathcal{S} \subseteq \mathcal{V}$. For the sake of convenience, without loss of generality, for a given algorithm, the vertices are relabeled after sampling so that their labels correspond to the order in which the vertices were chosen, $\mathcal{S} = \{1, 2, \cdots\}$. We denote $\mathbf{x}_{\mathcal{S}}$ and $\mathbf{x}_{\mathcal{S}^c}$ the known and unknown signals, respectively, where $\mathcal{S}^c$ is the complement of $\mathcal{S}$. Estimating $\mathbf{x}_{\mathcal{S}^c}$ from $\mathbf{x}_{\mathcal{S}}$ is the graph signal reconstruction problem [54]. We denote the reconstructed unknown signal as $\hat{\mathbf{x}}_{\mathcal{S}^c}$, and quantify its closeness with the original signal, $\mathbf{x}_{\mathcal{S}^c}$, using the ℓ_2 norm $\|\mathbf{x}_{\mathcal{S}^c} - \hat{\mathbf{x}}_{\mathcal{S}^c}\|_2^2$. A popular choice for a smooth graph signal is the bandlimited signal model [2].

Bandlimited signals are represented as $\mathbf{x} = \mathbf{U}_{\mathcal{F}}\boldsymbol{\alpha}$, where $\mathcal{F}$ is the set $\{1, \cdots, f\}$, and $\boldsymbol{\alpha}$ is an f-dimensional vector. We call f the bandwidth of the signal. For sampling bandlimited signals $\mathbf{x}$, a sampling set that satisfies the following two conditions: Condition i) the number of samples requested is larger than the bandwidth, that is $|\mathcal{S}| \geq f$, and Condition ii) the sampling set $\mathcal{S}$ is a uniqueness set [54] corresponding to the bandwidth support $\mathcal{F}$, will allow

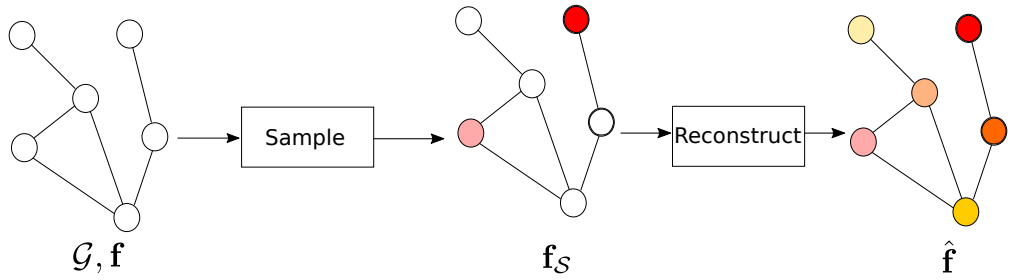


Figure 1.1: Pipeline for sampling and reconstruction of graph signals. Uncolored vertices indicate unknown signal values, and colored vertices indicate known or predicted signal values.

us to recover $\mathbf{x}$ exactly. Given the observed samples, $\mathbf{x}_{\mathcal{S}}$, the reconstruction is given by the least squares solution:

$$\hat{\mathbf{x}} = \mathbf{U}_{\mathcal{F}}(\mathbf{U}_{\mathcal{SF}}^T\mathbf{U}_{\mathcal{SF}})^{-1}\mathbf{U}_{\mathcal{SF}}^T\mathbf{x}_{\mathcal{S}}. \tag{1.1}$$

In practice, signals are never exactly bandlimited. In this thesis, we consider the widely-studied scenario of bandlimited signals with added noise and choose sampling rates that satisfy Condition i) for the underlying noise-free signal[1]. While Condition ii) is difficult to verify without computing the eigendecomposition of the Laplacian, it is likely to be satisfied if Condition i) holds. Indeed, for most graphs, except those that are either disconnected or have some symmetries (e.g., unweighted path or grid graphs), any sets such that $|\mathcal{S}| \geq f$ are uniqueness sets. Thus, similar to most practical sampling methods [2, 57, 59, 5], our sampling algorithms are not designed to return uniqueness sets satisfying Condition ii), which guarantee exact recovery, and instead, we assume that Condition i) is sufficient to guarantee exact recovery.

We consider the signal model

$$\mathbf{f} = \mathbf{x} + \mathbf{n}, \tag{1.2}$$

[1]We do not consider cases where signals are not bandlimited but can be sampled and reconstructed (refer to [67] and references therein). Exploring more general models for signal sampling is left for future work.

where $\mathbf{x}$ is bandlimited and $\mathbf{n}$ is an n-dimensional noise vector. The reconstruction from the sampled signal $\mathbf{f}_{\mathcal{S}} = \mathbf{x}_{\mathcal{S}} + \mathbf{n}_{\mathcal{S}}$ is then:

$$\hat{\mathbf{f}} = \mathbf{U}_{\mathcal{F}}(\mathbf{U}_{\mathcal{SF}}^T\mathbf{U}_{\mathcal{SF}})^{-1}\mathbf{U}_{\mathcal{SF}}^T(\mathbf{x}_{\mathcal{S}} + \mathbf{n}_{\mathcal{S}}). \tag{1.3}$$

This process is summarized in Figure 1.1. The original unknown signal is $\mathbf{f}$. After sampling it, we know a few signal values corresponding to $\mathbf{f}_{\mathcal{S}}$. Using those values, we reconstruct the signal, represented as $\hat{\mathbf{f}}$. Since (1.1) allows us to reconstruct $\mathbf{x}$ exactly, the error in the reconstructed signal is:

$$\hat{\mathbf{f}} - \mathbf{x} = \mathbf{U}_{\mathcal{F}}(\mathbf{U}_{\mathcal{SF}}^T\mathbf{U}_{\mathcal{SF}})^{-1}\mathbf{U}_{\mathcal{SF}}^T\mathbf{n}_{\mathcal{S}}. \tag{1.4}$$

The expected value of the corresponding error matrix, $(\hat{\mathbf{f}} - \mathbf{x})(\hat{\mathbf{f}} - \mathbf{x})^T$, is

$$\mathbb{E}[(\hat{\mathbf{f}} - \mathbf{x})(\hat{\mathbf{f}} - \mathbf{x})^T] = \mathbf{U}_{\mathcal{F}}(\mathbf{U}_{\mathcal{SF}}^T\mathbf{U}_{\mathcal{SF}})^{-1}\mathbf{U}_{\mathcal{SF}}^T E[\mathbf{n}_{\mathcal{S}}\mathbf{n}_{\mathcal{S}}^T]\mathbf{U}_{\mathcal{SF}}(\mathbf{U}_{\mathcal{SF}}^T\mathbf{U}_{\mathcal{SF}})^{-1}\mathbf{U}_{\mathcal{F}}{}^T. \tag{1.5}$$

Each choice of a sampling set $\mathcal{S}$ leads to a different error covariance matrix in (1.5). The sampling problem consists of finding the sampling set $\mathcal{S}$ that optimizes a scalar function h of the error covariance matrix:

$$\mathcal{S}^* = \arg\max_{\mathcal{S}} h\left(\mathbb{E}[(\hat{\mathbf{f}} - \mathbf{x})(\hat{\mathbf{f}} - \mathbf{x})^T]\right) \tag{1.6}$$

In the subsequent chapters, we will study the sampling problem for different functions h of the error covariance matrix and under different assumptions on the signal and the noise distributions.

Chapter 2

Practical graph signal sampling with log-linear size scaling

2.1 Introduction

Similar to traditional signals, a smooth graph signal can be sampled by making observations on a few graph nodes so that the signal at the remaining (non-observed) nodes can be estimated [14, 2, 67]. For this, we need to choose a set of vertices, $\mathcal{S}$, called the sampling set, on which we observe the signal values with the goal of predicting signal values on the other vertices (the complement of $\mathcal{S}$, $\mathcal{S}^c$). In the presence of noise, some sampling sets lead to better signal reconstructions than others: the goal of *sampling set selection* is to find the best such sampling set.

For traditional discrete signals such as images and audio, downsampling by an integer factor often works well because of the implicit ordering and regular spacing in the signals (e.g., observing every other sample in a discrete-time signal). Such a structure with ordered and evenly spaced out locations of the discretized signal is unavailable for most graph signals. As a result, the best sampling set is also unknown. Accurately reconstructing graph signals from observed samples usually relies on the assumption that the underlying signal is smooth. Intuitively, this means that signal values for neighboring vertices are not drastically different. This is a reasonable assumption in various scenarios, such as sensor networks modeling

temperature distribution, graph signals representing labels in semi-supervised learning, or preferences in social networks. This makes it possible to reconstruct these graph signals from a few observed values [54].

As we saw in Chapter 2, a common model for smooth graph signals assumes that most of their energy is localized in the subspace spanned by a subset of eigenvectors of the graph Laplacian or other graph operators [63]. Thus, the problem of selecting the best sampling set naturally translates to the problem of selecting a submatrix of the matrix of eigenvectors of the graph Laplacian [2]. Specifically, the problem reduces to a row/column subset selection similar to the linear measurement sensor selection problem [35]. In the graph signal sampling context, several papers leverage this knowledge to propose novel algorithms — [62, 14, 69, 68, 13, 71]. We refer the reader to [67] for a recent comprehensive literature review on this topic.

However, to solve the graph sampling set selection problem, row/column selection needs to be applied to the matrix of eigenvectors of the graph Laplacian (or those of some other suitable graph operator). The corresponding eigendecomposition is an $O(n^3)$ operation for an $n \times n$ matrix[1]. This makes it impractical for large graphs in machine learning, social networks, and other applications, for which the cost of eigendecomposition would be prohibitive. Thus, methods that solve this subset selection problem without explicitly requiring eigendecomposition are valuable.

2.1.1 Prior work

We can group sampling set selection methods into two main classes based on whether they require eigendecomposition or not. Some methods compute the *full eigendecomposition* [62, 14, 69], or instead require a sequential eigendecomposition, where one eigenvector is computed at each step [2]. Alternatively, *eigendecomposition-free* methods do not make use

[1]In practice if the signal is bandlimited to the lowest f frequencies, only f eigenvectors need to be computed, but even this can be a complex problem (e.g., a signal bandlimited to the top 10% frequencies of a graph with millions of nodes). We describe these as *full decomposition methods* for simplicity, even though in practice only a subset of eigenvectors is needed.

of an eigendecomposition of the Laplacian matrix [57, 71, 68, 59] and are usually faster. Weighted Random Sampling (WRS) [57] is the fastest method but provides only guarantees on average performance, which means that it may exhibit poor reconstruction accuracy for specific instances. It also needs more samples to match the reconstruction accuracy of other eigendecomposition-free methods. Among eigendecomposition-free methods discussed in [67], Neumann series-based sampling [71] has a high computational complexity, Binary Search with Gershgorin Disc Alignment (BS-GDA) [5] has low computational complexity for smaller graphs, but cannot compete with WRS for large graphs, and Localization operator based Sampling Set Selection (LSSS) [59] achieves good performance but requires some parameter tuning to achieve optimal performance. Our proposed method can overcome these limitations: similar to [71, 5, 59], it is eigendecomposition-free, but it has complexity closer to WRS while requiring fewer parameters to tune than WRS.

Other recently proposed sampling algorithms are eigendecomposition-free but involve a different setup than what we consider in this chapter. For example, the error diffusion sampling algorithm (Algorithm 5 from [50]) achieves complexity comparable to WRS. However, the sampling set and the number of samples chosen depend on the vertex numbering in the graph, which has to be done independently of the algorithm in question. In [50], no specific vertex numbering suitable for Algorithm 5 was recommended. A random vertex numbering algorithm would be fast but may lead to suboptimal sampling set choices (similar to what may happen with random sampling). Thus, more research may be needed to identify efficient numbering algorithms. Note that other blue noise sampling algorithms [51] do not require vertex numbering, but they involve distance computations on the graph similar to Distance-Coherence(DC) in [33]. In contrast, our proposed algorithm, AVM, is independent of the vertex numbering of the graph and does not require distance computations. As another example, the algorithms proposed in [7] and [1] are designed for sampling clustered piecewise constant graph signals. However, this chapter focuses on a bandlimited smoothness model for graph signals, with topologies not limited to clustered graphs.

2.1.2 Motivation

To motivate our methods consider first WRS, where vertices are sampled with a probability proportional to their *squared local coherence* [57]. However, selecting vertices having the highest coherence may not result in the best sampling set because some vertices may be "redundant" (e.g. if they are close to each other on the graph). Other sampling algorithms [59] improve performance by selecting vertices based on importance but avoid redundancy by minimizing a notion of the overlapped area between functions centered on the sampled vertices.

In our preliminary work [33], we proposed the Distance-Coherence (DC) algorithm, which mitigates the effect of redundancy between vertices by adding new vertices to the sampling set only if they are at a sufficient distance on the graph from the previously selected nodes. While this can eliminate redundancy, it leads to higher computation costs since distance computation is expensive. As an alternative, in this chapter, we propose a novel algorithm, Approximate Volume Maximization (AVM), that replaces the distance computation with a filtering operation. Loosely speaking, our proposed scheme in AVM precomputes squared coherences as [57], with an additional criterion to maintain separation between selected vertices using a filtering operation. The resulting complexity (see Section 2.3.4) has a log-linear dependence on the number of edges in a connected graph. The log-linear dependence is desired because it is similar to that of WRS, which is the fastest among algorithms in the literature that use spectral information, and second only to unweighted random sampling from [57] in terms of overall speed. AVM can also be considered an efficient approximation to the D-optimality criterion [4]. In this chapter, we review the main concepts in DC, introduce AVM, and demonstrate its benefits over existing algorithms.

2.1.3 Contributions

Our main contributions are:

1. We describe our distance-based sampling DC algorithm (Section 2.3) to illustrate how to balance the frequency and vertex domain information of graphs for sampling. DC provided us with key ideas to develop the AVM algorithm and can potentially serve as the basis for hybrid algorithms.

2. We introduce a new algorithm, AVM (Algorithm 2), which can be used for any graph size or topology while requiring only a few parameters to tune. Moreover, changing the values of these parameters allows us to achieve different trade-offs between speed and accuracy.

3. Using the framework of volume based sampling (Section 2.3), we interpret a series of algorithms — exact greedy [69], WRS, Spectral Proxies (SP) [2], LSSS, DC, and our proposed AVM as variations of the volume maximization problem formulation (Section 2.4), and explain critical differences between existing methods and AVM.

4. AVM provides competitive reconstruction performance on a variety of graphs and sampling scenarios, improving reconstruction signal-to-noise ratio (SNR) over WRS by at least 0.6dB and frequently significantly higher (e.g., 2dB) — Section 2.5. The practicality of AVM is apparent for larger graph sizes (e.g., of the order of a hundred thousand nodes). In the graph sizes chosen for some of our experiments (see Section 2.5.1.4), other state-of-the-art algorithms such as SP, LSSS and BS-GDA often fail, while a complete execution is always possible for AVM. At graph sizes small enough for the other algorithms to be applied, AVM is at least 2.5 times and often orders of magnitude faster compared to state-of-the-art algorithms such as SP, LSSS and BS-GDA, while sacrificing less than 0.01dB of reconstruction SNR — Section 2.6. We explain these advantages in terms of complexity towards the end of Section 2.3 by showing that compared to WRS, the additional computations needed by AVM scale linearly as a function of the number of edges in a connected graph.

As a summary, our proposed AVM sampling algorithm has complexity comparable to the WRS sampling algorithm along with a significantly better reconstruction accuracy. It achieves this without requiring prior knowledge of the signal bandwidth and can be used for different graphs while requiring a few easy-to-tune parameters.

2.2 Problem setup

2.2.1 Formulation

We follow the formulation of Section 1.2.1, with signal model (1.2) and least squares reconstruction from the signal samples as in (1.5)(see Section 1.2.1). If we assume further individual noise entries to be independent with zero mean and equal variance, the expected value, the error covariance matrix from (1.5) reduces to

$$\mathbf{K}_e = \mathbb{E}[(\hat{\mathbf{f}} - \mathbf{x})(\hat{\mathbf{f}} - \mathbf{x})^T] = c\mathbf{U}_{\mathcal{F}}(\mathbf{U}_{\mathcal{SF}}^T\mathbf{U}_{\mathcal{SF}})^{-1}\mathbf{U}_{\mathcal{F}}{}^T, \tag{2.1}$$

for a constant c. Different metrics of the reconstruction error $\hat{\mathbf{f}} - \mathbf{x}$ can be optimized by maximizing a function $h : M_{n,n}(\mathbb{R}) \to \mathbb{R}$ of $\mathbf{K}_e$, where $M_{n,n}(\mathbb{R})$ is an $n \times n$ matrix of real numbers. Since $\mathbf{K}_e$ is a function of the sampling set $\mathcal{S}$, we wish to find an $\mathcal{S}$ that maximizes a function of $\mathbf{K}_e$:

$$\mathcal{S} = \underset{\mathcal{S}\subset\mathcal{V},|\mathcal{S}|=s}{\arg\max}\, h\left(\mathbf{U}_{\mathcal{F}}(\mathbf{U}_{\mathcal{SF}}^T\mathbf{U}_{\mathcal{SF}})^{-1}\mathbf{U}_{\mathcal{F}}^T\right). \tag{2.2}$$

Note that the set $\mathcal{S}$ achieving optimality under general criteria in the form of (2.2) is a function of $\mathcal{F}$, so that $\mathcal{S}$ is optimized for reconstruction with that particular bandwidth support $\mathcal{F}$. While typically we do not know the bandwidth of the original signal, in what follows, we assume that a specific bandwidth for reconstructing the signal has been given. This assumption can be relaxed, as will be shown in Chapter 4.

A particular choice $h(\mathbf{K}_e)$ of interest is $h(\mathbf{K}_e) = 1/\operatorname{pdet}(\mathbf{K}_e)$, where $\operatorname{pdet}(\cdot)$ is the pseudo determinant [45]. Since $\mathbf{K}_e$ is singular, we used the pseudo-determinant instead of the

determinant. The pseudo-determinant differs from the determinant in that it is a product of only the non-zero eigenvalues instead of all eigenvalues. With our choice of $h(\cdot)$, (2.2) is equivalent to the following maximization:

$$\mathcal{S} = \underset{\mathcal{S}\subset\mathcal{V},|\mathcal{S}|=s}{\arg\max} \det(\mathbf{U}_{\mathcal{S}\mathcal{F}}^T \mathbf{U}_{\mathcal{S}\mathcal{F}}). \tag{2.3}$$

This is also known as the D-optimality criterion. Maximizing the determinant leads to minimizing the confidence interval of the solution $\hat{\mathbf{f}}$ [4], as will be seen in Appendix B. As a further advantage, the D-optimal objective leads to a novel unified view of different types of sampling algorithms proposed in the literature (Section 2.4.3). Moreover, the D-optimal objective is necessary for the approximations we need to develop algorithms achieving efficient eigendecomposition-free subset selection.

Sampling algorithms are designed to implicitly or explicitly optimize the sampling set for a particular bandwidth support. In this chapter, we denote by $\mathcal{R}$ the bandwidth support assumed by a sampling algorithm. $\mathcal{R}$ can be equal to the reconstruction bandwidth support $\mathcal{F}$, in which case the objective (2.3) can be rewritten as:

$$\mathcal{S} = \underset{\mathcal{S}\subset\mathcal{V},|\mathcal{S}|=s}{\arg\max} \det(\mathbf{U}_{\mathcal{S}\mathcal{R}}^T \mathbf{U}_{\mathcal{S}\mathcal{R}}), \quad \text{with} \qquad \mathcal{R} = \mathcal{F}. \tag{2.4}$$

However, there are advantages to choosing a different $\mathcal{R} \neq \mathcal{F}$. For example, if we consider $\mathcal{R} = \{1, \cdots, s\}$ so that $|\mathcal{R}| = |\mathcal{S}|$, we can rewrite the objective function (2.4) without changing its value, by permuting the order of the matrices:

$$\mathcal{S} = \underset{\mathcal{S}\subset\mathcal{V},|\mathcal{S}|=s}{\arg\max} \det(\mathbf{U}_{\mathcal{S}\mathcal{R}} \mathbf{U}_{\mathcal{S}\mathcal{R}}^T). \tag{2.5}$$

Essentially, instead of using the reconstruction frequency f as the sampling frequency, we use the number of samples requested, s, as a proxy for the sampling frequency. As we will see, this new form of (2.5) is easier to interpret and use.

Since choosing $|\mathcal{R}| = |\mathcal{S}|$ is required, it raises concerns about the optimality of our sampling set for the original objective function. This issue will be discussed in Appendix B.

2.2.2 Solving D-optimal objectives

As just discussed, D-optimal subsets for matrices are determinant maximizing subsets. The determinant measures the volume, and selecting a maximum volume submatrix is an NP-Hard problem [15]. Nearly-optimal methods have been proposed in the literature [29], [19], but these are based on selecting a submatrix of rows or columns of a known matrix. Similarly, in the graph signal processing literature, several contributions [69, 13] develop algorithms for D-optimal selection assuming that $\mathbf{U}$ is available. In contrast, the main novelty of our work is to develop greedy algorithms for approximate D-optimality, i.e., solving (2.3) without requiring explicit eigendecomposition to obtain $\mathbf{U}$. This is made possible by specific characteristics of our problem to be studied next.

Among graph signal sampling approaches that solve the D-optimal objective, the closest to our work is the application of Wilson's algorithm for Determinantal Point Process (WDPP) [68], which similarly does not require explicitly computing $\mathbf{U}$. However, our proposed technique, AVM, achieves this goal in a different way and leads to better performance. Specifically, WDPP avoids eigendecomposition while approximating the bandlimited kernel using Wilson's marginal kernel upfront [68]. This is a one-time approximation, which does not have to be updated each time nodes are added to the sampling set. This approach relies on a relation between Wilson's marginal kernel and random walks on the graph, leading to a probability of choosing sampling sets that is proportional to the determinant [68]. In contrast, AVM solves an approximate optimization at each iteration, i.e., each time a new vertex is added to the existing sampling set. Thus, AVM optimizes the cost function (2.3) at every iteration as opposed to WDPP which aims to optimize the expected value of the cost function.

The WDPP and AVM algorithms differ in their performance as well. AVM is a greedy algorithm, and the performance greedy determinant maximization algorithms is known to lie within a factor of the maximum determinant [15]. In contrast, WDPP samples with probabilities proportional to the determinants, so that its average performance depends on the distribution of the determinants. In fact, for certain graph types in [68], we observe that WDPP has a worse average performance than WRS. In comparison, in our experiments, for a wide variety of graph topologies and sizes, AVM consistently outperforms WRS [57] in terms of the average reconstruction error.

2.3 Efficient sampling set selection algorithms

In what follows we assume that the conditions for equivalence between the two objective function forms (2.4) and (2.5) are verified, so that we focus on solving (2.5).

2.3.1 Incremental subset selection

The bandwidth support for the purpose of sampling is assumed to be $\mathcal{R} = \{1, \cdots, s\}$. Let us start by defining $\mathbf{d}_v$, the signal obtained by applying an ideal low pass filter to the Kronecker delta function $\boldsymbol{\delta}_v$ localized at vertex v:

$$\mathbf{d}_v = \mathbf{U}_{\mathcal{R}} \mathbf{U}_{\mathcal{R}}^T \boldsymbol{\delta}_v. \tag{2.6}$$

With this definition, the objective in (2.5) can be written as:

$$\begin{aligned}\det(\mathbf{U}_{\mathcal{SR}}\mathbf{U}_{\mathcal{SR}}^T) &= \det(\mathbf{U}_{\mathcal{SR}}\mathbf{U}_{\mathcal{R}}^T\mathbf{U}_{\mathcal{R}}\mathbf{U}_{\mathcal{SR}}^T) \\ &= \det(\mathbf{I}_{\mathcal{S}}^T\mathbf{U}_{\mathcal{R}}\mathbf{U}_{\mathcal{R}}^T\mathbf{U}_{\mathcal{R}}\mathbf{U}_{\mathcal{R}}^T\mathbf{I}_{\mathcal{S}}) \\ &= \det\left(\begin{bmatrix}\mathbf{d}_1 & \cdots & \mathbf{d}_s\end{bmatrix}^T\begin{bmatrix}\mathbf{d}_1 & \cdots & \mathbf{d}_s\end{bmatrix}\right) \\ &= \mathrm{Vol}^2(\mathbf{d}_1, \cdots, \mathbf{d}_s). \end{aligned} \tag{2.7}$$

Here $\mathbf{I}_{\mathcal{S}}$ represents the submatrix obtained by selecting the columns of $\mathbf{I}$ indexed by set $\mathcal{S}$. Thus, maximizing the determinant $\det(\mathbf{U}_{\mathcal{SR}}\mathbf{U}_{\mathcal{SR}}^T)$ is equivalent to maximizing $\mathrm{Vol}(\mathbf{d}_1, \cdots, \mathbf{d}_s)$, and as a consequence the set maximizing (2.7) also maximizes (2.5).

In an iterative algorithm where the goal is to select s samples, consider a point where $m < s$ samples have been selected and we have to choose the next sample from among the remaining vertices. Throughout the rest of the chapter, we denote the sampling set at the end of the m^{th} iteration of an algorithm by $\mathcal{S}_m$. Given the first m chosen samples we define $\mathbf{D}_m = [\mathbf{d}_1 \cdots \mathbf{d}_m]$ and the space spanned by the vectors in $\mathbf{D}_m$ as $\mathcal{D}_m = \mathrm{span}(\mathbf{d}_1, \cdots, \mathbf{d}_m)$. We denote $\mathcal{D}$ and $\mathbf{D}$ at the end of the m^{th} iteration of an algorithm by $\mathcal{D}_m$ and $\mathbf{D}_m$. Note that both $\mathcal{D}_m$ and $\mathbf{D}_m$ are a function of the choice of the sampling bandwidth support $\mathcal{R}$. Next, the best column $\mathbf{d}_v$ to be added to $\mathbf{D}_m$ should maximize:

$$\det\left(\begin{bmatrix}\mathbf{D}_m & \mathbf{d}_v\end{bmatrix}^T\begin{bmatrix}\mathbf{D}_m & \mathbf{d}_v\end{bmatrix}\right) = \det\left(\begin{bmatrix}\mathbf{D}_m^T\mathbf{D}_m & \mathbf{D}_m^T\mathbf{d}_v \\ \mathbf{d}_v^T\mathbf{D}_m & \mathbf{d}_v^T\mathbf{d}_v\end{bmatrix}\right) \tag{2.8a}$$

$$= \det(\mathbf{D}_m^T\mathbf{D}_m)\det(\mathbf{d}_v^T\mathbf{d}_v - \mathbf{d}_v^T\mathbf{D}_m(\mathbf{D}_m^T\mathbf{D}_m)^{-1}\mathbf{D}_m^T\mathbf{d}_v) \tag{2.8b}$$

$$= \det(\mathbf{D}_m^T\mathbf{D}_m)(\|\mathbf{d}_v\|^2 - \mathbf{d}_v^T\mathbf{D}_m(\mathbf{D}_m^T\mathbf{D}_m)^{-1}\mathbf{D}_m^T\mathbf{d}_v) \tag{2.8c}$$

$$= \det(\mathbf{D}_m^T\mathbf{D}_m)\left(\|\mathbf{d}_v\|^2 - \|\mathbf{P}_{\mathcal{D}_m}\mathbf{d}_v\|^2\right). \tag{2.8d}$$

The effect on the determinant of adding a column to $\mathbf{D}_m$ can be represented according to a multiplicative update (Section 11.2 [4]) in our D-optimal design. (2.8b) follows from [31] (Section 0.8.5 in Second Edition), while (2.8d) follows because

$$\mathbf{P}_{\mathcal{D}_m} = \mathbf{D}_m(\mathbf{D}_m^T\mathbf{D}_m)^{-1}\mathbf{D}_m^T \tag{2.9}$$

is a projection onto the space $\mathcal{D}_m$. Direct greedy determinant maximization requires selecting a vertex that maximizes the update term in (2.8c):

$$v^* = \arg\max_{v\in\mathcal{S}_m^c} \left(\|\mathbf{d}_v\|^2 - \mathbf{d}_v^T\mathbf{D}_m(\mathbf{D}_m^T\mathbf{D}_m)^{-1}\mathbf{D}_m^T\mathbf{d}_v\right) \tag{2.10}$$

over all possible vertices $v \in \mathcal{S}_m^c$, which requires the expensive computation of $(\mathbf{D}_m^T\mathbf{D}_m)^{-1}$.

The first step towards a greedy incremental vertex selection is estimating the two components, $\|\mathbf{d}_v\|^2$ and $\mathbf{d}_v^T\mathbf{D}_m(\mathbf{D}_m^T\mathbf{D}_m)^{-1}\mathbf{D}_m^T\mathbf{d}_v$, of the multiplicative update. The first term $\|\mathbf{d}_v\|^2$ is the *squared coherence* introduced in [57], which is estimated here using the same techniques as in [57], and is defined as

$$\|\mathbf{d}_v\|^2 = \left\|\mathbf{U}_\mathcal{R}\mathbf{U}_\mathcal{R}^T\boldsymbol{\delta}_v\right\|^2 = \left\|\mathbf{U}_\mathcal{R}^T\boldsymbol{\delta}_v\right\|^2. \tag{2.11}$$

For the second term, the projection interpretation of (2.8d) will be useful to develop approximations to reduce complexity. Additionally, we will make use of the following property of our bandlimited space to develop an approximation.

Lemma 2.1. *The space of bandlimited signals* $\mathrm{span}(\mathbf{U}_\mathcal{R})$ *equipped with the dot product is a reproducing kernel Hilbert space (RKHS).*

Solution. Defining the inner product for signals $\mathbf{f}, \mathbf{g} \in \mathrm{span}(\mathbf{U}_\mathcal{R})$ as

$$\langle \mathbf{f}, \mathbf{g}\rangle = \sum_i f_i g_i, \tag{2.12}$$

span($\mathbf{U}_\mathcal{R}$) is a Hilbert space. A Hilbert space further needs an existing reproducing kernel to be an RKHS. Towards that end, consider a mapping to our bandlimited space $\phi : \mathbb{R}^n \to$ span($\mathbf{U}_\mathcal{R}$) given as:

$$\phi(\mathbf{x}) = \mathbf{U}_\mathcal{R}\mathbf{U}_\mathcal{R}^T\mathbf{x}, \tag{2.13}$$

where $\phi(\mathbf{x})$ is the orthogonal projection of $\mathbf{x}$ onto span($\mathbf{U}_\mathcal{R}$). A function $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ that uses that mapping and the scalar product in our Hilbert space is:

$$\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n, \qquad K(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle. \tag{2.14}$$

Now using Theorem 4 from [8], K is a reproducing kernel for our Hilbert space, and using Theorem 1 from [8] we conclude that our bandlimited space of signals is an RKHS. □

Corollary 2.1. *The dot product of a bandlimited signal* $\mathbf{f} \in$ span($\mathbf{U}_\mathcal{R}$) *with a filtered delta* $\mathbf{d}_v$ *is* $\mathbf{f}(v)$*, the entry at node* v *of signal* $\mathbf{f}$*:*

$$\langle \mathbf{f}, \mathbf{d}_v \rangle = \mathbf{f}(v). \tag{2.15}$$

Solution. The dot product $\langle \mathbf{f}, \mathbf{d}_v \rangle$ in our RHKS can be seen as the evaluation functional of $\mathbf{f}$ at the point v. Using the definition of reproducing kernel K, since $\mathbf{f} \in$ span($\mathbf{U}_\mathcal{R}$) we have that $\phi(\mathbf{f}) = \mathbf{f}$ and thus (using Section 2 definition and Theorem 1 Property b from [8]):

$$\langle \mathbf{f}, \mathbf{d}_v \rangle = \langle \phi(\mathbf{f}), \phi(\boldsymbol{\delta}_v) \rangle = \langle \mathbf{f}, \phi(\boldsymbol{\delta}_v) \rangle. \tag{2.16}$$

An evaluation functional $\langle \mathbf{f}, \phi(\mathbf{x}) \rangle$ for $\mathbf{f}$ bandlimited can be simplified as:

$$\begin{aligned} \langle \mathbf{f}, \phi(\mathbf{x}) \rangle = \langle \mathbf{f}, \mathbf{U}_\mathcal{R}\mathbf{U}_\mathcal{R}^T\mathbf{x} \rangle &= \langle \mathbf{U}_\mathcal{R}\mathbf{U}_\mathcal{R}^T\mathbf{f}, \mathbf{x} \rangle \\ &= \langle \mathbf{f}, \mathbf{x} \rangle. \end{aligned} \tag{2.17}$$

Thus, from (2.16) and (2.17):

$$\langle \mathbf{f}, \mathbf{d}_v \rangle = \langle \mathbf{f}, \phi(\boldsymbol{\delta}_v) \rangle = \langle \mathbf{f}, \boldsymbol{\delta}_v \rangle = \mathbf{f}(v).$$

□

As a consequence of (2.15), if $\mathbf{f} = \mathbf{d}_w$ we have:

$$\langle \mathbf{d}_w, \mathbf{d}_v \rangle = \mathbf{d}_v(w) = \mathbf{d}_w(v). \tag{2.18}$$

2.3.2 Approximation through distances

We start by proposing a distance-based algorithm (DC) based on the updates we derived in (2.8). While in principle those updates are valid only when $s = f$, in DC we apply them even when $s > f$. We assume f is known. We take the bandwidth support for the purpose of sampling to be $\mathcal{R} = \{1, \cdots, f\}$, which is the same as the signal reconstruction bandwidth support $\mathcal{F}$. To maximize the expression in (2.8d) we would like to select nodes that have:

1. Large squared graph coherence $\|\mathbf{d}_v\|^2$ with respect to f frequencies (the first term in (2.8d), which is a property of each node and independent of $\mathcal{S}_m$), and

2. small squared magnitude of projection onto the subspace $\mathcal{D}_m$ (which does depend on $\mathcal{S}_m$) $\left\|\mathbf{P}_{\mathcal{D}_m}\hat{\mathbf{d}}_v\right\|^2$ and thus would increase (2.8d).

$\|\mathbf{d}_v\|^2$ varies between 0 and 1, taking the largest values at vertices that are poorly connected to the rest of the graph [57]. On the other hand, the subspace $\mathcal{D}_m$ is a linear combination of filtered delta signals $\mathbf{d}_v$ corresponding to the vertices in $\mathcal{S}_m$. The energy of a signal $\mathbf{d}_v$ is expected to decay as a function of distance from v. Therefore, for a particular energy $\|\mathbf{d}_v\|^2$, a vertex whose inner-product is minimum with the filtered delta signals corresponding to vertices in $\mathcal{S}_m$ will have a small $\|\mathbf{P}_{\mathcal{D}_m}\mathbf{d}_v\|^2$. Thus, $\mathbf{d}_v$ for a vertex farther away from the already sampled vertices will have lesser overlap with their corresponding $\mathbf{d}_w$,

Algorithm 1 Distance-coherence (DC)

function DC($\mathbf{L}, s, f, d, \epsilon$)
  $\mathcal{S} \leftarrow \emptyset$,
  $\Delta \leftarrow 0.9$
  $\mathcal{R} \leftarrow \{1, \cdots, f\}$
  $\left[\|\mathbf{d}_1\|^2, \cdots, \|\mathbf{d}_n\|^2\right], \lambda_f, \text{coeffs} \leftarrow$ COMPUTE COHERENCE($\mathbf{L}, n, f, \epsilon$)
  while $|\mathcal{S}| < s$ **do**
    $\mathcal{V}_d(\mathcal{S}) \leftarrow \{v \in \mathcal{S}^c | d(\mathcal{S}, v) > \Delta . \max_{u \in \mathcal{V}} d(\mathcal{S}, u)\}$
    $v* \leftarrow \arg\max_{v \in \mathcal{V}_d(\mathcal{S})} \|\mathbf{d}_v\|^2$
    $\mathcal{S} \leftarrow \mathcal{S} \cup v^*$
  end while
  return $\mathcal{S}$
end function

which also span the space $\mathcal{D}_m$. Therefore for a vertex $v \in \mathcal{S}_m^c$ whose "distance" to the vertices $\mathcal{S}_m$ is large, the corresponding $\mathbf{d}_v$ will have a small projection on the space $\mathcal{D}_m$.

Our proposed DC algorithm (Algorithm 1) consists of two stages; it first identifies vertices that are at a sufficiently large distance from already chosen vertices in $\mathcal{S}_m$. This helps in reducing the set size, by including only those $v \in \mathcal{V}_d(\mathcal{S}_m)$ that are expected to have a small $\|\mathbf{P}_{\mathcal{D}_m}\mathbf{d}_v\|^2$. From among those selected vertices it chooses the one with the largest value of $\|\mathbf{d}_v\|^2$.

The nodes with a sufficiently large distance from S are defined as follows

$$\mathcal{V}_d(\mathcal{S}_m) = \{v \in \mathcal{S}_m^c | d(\mathcal{S}_m, v) > \Delta . \max_{u \in \mathcal{V}} d(\mathcal{S}_m, u)\},$$

where $\Delta \in [0\ 1]$,

$$d(\mathcal{S}_m, v) = \min_{u \in \mathcal{S}_m} d(u, v) \tag{2.19}$$

and d is the geodesic distance on the graph. The distance between two adjacent vertices i, j is given by

$$d(i, j) = 1/w(i, j). \tag{2.20}$$

The parameter Δ is used to control how many nodes can be included in $\mathcal{V}_d(\mathcal{S}_m)$. With a small Δ, more nodes will be considered at the cost of increased computations; while with a

large Δ, lesser nodes will be considered with the benefit of reduced computations. For small Δ, the DC algorithm becomes similar to WRS, except the vertices are picked in the order of their squared coherence, rather than randomly with probability proportional to their squared coherence as in [57].

The DC algorithm (Algorithm 1) provides a proof-of-concept of the volume maximization interpretation using coherences and distances for sampling. However, it involves obtaining geodesic distances on the graph, which is a computationally expensive task. Eliminating this bottleneck is possible by employing simpler distances such as hop distance, or doing away with distances altogether. We leave the first approach open for future work and develop the second approach here as the AVM algorithm (Algorithm 2).

2.3.3 Approximate volume maximization (AVM) through inner products

In this section, we use a more efficient technique based on filtering, instead of computing the distance between nodes as in DC, here we assumed that the signal bandwidth for sampling was the same as the reconstruction bandwidth f. In practice, we do not know the signal bandwidth and thus also do not know the reconstruction bandwidth. To remedy this, in AVM, we use the number of samples, s, as a proxy for the signal's bandwidth. As a result, the bandwidth support used for sampling is $\mathcal{R} = \{1, \cdots, s\}$. We explained the reason behind this decoupling of the sampling and the reconstruction bandwidth in Section 2.2.1 through equations (2.3) and (2.4). AVM has the following advantages:

- We can use the optimization framework we defined in Section 2.3.1.
- By not assuming knowledge of the reconstruction bandwidth for sampling, AVM models real-world sampling scenarios better.
- For our chosen set of samples, we do not have to limit ourselves to one reconstruction bandwidth.

Algorithm 2 Approximate volume maximization (AVM)

function AVM($\mathbf{L}, s, d, \epsilon$)
  $\mathcal{S} \leftarrow \emptyset$
  $\mathcal{R} \leftarrow \{1, \cdots, s\}$
  $\left[\|\mathbf{d}_1\|^2, \cdots, \|\mathbf{d}_n\|^2\right], \lambda_s, \text{coeffs} \leftarrow$ COMPUTE COHERENCE($\mathbf{L}, n, s, \epsilon$)
  while $|\mathcal{S}| < s$ **do**
    $v* \leftarrow \arg\max_{v \in \mathcal{S}^c} \|\mathbf{d}_v\|^2 - \sum_{w \in \mathcal{S}} \frac{\mathbf{d}_w^2(v)}{\|\mathbf{d}_w\|^2}$
    $\mathbf{d}_{v^*} \leftarrow$ FILTER($\mathbf{L}, \text{coeffs}, \boldsymbol{\delta}_{v^*}$)
    $\mathcal{S} \leftarrow \mathcal{S} \cup v^*$
  end while
  return $\mathcal{S}$
end function

AVM successively simplifies the greedy volume maximization step (2.10) in four stages:

1. Approximating squared coherences, $\|\mathbf{d}_v\|^2$
2. Approximating the inner product matrix, $\mathbf{D}_m^T \mathbf{D}_m$
3. Computing low pass filtered delta signals, $\|\mathbf{d}_v\|^2$
4. Fast inner product computations, $\langle \mathbf{d}_w, \mathbf{d}_v \rangle$

which we will study next.

2.3.3.1 Approximate squared coherence

Algorithm 2 estimates the squared coherence, $\|\mathbf{d}_v\|^2, v \in \mathcal{S}_m$, using the method of random projections method from Section 4.1 in [57] in the same way as in Algorithm 1. This approach avoids explicitly finding $\mathbf{d}_v$ to compute $\|\mathbf{d}_v\|^2$.

For completeness, we include the approach from [57] to find squared coherences as Function 1. For implementations of the functions APPROXIMATE LARGEST EIGENVALUE, POLYNOMIAL FILTER COEFFICIENTS, and POLYNOMIAL FILTER, which Function 1 calls, we refer the reader to the GSP toolbox [53].

Function 1 Compute coherence [57]

function COMPUTE COHERENCE($\mathbf{L}, n, k, \epsilon$)
  $L \leftarrow \text{round}(10\log(n))$
  $\left[\mathbf{r}^1, \cdots, \mathbf{r}^L\right] \leftarrow [\mathcal{N}(\mathbf{0}_{n\times 1}, \mathbf{I}_{n\times n}), \cdots, \mathcal{N}(\mathbf{0}_{n\times 1}, \mathbf{I}_{n\times n})]$
  $\lambda_n \leftarrow$ APPROXIMATE LARGEST EIGENVALUE($\mathbf{L}$)
  $\underline{\lambda} \leftarrow 0$, $\bar{\lambda} \leftarrow \lambda_n$, $\lambda \leftarrow \lambda_n/2$.
  coeffs $\leftarrow$ POLYNOMIAL FILTER COEFFICIENTS$(0, \lambda_n, \lambda, d)$
  $\left[\mathbf{r}^1_{\text{filt}}, \cdots, \mathbf{r}^L_{\text{filt}}\right] \leftarrow \left[\text{POLYNOMIAL FILTER}(\mathbf{L}, \text{coeffs}, \mathbf{r}^1), \cdots, \text{POLY. FILTER}(\mathbf{L}, \text{coeffs}, \mathbf{r}^L)\right]$
  $SS \leftarrow \sum_{i=1}^{n}\sum_{l=1}^{L}\left(\mathbf{r}^l_{\text{filt}}\right)_i^2$
  while round $(SS) \neq k$ or $|\underline{\lambda} - \bar{\lambda}| > \varepsilon \cdot \bar{\lambda}$ **do**
    if round $(SS) \geq k$ **then**
      $\bar{\lambda} \leftarrow \lambda$.
    else
      $\underline{\lambda} \leftarrow \lambda$.
    end if
    $\lambda \leftarrow (\underline{\lambda} + \bar{\lambda})/2$
    coeffs $\leftarrow$ POLYNOMIAL FILTER COEFFICIENTS$(0, \lambda_n, \lambda, d)$
    $\left[\mathbf{r}^1_{\text{filt}}, \cdots, \mathbf{r}^L_{\text{filt}}\right] \leftarrow \left[\text{POLY. FILTER}(\mathbf{L}, \text{coeffs}, \mathbf{r}^1), \cdots, \text{POLY. FILTER}(\mathbf{L}, \text{coeffs}, \mathbf{r}^L)\right]$
    $SS \leftarrow \sum_{i=1}^{n}\sum_{l=1}^{L}\left(\mathbf{r}^l_{\text{filt}}\right)_i^2$
  end while
  $\left[\|\mathbf{d}_1\|^2, \cdots, \|\mathbf{d}_n\|^2\right] \leftarrow \left[\left(\sum_{l=1}^{L}\left(\mathbf{r}^l_{\text{filt}}\right)_1^2\right), \cdots, \left(\sum_{l=1}^{L}\left(\mathbf{r}^l_{\text{filt}}\right)_n^2\right)\right]/SS$
  return $\left[\|\mathbf{d}_1\|^2, \cdots, \|\mathbf{d}_n\|^2\right], \lambda, \text{coeffs}$
end function

2.3.3.2 Approximate inner product matrix

We know that the volume of the parallelepiped formed by two fixed-length vectors is maximized when the vectors are orthogonal to each other. Now, since vectors that optimize (2.10) also approximately maximize the volume, we expect them to be close to orthogonal. Thus, we approximate $\mathbf{D}_m^T\mathbf{D}_m$ by an orthogonal matrix (Appendix C). That is, assuming that the filtered delta signals corresponding to the previously selected vertices are approximately orthogonal we can write:

$$\mathbf{D}_m^T\mathbf{D}_m \approx \text{diag}\left(\|\mathbf{d}_1\|^2, \cdots, \|\mathbf{d}_m\|^2\right)$$

and

$$(\mathbf{D}_m^T\mathbf{D}_m)^{-1} \approx \operatorname{diag}\left(\frac{1}{\|\mathbf{d}_1\|^2}, \cdots, \frac{1}{\|\mathbf{d}_m\|^2}\right),$$

which leads to an approximation of the determinant:

$$\det\left(\begin{bmatrix}\mathbf{D}_m^T\mathbf{D}_m & \mathbf{D}_m^T\mathbf{d}_v \\ \mathbf{d}_v^T\mathbf{D}_m & \mathbf{d}_v^T\mathbf{d}_v\end{bmatrix}\right) \approx \det(\mathbf{D}_m^T\mathbf{D}_m)\det(\mathbf{d}_v^T\mathbf{d}_v - \mathbf{d}_v^T\hat{\mathbf{D}}_m\hat{\mathbf{D}}_m^T\mathbf{d}_v), \tag{2.21}$$

where $\hat{\mathbf{D}}_m$ is obtained from $\mathbf{D}_m$ by normalizing the columns:

$$\hat{\mathbf{D}}_m = \mathbf{D}_m \operatorname{diag}\left(1/\|\mathbf{d}_1\|, \cdots, 1/\|\mathbf{d}_m\|\right). \tag{2.22}$$

The second term in (2.21) can be written as:

$$\mathbf{d}_v^T\hat{\mathbf{D}}_m\hat{\mathbf{D}}_m^T\mathbf{d}_v = \frac{\langle\mathbf{d}_v, \mathbf{d}_1\rangle^2}{\|\mathbf{d}_1\|^2} + \cdots + \frac{\langle\mathbf{d}_v, \mathbf{d}_m\rangle^2}{\|\mathbf{d}_m\|^2}, \tag{2.23}$$

which would be the signal energy of projected signal $\mathbf{d}_v$ on to $\operatorname{span}(\mathbf{d}_1, \cdots, \mathbf{d}_m)$, if the vectors $\mathbf{d}_1, \cdots, \mathbf{d}_m$ were mutually orthogonal. This is consistent with our assumption that $\mathbf{D}_m^T\mathbf{D}_m$ is approximately diagonal, which would only hold exactly if the vectors form an orthogonal set.

2.3.3.3 Computing low pass filtered delta signals

If $\mathbf{U}$ is known, then computing the low pass filtered delta signal $\mathbf{d}_v$ is straightforward by simply using the ideal low pass filter as in (2.6). However, since we would like to avoid the cost of the eigendecomposition, $\mathbf{U}$ is unknown. A polynomial approximation of the ideal low pass filter with the frequency λ_s can be computed using Function 1. Using this polynomial approximation, $\boldsymbol{\delta}_v$ is filtered to obtain $\mathbf{d}_v$.

2.3.3.4 Fast inner product computations

Maximization of (2.21) requires evaluating the inner products $\langle \mathbf{d}_v, \mathbf{d}_i \rangle$ in (2.23) for all $i \in \mathcal{S}_m$ and all vertices v outside $\mathcal{S}_m$. Suppose we knew $\mathbf{d}_i$ for sampled vertices, $i \in \mathcal{S}_m$, and the inner products from the past iteration. The current $(m+1)^{\text{th}}$ iteration would still need to compute $n - m$ new inner products.

To avoid this computation, we use the inner product property of (2.18), which allows us to simplify (2.23) as follows:

$$\mathbf{d}_v^T \hat{\mathbf{D}}_m \hat{\mathbf{D}}_m^T \mathbf{d}_v = \frac{\mathbf{d}_1^2(v)}{\|\mathbf{d}_1\|^2} + \cdots + \frac{\mathbf{d}_m^2(v)}{\|\mathbf{d}_m\|^2}.$$

By doing this we avoid computing $\langle \mathbf{d}_v, \mathbf{d}_m \rangle$ for $v \in \mathcal{S}^c$ for the newly added vertex m. Thus, there is no need to compute $n - m$ new inner products, while we also avoid computing $\mathbf{d}_v, v \in \mathcal{S}_m^c$. With this, our greedy optimization step becomes:

$$v* \leftarrow \arg\max_{v \in \mathcal{S}_m^c} \|\mathbf{d}_v\|^2 - \sum_{w \in \mathcal{S}_m} \frac{\mathbf{d}_w^2(v)}{\|\mathbf{d}_w\|^2}.$$

2.3.3.5 Summary of approximations

In summary, thanks to the approximations from Section 2.3.3.2 to Section 2.3.3.4, we do not need to compute distances and no longer rely on the choice of a parameter Δ, as in Algorithm 1. Algorithm 2 only requires the following inputs:

1. The number of samples requested, s,
2. The constant c specifying the number of random projections, $cs \log s$,
3. The scalar ϵ specifying the convergence criteria for random projection iterations while computing squared coherences.

The last two inputs are specifically needed by Algorithm 1 in [57], which we use in Step 1 (Section 2.3.3.1) to compute squared coherences.

While the inner product property is defined based on the assumption that we use an "ideal" low pass filter for reconstruction, it can also be used to maximize the volume formed by the samples of more generic kernels — see Appendix D. The approximations proposed in this section towards designing AVM can be justified if they lead to a scalable and fast algorithm. In what follows, we study the computational complexity of AVM to assess its scalability.

2.3.4 Computational complexity of AVM

The computational complexity of AVM depends on the number of vertices and edges in the graph — $|\mathcal{V}|$ and $|\mathcal{E}|$, the degree of the polynomial d, the number of samples s, and the number of iterations T_1 to converge to the right λ_s for computing squared coherences. In practice, we observe that a small number of iterations T_1 are required to converge.

AVM starts by computing squared coherences, with complexity $O(|\mathcal{E}|dT_1 \log|\mathcal{V}|)$. Finding and normalizing filtered signal requires $O(d(|\mathcal{E}| + |\mathcal{V}|))$ computations. Subtraction and finding the maximum requires $O(|\mathcal{V}|)$ computations. We repeat this s times which results in $O(s|\mathcal{V}| + s(|\mathcal{E}| + |\mathcal{V}|)d)$ computations in Stage 2 of AVM. This leads to Algorithm 2 having a computational complexity of $O((|\mathcal{E}| + |\mathcal{V}|)dT_1 \log|\mathcal{V}| + s(|\mathcal{E}| + |\mathcal{V}|)d)$. For a connected graph we know that $|\mathcal{E}| \geq |\mathcal{V}| - 1$, so then the complexity is simply $O(|\mathcal{E}|dT_1 \log|\mathcal{V}| + s|\mathcal{E}|d)$.

2.3.4.1 Dependence on coherence estimation accuracy

Stage 1 is the bottleneck in the AVM algorithm because it involves T_1 iterations to find the squared coherences, with computations in each iteration scaling as $|\mathcal{E}| \log|\mathcal{V}|$, where both the factors $|\mathcal{E}|$ and $|\mathcal{V}|$ scale with the graph size. A limitation in the number of computations we can do at this stage may cap the graph sizes we can consider. In this situation, we note that Stage 1 (computing squared coherences and λ_s) is an approximation, and we could select an alternative approximation requiring fewer computations instead.

2.3.4.2 Dependence on the number of samples

The complexity of sampling algorithms naturally depends on the number of samples requested at the input, and it is reasonable to assume that an ideal sampling algorithm cannot grow sublinearly in complexity as the number of samples increases. This is because simply adding one sample requires $O(1)$ computations. While a sampling algorithm's complexity may grow superlinearly with the number of samples requested — see Table III in [2], algorithms we compare in Section 2.6.2 grow linearly with the number of samples. Additionally, AVM's complexity also scales linearly with the number of samples as the complexity factor $O(s|\mathcal{E}|d)$ suggests.

2.3.4.3 Log-linear dependence on graph size

The computational complexity of $O(|\mathcal{E}|dT_1 \log|\mathcal{V}| + s|\mathcal{E}|d)$ suggests that AVM has a log-linear dependence on the graph size, specifically with a linear dependence on the number of edges and log dependence on the number of vertices. This can further be seen as complexity with log-linear dependence on the number of edges as $O(|\mathcal{E}|dT_1 \log|\mathcal{E}| + s|\mathcal{E}|d)$, but $O(|\mathcal{E}|dT_1 \log|\mathcal{V}| + s|\mathcal{E}|d)$ is a more accurate estimate.

So far, approximations to the volume maximization objective (2.8d) were useful to develop DC and AVM[2] algorithms. In the following sections, we will show how other eigendecomposition-free algorithms can also be interpreted as approximations to the greedy volume maximization objective.

2.4 Volume maximization interpretation of sampling

We next study how existing graph signal sampling methods are related to volume maximization. We start by focusing on the SP algorithm from [2] and show how it can be seen as a volume maximization method. This idea is developed in Section 2.4.1 and Section 2.4.2.

[2]Code: `https://github.com/STAC-USC/Graph-signal-sampling-AVM`

Section 2.4.3 then considers other eigendecomposition-free methods and draws parallels with our volume maximization approach.

2.4.1 The SP algorithm as Gaussian elimination

The SP algorithm is based on the following theorem [2].

Theorem 2.1. *Let* $\mathbf{L}$ *be the combinatorial Laplacian of an undirected graph. For a set* $\mathcal{S}_m$ *of size* m*, let* $\mathbf{U}_{\mathcal{S}_m,1:m}$ *be full rank. Let* $\boldsymbol{\psi}_k^*$ *be zero over* $\mathcal{S}_m$ *and a minimizing signal in the Rayleigh quotient of* L^k *for a positive integer* k*.*

$$\boldsymbol{\psi}_k^* = \underset{\boldsymbol{\psi},\boldsymbol{\psi}(\mathcal{S}_m)=\mathbf{0}}{\arg\min} \frac{\boldsymbol{\psi}^T\mathbf{L}^k\boldsymbol{\psi}}{\boldsymbol{\psi}^T\boldsymbol{\psi}}. \tag{2.24}$$

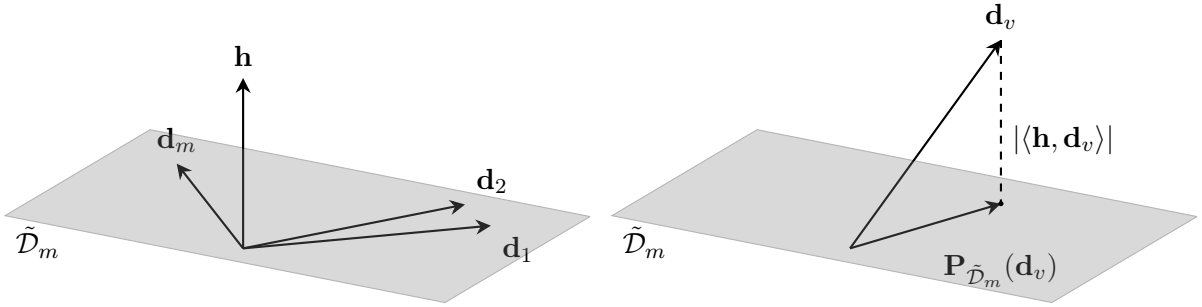
Let the signal $\boldsymbol{\psi}^*$ *be a linear combination of first* $m+1$ *eigenvectors such that* $\boldsymbol{\psi}^*(\mathcal{S}_m) = \mathbf{0}$*. Now if there is a gap between the singular values* $\sigma_{m+2} > \sigma_{m+1}$*, then* $\|\boldsymbol{\psi}_k^* - \boldsymbol{\psi}^*\|_2 \to 0$ *as* $k \to \infty$*.*

Solution. Refer to [2]. For the proof of ℓ_2 convergence, see Appendix A. □

Following (2.24), the step in the SP algorithm that leads to sampling a new vertex is

$$v^* = \underset{v\in\mathcal{S}_m{}^c}{\arg\max}|\boldsymbol{\psi}_k^*|,$$

where $\boldsymbol{\psi}_k^*$ is from (2.24). Consider first the ideal SP algorithm, where $k \to \infty$ and the solution tends to the ideal bandlimited solution. In the ideal case, given a full rank $\mathbf{U}_{\mathcal{S}_m,1:m}$, from Theorem 2.1 we can always get another vertex v such that $\mathbf{U}_{\mathcal{S}_m\cup v,1:m+1}$ is also full rank. Thus, at all iterations, any submatrix of $\mathbf{U}_{\mathcal{S}_m,1:m}$ will have full rank.



(a) Orthogonality of $\mathbf{h}$ with $\tilde{\mathcal{D}}_m$.

(b) Components of $\mathbf{d}_v$.

Figure 2.1: Geometry of SP.

When $k \to \infty$ and $\mathcal{S}_m$ vertices are selected, $\boldsymbol{\psi}^*$ is given by the $m+1^{\text{th}}$ column of $\mathbf{U}'$, where $\mathbf{U}'$ is obtained by applying Gaussian elimination to the columns of $\mathbf{U}$ such that the $m+1^{\text{th}}$ column has zeros at indexes given by $\mathcal{S}_m$ [2]. $\mathbf{U}'$ can be written as:

$$\mathbf{U}' = \begin{bmatrix} \boldsymbol{u}_1(1) & & & & & \\ & \boldsymbol{u}_2'(2) & & \mathbf{0} & & \\ & & \ddots & & \mathbf{u}_{m+2} & \cdots & \mathbf{u}_n \\ & & & \boldsymbol{u}_{m+1}'(m+1) & & & \\ & & & \vdots & & & \\ & \bigstar & & \boldsymbol{u}_{m+1}'(n) & & & \end{bmatrix}, \tag{2.25}$$

where $\star$ denotes arbitrary entries and $\mathbf{0}$ denotes zero entries in the corresponding matrix regions. Because we have non-zero pivots, $\boldsymbol{u}_1(1)$ to $\boldsymbol{u}_{m+1}'(m+1)$, $\mathbf{U}_{\mathcal{S}_m,1:m}$ is full rank. The columns in $\mathbf{U}$ from $m+2$ to n remain intact. The concept is the same as that in [62]. Next, we explain how an iteration in the ideal SP algorithm can be seen as a volume maximization step.

2.4.2 SP algorithm as volume maximization

Consider a single stage in the SP algorithm, where the current sampling set is $\mathcal{S}_m$, $|\mathcal{S}_m| = m$ is the number of sampled vertices, and the bandwidth support is $\mathcal{R}_m = \{1, \cdots, m+1\}$.

At this stage, choosing a vertex v that maximizes $\det(\mathbf{U}_{\mathcal{S}_m\cup v,\mathcal{R}_m}\mathbf{U}^T_{\mathcal{S}_m\cup v,\mathcal{R}_m})$ is equivalent to choosing a vertex that maximizes $|\det(\mathbf{U}_{\mathcal{S}_m\cup v,\mathcal{R}_m})|$. We briefly state our results in terms of $|\det(\mathbf{U}_{\mathcal{S}_m\cup v,\mathcal{R}_m})|$ as this gives us the added advantage of making the connection with the Gaussian elimination perspective from Section 2.4.1. Now, focusing on the selection of the $(m+1)^{\text{th}}$ sample, we have the following result.

Proposition 2.1. *The sample v^* selected in the $(m+1)^{th}$ iteration of the ideal SP algorithm is the vertex v from $\mathcal{S}_m^c$ that maximizes $|\det(\mathbf{U}_{\mathcal{S}_m\cup v,\mathcal{R}_m})|$.*

Solution. The ideal SP algorithm selects the vertex corresponding to the maximum value in $|\boldsymbol{u}'_{m+1}(m+1)|, \cdots, |\boldsymbol{u}'_{m+1}(n)|$. Since $\mathcal{S}_m$ is given and $\mathbf{U}'_{\mathcal{S}_m\cup v,\mathcal{R}_m}$ is a diagonal matrix, this also corresponds to the selection of v such that magnitude value of the $\det(\mathbf{U}'_{\mathcal{S}_m\cup v,\mathcal{R}_m})$ is the maximum among all possible v selections.

But because $\mathbf{U}'_{\mathcal{S}_m\cup v,\mathcal{R}_m}$ is obtained from $\mathbf{U}$ by doing Gaussian elimination, the two determinants are equal, i.e., $|\det(\mathbf{U}_{\mathcal{S}_m\cup v,\mathcal{R}_m})| = |\det(\mathbf{U}'_{\mathcal{S}_m\cup v,\mathcal{R}_m})|$, and since the current $(m+1)^{\text{th}}$ iteration chooses the maximum absolute value pivot, given $\mathcal{S}_m$, the selected sample maximizes $|\det(\mathbf{U}_{\mathcal{S}_m\cup v,\mathcal{R}_m})|$. □

We now show that the vertex v^* is selected in the $(m+1)^{\text{th}}$ iteration according to the following rule:

$$v^* = \underset{v\in\mathcal{S}_m^c}{\arg\max}\, \text{dist}(\mathbf{d}_v, \text{span}(\mathbf{d}_1, \cdots, \mathbf{d}_m)),$$

where $\text{dist}(\cdot,\cdot)$ is the distance between a vector and its orthogonal projection onto a vector subspace. Thus, this optimization is equivalent to selecting a vertex v that maximizes the volume of the parallelepiped formed by $\mathbf{d}_1, \cdots, \mathbf{d}_m, \mathbf{d}_v$, i.e., $\text{Vol}(\mathbf{d}_1, \cdots, \mathbf{d}_m, \mathbf{d}_v)$.

Let $\mathbf{h}$ be a unit vector along the direction of $(m+1)^{\text{th}}$ column of $\mathbf{U}'$ in (2.25). We are interested in finding the vertex v that maximizes $|\mathbf{h}(v)|$.

Proposition 2.2. *The signal value $\mathbf{h}(v)$ is the length of projection of $\mathbf{d}_v$ on $\mathbf{h}$.*

Solution. The signal $\mathbf{h}$ belongs to the bandlimited space, $\mathbf{h} \in \text{span}(\mathbf{U}_{\mathcal{R}_m})$. Thus, using (2.15) we have that:

$$\mathbf{h}(v) = \langle \mathbf{h}, \mathbf{d}_v \rangle.$$

Since $\mathbf{h}$ is a unit vector, the last expression in the equation above is the projection length of $\mathbf{d}_v$ on $\mathbf{h}$. □

In summary, $|\mathbf{h}(v)|$ is maximized when $|\langle \mathbf{d}_v, \mathbf{h} \rangle|$ is maximized.

Proposition 2.3. *The signal* $\mathbf{h}$ *is such that* $\mathbf{h} \in \text{span}(\mathbf{d}_1, \cdots, \mathbf{d}_m)^{\perp} \cap \text{span}(\mathbf{U}_{\mathcal{R}_m})$.

Solution. All the diagonal elements in the Gaussian elimination of $\mathbf{U}_{\mathcal{S}_m \cup v, \mathcal{R}_m}$ are non-zero, as seen in (2.25), so that the following equivalent statements follow:

- $\mathbf{U}_{\mathcal{S}_m, 1:m}$ is full rank.
- $\mathbf{U}_{\mathcal{R}_m} \mathbf{U}_{\mathcal{R}_m}^T \mathbf{I}_{\mathcal{S}_m}$ is full column rank.
- $\text{span}(\mathbf{d}_1, \cdots, \mathbf{d}_m)$ has dimension m.

The second statement follows (Section 0.4.6 (b) [31]) from the first because $\mathbf{U}_{\mathcal{R}_m}$ has full column rank and $\mathbf{U}_{\mathcal{S}_m, 1:m}$ is nonsingular. Given that $\text{span}(\mathbf{d}_1, \cdots, \mathbf{d}_m)$ has dimension m we can proceed to the orthogonality arguments.

By definition, $\mathbf{h}$ obtained from (2.25) is zero over the set $\mathcal{S}_m$ so that, from Proposition 2.1:

$$\begin{aligned} \mathbf{h}(1) = 0 &\implies \langle \mathbf{h}, \mathbf{d}_1 \rangle = 0, \\ &\vdots \\ \mathbf{h}(m) = 0 &\implies \langle \mathbf{h}, \mathbf{d}_m \rangle = 0, \end{aligned}$$

and therefore $\mathbf{h}$ is orthogonal to each of the vectors $\mathbf{d}_1, \cdots, \mathbf{d}_m$. We call the space spanned by those vectors $\tilde{\mathcal{D}}_m$, defined as

$$\tilde{\mathcal{D}}_m = \{\text{span}(\mathbf{d}_i) | i \in \{1, 2, \cdots, m\}\}.$$

Since dimension of $\mathbf{U}_{\mathcal{R}_m}$ is $m+1$ and $\mathbf{h}$ is orthogonal to

$$\tilde{\mathcal{D}}_m = \text{span}(\mathbf{d}_1, \cdots, \mathbf{d}_m) \tag{2.26}$$

of dimension m, span $(\mathbf{h})$ is the orthogonal complement of $\tilde{\mathcal{D}}_m$ (see Fig. 2.1a for an illustration). □

For this particular algorithm, $\mathcal{R}$ changes with the number of samples in the sampling set. At the end of m^{th} iteration, the bandwidth support $\mathcal{R}$ can be represented as

$$\mathcal{R}_m = \{1, \cdots, m+1\}, \tag{2.27}$$

where m is the number of samples in the current sampling set. We use $\tilde{\mathcal{D}}$ and $\tilde{\mathcal{D}}_m$ to denote a dependence of $\mathcal{D}$ and $\mathcal{D}_m$ on $\mathcal{R}_m$ in addition to $\mathcal{S}_m$.

Proposition 2.4. *The sample v^* selected in the $(m+1)^{th}$ iteration of SP maximizes the distance between $\mathbf{d}_v$ and its orthogonal projection on $\tilde{\mathcal{D}}_m$.*

Solution. Since $\mathbf{d}_v \in \text{span}(\mathbf{U}_{\mathcal{R}_m})$, it can be resolved into two orthogonal components belonging to the two orthogonal spaces $\tilde{\mathcal{D}}_m$ and span $(\mathbf{h})$ (Prop. 2.3):

$$\mathbf{d}_v = \mathbf{P}_{\tilde{\mathcal{D}}_m} \mathbf{d}_v + \langle \mathbf{d}_v, \mathbf{h} \rangle \mathbf{h},$$

where $\mathbf{h}$ has unit magnitude and $\mathbf{P}_{\tilde{\mathcal{D}}_m}$ is the projection matrix onto the subspace $\tilde{\mathcal{D}}_m$.

Maximizing $|\mathbf{h}(v)|$ is equivalent to maximizing $|\langle \mathbf{d}_v, \mathbf{h}\rangle|$ which can be expressed in terms of magnitude of $\mathbf{d}_v$ and the magnitude of its projection on $\tilde{\mathcal{D}}_m$ as:

$$\arg\max_v \langle \mathbf{d}_v, \mathbf{h}\rangle^2 = \arg\max_v \|\mathbf{d}_v\|^2 - \left\|\mathbf{P}_{\tilde{\mathcal{D}}_m}\mathbf{d}_v\right\|^2. \tag{2.28}$$

Fig. 2.1b shows this orthogonality relation between $\mathbf{d}_v$, $|\langle \mathbf{h}, \mathbf{d}_v\rangle|$, and $\mathbf{P}_{\tilde{\mathcal{D}}_m}(\mathbf{d}_v)$. □

Thus, the v^* chosen at the $(m+1)$th iteration is the one that maximizes the volume of the space spanned by the filtered delta signals.

$$v^* = \arg\max_{v\in\mathcal{S}_m^c} \mathrm{Vol}(\mathbf{d}_1, \cdots, \mathbf{d}_m, \mathbf{d}_v).$$

(2.4.2) follows from using the definition of volume of parallelepiped [52]. Although Proposition 2.4 could have been derived from the determinant property in Proposition 2.1 using (2.7), we observe that the approach using the orthogonal vector to the subspace in Proposition 2.2, Proposition 2.3 and Proposition 2.4 makes more explicit the geometry of the problem.

Algorithm 3 summarizes this new volume maximization interpretation of SP. Although Algorithm 3 requires eigendecomposition, it is helpful to see its conceptual similarity with Algorithms 1 and 2. Algorithm 3 updates $\mathcal{R}$ in each iteration and that can be seen as an approximation of a greedy volume maximization approach where $\mathcal{R}$ is kept fixed. Hence Algorithm 3 is expected to have sub-optimal performance compared to the greedy volume maximization approach for the D-optimality criteria. For an empirical comparison, in Section 2.5, we compare SP which is Algorithm 3 relaxed with a finite value of k and without requiring the full eigendecomposition.

Algorithm 3 Volume interpretation of SP algorithm as $k \to \infty$

function SP($\mathbf{L}, s$)
  $\mathcal{S} \leftarrow \emptyset$
  $\mathcal{R} \leftarrow \{1\}$
  while $|\mathcal{S}| < s$ **do**
    $\left[\mathbf{d}_1, \cdots, \mathbf{d}_{|\mathcal{S}|}\right] \leftarrow \left[\mathbf{U}_{\mathcal{R}}\mathbf{U}_{\mathcal{R}}^T\boldsymbol{\delta}_1, \cdots, \mathbf{U}_{\mathcal{R}}\mathbf{U}_{\mathcal{R}}^T\mathbf{d}_{|\mathcal{S}|}\right]$
    $\tilde{\mathcal{D}} \leftarrow \text{span}_{v\in\mathcal{S}}(\mathbf{d}_v)$
    $v* \leftarrow \arg\max_v \|\mathbf{d}_v\|^2 - \|\mathbf{P}_{\tilde{\mathcal{D}}}\mathbf{d}_v\|^2$
    $\mathcal{S} \leftarrow \mathcal{S} \cup v^*$
    $\mathcal{R} \leftarrow \mathcal{R} \cup |\mathcal{R}| + 1$
  end while
  return $\mathcal{S}$
end function

2.4.3 Eigendecomposition-free methods as volume maximization

We now revisit some eigendecomposition-free graph signal sampling algorithms from the perspective of volume maximization. So far, we have covered existing literature on D-optimality as it relates to graphs and proposed two fast algorithms for sampling of graph signals. From (2.8d), note that the greedy update for approximate volume maximization is

$$v^* = \underset{v\in\mathcal{S}_m^c}{\arg\max} \|\mathbf{d}_v\|^2 - \|\mathbf{P}_{\mathcal{D}_m}\mathbf{d}_v\|^2 . \tag{2.29}$$

For each of these eigendecomposition-free algorithms, we consider the criterion to add a vertex to the sampling set in the $(m+1)^{\text{th}}$ iteration.

First, the WRS algorithm can be seen as neglecting the projection term in (2.29) and sampling by considering only the $\|\mathbf{d}_v\|^2$ term. Alternatively, as shown in Section 2.4.2, the SP algorithm approximates this by

$$v^* = \underset{v\in\mathcal{S}_m^c}{\arg\max} \|\mathbf{d}_v\|^2 - \left\|\mathbf{P}_{\tilde{\mathcal{D}}_m}\mathbf{d}_v\right\|^2 \tag{2.30}$$

Table 2.1: Approximation to greedy maximization of determinant. LSSS - For implementation details refer to Section 2.4.3

Sampling method	Selection process	Approximation
Exact greedy	$\arg\max_{v\in\mathcal{S}_m^c} \|\mathbf{d}_v\|^2 - \|\mathbf{P}_{\mathcal{D}_m}\mathbf{d}_v\|^2$	-
WRS [57]	$p(v) \propto \|\mathbf{d}_v\|^2$	No projection.
SP [3]	$\arg\max_{v\in\mathcal{S}_m^c} \|\mathbf{d}_v\|^2 - \|\mathbf{P}_{\tilde{\mathcal{D}}_m}\mathbf{d}_v\|^2$	Projec. space approximate and increasing in size.
LSSS[5]	$\arg\max_{v\in\mathcal{S}_m^c} \|\mathbf{d}_v\|^2 - 2\sum_{w\in\mathcal{S}_m}\langle\|\mathbf{d}_w\|,\|\mathbf{d}_v\|\rangle$	Inner product approx for projection.
AVM [34]	$\arg\max_{v\in\mathcal{S}_m^c} \|\mathbf{d}_v\|^2 - \sum_{w\in\mathcal{S}_m}\frac{\mathbf{d}_w^2(v)}{\|\mathbf{d}_w\|^2}$	Orthogonality assumption.

for a finite value of parameter k and a varying $\tilde{\mathcal{D}}_m$ in place of $\mathcal{D}_m$. The second eigen decomposition-free approach presented in [58] (V2) proposes to maximize (using the greedy selection in Equation (31) in [58]):

$$v^* = \arg\max_{v\in\mathcal{S}_m^c} \|\mathbf{d}_v\|^2 - 2\sum_{w\in\mathcal{S}_m}\langle|\mathbf{d}_w|,|\mathbf{d}_v|\rangle,$$

but in practice maximizes a different expression — see (32) in [59].

The crucial difference between our proposed method and [59] is that we obtain a specific expression to be maximized through D-optimality. Whereas [59] clearly shows the relation between various experiment design objective functions and their corresponding localization operators, the relation between the algorithm proposed in [59], LSSS, and the experiment design objective functions is unclear.

We do not attempt to explain methods such as [36] under the volume maximization framework as they define the signal smoothness through the total variation operator as opposed to squared differences through the graph Laplacian operator, which is necessary for the volume maximization interpretation. The similarities in the optimization objective function for various eigendecomposition-free sampling methods that we studied are summarized in Table 2.1. The differences between various sampling methods will be apparent when we compare their performance for various sampling and reconstruction settings.

2.5 Experimental settings

To evaluate our sampling algorithms, we assess their sampling performance on different graph topologies at different sampling rates.

2.5.1 Signal, Graph Models and Sampling setups

With a perfectly bandlimited signal, most sampling schemes can reconstruct the signal exactly. However, in practice, signals are rarely perfectly bandlimited and noise-free. Therefore, it is necessary to compare the performance of the sampling methods on non-ideal signals.

2.5.1.1 Signal smoothness and graph topologies

Consider first a synthetic noisy signal model from (1.2). The resulting signal is $\mathbf{f} = \mathbf{x} + \mathbf{n}$ which can be expressed as $\mathbf{U}_{\mathcal{F}}\tilde{\mathbf{f}}_{\mathcal{F}} + \mathbf{n}$. The frequency components of $\mathbf{x}$ and the noise term are assumed to be random variables with multivariate normal distributions, $\tilde{\mathbf{f}}_{\mathcal{F}} \sim \mathcal{N}(\mathbf{0}, c_1\mathbf{I}_{\mathcal{F}\mathcal{F}})$, $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, c_2\mathbf{I}_{\mathcal{V}\mathcal{V}})$, respectively. The constants c_1 and c_2 are chosen so that the expected signal power is 1 and the expected noise power is 0.1. Since our main objective is to study the effect of varying the number of samples, the graph topologies, and the graph size on DC and AVM, we fix the signal bandwidth to 50.

We compare our algorithms against three established algorithms — WRS, SP, and LSSS All methods except WRS return unique samples. For a fair comparison, all sampling methods are evaluated under conditions where the same number of samples is obtained, irrespective of whether the returned ones are unique or not (which could occur in the case of WRS [57]). We use the combinatorial Laplacian for our sampling and reconstruction experiments, except for the classification experiment where the normalized Laplacian of the nearest neighbors graph is used, as it achieves overall better classification accuracy.

2.5.1.2 Sampling set sizes

We use two graph sizes, 500 and 1000. Except for the Erdős Rényi graph model, we use the Grasp [27] and GSPBox [53] MATLAB toolboxes to generate the graph instances — see Table 2.2.

We use sampling set sizes ranging from 60 to 200 samples to compare the variation of the reconstruction error. For comparing algorithms in this setting, we do not show the full range of reconstruction SNRs from WRS because its SNR is usually 5-10 dBs lower than other methods at starting sampling rate of 60 (see Tables 2.4 and 2.6 for performance at higher sampling rates).

2.5.1.3 Classification on real-world datasets

In this experiment, we evaluate sampling algorithms in a transductive semi-supervised learning setting for a digit classification task (USPS dataset). We randomly select 10 smaller datasets of size 1000 from the original dataset, such that each smaller dataset contains 100 elements from each category, i.e., the 10 digits. Using those smaller datasets, we construct a nearest neighbors graph with 10 neighbors. This setup is the same as in [2]. The graph sampling and reconstruction algorithms then select a number of samples ranging from 60 to 200. Using the one-vs-all strategy, we reconstruct the class signals and then classify them by selecting the class which gives the maximum reconstruction in magnitude at a vertex. We then report the average accuracy of the classification over the 10 smaller sets.

2.5.1.4 Effect of scaling graph sizes

One of our primary goals is to develop fast and scalable algorithms. To evaluate these properties, we use a random sensor nearest neighbors graph with 20 nearest neighbors and a community graph with 10 communities with different graph sizes (500, 1000, 2000, 4000, and 8000). For each graph size, we sample 150 vertices, and the signal model remains the same

Table 2.2: Types of graphs in the experiments

Graph model	Instance	Construction comments
Random sensor knn	`grasp_plane_knn(n, k),k=8 or 15` `gsp_sensor(n, 20)`	Uniformly sampled vertices on 2d plane with k nearest neighbors Uniformly sampled vertices on 2d plane with 20 nearest neighbors
Scale-free	`grasp_barabasi_albert(n,8)`	Initial 8 nodes
Community	`param.Nc=5 or 10` `gsp_community(n, param)`	`param.Nc` communities
WS/Small world	`grasp_watts_strogatz(n, 5, 0.2)`	Average degree 5 rewiring probability 0.2
Erdős Rényi	`erdos_renyi(n,0.02)`	Probability of connection 0.02

as in Section 2.5.1.1. We report the reconstruction SNR and the time required to sample averaged over 50 graph and signal realizations for a given graph type.

The feasibility of different sampling algorithms on graphs with sizes that are orders of magnitude larger than a thousand vertices is an indicator of scalability. Thus, we also test the sampling algorithms on larger graph sizes (50,000 and 100,000). Both the graph parameters, such as the number of nearest neighbors or the number of communities, and the signal model parameters, such as the noise power, remain the same as those we use for smaller graph sizes, while we use a bandwidth of 100, and sample 5000 vertices for the two larger graphs sizes. At these graph sizes, some sampling algorithms require more than 10 times the time required by AVM and more than 64GB of random-access memory (RAM). Because of this, it is not possible to run 50 graph and signal realizations for all the sampling algorithms as we did earlier. Least squares reconstruction, which we used for smaller graphs, is also not feasible at these graph sizes, so we reconstruct using projection onto convex sets(POCS) from [46], which is tailored for bandlimited signals on graphs. We include the execution times and the SNRs for this setting in the tables along with smaller graphs. However, due to fundamental differences in the reconstruction method, we do not plot the SNRs together with those of smaller graphs. The execution times measured in seconds are rounded to one decimal precision for display.

2.5.2 Initialization details

We wish to evaluate all the algorithms on an equal footing. Thus, for evaluating the squared coherences using Function 1 we use the same number of random vectors $10\log(n)$ corresponding to $c = 10$, $\epsilon = 0.01$, and an order 30 polynomial wherever filtering is needed for the WRS, DC, and AVM methods. Larger c and smaller ϵ values result in a more accurate approximation of squared coherences but also require more computations. We choose those particular values to achieve a good balance between the approximation accuracy and the number of computations. The degree of the polynomial is selected to be larger than the

diameter of most graphs we consider. However, we get marginal returns in performance by increasing the degree of the polynomial. Thus, in future work, it might be interesting to study adaptive schemes where the degree of the polynomial is chosen to be similar to the graph diameter.

The various algorithms we consider have some "hard-coded" parameters. SP has just one parameter k to which we assign $k = 4$. LSSS has a few more parameters to tune such as ν, η. In [59] the parameter $\nu = 75$ is chosen experimentally, but in our experiments, we run the LSSS algorithm on a wide range of ν values around 75 — $\nu = [0.075, 7.5, 75, 750, 75000]$, and select the value of ν that maximizes SNR. We chose this wide range of values experimentally, as we observed that for some topologies, graph sizes and Laplacians, optimal reconstruction SNR was sometimes achieved at ν values differing from the proposed 75 by several orders of magnitude. As for the sampling times, we choose the sampling time corresponding to the ν achieving the maximum SNR. In most cases, there was a non-negligible increase in SNR while choosing ν without a significant change in the execution time. We experimentally determine η the same way as in the original implementation. For the DC algorithm, we choose $\Delta = 0.9$.

2.5.3 Reconstruction techniques

We denote the sampled signal $\mathbf{f}_{\mathcal{S}}$ and the lowpass frequencies of the original signal by $\tilde{\mathbf{f}}_{\mathcal{F}} = \mathbf{U}_{\mathcal{F}}^T \mathbf{f}$. The ideal reconstruction which minimizes the mean square error using the sampled signal is given by the least squares solution to $\left\| \mathbf{U}_{\mathcal{S}\mathcal{F}} \tilde{\mathbf{f}}_{\mathcal{F}} - \mathbf{f}_{\mathcal{S}} \right\|_2$. Other existing methods of reconstruction are (i) using a linear combination of tailored kernels as seen in LSSS and (ii) solving a regularized least squares problem as in BS-GDA. However, since we are primarily interested in comparing the sampling sets generated by various algorithms on an even footing, we use the least squares solution for reconstruction which we compute by assuming that we know the graph Fourier basis. To achieve the best results for WRS, instead of the least squares solution we use the recommended weighted least squares [57],

although it is slightly different from what we use for all other algorithms. The weighted least squares solution is tailored to account for sampling probabilities in WRS and does provide a marginal improvement in performance compared to least squares for most graphs. We use the Moore-Penrose pseudo inverse for all our least squares solutions.

2.6 Results

We now evaluate the performance of our algorithm based on its speed and on how well it can reconstruct the original signal.

2.6.1 Reconstruction error

We evaluate performance based on the mean squared error in reconstructing the signal $\mathbf{f}$. Thus, we measure the error $\left\|\hat{\mathbf{f}} - \mathbf{f}\right\|^2$ between the reconstructed signal and the original noisy signal, where $\hat{\mathbf{f}}$ is the reconstructed signal.

For our experiments, we plot the SNR averaged over 50 different graph instances of each type of graph, with a new signal generated for each graph instance (see Figure 2.2). The two graph models where we observe a lesser SNR for the AVM algorithm are the random sensor nearest neighbors and the community graph models, which we discuss next. For the remaining graph models the reconstruction SNR from DC and AVM sampling is comparable to other algorithms, such as SP and LSSS. As mentioned, we find the sampling from AVM to be comparatively satisfactory considering that we are reporting the maximum SNRs for LSSS over 5 different parameter values.

In Fig. 2.2f, we notice that for random-sensor nearest neighbor graphs of size 500, we need more samples to achieve competitive performance. To better understand this, consider a graph consisting of two communities. In the original volume maximization, a sampling set from only one community would give a volume of zero and that sampling set would never be selected by a greedy exact volume maximization. However, because AVM is only

an approximation to the greedy volume maximization, sampling from only one community is possible, although unlikely. More generally, this approximation affects weakly connected graphs such as random sensor graphs with a kNN construction with a small number of nearest neighbors. More specifically, this approximation affects community graphs at low sampling rates, as shown in Figures 2.2c and 2.2h.

The issue of lower performance at lower sampling rates for smaller graphs, however, is no longer critical for larger graphs as we see when we increase the number of samples to 150 — (Table 2.6), our algorithm performance is comparable to that of other state-of-the-art algorithms. Note that we do not face this issue for Erdős Rényi graph instances in our experiments since these are almost surely connected as the probability of connection exceeds the sharp threshold [22].

For the USPS dataset classification, we observe a significant drop in the classification accuracy for the samples chosen using the DC algorithm. However, the classification accuracy for AVM on the USPS dataset is at par with the remaining algorithms.

Next, we evaluate how the complexity of AVM scales with the graph size.

2.6.2 Speed

Using the setup from Section 2.5.1, we compare the sampling times for WRS, SP, LSSS, BS-GDA and AVM algorithms. We exclude the DC algorithm from these comparisons because the distance evaluations in DC, which provide good intuition, make DC significantly slower as compared to AVM. We include BS-GDA since it is one of the lowest complexity approaches among eigendecomposition-free algorithms. We use the random sensor graph from the GSPbox [53] with 20 nearest neighbors.

In our comparison, we use the implementations of WRS, SP, LSSS and BS-GDA distributed by their respective authors and run them on MATLAB 2019b along with our proposed algorithm. We could possibly improve on the existing implementations using specialized packages for functionalities such as eigendecomposition, but to remain faithful to

Table 2.3: Execution time(secs.) for sampling, random sensor graphs

$\lvert\mathcal{V}\rvert$	WRS	SP	LSSS	BS-GDA	AVM	Overhead(AVM)
500	0.2	2.7	0.6	0.1	0.5	2.53
1,000	0.5	7.4	2.5	0.6	0.9	1.98
2,000	1.1	21.6	9.8	3.3	1.9	1.7
4,000	2.5	65.1	38.6	18.1	4.2	1.65
8,000	6.2	165.9	115.3	96.5	9.5	1.54
50,000	71.9	–	11,066.8	1,759.3	699.5	9.73
100,000	167.8	–	–	–	1,525.4	9.09

Table 2.4: SNRs, random sensor graphs

$\lvert\mathcal{V}\rvert$	WRS	SP	LSSS	BS-GDA	AVM
500	6.8	9.78	9.83	8.96	9.05
1,000	7.89	9.69	9.81	9.25	9.36
2,000	8.23	9.38	9.39	9.24	9.33
4,000	7.99	9.54	9.51	9.36	9.47
8,000	8.11	8.94	8.98	8.94	8.92
50,000	2.11	–	2.14	2.13	2.13
100,000	2.86	–	–	–	2.88

the original papers, we use their codes with minimal changes. Wherever the theoretical algorithms in the papers conflict with the provided implementations, we go with the implementation since that was presumably what the algorithms in the papers were timed on.

To minimize the effect of other processes running at different times, we run the sampling algorithms round-robin. We do this process for multiple iterations and different graph topologies. We time the implementations on an Ubuntu HP Z840 Workstation, which naturally has plenty of background processes running. The changes in their resource consumption affect our timing. It is impossible to stop virtually all background processes, so we try to reduce their impact in two ways. First, we iterate over each sampling scheme 50 times and report the averages. Second, instead of completing iterations over the sampling schemes one by one, we call all the different sampling schemes in the same iteration. These minor precautions help us mitigate any effects of background processes on our timing.

For 500-8,000 graph sizes, we observe that as the graph size increases, AVM is only slightly slower compared to WRS. It is orders of magnitude faster than SP, LSSS and the

Table 2.5: Execution time(secs.) for sampling, Community graphs

$\lvert\mathcal{V}\rvert$	WRS	SP	LSSS	BS-GDA	AVM	Overhead(AVM)
500	0.2	3.9	0.6	0.1	0.5	2.59
1,000	0.4	21.7	3	0.7	0.8	1.89
2,000	1	125.9	13.5	3.9	1.8	1.72
4,000	2.6	597.2	65.8	28.6	4.3	1.66
8,000	6.8	2,377.9	272.3	188.8	9.4	1.38
50,000	89.1	–	–	2,504.6	957.7	10.75
100,000	267.8	–	–	–	2,485.7	9.28

Table 2.6: SNRs, Community graphs

$\lvert\mathcal{V}\rvert$	WRS	SP	LSSS	BS-GDA	AVM
500	6.41	10.14	10.02	−5.61	9.42
1,000	7.09	9.7	9.62	7.13	8.61
2,000	7.16	9.61	9.65	9.13	9.16
4,000	7.53	9.43	9.44	9.06	9.3
8,000	8.04	9.58	9.56	9.16	9.48
50,000	0.92	–	–	0.83	1.11
100,000	0.73	–	–	–	0.79

BS-GDA algorithm — see Tables 2.3 and 2.5, while having a very small impact on the SNR of the reconstructed signal — Tables 2.4 and 2.6. The execution time also scales well with the graph size, as shown in Fig. 2.3a.

We also report the relative execution times using the overhead rate, the ratio of execution times of two algorithms. We compute this overhead for the AVM algorithm vs the WRS algorithm pair.

$$\text{Overhead(AVM)} = \frac{\text{Execution time of proposed AVM algorithm}}{\text{Execution time of WRS}}$$

Most authors consider WRS as the fastest sampling algorithm and benchmark against it. By specifying our overhead rates relative to WRS we can indirectly compare our algorithm with myriad others without doing so individually. We report these factors for various graph sizes in Tables 2.3 and 2.5.

To justify the increase in the speed of execution compared to the slight decrease in the SNR, we plot the SNR versus the execution time for the different algorithms we compared. Ideally, we want an algorithm with fast execution and good SNR. From Figs. 2.4a, 2.4b, we see that our algorithm fits that requirement very well.

For experiments on graph sizes 500-8000, we reduced the variability in the execution time and SNR observations by reporting means over 50 randomly initialized graph and signal realizations. However, for graphs of sizes 50,000 and 100,000, running 50 realizations of each sampling strategy is impractical because of the time required. To determine if 10 realizations are sufficient, we compute the ratio of standard deviation to the mean for execution times and SNRs. Except for one setting of BS-GDA where the ratio is 0.26, it does not exceed 0.12 in all experiments. So for graphs of size 50,000 and 100,000, we report the execution times and SNRs averaged over 10 randomly initialized realizations in Tables 2.3, 2.4, 2.5, and 2.6.

In those tables of algorithm comparisons, we look for scalable algorithms with low execution times and high SNRs, or which at least finish execution within our limits as mentioned in Section 2.5.1.4. For graphs with size 50,000, WRS, LSSS, BS-GDA, and AVM, finish within our limits, while for graphs with size 100,000 only WRS, and AVM can finish. We fill the table entries corresponding to the algorithms that did not finish with a $-$. Among the algorithms that finish, AVM provides up to 20% improvement in SNR over that of WRS, and at most 0.46% decrease compared to other algorithms. However, the SNRs are lower than for smaller graph sizes because of the POCS-based reconstruction. The execution times of AVM are at least 60% less and as much as 93% less compared to other state-of-the-art algorithms except WRS. The overhead of AVM relative to WRS is larger compared to smaller graph sizes because of the 5000 sampled vertices for 50,000 and 100,000 graph sizes as opposed to 150 samples for 500 to 8,000 graph sizes. We see a further increase of about 62% in the execution time for community graphs due to the larger number of edges. So for graph sizes 50,000 and 100,000, AVM not only finishes execution within our limits but maintains

SNR at par with other algorithms for two different graph topologies, while being the fastest algorithm second only to WRS.

Of course, with different graph types, the SNR vs execution time performance of AVM may vary. But what we always expect this algorithm to deliver is execution times similar to WRS while having a significant improvement in the SNR. In a way, this algorithm bridges the gap between existing Eigendecomposition-free algorithms and WRS.

2.6.3 Effect of number of samples on execution time

A fast graph signal sampling algorithm should be scalable with respect to the number of sampled vertices. To experiment and compare the scalability of different graph signal sampling algorithms, we set up sensor graphs with 20 nearest neighbors and community graphs with 10 communities, both of size 8192. For each graph type, we sample a varying number of vertices ranging from 64 to 4096 samples in multiples of 2 and measure the corresponding execution times. We display the results of this scalability experiment as execution times versus the number of samples in plots.

Doing this experiment for different sampling methods helps us compare their robustness to a varying number of samples. In figures 2.5 and 2.7, we observe the effect of varying the number of sampled vertices on the execution times of the algorithms WRS, SP, LSSS, BS-GDA, and AVM. With an increase in the number of sampled vertices, WRS's execution time does not show a significant dependence, SP and LSSS's execution time increases, whereas BS-GDA's execution time decreases. The minimal dependence of WRS's execution time is due to the computationally cheap random sampling step once the graph coherences are computed by Function 1. The increase in the execution time of SP and LSSS is due to the extra computations needed for sampling a new vertex. However, for BS-GDA, we believe that the decrease in the execution time is due to the decrease in the coverage set sizes with the increase in the number of requested samples. We see that AVM's execution time is close to WRS's for smaller sampling sets, and it increases with the number of requested samples.

Except for sample set sizes in the order of graph size, AVM has the second-lowest execution times for a range of sample set sizes.

Since we saw that the execution time of AVM increases with the number of sampled vertices, we wish to assess further the rate of increase of the execution time. For this purpose, we consider sensor graphs with 20 nearest neighbors and community graphs with 10 communities, both of size 50,000. The number of samples requested ranges from 64 samples to 32768 samples in multiples of 2. We display the results of this experiment as execution times versus the number of samples plots limited to AVM.

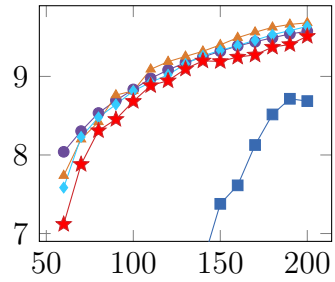
In Figures 2.6 and 2.8, apart from minor fluctuations, we see a linear relationship between the execution times and the number of samples for a number of samples ranging from 64 to 4096. This observation agrees with our theoretical analysis of AVM complexity in Section 2.3.4.2 explaining an additional $O(s|\mathcal{E}|d)$ dependence on the number of samples compared to WRS.

2.7 Conclusion

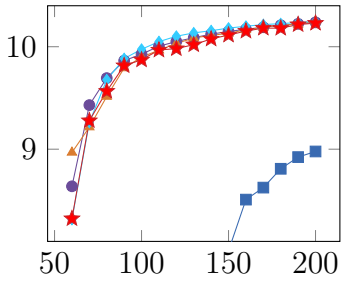
Most sampling schemes perform reasonably well when dealing with perfectly bandlimited signals. However, in the presence of noise or the signal not being perfectly bandlimited, some schemes perform much better. In the scenario that only a limited number of samples can be chosen, we would like to use an algorithm that can perform well without requiring computationally expensive procedures such as eigendecomposition.

The algorithms presented in this chapter rely on the intuition of looking at the problem as maximizing the volume of the parallelepiped formed by the lowpass signals corresponding to the sampled vertices. This helps us to develop intuitive and fast graph signal sampling algorithms. The volume maximization framework also helped to connect various existing algorithms.

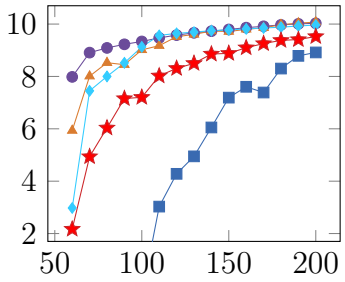
The sampling algorithm we developed reaches speeds achieved by WRS but with a large improvement in reconstruction accuracies. The accuracies are comparable with other contemporary algorithms but, at the same time, provide significant speed improvements.



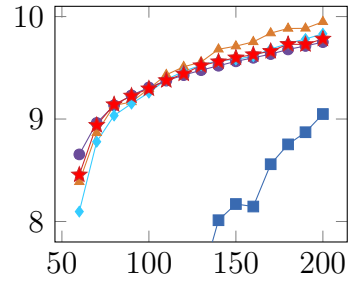
(a) Random sensor knn graph, size 1000



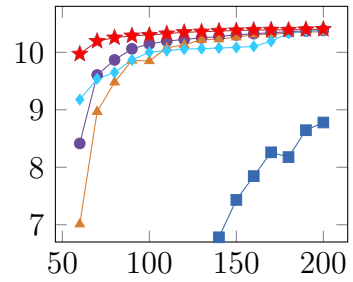
(b) Scale-free graph, size 1000



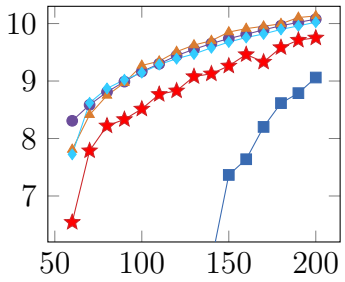
(c) Community graph, size 1000, 10 communities



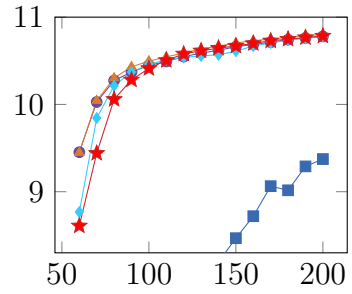
(d) WS graph, size 1000



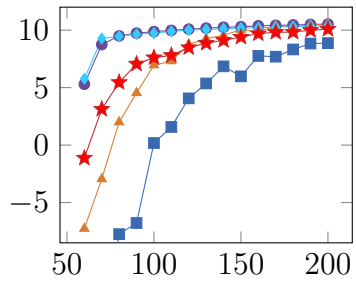
(e) Erdős Rényi graph, size 1000



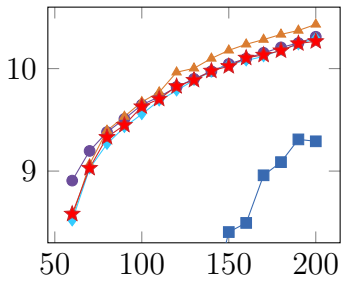
(f) Random sensor knn graph, size 500



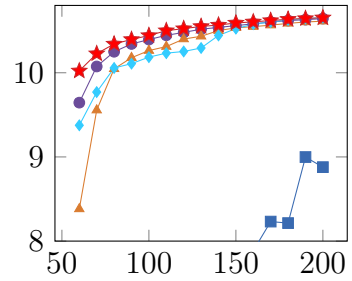
(g) Scale-free graph, size 500



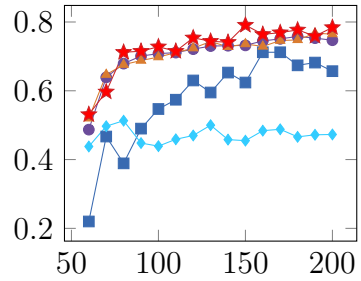
(h) Community graph, size 500, 5 communities



(i) WS graph, size 500



(j) Erdős Rényi graph, size 500



(k) Classification accuracies

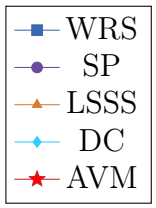
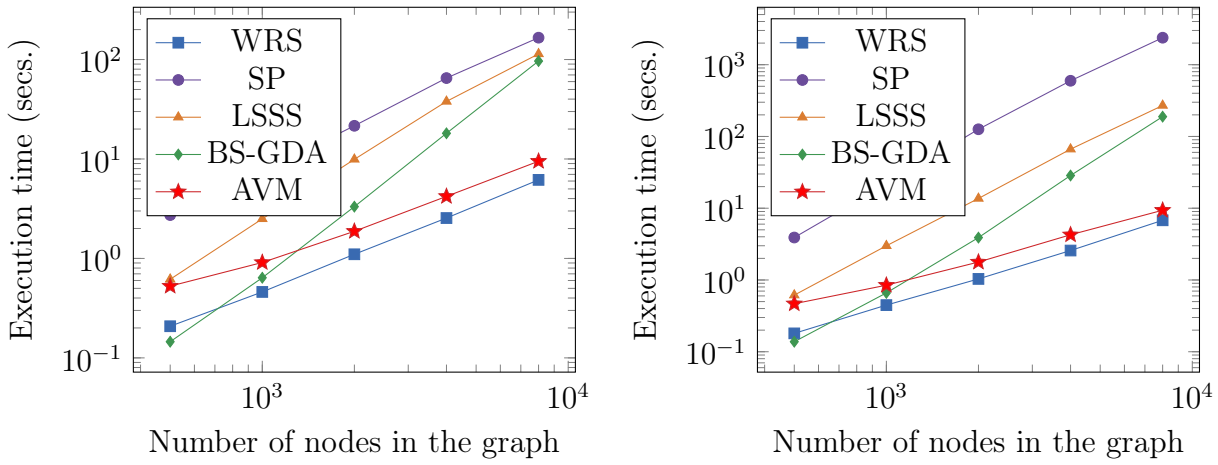


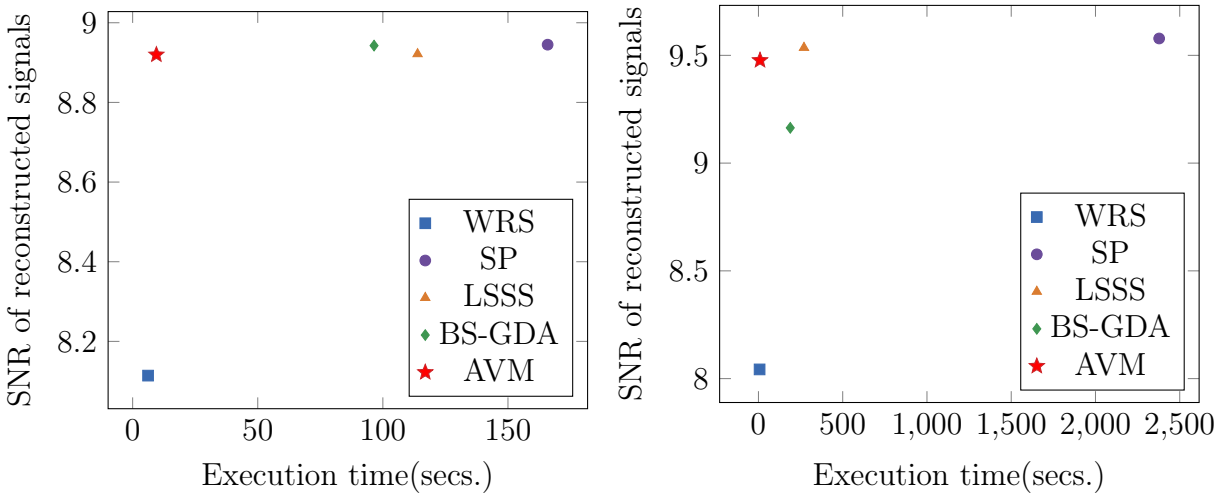
Figure 2.2: Comparison of eigendecomposition-free methods in the literature. x-axis: number of samples selected. y-axis: average SNR of the reconstructed signals. We do not include the entire range of SNR from WRS based reconstruction because of its comparatively wider range.



(a) Random sensor graphs with 20 nearest neighbour connections

(b) Community graphs with 10 communities

Figure 2.3: Visualizing average sampling times of four algorithms over 50 iterations on community graphs with 10 communities. Execution times for LSSS are averaged over executions for different parameter values.



(a) Random sensor graphs with 20 nearest neighbour connections.

(b) Community graphs with 10 communities

Figure 2.4: Scatter plot of the SNR vs execution time for graph size 8000. Axis for execution time is reversed, so results on the top right are desirable.

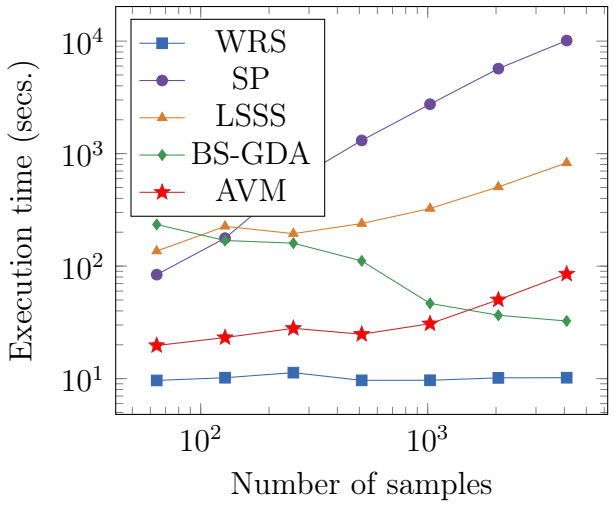


Figure 2.5: Random sensor graphs with 8192 vertices and 20 nearest neighbour connections.

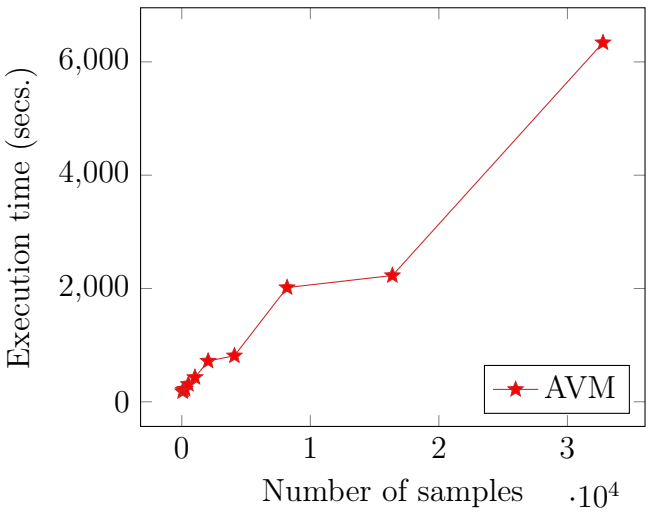


Figure 2.6: Random sensor graphs with 50,000 vertices and 20 nearest neighbour connections.

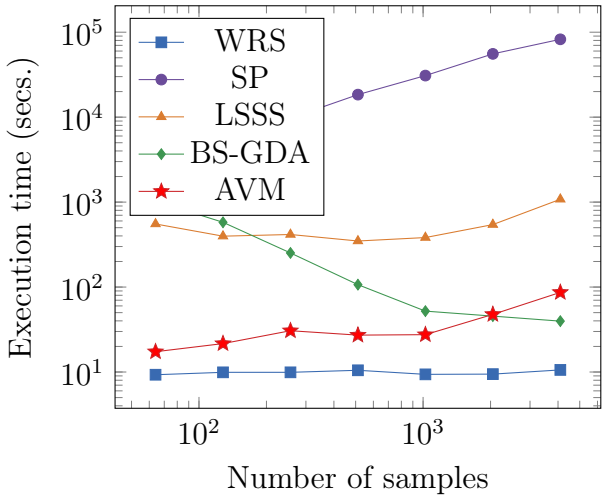


Figure 2.7: Community graphs with 8192 vertices and 10 communities.

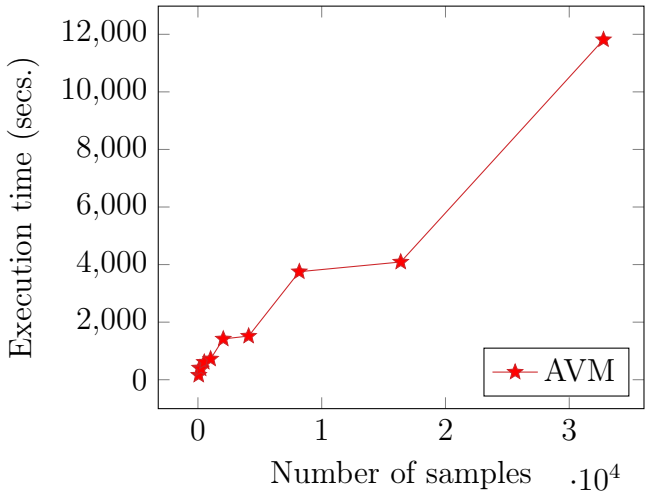


Figure 2.8: Community graphs with 50,000 vertices and 10 communities.

Chapter 3

Robust graph signal sampling

3.1 Introduction

Most graph signal sampling methodologies, including those we presented in Chapter 2, assume that all samples from the selected sampling set can be observed. In environments where sensors can fail or an adversary can delete samples, however, one may receive only an unknown subset of the selected samples. This phenomenon can also be observed in social sensing scenarios such as crowdsourcing [44], where one may query a large group of users only to receive answers from a small subset of them. Similarly, in the case of sample set selection on graphs, some samples may be lost.

The effects of losing some samples depend on the application. Since the conventional greedy sample selection algorithm is based on an iterative setup where the next sample is chosen to maximize the performance given all the previously selected samples, the loss of a few important samples may be highly detrimental to overall performance. On a graph with n disconnected components, the greedy sample set so found will have one sample in each disconnected subgraph. Losing any of these samples means the signal values over that subgraph cannot be recovered (since there is only one sample per connected component).

Motivated by such scenarios, in this work, we consider a further generalization of the sampling problem from Section 1.2.1 wherein one selects a set of k samples but only receives $k - \tau$ samples, with the remaining τ samples being lost. We call this the *robust graph signal*

sampling problem. Furthermore, in general, it cannot be known beforehand which τ samples will be lost. Then, the goal should be to select the sampling set of size k that achieves the best performance in the worst-case scenario, i.e., when out of all possible subsets of size $k - \tau$ samples, the one leading to maximum degradation is removed from the original set of k samples.

We propose a robust greedy sample set selection algorithm that optimizes worst-case performance under sample losses, and we study its performance guarantees. Our work extends the greedy sample set selection setup from Chapter 2 to sampling scenarios in which one receives only a subset of the selected samples. Our robust graph signal sampling problem is also related to existing robust optimization frameworks [39, 47, 9], which mainly focus on the optimization of a monotone function with a given approximate submodularity parameter. Our focus is on the functions used as an optimality criterion for graph signal sampling, and through numerical evaluations, we demonstrate how in this setup the proposed approach outperforms state-of-the-art robust maximization techniques.

3.2 System Model

We consider the signal model from (1.2) with additional restrictions. In particular, we study the class of graph signals for which $\tilde{\mathbf{x}}_{\mathcal{F}}$ has mean $\mathbb{E}[\tilde{\mathbf{x}}_{\mathcal{F}}] = \mathbf{0}$ and covariance

$$\mathbb{E}[\tilde{\mathbf{x}}_{\mathcal{F}}\tilde{\mathbf{x}}_{\mathcal{F}}^T] = \mathbf{\Gamma} = \text{diag}(\gamma_1, \ldots, \gamma_{\mathcal{F}}) \tag{3.1}$$

with added noise that can be represented by the noise vector $\mathbf{n}$, which has mean $\mathbb{E}[\mathbf{n}] = \mathbf{0}$ and covariance

$$\mathbb{E}[\mathbf{n}\mathbf{n}^T] = \mathbf{M} = \text{diag}(\mu_1, \ldots, \mu_n). \tag{3.2}$$

Without loss of generality, we assume $\mathbf{\Gamma}$ has full rank, as one can always adjust $\mathcal{F}$ to remove the elements for which $\gamma_i = 0$ otherwise.

We represent the sampling process explained in Section 1.2.1 by multiplication with a sampling matrix $\mathbf{S} \in \{0,1\}^{|\mathcal{S}| \times n}$ in which the i^{th} row corresponds to the i^{th} row of the $n \times n$ identity matrix $\mathbf{I}$. The samples are given by:

$$\mathbf{f}_{\mathcal{S}} = \mathbf{S}\mathbf{f}. \tag{3.3}$$

We represent the reconstructed signal by

$$\hat{\mathbf{f}} = \mathbf{W}\mathbf{f}_{\mathcal{S}}, \tag{3.4}$$

where $\mathbf{W}$ is an $n \times |\mathcal{S}|$ reconstruction matrix. The corresponding error covariance matrix is given by:

$$\mathbf{K} = \mathbb{E}[(\mathbf{f} - \hat{\mathbf{f}})(\mathbf{f} - \hat{\mathbf{f}})^T]. \tag{3.5}$$

The optimal reconstruction matrix $\mathbf{W}$ is chosen as to minimize the scalar cost function

$$J(\mathbf{W}) = \mathbf{z}^T\mathbf{K}\mathbf{z} \tag{3.6}$$

over the error covariance matrix for all $\mathbf{z} \in \mathbb{R}^n$, as detailed in [12], leading to the following error covariance matrix:

$$\mathbf{K} = \mathbf{U}_{\mathcal{F}}(\mathbf{\Gamma}^{-1} + \mathbf{U}_{\mathcal{F}}^T\mathbf{S}^T\mathbf{S}\mathbf{M}^{-1}\mathbf{S}^T\mathbf{S}\mathbf{U}_{\mathcal{F}})^{-1}\mathbf{U}_{\mathcal{F}}^T. \tag{3.7}$$

As can be observed from (3.7), the chosen sampling set $\mathcal{S}$ affects the error covariance matrix through the following term:

$$\mathbf{K}(\mathcal{S}) = (\mathbf{\Gamma}^{-1} + \mathbf{U}_{\mathcal{F}}^T\mathbf{S}^T\mathbf{S}\mathbf{M}^{-1}\mathbf{S}^T\mathbf{S}\mathbf{U}_{\mathcal{F}})^{-1} \tag{3.8}$$

$$= (\mathbf{\Gamma}^{-1} + \sum_{i \in \mathcal{S}} \mu_i^{-1}\mathbf{r}_i\mathbf{r}_i^T)^{-1}. \tag{3.9}$$

where $\mathbf{r}_i^T$ is the i^{th} row of $\mathbf{U}_\mathcal{F}$. Then, the sampling set selection problem is to choose the best sampling set $\mathcal{S}^*$ of a given size k with respect to an objective function defined over the matrix from (3.9),

$$\mathcal{S}^* = \arg \max_{\mathcal{S}:|\mathcal{S}|=k} f((\mathbf{\Gamma}^{-1} + \sum_{i \in \mathcal{S}} \mu_i^{-1} \mathbf{r}_i \mathbf{r}_i^T)^{-1}). \tag{3.10}$$

It is useful to note that the formulation from (3.10) depends on the choice of function $f(.)$ In this work, we let $f(.)$ be a monotonically non-decreasing, non-negative function with $f(\emptyset) = 0$.

Finding $\mathcal{S}^*$ is in general an NP-hard problem [38]. When the objective function (3.10) is submodular, greedy algorithms can approximate it in polynomial time. However, most functions used as optimality criteria in graph signal sampling are not submodular. In such cases, performance of greedy algorithms has been studied using the notion of approximate submodularity, which measures how close a function is to being submodular [12].

3.3 Problem formulation: Lost samples

The problem defined in (3.10) identifies the best sampling set under the assumption that all the samples from the selected set $\mathcal{S}^*$ are receied. If there are sensor failures, or in adversarial environments, not all selected sensors will provide samples. Accordingly, we define the *robust graph sampling* problem as the selection of a set of k samples under the assumption that one receives only $k - \tau$ of them, where we do not have prior knowledge of which samples will be lost.

Problem 3.1. *Select a set of samples $\mathcal{S}$ to maximize the worst-case performance:*

$$\mathcal{S}^* = \arg \max_{\mathcal{S}:|\mathcal{S}|=k} \min_{\substack{\mathcal{A}:\mathcal{A}\subseteq\mathcal{S} \\ |\mathcal{A}|=k-\tau}} f((\mathbf{\Gamma}^{-1} + \sum_{i \in \mathcal{A}} \mu_i^{-1} \mathbf{r}_i \mathbf{r}_i^T)^{-1}). \tag{3.11}$$

To solve Problem 3.1, we propose a greedy algorithm consisting of two stages (see Algorithm 4). In the first stage, the algorithm selects τ nodes in an oblivious manner, similar to

Algorithm 4 Robust Graph Sampling

Initialize graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and sampling set $\mathcal{S} = \emptyset$.
Stage 1:
for $i = 1, \ldots, \tau$
    Choose node u^* such that:

$$u^* \leftarrow \arg \max_{u \in \mathcal{V} \setminus \mathcal{S}} f(\{u\}) \tag{3.12}$$

▷ $\mathcal{S}$ is the set of already selected nodes at the beginning of iteration i.

    $\mathcal{S} = \mathcal{S} \cup \{u^*\}$
end for
Set $j := 0$
Stage 2:
for $i = \tau + 1, \ldots, k$
    Choose node u^* such that:

$$u^* \leftarrow \arg \max_{u \in \mathcal{V} \setminus \mathcal{S}} \min_{\mathcal{A} \in \mathcal{S}^j} f\big(\mathcal{A} \cup \{u\}\big) \tag{3.13}$$

▷ $\mathcal{S}^j$ denotes the set of j-element subsets of $\mathcal{S}$.

    $\mathcal{S} = \mathcal{S} \cup \{u^*\}$
    $j = j + 1$
end for

the first stage of existing two-stage algorithms [47, 9]; at each step, the best node is selected from the set of available nodes, by assuming all the previous ones may be lost. In the second stage, we know that at least j nodes will be received from the first stage. The algorithm then selects the next node to maximize the worst-case performance when combined with any j nodes from the samples selected so far. This is unlike the selection criterion used in [9], which discards the nodes selected in the first stage. For $\tau = 0$, Algorithm 4 reduces to the conventional greedy algorithm [12].

To study the performance of Algorithm 4, we utilize the following definitions. The first one is the approximate submodularity notion from [12]. In the sequel, we use the shorthand notation:

$$f(\mathcal{A}) = f((\mathbf{\Gamma}^{-1} + \sum_{i \in \mathcal{A}} \mu_i^{-1} \mathbf{r}_i \mathbf{r}_i^T)^{-1}). \tag{3.14}$$

Definition 3.1. *(Approximate submodularity) A function f is α-approximately submodular if,*

$$f(\mathcal{A} \cup \{i\}) - f(\mathcal{A}) \geq \alpha(f(\mathcal{B} \cup \{i\}) - f(\mathcal{B})) \tag{3.15}$$

for all $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}$ *and* $i \in \mathcal{V} \backslash \mathcal{B}$. *where* $\alpha \in [0, 1]$ *is chosen as the largest scalar satisfying* (3.15).

Note that $\alpha = 0$ always holds since f is monotonically non-decreasing, and f becomes submodular when $\alpha = 1$. Another useful property of a class of functions that puts a limit to their growth in terms of the union of multiple input sets is their property of subadditivity.

Definition 3.2. *(Bipartite subadditivity ratio)[9] The bipartite subadditivity ratio of* f *is the largest* $\theta \in [0, 1]$ *such that*

$$\frac{f(\mathcal{A}) + f(\mathcal{B})}{f(\mathcal{A} \cup \mathcal{B})} \geq \theta, \quad \forall \mathcal{A}, \mathcal{B} \subseteq \mathcal{V} \text{ such that } \mathcal{A} \cap \mathcal{B} = \emptyset. \tag{3.16}$$

Next, we provide a lower bound on the performance of the robust greedy sample set selection Algorithm 4.

Theorem 3.1. *Let* $\mathcal{S}$ *be the sampling set selected by Algorithm 4, with* $\mathcal{S}_0$ *and* $\mathcal{S}_1$ *representing the samples selected in the first and second stages, respectively. Denote the worst-case performance of* $\mathcal{S}$ *by,*

$$\phi(\mathcal{S}) = \min_{\substack{\mathcal{A}: \mathcal{A} \subseteq \mathcal{S} \\ |\mathcal{A}| = k - \tau}} f(\mathcal{A}) \tag{3.17}$$

and let $\phi(\mathcal{S}^*)$ *denote the optimal solution of Problem 3.1. Then,*

$$\phi(\mathcal{S}) \geq \left(1 - e^{-\bar{\alpha}}\right) \left(\theta \, \phi(\mathcal{S}^*) - \tau \frac{\beta}{\alpha}\right) \tag{3.18}$$

where $\bar{\alpha}$ *is the approximate submodularity ratio from* (3.15) *of the function,*

$$g_{\mathcal{S}_0}(\mathcal{S}_1) = \min_{\substack{\mathcal{A}: \mathcal{A} \subseteq \mathcal{S}_0 \cup \mathcal{S}_1 \\ |\mathcal{A}| = |\mathcal{S}_1|}} f(\mathcal{A}). \tag{3.19}$$

where $\mathcal{S}_1 \subseteq \mathcal{V} \backslash \mathcal{S}_0$, *and* $\beta = \max_{i \in \mathcal{V}} f(\{i\})$.

Solution. We wish to prove a lower bound for $\phi(\mathcal{S})$ and relate it to global optimum $\phi(\mathcal{S}^*)$. We will divide the proof into three parts,

1. Provide an upper bound for $\phi(\mathcal{S}^*)$.

2. Provide a lower bound for $\phi(\mathcal{S})$.

3. Relate the quantities $\phi(\mathcal{S}^*)$ and $\phi(\mathcal{S})$.

Upper bound for $\phi(\mathcal{S}^*)$: Consider the optimal sampling set $\mathcal{S}^*$ from Problem 3.1, and let $\mathcal{W} \subseteq \mathcal{S}^*$ be its worst-case subset of size $k-\tau$, i.e., $f(\mathcal{W}) \leq f(\mathcal{W}')$ for all $\mathcal{W}' \subseteq \mathcal{S}^*$ with $|\mathcal{W}'| = k-\tau$. Then,

$$\phi(\mathcal{S}^*) = f(\mathcal{W}) \leq f(\mathcal{W} \cup \mathcal{S}_0 \cup \mathcal{R}) \tag{3.20}$$

where $\mathcal{R} \subseteq \mathcal{V} \backslash (\mathcal{W} \cup \mathcal{S}_0)$ is an arbitrary set that satisfies the condition $|\mathcal{W} \cup \mathcal{S}_0 \cup \mathcal{R}| = k$, and (3.20) follows from the fact that f is monotonically non-decreasing. Let $\mathcal{M} \subseteq \mathcal{W} \cup \mathcal{S}_0 \cup \mathcal{R}$ be a set of size $|\mathcal{M}| = k-\tau$ such that $f(\mathcal{M}) \leq f(\mathcal{M}')$ for all $\mathcal{M}' \subseteq \mathcal{W} \cup \mathcal{S}_0 \cup \mathcal{R}$ with $|\mathcal{M}'| = k-\tau$. Then, we have from (3.20),

$$\begin{aligned} f(\mathcal{W} \cup \mathcal{S}_0 \cup \mathcal{R}) &= f(\mathcal{M} \cup (\mathcal{W} \cup \mathcal{S}_0 \cup \mathcal{R}) \backslash \mathcal{M}) && (3.21) \\ &\leq \frac{1}{\theta}(f(\mathcal{M}) + f((\mathcal{W} \cup \mathcal{S}_0 \cup \mathcal{R}) \backslash \mathcal{M})) && (3.22) \\ &\leq \frac{1}{\theta}(g_{\mathcal{S}_0}(\mathcal{S}_1^*) + f((\mathcal{W} \cup \mathcal{S}_0 \cup \mathcal{R}) \backslash \mathcal{M})) && (3.23) \end{aligned}$$

where (3.22) follows from (3.16), and $g_{\mathcal{S}_0}(\mathcal{S}_1^*)$ in (3.23) is defined as,

$$g_{\mathcal{S}_0}(\mathcal{S}_1^*) = \max_{\substack{\mathcal{S}_1 : \mathcal{S}_1 \subseteq \mathcal{V} \backslash \mathcal{S}_0 \\ |\mathcal{S}_1| = k-\tau}} \min_{\substack{\mathcal{A} : \mathcal{A} \subseteq \mathcal{S}_0 \cup \mathcal{S}_1 \\ |\mathcal{A}| = |\mathcal{S}_1|}} f(\mathcal{A}), \tag{3.24}$$

hence,

$$\begin{aligned} g_{\mathcal{S}_0}(\mathcal{S}_1^*) &\geq \min_{\substack{\mathcal{A} : \mathcal{A} \subseteq \mathcal{S}_0 \cup \mathcal{T} \\ |\mathcal{A}| = k-\tau}} f(\mathcal{A}), \ \forall\, \mathcal{T} \subseteq \mathcal{V} \backslash \mathcal{S}_0 \text{ s.t. } |\mathcal{T}| = k-\tau && (3.25) \\ &\geq f(\mathcal{M}) && (3.26) \end{aligned}$$

from which (3.23) follows. By denoting $(\mathcal{W}\cup\mathcal{S}_0\cup\mathcal{R})\backslash\mathcal{M}\triangleq\{e_1,\ldots,e_\tau\}$, we find from (3.23) that

$$\begin{aligned}
&\frac{1}{\theta}(g_{\mathcal{S}_0}(\mathcal{S}_1^*)+f((\mathcal{W}\cup\mathcal{S}_0\cup\mathcal{R})\backslash\mathcal{M}))\\
&=\frac{1}{\theta}(g_{\mathcal{S}_0}(\mathcal{S}_1^*)+f(\{e_1,\ldots,e_\tau\}) &(3.27)\\
&=\frac{1}{\theta}(g_{\mathcal{S}_0}(\mathcal{S}_1^*)+\sum_{i=1}^{\tau}(f(\{e_i\}\cup\{e_{i+1},\ldots,e_\tau\})-f(\{e_{i+1},\ldots,e_\tau\})) &(3.28)\\
&\leq\frac{1}{\theta}(g_{\mathcal{S}_0}(\mathcal{S}_1^*)+\sum_{i=1}^{\tau}\frac{1}{\alpha}f(\{e_i\}) &(3.29)\\
&\leq\frac{1}{\theta}\left(g_{\mathcal{S}_0}(\mathcal{S}_1^*)+\tau\frac{\beta}{\alpha}\right) &(3.30)
\end{aligned}$$

where (3.28) is from telescopic sum, (3.29) is from (3.15) and $f(\emptyset)=0$.

Lower bound for $\phi(\mathcal{S})$: Next, note that $\mathcal{S}_0$ is the set of nodes selected in the first stage of Algorithm 4. In the second stage, the algorithm aims to solve the following set selection problem:

$$g_{\mathcal{S}_0}(\mathcal{S}_1^*)=\max_{\substack{\mathcal{S}_1:\mathcal{S}_1\subseteq\mathcal{V}\backslash\mathcal{S}_0\\|\mathcal{S}_1|=k-\tau}}\min_{\substack{\mathcal{A}:\mathcal{A}\subseteq\mathcal{S}_0\cup\mathcal{S}_1\\|\mathcal{A}|=|\mathcal{S}_1|}}f(\mathcal{A}). \tag{3.31}$$

in a greedy manner. That is, (3.13) constructs a set $\mathcal{S}_1$ of size $k-\tau$ iteratively, at each iteration by selecting the node u that essentially maximizes the function,

$$\begin{aligned}
u^*&=\arg\max_u g_{\mathcal{S}_0}(\mathcal{U}_i\cup\{u\}) &(3.32)\\
&=\arg\max_u\min_{\substack{\mathcal{A}:\mathcal{A}\subseteq\mathcal{S}_0\cup\mathcal{U}_i\cup\{u\}\\|\mathcal{A}|=|\mathcal{U}_i|+1}}f(\mathcal{A}) &(3.33)\\
&=\arg\max_u\min_{\substack{\mathcal{A}:\mathcal{A}\subseteq\mathcal{S}_0\cup\mathcal{U}_i\\|\mathcal{A}|=|\mathcal{U}_i|}}f(\mathcal{A}\cup\{u\}) &(3.34)
\end{aligned}$$

where $\mathcal{U}_i$ denotes the already selected nodes at iteration i, accordingly, $\mathcal{S}_1=\mathcal{U}_{k-\tau}$. Function $g_{\mathcal{S}_0}(\cdot)$ in (3.32) is monotonically non-decreasing, which can be proved by contradiction. Then,

by letting $\bar{\alpha}$ denote the approximate submodularity ratio of $g_{\mathcal{S}_0}(\cdot)$, one can show through similar steps from [17, 12] for bounding the performance of greedy algorithms that,

$$\phi(\mathcal{S}) = g_{\mathcal{S}_0}(\mathcal{S}_1) \geq (1 - e^{\bar{\alpha}}) g_{\mathcal{S}_0}(\mathcal{S}_1^*). \tag{3.35}$$

Relating $\phi(\mathcal{S}^*)$ and $\phi(\mathcal{S})$: (3.35) when combined with (3.30), leads to (3.18). □

We will now specify the f needed for our application. An important optimality criterion based on the error covariance $\mathbf{K}(\mathcal{S})$ from (3.9) is minimizing the mean-squared error (MSE) of the reconstructed signal. Also known as A-optimality [4], the MSE criterion is quantified by $\text{tr}(\mathbf{K}(\mathcal{S}))$. Minimizing $\text{tr}(\mathbf{K}(\mathcal{S}))$ can be equivalently represented as the maximization of $f(\mathcal{S})$ where $f(\mathcal{S})$ is defined as:

$$f(\mathcal{S}) = \text{tr}(\mathbf{\Gamma}) - \text{tr}(\mathbf{K}(\mathcal{S})) \tag{3.36}$$

$$= \text{tr}(\mathbf{\Gamma}) - \text{tr}((\mathbf{\Gamma}^{-1} + \sum_{i \in \mathcal{S}} \mu_i^{-1} \mathbf{r}_i \mathbf{r}_i^T)^{-1}) \tag{3.37}$$

which is a non-negative, monotonically non-decreasing set function, with $f(\emptyset) = 0$. In the following, we study the approximate submodularity characteristics of the MSE function.

For tractability of our further analysis of maximizing $f(\mathcal{S})$, we let $\mathbf{\Gamma} = \sigma_x^2 \mathbf{I}$ and $\mathbf{M} = \sigma_n^2 \mathbf{I}$, from which (3.37) can be written as

$$f(\mathcal{S}) = \sigma_x^2 \left(|\mathcal{F}| - \text{tr}((\mathbf{I} + \gamma \sum_{i \in \mathcal{S}} \mathbf{r}_i \mathbf{r}_i^T)^{-1}) \right), \tag{3.38}$$

where $\gamma = \sigma_x^2 / \sigma_n^2$ is the SNR of the graph signals.

The submodularity ratio α for the function f in (3.38) can be bounded following the same steps in [12][Theorem 3]. The submodularity ratio $\bar{\alpha}$, however, is based on function (3.19) instead, which may be different from f in general. As such, the next result provides a lower bound on the approximate submodularity ratio $\bar{\alpha}$.

Proposition 3.1. *The approximate submodularity ratio* $\bar{\alpha}$ *in* (3.19)*, where* f *is the MSE criterion from* (3.38)*, can be bounded below by,*

$$\bar{\alpha} \geq \frac{\gamma^{-1} + (1+\gamma)^{-1}}{\gamma^{-1} + \rho} \rho (1+\gamma)^{-2} \tag{3.39}$$

where $\rho = \min_{i \in \mathcal{V}} ||\mathbf{r}_i||^2$.

Solution. The proof follows the lines of [12], but over the worst-case solutions in (3.19), and is omitted due to space considerations. □

We observe that $\bar{\alpha}$ increases as SNR decreases. Hence the function becomes more submodular. As a result, the greedy algorithm provides a good approximation of the optimal solution in low SNR environments. In contrast, for the noiseless case, almost every subset of $|\mathcal{F}|$ samples provides perfect reconstruction. Hence, in high SNR scenarios, the specific choice of sampling set impacts the reconstruction error to a lesser extent [12].

3.4 Performance Evaluation

In our simulations, we let $\mathbf{\Gamma} = \mathbf{I}$ and $\mathbf{M} = \sigma_n^2 \mathbf{I}$ with $\sigma_n^2 = 10^{-2}$, and $|\mathcal{F}| = 5$. We compare three algorithms, Algorithm 4, the conventional greedy set selection algorithm, and the robust greedy optimization algorithm – termed as oblivious-greedy (OG) algorithm – from [9, Algorithm 1]. The OG algorithm is a 2-stage algorithm as Algorithm 4. In stage 1, nodes are selected obliviously as in Algorithm 4. In stage 2, however, the two algorithms differ. The oblivious-greedy algorithm discards the τ nodes selected in the first stage and, starting from an empty set, applies the conventional greedy algorithm for selecting $k - \tau$ nodes in the second stage. On the other hand, Algorithm 4 combines the information from samples selected in the first stage and selects the next node to maximize the performance

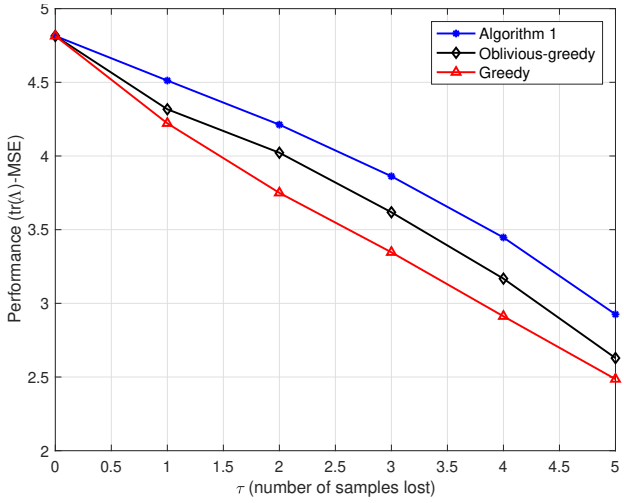
of the worst-case subset. The performance of a selected set $\mathcal{S}$ is measured by the worst-case performance:

$$\min_{\substack{\mathcal{A}:\mathcal{A}\subseteq\mathcal{S}\\|\mathcal{A}|=k-\tau}} (\mathrm{tr}(\mathbf{\Gamma}) - \mathrm{tr}(\mathbf{K}(\mathcal{A}))) \triangleq \mathrm{tr}(\mathbf{\Gamma}) - MSE. \tag{3.40}$$

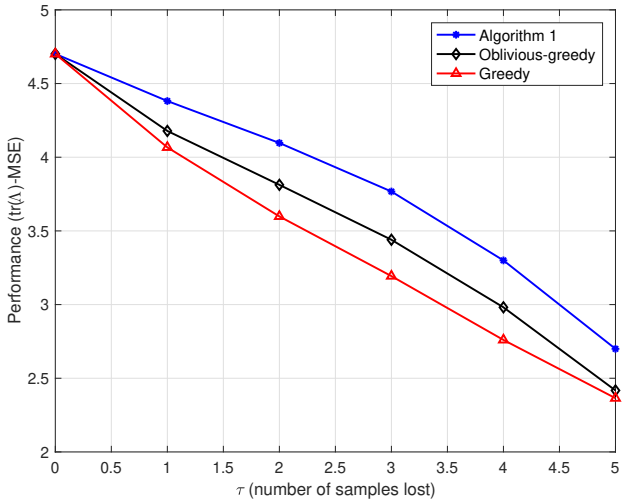
We first consider a Barabasi-Albert graph created from 4 seed nodes, which has a scale-free degree structure like many real-world topologies such as the web. The results, given in Figure 3.1, show that Algorithm 4 can provide performance gains of up to 20% improvement over the greedy algorithm and up to 12% over the OG algorithm. Next, we consider an Erdös-Rényi graph with each edge drawn with probability $p = 0.2$. The results are illustrated in Figure 3.2. In this setup Algorithm 4 can provide performance gains of up to 54% over the conventional greedy algorithm and up to 7% improvement over the OG algorithm. In our performances $\mathrm{tr}(\mathbf{\Gamma}) = 5$. Thus, in Figure 3.1 and Figure 3.2 at a performance level of 4.5, a drop in the performance by 1 for a different method indicates a difference in the SNR of 4.7dB between those two methods whereas at the performance level of 3 a drop in the performance by 1 unit indicates a drop in the SNR by 1.76dB. As expected, the performance of all three algorithms is the same when $\tau = 0$, since in this case, both Algorithm 4 and the OG algorithm reduce to the greedy algorithm. The performance improvement of Algorithm 4 over the OG algorithm becomes more significant for the scale-free network topology.

3.5 Conclusion

In this chapter, we considered a graph sampling problem in which one receives only a subset of the selected samples. For this problem, we proposed a greedy robust sample selection algorithm and investigated its performance guarantees. Numerical experiments show that the proposed setup can significantly improve the performance over conventional greedy sample selection algorithms and state-of-the-art robust set selection algorithms.

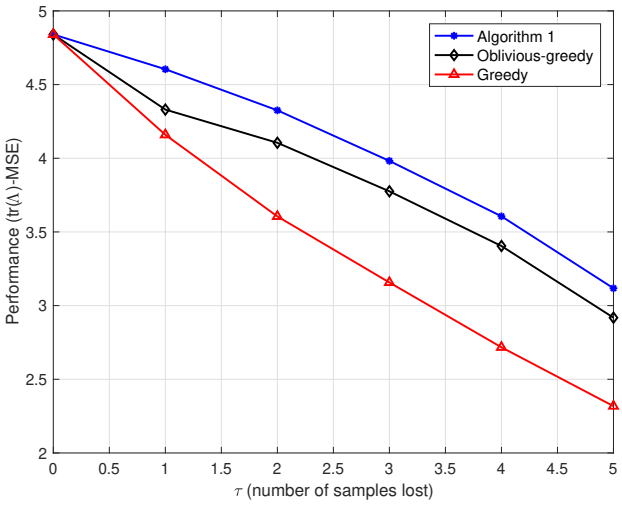


(a)

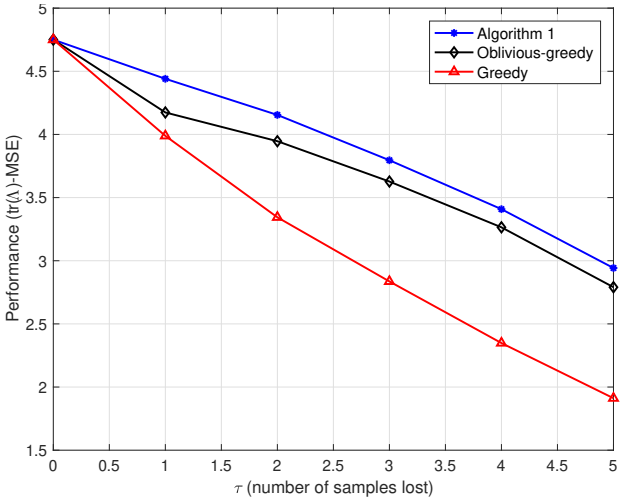


(b)

Figure 3.1: Performance comparisons for Barabasi-Albert graph with (a) $n = 100$, (b) $n = 200$.



(a)



(b)

Figure 3.2: Performance comparisons for Erdös-Rényi graph with (a) $n = 100$, (b) $n = 200$.

Chapter 4

Graph signal reconstruction with unknown signal bandwidth

4.1 Introduction

In Chapter 2, we saw that the original signal can be predicted or reconstructed by solving an inverse problem. However, note that reconstructing the signal using the bandlimited signal model requires knowledge of the signal's bandwidth. Most papers in the graph signal processing literature [14, 57, 2, 34] assume this bandwidth is known. However, in many real scenarios, the signal bandwidth is unknown. To add to this difficulty, in reality, signals are not bandlimited to a certain maximum frequency because the signal may contain added noise. Even if the signal is bandlimited for a particular graph construction, since the definition of bandwidth is dependent on graph topology a slightly different graph construction may lead to a non-bandlimited signal.

Even if the exact signal bandwidth is unknown (or if the signal is not exactly bandlimited), reconstruction using the bandlimited model remains useful because it is based on signal smoothness which is a reasonable assumption for many real-life signals like temperature. This means that it is important to optimize the choice of the bandwidth of the signal for reconstruction, regardless of whether the original signal is bandlimited or not. Since the

reconstruction error of the signal usually varies with the choice of the reconstruction bandwidth, the bandwidth which gives us the smallest error over a choice of different bandwidths would be the right choice for reconstruction. To select a bandwidth in such a way, we need to know the reconstruction error for different bandwidth choices. However, we cannot calculate the overall reconstruction error since there are unknown signal values. Thus, we need an estimate of the actual reconstruction error.

Our main contribution is to formulate the problem of selecting a reconstruction bandwidth from data, without knowledge of the actual graph signal bandwidth, a problem as yet not considered in the graph signal sampling and reconstruction literature (see [67] for a review). We propose a solution that uses a novel cross-validation methodology based on graph signal sampling concepts. Specifically, we solve the problem of estimating the reconstruction error which is essential to select a reconstruction bandwidth.

In a standard cross-validation setting [30], multiple random subsets are used to validate parameter choices. In Chapter 2, we saw that random subset selection leads to poor reconstruction performance (Figure 2.2). Similarly, using random subsets for cross-validation can result in ill-conditioned reconstruction operators. To resolve that we propose a technique that mitigates the effects of ill-conditioning by giving different importance to each random subset. This approach significantly improves error estimation, and our proposed method estimates the squared reconstruction error with good accuracy for a wide variety of both synthetic and real-life graphs and signals.

4.2 Problem formulation

4.2.1 Signal model

We consider the same bandlimited signal with noise model for the signal as in (1.2), with explicitly defined bandlimited frequency coefficients given by $\boldsymbol{\alpha}$ as follows

$$\mathbf{x} = \mathbf{U}_{\mathcal{F}}\boldsymbol{\alpha} + \mathbf{n}. \tag{4.1}$$

However, for signal reconstruction, a signal model needs to be chosen in addition to the graph.

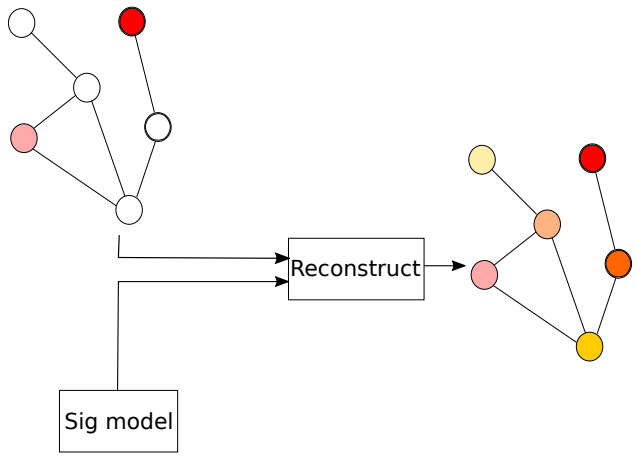


Figure 4.1: Inputs to the reconstruction algorithm

Figure 4.1 shows the reconstruction process that we will consider in this chapter. Given a graph $\mathcal{G}$, a few known samples(colored in the figure) $\mathbf{x}_{\mathcal{S}}$, the signal bandwidth f, the signal is reconstructed. This reconstruction is done for every value of signal bandwidth that we wish to test on.

4.2.2 Model selection for reconstruction

With the signal model in (4.1) and known signal values $\mathbf{x}_{\mathcal{S}}$, the signal on $\mathcal{S}^c$ can be reconstructed as:

$$\hat{\mathbf{x}}_{\mathcal{S}^c} = \mathbf{U}_{\mathcal{S}^c\mathcal{F}}(\mathbf{U}_{\mathcal{S}\mathcal{F}}^T\mathbf{U}_{\mathcal{S}\mathcal{F}})^{-1}\mathbf{U}_{\mathcal{S}\mathcal{F}}^T\mathbf{x}_{\mathcal{S}}.$$

This is a least squares reconstruction from Section 1.2.1 when the size of the known signal set is larger than the signal bandwidth used for reconstruction, $|\mathcal{S}| > f$, which is the setting we consider in this chapter. Note that this reconstruction requires the signal bandwidth f to be known, regardless of whether the signal is bandlimited or bandlimited with additional noise. Most reconstruction algorithms assume that this bandwidth is known [14, 57, 72]. However, fundamentally this is a model selection problem where an appropriate bandlimited signal model with a fixed bandwidth f must be chosen.

4.2.3 Bandwidth selection through reconstruction errors

Although the goal of model selection for signal reconstruction is to choose f, the signal itself might not be bandlimited. As a result, there may not be any prior for signal bandwidth. However, our primary goal is to minimize the reconstruction error:

$$E_{\mathcal{S}^c} = \left\|\mathbf{x}_{\mathcal{S}^c} - \hat{\mathbf{x}}_{\mathcal{S}^c}\right\|^2, \tag{4.2}$$

where the estimate $\hat{\mathbf{x}}_{\mathcal{S}^c}$ is a function of f, and so is $E_{\mathcal{S}^c}$. To select f we propose a minimization of $\left\|\hat{\mathbf{x}}_{\mathcal{S}^c} - \mathbf{x}_{\mathcal{S}^c}\right\|^2$ over a set of possible values of f, so that whichever bandwidth f minimized the error will be used as the reconstruction bandwidth, $f^* = \min_f E_{\mathcal{S}^c}$.

However, minimizing $\left\|\hat{\mathbf{x}}_{\mathcal{S}^c} - \mathbf{x}_{\mathcal{S}^c}\right\|^2$ is impossible without knowing $\mathbf{x}_{\mathcal{S}^c}$. For that reason, we propose estimating the error $\left\|\hat{\mathbf{x}}_{\mathcal{S}^c} - \mathbf{x}_{\mathcal{S}^c}\right\|^2$ for different values of f using the known signal values, $\mathbf{x}_{\mathcal{S}}$. We limit the scope of this chapter to estimating this reconstruction error and leave the bandwidth selection for future work. Towards that end, we propose an estimate,

$\hat{E}_{\mathcal{S}^c}$, of the reconstruction error $E_{\mathcal{S}^c}$ for different values of f, such that $|E_{\mathcal{S}^c} - \hat{E}_{\mathcal{S}^c}|$ is as small as possible. Cross-validation is a suitable tool for such estimations and we will consider that next.

4.3 Cross-validation theory for graph signals

In order to accurately estimate the reconstruction error as a function of the signal bandwidth f, it is essential to analyze in more detail the error with respect to subset selection on the set of graph vertices.

4.3.1 Conventional error estimation and shortcomings

The reconstruction error, $\mathbf{e}(\mathcal{S}^c)$, measured over the unknown nodes is the following:

$$\mathbf{e}(\mathcal{S}^c) = \mathbf{x}_{\mathcal{S}^c} - \hat{\mathbf{x}}_{\mathcal{S}^c} = \mathbf{x}_{\mathcal{S}^c} - \mathbf{U}_{\mathcal{S}^c\mathcal{F}}(\mathbf{U}_{\mathcal{S}\mathcal{F}}^T\mathbf{U}_{\mathcal{S}\mathcal{F}})^{-1}\mathbf{U}_{\mathcal{S}\mathcal{F}}^T\mathbf{x}_{\mathcal{S}}.$$

To estimate this error we could split the set $\mathcal{S}$ further into the sets $\{\mathcal{S}_1, \mathcal{S}_1^c\}, \cdots, \{\mathcal{S}_k, \mathcal{S}_k^c\}$ such that $\mathcal{S}_i \cup \mathcal{S}_i^c = \mathcal{S}$ for $i \in \{1, \cdots, k\}$, estimate

$$\mathbf{e}(\mathcal{S}_i^c) = \mathbf{x}_{\mathcal{S}_i^c} - \mathbf{U}_{\mathcal{S}_i^c\mathcal{F}}(\mathbf{U}_{\mathcal{S}_i\mathcal{F}}^T\mathbf{U}_{\mathcal{S}_i\mathcal{F}})^{-1}\mathbf{U}_{\mathcal{S}_i\mathcal{F}}^T\mathbf{x}_{\mathcal{S}_i},$$

and use the estimate

$$\hat{E}_{\mathcal{S}^c} = \sum_{i\in\{1,\cdots,k\}} \|\mathbf{e}(\mathcal{S}_i^c)\|^2 / k.$$

This would be equivalent to using the standard cross-validation approach that is typical in linear model selection [60].

Suppose that the noise vector has some representation,

$$\mathbf{n} = \mathbf{U}_{\mathcal{F}}\boldsymbol{\gamma} + \mathbf{U}_{\mathcal{F}^c}\boldsymbol{\beta}, \tag{4.3}$$

we can conveniently separate the bandlimited and non-bandlimited components of the signal using the following representation:

$$\mathbf{x} = \mathbf{U}_{\mathcal{F}}\boldsymbol{\alpha}' + \mathbf{U}_{\mathcal{F}^c}\boldsymbol{\beta}, \tag{4.4}$$

where

$$\boldsymbol{\alpha}' = \boldsymbol{\alpha} + \boldsymbol{\gamma}. \tag{4.5}$$

The bandlimited component of the signal has no effect on either $\mathbf{e}(\mathcal{S}^c)$ or $\mathbf{e}(\mathcal{S}_i^c)$.

Using the new representation of the signal, to simplify the notation we define

$$\mathbf{M} = \mathbf{U}_{\mathcal{S}^c\mathcal{F}^c} - \mathbf{U}_{\mathcal{S}^c\mathcal{F}}(\mathbf{U}_{\mathcal{S}\mathcal{F}}^T\mathbf{U}_{\mathcal{S}\mathcal{F}})^{-1}\mathbf{U}_{\mathcal{S}\mathcal{F}}^T\mathbf{U}_{\mathcal{S}\mathcal{F}^c}$$
$$\mathbf{M}_i = \mathbf{U}_{\mathcal{S}_i^c\mathcal{F}^c} - \mathbf{U}_{\mathcal{S}_i^c\mathcal{F}}(\mathbf{U}_{\mathcal{S}_i\mathcal{F}}^T\mathbf{U}_{\mathcal{S}_i\mathcal{F}})^{-1}\mathbf{U}_{\mathcal{S}_i\mathcal{F}}^T\mathbf{U}_{\mathcal{S}_i\mathcal{F}^c}.$$

Thus, our errors are

$$\mathbf{e}(\mathcal{S}^c) = \mathbf{M}\boldsymbol{\beta}, \tag{4.6}$$

$$\mathbf{e}(\mathcal{S}_i^c) = \mathbf{M}_i\boldsymbol{\beta} \quad i \in \{1, \cdots, k\}. \tag{4.7}$$

The matrices $\mathbf{M}$ and $\mathbf{M}_i$ are what mainly differentiate the errors $\mathbf{e}(\mathcal{S}^c)$ and $\mathbf{e}(\mathcal{S}_i^c)$. Because the subsets $\mathcal{S}_i$ are selected randomly, $\mathbf{M}_i$ can be ill-conditioned although $\mathbf{M}$ is well-conditioned. This ill-conditioning often causes the estimate of the cross-validation error to be orders of magnitude higher than the actual error.

Intuitively, this can happen in cases where $\mathcal{S}$ is well connected to $\mathcal{S}^c$ but $\mathcal{S}_i$ is poorly connected to $\mathcal{S}_i^c$. See Figure 4.2 for a toy example where all vertices in the graph are within

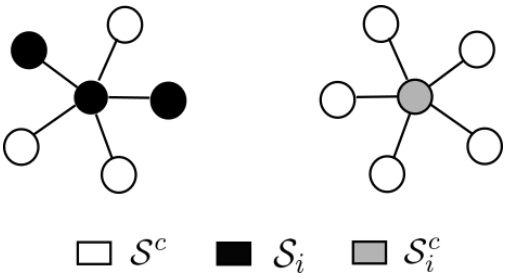


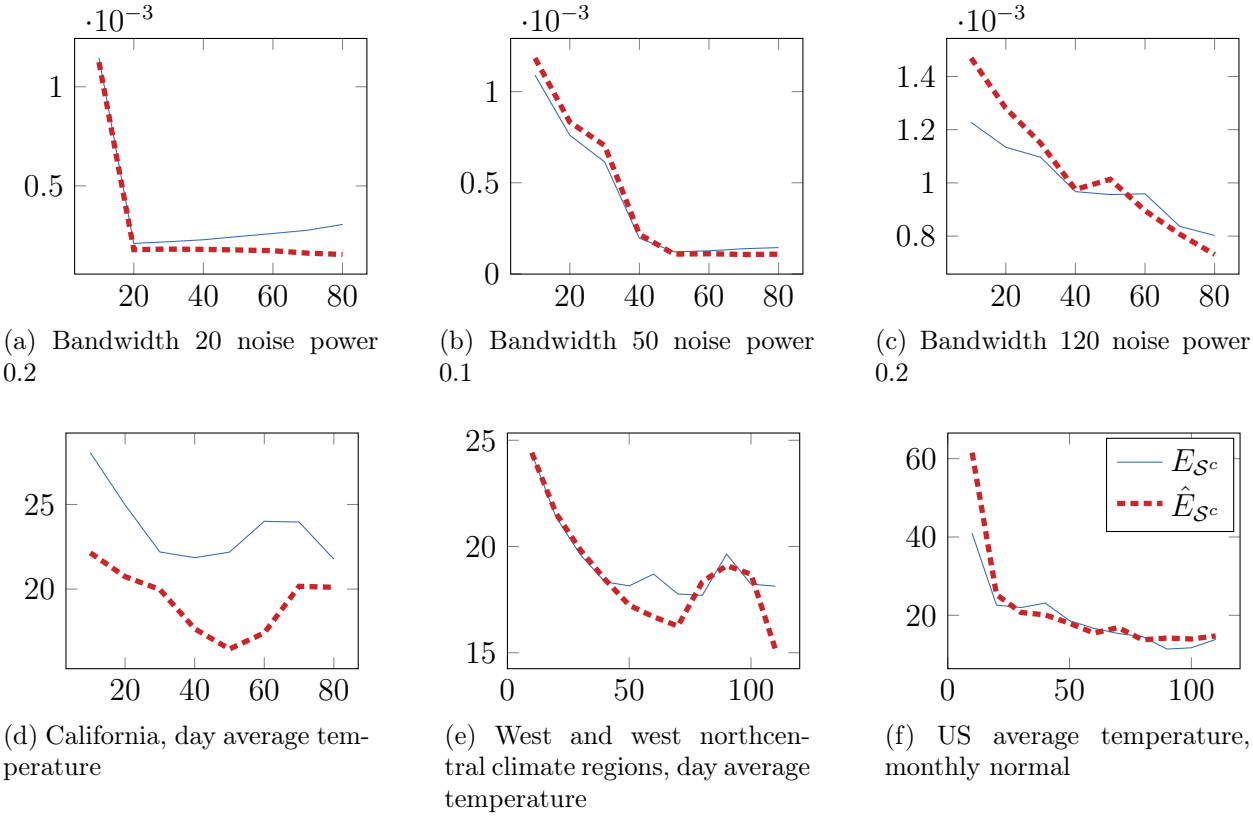
Figure 4.2: Disconnected graph case for cross-validation. In this example, we sample a single graph containing two connected components (left and right). Since the random subset $\mathcal{S}_i^c$ is disconnected from $\mathcal{S}_i$, signal values on $\mathcal{S}_i^c$ cannot be inferred from signal values on $\mathcal{S}_i$.

one hop of $\mathcal{S}$, but $\mathcal{S}_i$ is disconnected from $\mathcal{S}_i^c$. In this example, trying to reconstruct the signal on $\mathcal{S}_i^c$ using the known signal values on $\mathcal{S}_i$ is impossible because the graph is disconnected and we only observe samples from one of the connected components. Thus, we cannot achieve a meaningful reconstruction for nodes in the other (unobserved) connected component. This can be viewed as an extreme case of ill-conditioning since the graph is disconnected graphs, but this issue manifests itself for connected graphs and would be reflected as an ill-conditioned $\mathbf{M}_i$.

4.3.2 Proposed error estimation

As we noted in Section 4.3.1, averaging the error over random subsets may lead to the blowing up of the error estimate due to the ill-conditioning of the reconstruction matrices. Such ill-conditioning is caused when the magnitude of $\|\mathbf{M}_i\|$ in (4.7) is large. To obtain reliable estimates in the presence of ill-conditioning, we propose a weighted averaging to estimate the bandwidth, where we assign different importance (weight) to the reconstruction errors estimated from different random subsets. Consider the singular value decomposition $\mathbf{M}_i$ and the resulting expression for the reconstruction error on the set $\mathcal{S}_i^c$:

$$\mathbf{M}_i = \mathbf{V}_i \mathbf{\Sigma}_i \mathbf{W}_i^T, \quad \mathbf{e}(\mathcal{S}_i^c) = \mathbf{V}_i \mathbf{\Sigma}_i \mathbf{W}_i^T \boldsymbol{\beta}. \tag{4.8}$$



(a) Bandwidth 20 noise power 0.2

(b) Bandwidth 50 noise power 0.1

(c) Bandwidth 120 noise power 0.2

(d) California, day average temperature

(e) West and west northcentral climate regions, day average temperature

(f) US average temperature, monthly normal

Figure 4.3: Squared reconstruction errors vs bandwidth for bandlimited signal model. Legend is common for all the plots.

We can see that $\|\mathbf{M}_i\|$ will be large when the singular values in $\mathbf{\Sigma}_i$ are large. Thus, to limit the increase in $\|\mathbf{M}_i\|$ due to $\mathbf{\Sigma}_i$, we propose to clip the singular values: We leave the singular value σ in $\mathbf{\Sigma}_i$ unchanged if $\sigma < 1$, and clip it to 1 if $\sigma \geq 1$. Define the singular value matrix $\mathbf{\Sigma}_i'$ with the singular values as follows:

$$\sigma' = \begin{cases} \sigma, & \sigma > 1 \\ 1, & \sigma \leq 1 \end{cases} \tag{4.9}$$

This helps in preserving the changes in the magnitude of $\|\mathbf{e}(\mathcal{S}_i^c)\|$ due to $\|\boldsymbol{\beta}\|$ but limits the effect of $\mathbf{M}_i$ having a large condition number. Although we decomposed $\mathbf{M}_i$, it is worth keeping in mind that we only have access to $\mathbf{e}(\mathcal{S}_i^c)$ and to control the magnitude of this error, we can pre-multiply with a matrix. To achieve the transformation in the singular values, we

multiply $\mathbf{e}(\mathcal{S}_i^c)$ with $\mathbf{\Sigma}_i'$, the inverse of the clipped matrix of singular values, to obtain a new error

$$\mathbf{e}_{\text{new}}(\mathcal{S}_i^c) = (\mathbf{\Sigma}_i')^{-1}\mathbf{V}_i^T\mathbf{e}(\mathcal{S}_i^c), \tag{4.10}$$

In (4.10), pre-multiplication of the existing error vector $\mathbf{e}(\mathcal{S}_i^c)$ by $(\mathbf{\Sigma}_i')^{-1}\mathbf{V}_i^T$ can be interpreted as giving more importance to certain vertices while ignoring others. Given that $\mathbf{V}_i^T$ is not diagonal, the weights in $(\mathbf{\Sigma}_i')^{-1}$ are not directly applied to individual vertices. The weights on individual vertices can be seen as weighted averaging by the matrix $\mathbf{V}_i(\mathbf{\Sigma}_i')^{-1}\mathbf{V}_i^T$. Finally, we estimate the error using

$$\hat{E}_{\mathcal{S}^c} = \frac{\sum_{i\in\{1,\cdots,k\}} \left\|(\mathbf{\Sigma}_i')^{-1}\mathbf{V}_i^T\mathbf{e}(\mathcal{S}_i^c)\right\|^2}{k} \tag{4.11}$$

$$\hat{E}_{\mathcal{S}^c} = \frac{\sum_{i\in\{1,\cdots,k\}} \left\|\mathbf{e}_{\text{new}}(\mathcal{S}_i^c)\right\|^2}{k}. \tag{4.12}$$

4.4 Experiments

4.4.1 Graph construction

For the initial verification of our error estimation approach, we construct random regular graphs with 1000 vertices according to the model `RandomRegular` from [18]. We define noisy bandlimited signals with bandwidths $\{20, 50, 120\}$ and power 1 and noise power levels 0.1 and 0.2 according to the model in (4.1). We call these graphs and signals *synthetic* graphs and signals for our experiments.

For the next experimental validation, we use publicly available climate data from the National Oceanic and Atmospheric Administration (NOAA) [70] measured by sensors throughout the United States. The sensor data consists of different weather measurements such as average daily temperature or precipitation, along with the corresponding sensors' latitudes, longitudes, and altitudes.

Geographical region	Signal	Number of samples
California	Avg. day temperature	100
West and west north central	Avg. day temperature	200
US	Avg. temperature monthly normal	200

Table 4.1: Setting for cross-validation experiments on sensor networks and weather data.

Using the locations, we construct graphs by connecting the five nearest sensor locations to each sensor. The edge weights of the graph are given by $e^{-d^2/2\sigma^2}$, where we experimentally choose $\sigma = 50$. We calculate the distance d between the measurement locations using the latitude, longitude, and altitude of the measuring station using $\sqrt{d_f^2 + d_a^2}$. d_f is the flat distance computed using the `distance` package from `geopy` library, and d_a is the altitude. While constructing the graph, we drop sensors whose measurements are missing because there is no way to verify our predictions for those sensors. The measurements we include as signals are day averages measured on 3^{rd} January 2020 and monthly normals [20], which are average measurements for January 2010.

4.4.2 Set selection

In Section 4.2, we assume that the signal values on a vertex set $\mathcal{S}$ are known. To select this set for the constructed graphs on which we assume signal values are known, we use the AVM algorithm from Chapter 2 to sample 200 vertices from each graph and observe the reconstruction errors on the frequencies $\{10, 20, \cdots, 110\}$. The only exception is the California sensor network graph, where we sample 100 vertices and observe the reconstruction errors over the frequencies $\{10, 20, \cdots, 80\}$ because the graph contains only 300 vertices. We summarize this in Table 4.1.

To estimate the reconstruction error using cross-validation, we partition each sampling set $\mathcal{S}$ into 10 subsets using `RepeatedKFold` function from `model_selection` package of `sklearn`. We measure the squared reconstruction error on each subset of the partition, for 50 different random partitions, and average the squared reconstruction errors using (4.12).

4.4.3 Results

We can see the results of our estimation in Figure 4.3. The estimated cross-validation error tracks the actual error in the wide variety of the graphs and graph signals that we experiment with. We note that in 4.3a the actual error increases slightly. However, the estimated error does not increase with it. This is due to the error weighting strategy proposed in (4.10). Since for the problem of choosing the bandwidth we are interested in correctly locating the lowest value of the actual error, the ability of the error estimate to track the actual error as it increases should be of lesser importance than its ability to track the actual error as it decreases. A more accurate error estimation could be achieved with different set selection or error weighting strategies for cross-validation, which we reserve for future work.

4.5 Conclusion

In this chapter, we proposed a way to minimize graph signal reconstruction error without assuming the knowledge of the signal bandwidth. In the process, we tailored the cross-validation method for the problem of reconstruction error estimation. Our technique accurately estimated the error as a function of the signal bandwidth on various bandlimited signals with noise and also for sensor networks measuring weather.

Chapter 5

Subgraph-based parallel sampling for large point clouds

5.1 Introduction

Analyzing and processing 3D structures is important for many applications such as autonomous driving [32], geological elevation models [65], and preserving information from historic sites [6]. Point clouds are one of the simplest data structures created when scanning such 3D structures. For these reasons, processing point clouds is important for success in these applications.

To prevent loss of information at the source while capturing 3D structures, point clouds typically consist of closely spaced points. Realistic point clouds easily contain a hundred thousand to a million nodes. However, not all parts of a point cloud need to be rendered in the highest resolution for various applications such as detecting defects in terrestrial scans [66] or segmentation [41]. Moreover, transmitting the point cloud requires the bandwidth to be minimized. Point cloud subsampling is a popular approach to reduce the size of the point cloud to process it for downstream tasks. In the past, point cloud subsampling approaches have been used for downsampling the geometry of the point cloud. However, point cloud attribute subsampling is equally important for applications such as attribute compression for immersive communication [42]. We will consider attribute subsampling in this chapter.

Point clouds are often scans of 3D objects. As a result, most points lie on the surface of the scanned objects. Since graphs are a good representation for manifolds, we consider graphs

for this application of subsampling point clouds. We will consider the graph sampling and reconstruction problem for graph signals. Point clouds often contain millions of data points. Graph sampling methods in the literature, including those presented in this thesis, are often not scalable to the size of millions of nodes in the graph. Therefore fast graph algorithms need to be developed for sampling point clouds. We devote this chapter to developing such algorithms. One way to speed up the sampling of point clouds is by parallelizing the sampling process. For that, we propose dividing the point clouds into smaller point clouds, constructing graphs on the smaller point clouds, and sampling graph signals on those smaller subgraphs.

This chapter makes the following contributions to the problem of sampling point cloud attributes:

- We propose a method of parallelizing graph sampling algorithms for large graphs through partitioning of the entire graph into subgraphs
- The proposed algorithm provides at least 0.8dB gain in performance compared to uniform sampling for all datasets we consider
- As future work, we propose approximations to speed up the algorithm even further over the approach from Chapter 2

5.2 Problem formulation

A point cloud is a set of coordinates (x, y, z) and their corresponding attributes. Attributes can contain object information, such as color, or structural information, such as surface normals. We consider the kNN graph constructed using coordinates of the point cloud and use the conventions and notations defined in Section 1.1. For the point cloud attribute signals that we consider, we use the bandlimited model with noise from (1.2) with the luminance channel as our signal $\mathbf{x}$. We want to be able to measure the values of the signal at a few

nodes $\mathcal{S}$ and predict the entire signal $\hat{\mathbf{x}}$ from those values. We measure the error as in (4.2) on the vertices where the signal value is unknown.

$$E_{\mathcal{S}^c} = \|\mathbf{x}(\mathcal{S}^c) - \hat{\mathbf{x}}(\mathcal{S}^c)\|^2 \tag{5.1}$$

The least squares reconstruction in (1.1) is computationally expensive for the entire graph. We will adopt a simpler reconstruction for signals on point clouds. We will use weighted edges to connect each unknown-signal point to its k nearest known-signal points. These weights are used for reconstruction, with each unknown value estimated as a weighted average of the attributes of its nearest neighbors with known attributes. More specifically,

$$\hat{\mathbf{x}}_i = \frac{w_{1i}x_1 + \cdots + w_{ki}x_k}{w_{1i} + \cdots + w_{ki}},$$

where w_{ij} are the edge weights in the graph constructed for the point cloud, and x_i are luminance values at the points in the cloud.

(a) $\mathcal{G}$ (b) $\mathcal{G}_1$ and $\mathcal{G}_2$

Figure 5.1: Original graph and its division into subgraphs for enabling the parallelized sampling algorithm.

5.3 Distributed sampling

Towards the goal of distributed sampling for graph signals, we propose dividing the point cloud into smaller point clouds. We construct graphs from these smaller point clouds which we call subgraphs and apply a graph sampling algorithm to each subgraph.

5.3.1 Problem formulation

5.3.1.1 Local sampling global reconstruction

We will assume that the partition of the graph into subgraphs is given. In practice, in this chapter, point clouds are partitioned using octrees with suitable depth and each subgraph connects all points within one of the octree volumes forming the partition. For simplicity, consider Figure 5.1 where the graph is divided into two subgraphs, $\mathcal{G}_1$ and $\mathcal{G}_2$, with corresponding vertex sets $\mathcal{V}_1$ and $\mathcal{V}_2$, respectively. Instead of solving the problem of selecting the best sampling set on the entire graph, we solve the problem of sampling in the individual graphs such that the original objective (5.1) is minimized. For a sampling set $\mathcal{S}_1$ selected from graph $\mathcal{G}_1$, and a sampling set $\mathcal{S}_2$ selected from graph $\mathcal{G}_2$, we can reconstruct $\mathbf{x}(\hat{\mathcal{S}}_1^c)$ and $\hat{\mathbf{x}}(\mathcal{S}_2^c)$. From (2.5) the sampling set selection problem for the entire graph can be formulated as:

$$\mathcal{S}^* = \underset{\mathcal{S}\subset\mathcal{V},|\mathcal{S}|=s}{\arg\max} \det(\mathbf{U}_{\mathcal{S}\mathcal{R}}\mathbf{U}_{\mathcal{S}\mathcal{R}}^T). \tag{5.2}$$

After partitioning the graph, we wish to formulate separate optimization problems on the two subgraphs of the original graph:

$$\mathcal{S}_1^* = \underset{\mathcal{S}_1\subset\mathcal{V},|\mathcal{S}_1|=s_1}{\arg\max} \det(\mathbf{U}_{\mathcal{S}_1\mathcal{R}}\mathbf{U}_{\mathcal{S}_1\mathcal{R}}^T) \tag{5.3}$$

$$\mathcal{S}_2^* = \underset{\mathcal{S}_2\subset\mathcal{V},|\mathcal{S}_2|=s_2}{\arg\max} \det(\mathbf{U}_{\mathcal{S}_2\mathcal{R}}\mathbf{U}_{\mathcal{S}_2\mathcal{R}}^T) \tag{5.4}$$

$$\text{such that,} \tag{5.5}$$

$$s = s_1 + s_2, \text{and} \tag{5.6}$$

$$\mathcal{S}_1^* \cup \mathcal{S}_2^* = \mathcal{S}^* = \underset{\mathcal{S}\subset\mathcal{V}_2,|\mathcal{S}|=s}{\arg\max} \det(\mathbf{U}_{\mathcal{S}\mathcal{R}}\mathbf{U}_{\mathcal{S}\mathcal{R}}^T) \tag{5.7}$$

This leaves the problem that optimizing problem (5.3) and (5.4) generally does not provide an optimal solution for (5.2). This is illustrated by Figures 5.3band 5.3b, where we can see that independent sampling in each subgraph (Fig. 5.3b) would lead to inefficiencies in

a global interpolation, since points that are neighbors in the joint graph (but happen to be in different subgraphs) have been sampled. To ameliorate this, we propose modifying the subgraphs so that sampling the individual subgraphs independently can better mimic the global sampling over the entire graph.

(a) Uniformly sampled graph

(b) Uniformly sampled graphs after partitioning

Figure 5.2: Illustration of how uniform sampling works before and after partitioning.

5.3.1.2 Proxy for optimizing the sampling over subgraphs

We designed AVM in Chapter 2 to solve (5.2). The solution $\mathcal{S}$ obtained through AVM depends on the underlying graph predominantly through $\mathbf{d}_v$s. A proxy for obtaining similar sampling results using AVM on $\mathcal{G}_1$ as that on $\mathcal{G}$ is to make $\mathbf{d}_v$ the same on both the graphs. More specifically, we would want

$$\mathbf{d}_{v,\mathcal{G}}(\mathcal{V}_1) = \mathbf{d}_{v,\mathcal{G}_1}, \quad \forall v \in \mathcal{V}_1, \tag{5.8}$$

where we have used an additional subscript to denote the graph that $\mathbf{d}_v$ is evaluated on. Note that $\mathbf{d}_{v,\mathcal{G}}$ has length n but we only wish to compare the vectors over $\mathcal{G}_1$. Hence, we consider a subset of the vector that corresponds to $\mathcal{V}_1$. In general (5.8) cannot be achieved exactly. Here we propose to modify the graph $\mathcal{G}_1$ to obtain a new graph $\mathcal{G}_1'$ for which (5.8) can be approximately satisfied, that is,

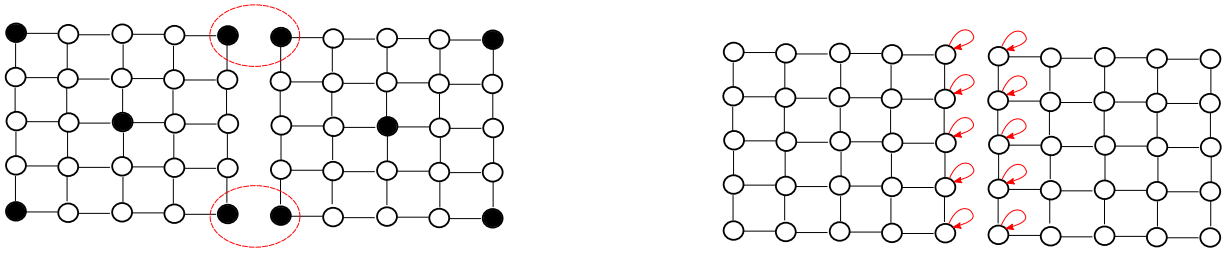
$$\mathcal{G}_1'^{*} = \arg\min_{\mathcal{G}_1'} \sum_{v \in \mathcal{V}_1} \left\| \mathbf{d}_{v,\mathcal{G}}(\mathcal{V}_1) - \mathbf{d}_{v,\mathcal{G}_1'} \right\|^2. \tag{5.9}$$

The operators can be seen as low-pass filters with a cut-off frequency. The first step to having the same operators for the original graph and the subgraphs is to have the same cutoff frequency for the operator on the subgraph and on the original graph. Using polynomials to synthesize the filter, this can be achieved by setting the cutoff frequency before computing the polynomial filter coefficients.

(a) AVM sampled graph

(b) AVM sampled graphs after partitioning

Figure 5.3: Illustration of how AVM sampling works before and after partitioning.



(a) Issue with AVM sampling after partitioning

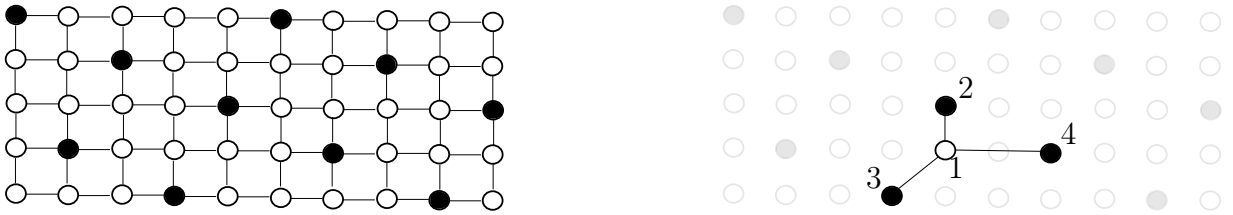
(b) Proposed graph modification for AVM sampling after partitioning

Figure 5.4: AVM tends to favor sampling more isolated (lower degree) nodes. Our proposed solution of adding self-loops prevents this bias towards boundary nodes.

Since we represent filters by their corresponding polynomials, we can make the filter operator the same by using the same polynomial coefficients corresponding to a single cutoff frequency. However, we also want the result of filtering operations on the graph signals to be the same. This is more difficult to achieve because the domain of the filter has changed. We still want the effects of the filter to be the same on the smaller domain. To ensure it is enough to ensure that the signals obtained by filtering delta signals to be the same.

Consider a polynomial representation of a filter $p(\mathbf{L}_1)$ where the order of the polynomial is k. If the boundary of $\mathcal{G}_1$ is at a distance of k-hops or more from a vertex 1, $p(\mathbf{L}_1)\boldsymbol{\delta}_\mathbf{1} = p(\mathbf{L})\boldsymbol{\delta}_\mathbf{1}$.

However, if vertex 1 is less than k-hops away from the boundary, the effects of the filter will not be the same. The effects of the filter only depend on two factors, the frequency response of the filter and the graph input to the filter. If we want to use the same filter for the entire signal, and also keep the effects of the filter the same for delta signals localized in the interior of the graph, we have to modify the graph Laplacian.



(a) Uniformly sampled graph

(b) Uniformly sampled graphs after partitioning

Figure 5.5: Illustration of how uniform sampling works before and after partitioning.

5.3.2 Distributed sampling through graph modifications

To improve the performance of independent sampling of subgraphs, we propose to modify the subgraphs such that the effect of the polynomial filtering on the subgraphs is as close as possible to filtering on the original graph. Essentially this entails introducing graph modifications at the boundaries between subgraphs, i.e., where edges were removed to partition the original graph.

For concreteness, we will consider the graph $\mathcal{G}$ which is split into graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ as in Figure 5.1. Without loss of generality, we will solve the problem for $\mathcal{G}_1$. We consider changing edge weights for all edges in the graph $\mathcal{G}_1$ and adding self-loops to all vertices. Since our goal is to compensate for the effect of partitioning on filtering operations, we consider the simplest possible graph filter, i.e., a first-order polynomial filter:

$$p(\mathbf{L}) = c_0\mathbf{I} + c_1\mathbf{L}. \tag{5.10}$$

Problem 5.1. *We propose to modify edge weights between vertices and add self-loops to vertices of $\mathcal{G}_1$ such that Laplacians of $\mathcal{G}$, and the modified graph $\mathcal{G}_1'$ satisfy*

$$(c_0\mathbf{I} + c_1\mathbf{L})\boldsymbol{\delta_i} = (c_0\mathbf{I} + c_1\mathbf{L}_1')\boldsymbol{\delta_i}, \quad \forall i \in \mathcal{V}_1. \tag{5.11}$$

Solution. We want to compare the effects of this filter on graph vertices in $\mathcal{G}_1$. Consider Figure 5.2b, where one edge connects graph $\mathcal{G}_1$ with graph $\mathcal{G}_2$. Let the change in edge weights be ϕ_{ij} and self-loops of degrees ϕ_i be added to each vertex.

$$\mathbf{L}_1' = \begin{bmatrix} w_{12} + w_{13} + \phi_{12} + \phi_{13} + \phi_1 & -w_{12} - \phi_{12} & -w_{13} - \phi_{13} \\ -w_{12} - \phi_{12} & w_{12} + \phi_{12} + \phi_2 & 0 \\ -w_{13} - \phi_{13} & 0 & w_{13} + \phi_{13} + \phi_3 \end{bmatrix} \tag{5.12}$$

First, consider the case a filtered delta signal localized in the interior of $\mathcal{G}_1$.

$$\mathbf{L}_1'\boldsymbol{\delta_2} = \begin{bmatrix} -w_{12} - \phi_{12} \\ w_{12} + \phi_{12} + \phi_2 \\ 0 \end{bmatrix} \tag{5.13}$$

For these points, we can see that equality (5.1) can be achieved, i.e.,

$$\mathbf{L}_1'\boldsymbol{\delta_2} = \mathbf{L}\boldsymbol{\delta_2}(\mathcal{V}_1), \tag{5.14}$$

for $\phi_{12} = 0$ and $\phi_2 = 0$. This means that when comparing the effects of a first-order filter, the edges and vertices in the interior of the subgraph should remain the same.

Next, consider the effects of the filter on a boundary vertex.

$$\mathbf{L}_1'\boldsymbol{\delta_1} = \begin{bmatrix} w_{12} + w_{13} + \phi_{12} + \phi_{13} + \phi_1 \\ -w_{12} - \phi_{12} \\ -w_{13} - \phi_{13} \end{bmatrix} \tag{5.15}$$

On equating the filtered delta signal using $\mathbf{L}$ and $\mathbf{L}_1'$, where $\mathbf{\Phi}$ is unknown,

$$\mathbf{L}_1'\boldsymbol{\delta_1} = \mathbf{L}\boldsymbol{\delta_1}(\mathcal{V}_1), \tag{5.16}$$

we observe that we need to add a self-loop with edge weight w_{14} to achieve equality while the edge weights remain unchanged.

$$\mathbf{L}_1' = \begin{bmatrix} w_{12} + w_{13} + w_{14} & -w_{12} & -w_{13} \\ -w_{12} & w_{12} & 0 \\ -w_{13} & 0 & w_{13} \end{bmatrix} \tag{5.17}$$

More generally, the weight of the self-loop added to a vertex v in $\mathcal{G}_1$ is the sum total of all edges connecting v to $\mathcal{G}_2$ in $\mathcal{G}$.

□

We will use the self-loop modification for vertices on the boundary of the partitioned graphs for experiments and demonstrate that the performance comparably improves while providing us with significant parallelization capabilities.

5.4 Experiments and Results

We consider the point clouds in the Microsoft voxelized upper bodies dataset [43] and 8i Voxelized Full Bodies dataset [21]. The dataset consists of point clouds for different people, with multiple point clouds of each person in different postures. For our experiments, we will

use one point cloud for each unique person in the dataset, and we consider the luminance channel of the point cloud as our signal.

Figure 5.6: Point clouds for experiments

Table 5.1: Reconstruction PSNRs for various sampling algorithms

Dataset	Uniform	AVM	AVM-SL
Sarah	46.62	47.43	**47.66**
Andrew	34.5	35.19	**35.31**
Longdress	31.5	32.23	**32.43**
Redandblack	37.87	38.94	**39.15**
Soldier	35.8	37.05	**37.29**
Loot	39.91	40.9	**41.14**
David	45.4	46.36	**46.53**
Phil	36.62	37.33	**37.42**
Ricardo	46.59	47.23	**47.55**

We divide each point cloud using octrees in blocks of size $2^{64} \times 2^{64}$. Increasing the subgraph sizes leads to improvement in the performance, at the expense of more computational time however we don't focus on that in this chapter. We construct k nearest neighbor graphs over each of the smaller point clouds with $k = 5$. We sample the subgraphs created by each block using multiple algorithms and reconstruct them using (5.2) from 10 nearest known

Table 5.2: Algorithm execution times in secs.

Dataset	Uniform	AVM	AVM-SL
Sarah	0.013	284	297
Andrew	0.021	457	468
Longdress	0.013	194	193
Redandblack	0.011	161	160
Soldier	0.014	226	227
Loot	0.011	141	144
David	0.023	371	380
Phil	0.024	425	398
Ricardo	0.013	229	230

samples. We compare the PSNR resulting from samples from three sampling algorithms — Uniform sampling [64], AVM, AVM with self-loops (AVM-SL). We run all the AVM algorithms in parallel on 8 CPU cores.

From Table 5.1, we can see that AVM outperforms uniform sampling in terms of reconstruction performance by 1dB in most cases. Additionally, AVM SL attains about 0.3dB gain in performance in most cases. In table 5.2 we see that Uniform sampling is faster than all the AVM-based algorithms. AVM and AVM-SL, on the other hand, take the longest to finish. However, it is worth noting that point clouds like Phil contain more than a million points, and AVM and AVL-SL finish executing within 8 minutes for all datasets we considered. In comparison, executing an unparallelized version of a fast algorithm like AVM takes time in hours and the algorithms we developed finish at least ten times faster.

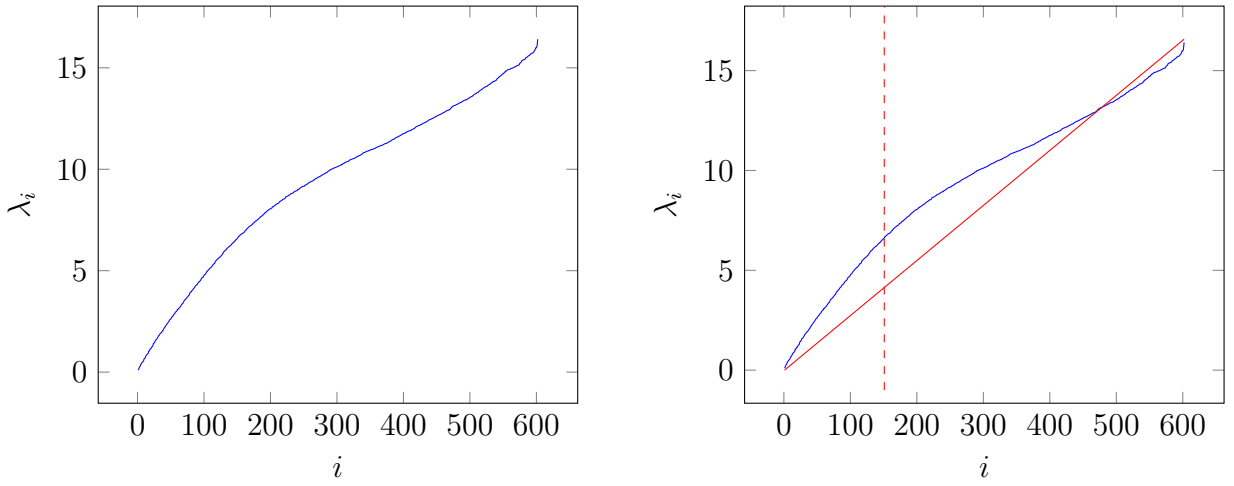
5.4.1 Further approximations for faster sampling

To speed up AVM-SL even further, we propose the following modifications as future work.

5.4.1.1 Leveraging degree information

Even after sampling in parallel after the self-loop modification, there is scope for making the sampling algorithm faster. The three sources of complexity in AVM come from

1. Frequency estimation



(a) Eigenvalue spectrum

(b) Eigenvalue and spectrum estimation

Figure 5.7: Estimating λ_k using $\lambda_{\max}$

2. Coherence estimation

3. Evaluating polynomials on the graph

To solve these issues, we propose the following modifications to the AVM algorithm:

1. First order approximation to the frequency spectrum

2. Coherence estimation using degree and neighbors

3. Reducing polynomial degrees

These modifications can also be used for applications other than point clouds.

5.4.1.2 First order approximation to frequency spectrum

The computation of coherences in Algorithm 2 involves the estimation of two frequencies - $\lambda_{\max}$ and λ_k. Extreme eigenvalues like $\lambda_{\max}$ can be iteratively computed with the complexity of each iteration linear in the number of edges of the graph — [37], but estimating λ_k is more computationally expensive. To address this λ_k can be estimated as follows:

$$\hat{\lambda}_k = \frac{k\lambda_{\max}}{n}. \tag{5.18}$$

For graphs, such as kNN, constructed from points whose pairwise distances do not show extreme variations such as those we study here, the frequency spectrum can be reasonably estimated using a linear approximation. Figures 5.7a and 5.7b illustrate this estimation for a block in the soldier point cloud.

5.4.1.3 Coherence estimation using degree and neighbors

Estimating coherence in Algorithm 2 is another computationally expensive task. However, we note that coherences, graph vertex degrees, and neighborhoods are highly correlated. This is because for higher filter frequencies corresponding to larger sampling rates, $p(\mathbf{L})\boldsymbol{\delta_i}$ is localized to the neighborhood around the vertex. As a result, the polynomial filters have dominant weights corresponding to the zeroeth and first-order coefficients. These zeroeth and first-order coefficients correspond to the edge weights of the neighbors of a vertex and vertex degrees. We propose estimating the coherences using a function of the vertex degrees and the 1-hop neighborhood of the vertices.

We estimate the coefficients of the polynomial using the Chebyshev polynomial approximation [55]. We know that the diagonal of the filter operator matrix gives us the squared coherence:

$$p(\mathbf{L})(i,i) = \|\mathbf{d}_i\|^2 \tag{5.19}$$

5.4.1.4 Reducing polynomial degrees

Finally, filtering signals using polynomials over the graph is another expensive computation. The computational requirement is linear in the order of the polynomial. To alleviate the computational cost, we reduce the polynomial order to 1.

5.5 Conclusion

In this chapter, we considered applications of the AVM algorithm to a million node graphs with high sampling rates. In the process, we extended AVM to be parallelizable. To improve the performance of parallelized algorithm, we proposed modifications to the graph of the point cloud and demostrated the performance improvements on several large-scale point clouds. In addition, we provided modifications to speed up the existing parallelized implementation by more than ten times.

Chapter 6

Conclusion

The sampling problem for traditional signals like audio and images is well-studied. In this thesis, we considered the setting of sampling and reconstruction (Figure 6.1) of unstructured data represented as graph signals.

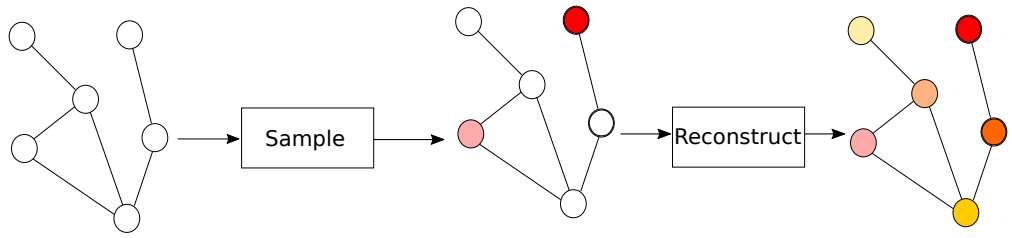


Figure 6.1: Sampling and reconstruction pipeline

It is necessary that graph signal sampling and reconstruction algorithms should work on a diverse set of graph types and sizes, unknown signal models, corruption of input signal, and real-time applications. We devote various chapters — see Table 6.1, for these problems.

Table 6.1: Proposed algorithms and contributions

Chapter	Problem considered
2	Computationally scalable sampling algorithm
3	Signal corruption and sample loss
4	Unknown signal model for reconstruction
5	Towards real-time sampling algorithms through parallelization

Through considering the sampling and reconstruction problem for graph signals in various settings such as semi-supervised learning, sensor networks, point clouds, we showed that the sampling algorithms that traditionally existed in for structured signals can also be extended for graph signals and made scalable and robust. We proposed several algorithms that improve the current state-of-art sampling and reconstruction strategies. Table 6.2 summarizes those algorithm contributions.

Table 6.2: Proposed algorithms and contributions

Algorithm	Contribution
DC	Sampling using graph distances
AVM	Scalable graph sampling algorithm
Robust sampling	Sampling preemptively against data loss
Error estimation	Towards signal reconstruction with unknown signal bandwidth
AVM-SL	Sampling algorithm parallelization

Through the proposed algorithms which are faster and more robust for sampling and reconstruction for graph signals, we pushed the limits of graph signal processing in this thesis.

References

[1] Oleksii Abramenko and Alexander Jung. Graph signal sampling via reinforcement learning. In ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 3077–3081. IEEE, 2019.

[2] Aamir Anis, Akshay Gadde, and Antonio Ortega. Efficient sampling set selection for bandlimited graph signals using graph spectral proxies. IEEE Transactions on Signal Processing, 64(14):3775–3789, 2015.

[3] Aamir Anis, Akshay Gadde, and Antonio Ortega. Efficient sampling set selection for bandlimited graph signals using graph spectral proxies. IEEE Transactions on Signal Processing, 64(14):3775–3789, 2016.

[4] Anthony Atkinson, Alexander Donev, and Randall Tobias. Optimum experimental designs, with SAS, volume 34. Oxford University Press, 2007.

[5] Yuanchao Bai, Fen Wang, Gene Cheung, Yuji Nakatsukasa, and Wen Gao. Fast graph sampling set selection using gershgorin disc alignment. IEEE Transactions on Signal Processing, 68:2419–2434, 2020.

[6] Ahmad Baik. From point cloud to jeddah heritage bim nasif historical house – case study. Digital Applications in Archaeology and Cultural Heritage, 4:1–18, 2017.

[7] Saeed Basirian and Alexander Jung. Random walk sampling for big data over networks. In 2017 International Conference on Sampling Theory and Applications (SampTA), pages 427–431. IEEE, 2017.

[8] Alain Berlinet and Christine Thomas-Agnan. Reproducing kernel Hilbert spaces in probability and statistics. Springer Science & Business Media, 2011.

[9] Ilija Bogunovic, Junyao Zhao, and Volkan Cevher. Robust maximization of non-submodular objectives. International Conference on Artificial Intelligence and Statistics (AISTATS'18), 84, 2018.

[10] Béla Bollobás. Modern graph theory, volume 184. Springer Science & Business Media, 2013.

[11] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. Convex optimization. Cambridge university press, 2004.

[12] Luiz F. O. Chamon and Alejandro Ribeiro. Greedy sampling of graph signals. IEEE Transactions on Signal Processing, 66(1):34–47, 2018.

[13] Luiz FO Chamon and Alejandro Ribeiro. Greedy sampling of graph signals. arXiv preprint arXiv:1704.01223, 2017.

[14] Siheng Chen, Rohan Varma, Aliaksei Sandryhaila, and Jelena Kovačević. Discrete signal processing on graphs: Sampling theory. IEEE Transactions on Signal Processing, 63(24):6510–6523, 2015.

[15] Ali Çivril and Malik Magdon-Ismail. On selecting a maximum volume sub-matrix of a matrix and related problems. Theoretical Computer Science, 410(47-49):4801–4811, 2009.

[16] Mark Crovella and Eric Kolaczyk. Graph wavelets for spatial traffic analysis. In INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, volume 3, pages 1848–1857. IEEE, 2003.

[17] Abhimanyu Das and David Kempe. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. arXiv preprint arXiv:1102.3975, 2011.

[18] Michaël Defferrard, Lionel Martin, Rodrigo Pena, and Nathanaël Perraudin. Pygsp: Graph signal processing in python.

[19] Amit Deshpande and Luis Rademacher. Efficient volume sampling for row/column subset selection. In 2010 IEEE 51st Annual Symposium on Foundations of Computer Science, pages 329–338. IEEE, 2010.

[20] Storm Dunlop. A dictionary of weather. OUP Oxford, 2008.

[21] Eugene d'Eon, Bob Harrison, Taos Myers, and Philip A Chou. 8i voxelized full bodies-a voxelized point cloud dataset. ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006, 7(8):11, 2017.

[22] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. Publ. Math. Inst. Hung. Acad. Sci, 5(1):17–60, 1960.

[23] Santo Fortunato. Community detection in graphs. Physics reports, 486(3):75–174, 2010.

[24] Akshay Gadde, Aamir Anis, and Antonio 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, pages 492–501, 2014.

[25] Akshay Gadde and Antonio Ortega. A probabilistic interpretation of sampling theory of graph signals. In 2015 IEEE international conference on Acoustics, Speech and Signal Processing (ICASSP), pages 3257–3261. IEEE, 2015.

[26] Fernando Gama, Antonio G Marques, Geert Leus, and Alejandro Ribeiro. Convolutional neural network architectures for signals supported on graphs. IEEE Transactions on Signal Processing, 67(4):1034–1049, 2019.

[27] Benjamin Girault, Shrikanth Narayanan, Paulo Gonçalves, Antonio Ortega, and Eric Fleury. Grasp: A matlab toolbox for graph signal processing. In 42nd IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2017), 2017.

[28] Gene H Golub and Charles F Van Loan. Matrix computations, volume 3. JHU Press, 2012.

[29] Sergei A Goreinov, Ivan V Oseledets, Dimitry V Savostyanov, Eugene E Tyrtyshnikov, and Nikolay L Zamarashkin. How to find a good submatrix. In Matrix Methods: Theory, Algorithms And Applications: Dedicated to the Memory of Gene Golub, pages 247–256. World Scientific, 2010.

[30] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. The elements of statistical learning: data mining, inference, and prediction, volume 2. Springer, 2009.

[31] Roger A Horn and Charles R Johnson. Matrix analysis. Cambridge University Press, 2012.

[32] Xinyu Huang, Xinjing Cheng, Qichuan Geng, Binbin Cao, Dingfu Zhou, Peng Wang, Yuanqing Lin, and Ruigang Yang. The apolloscape dataset for autonomous driving. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pages 954–960, 2018.

[33] Ajinkya Jayawant and Antonio Ortega. A distance-based formulation for sampling signals on graphs. In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 6318–6322. IEEE, 2018.

[34] Ajinkya Jayawant and Antonio Ortega. Practical graph signal sampling with log-linear size scaling. Signal Processing, 194:108436, 2022.

[35] Siddharth Joshi and Stephen Boyd. Sensor selection via convex optimization. IEEE Transactions on Signal Processing, 57(2):451–462, 2008.

[36] Alexander Jung and Nguyen Tran. Localized linear regression in networked data. IEEE Signal Processing Letters, 26(7):1090–1094, 2019.

[37] Andrew V Knyazev. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. SIAM journal on scientific computing, 23(2):517–541, 2001.

[38] Andreas Krause and Daniel Golovin. Submodular function maximization. Cambridge University Press, 2014.

[39] Andreas Krause, H Brendan McMahan, Carlos Guestrin, and Anupam Gupta. Robust submodular observation selection. Journal of Machine Learning Research, 9(Dec):2761–2801, 2008.

[40] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, Dandapani Sivakumar, Andrew Tompkins, and Eli Upfal. The web as a graph. In Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pages 1–10. ACM, 2000.

[41] Yun-Jou Lin, Ronald R Benziger, and Ayman Habib. Planar-based adaptive downsampling of point clouds. Photogrammetric Engineering & Remote Sensing, 82(12):955–966, 2016.

[42] Zhi Liu, Qiyue Li, Xianfu Chen, Celimuge Wu, Susumu Ishihara, Jie Li, and Yusheng Ji. Point cloud video streaming: Challenges and solutions. IEEE Network, 35(5):202–209, 2021.

[43] Charles Loop, Qin Cai, S Orts Escolano, and Philip A Chou. Microsoft voxelized upper bodies-a voxelized point cloud dataset. ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document m38673 M, 72012:2016, 2016.

[44] Javier Maroto and Antonio Ortega. Efficient worker assignment in crowdsourced data labeling using graph signal processing. In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 2271–2275. IEEE, 2018.

[45] Thomas Minka. Inferring a gaussian distribution. Media Lab Note, 1998.

[46] Sunil K Narang, Akshay Gadde, Eduard Sanou, and Antonio Ortega. Localized iterative methods for interpolation in graph structured data. In 2013 IEEE Global Conference on Signal and Information Processing, pages 491–494. IEEE, 2013.

[47] James B Orlin, Andreas S Schulz, and Rajan Udwani. Robust monotone submodular function maximization. In International Conference on Integer Programming and Combinatorial Optimization, pages 312–324. Springer, 2016.

[48] Antonio Ortega. Introduction to Graph Signal Processing. Cambridge University Press, 2022.

[49] Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst. Graph signal processing: Overview, challenges, and applications. Proceedings of the IEEE, 106(5):808–828, 2018.

[50] Alejandro Parada-Mayorga. Blue noise and optimal sampling on graphs. PhD thesis, University of Delaware, 2019.

[51] Alejandro Parada-Mayorga, Daniel L Lau, Jhony H Giraldo, and Gonzalo R Arce. Blue-noise sampling on graphs. IEEE Transactions on Signal and Information Processing over Networks, 5(3):554–569, 2019.

[52] Bo Peng. The determinant: A means to calculate volume. Recall, 21:a22, 2007.

[53] Nathanaël Perraudin, Johan Paratte, David Shuman, Lionel Martin, Vassilis Kalofolias, Pierre Vandergheynst, and David K. Hammond. Gspbox: A toolbox for signal processing on graphs, 2016.

[54] Isaac Pesenson. Sampling in paley-wiener spaces on combinatorial graphs. Transactions of the American Mathematical Society, 360(10):5603–5627, 2008.

[55] William H Press, William T Vetterling, Saul A Teukolsky, and Brian P Flannery. Numerical Recipes Example Book (FORTRAN). Cambridge University Press Cambridge, 1992.

[56] Friedrich Pukelsheim. Optimal design of experiments. SIAM, 2006.

[57] Gilles Puy, Nicolas Tremblay, Rémi Gribonval, and Pierre Vandergheynst. Random sampling of bandlimited signals on graphs. Applied and Computational Harmonic Analysis, 2016.

[58] Akie Sakiyama, Yuichi Tanaka, Toshihisa Tanaka, and Antonio Ortega. Eigendecomposition-free sampling set selection for graph signals. arXiv preprint arXiv:1809.01827, 2018.

[59] Akie Sakiyama, Yuichi Tanaka, Toshihisa Tanaka, and Antonio Ortega. Eigendecomposition-free sampling set selection for graph signals. IEEE Transactions on Signal Processing, 67(10):2679–2692, 2019.

[60] Jun Shao. Linear model selection by cross-validation. Journal of the American statistical Association, 88(422):486–494, 1993.

[61] Michael C Shewry and Henry P Wynn. Maximum entropy sampling. Journal of applied statistics, 14(2):165–170, 1987.

[62] Han Shomorony and A Salman Avestimehr. Sampling large data on graphs. In Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conference on, pages 933–936. IEEE, 2014.

[63] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. IEEE Signal Processing Magazine, 30(3):83–98, 2013.

[64] Shashank N Sridhara, Eduardo Pavez, Antonio Ortega, Ryosuke Watanabe, and Keisuke Nonaka. Point cloud attribute compression via chroma subsampling. In ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 2579–2583. IEEE, 2022.

[65] Jason M Stoker, John C Brock, Christopher E Soulard, Kernell G Ries, Larry Sugarbaker, Wesley E Newton, Patricia K Haggerty, Kathy E Lee, and John A Young. USGS lidar science strategy: mapping the technology to the science, volume 10. US Department of the Interior, US Geological Survey, 2015.

[66] Czesław Suchocki and Wioleta Błaszczak-Bąk. Down-sampling of point clouds for the technical diagnostics of buildings and structures. Geosciences, 9(2):70, 2019.

[67] Yuichi Tanaka, Yonina C Eldar, Antonio Ortega, and Gene Cheung. Sampling signals on graphs: From theory to applications. IEEE Signal Processing Magazine, 37(6):14–30, 2020.

[68] Nicolas Tremblay, Pierre-Olivier Amblard, and Simon Barthelme. Graph sampling with determinantal processes. arXiv preprint arXiv:1703.01594, 2017.

[69] Mikhail Tsitsvero, Sergio Barbarossa, and Paolo Di Lorenzo. Signals on graphs: Uncertainty principle and sampling. IEEE Transactions on Signal Processing, 64(18):4845–4860, 2016.

[70] Russell S. Vose, Scott Applequist, Mike Squires, Imke Durre, Matthew J. Menne, Claude N. Williams, Chris Fenimore, Karin Gleason, and Derek Arndt. Improved historical temperature and precipitation time series for u.s. climate divisions. Journal of Applied Meteorology and Climatology, 53(5):1232 – 1251, 2014.

[71] Fen Wang, Yongchao Wang, and Gene Cheung. A-optimal sampling and robust reconstruction for graph signals via truncated neumann series. IEEE Signal Processing Letters, 25(5):680–684, 2018.

[72] Xiaohan Wang, Pengfei Liu, and Yuantao Gu. Local-set-based graph signal reconstruction. IEEE transactions on signal processing, 63(9):2432–2444, 2015.

[73] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In Proceedings of the 20th International conference on Machine learning (ICML-03), pages 912–919, 2003.

Appendices

A Proof of eigenvector convergence

Lemma 6.1. *There exists a signal $\boldsymbol{\phi}$ in the orthogonal subspace to $\boldsymbol{\psi}^*$ with $\boldsymbol{\phi}(\mathcal{S}_m) = \mathbf{0}, \|\boldsymbol{\phi}\| = 1$ whose out of bandwidth energy is a minimum value $c_0 \neq 0$.*

Solution. The set of signals $\{\boldsymbol{\phi} : \boldsymbol{\phi}(\mathcal{S}_m) = \mathbf{0}, \|\boldsymbol{\phi}\| = 1\}$ is a closed set. Let $\begin{pmatrix} x_1 & \cdots & x_n \end{pmatrix}^T$ be in the set for any ϵ. Then $\begin{pmatrix} x_1 + \epsilon/2 & x_2 & \cdots & x_n \end{pmatrix}^T$ is in the ϵ neighborhood. Distance exists because it is a normed vector space. That vector does not have $\|\| = 1$ so it is not in the set. So for every ϵ-neighborhood $\exists$ a point not in the set. So every point is a limit point and the set is a closed set.

Out of bandwidth energy is a continuous function on our set. Let $\mathbf{v}_1, \mathbf{v}_2$ be such that $\mathbf{v}_1, \mathbf{v}_2 \perp \boldsymbol{\psi}^*$ and $\mathbf{v}_1(\mathcal{S}_m) = \mathbf{0}, \mathbf{v}_2(\mathcal{S}_m) = \mathbf{0}$. Let us suppose that the Fourier coefficients for $\mathbf{v}_1$ and $\mathbf{v}_2$ are $(\alpha_1, \cdots, \alpha_n)^T$ and $(\beta_1, \cdots, \beta_n)^T$. Then we want

$$\sum_{i=m+2}^{n} (\alpha_i^2 - \beta_i^2) < \epsilon \tag{A.1}$$

for some δ where $\|\mathbf{v}_1 - \mathbf{v}_2\| < \delta$. We can show that (A.1) holds when $\delta = \epsilon/2$.

Since the set is closed and the function is continuous on the set, the function attains a minimum value. Minimum value cannot be zero because there is a unique signal $\boldsymbol{\psi}^*$ with that property, and we are looking in a space orthogonal to $\boldsymbol{\psi}^*$. So there is a signal with minimum out of bandwidth energy of c_0 where $c_0 > 0$. □

Next, this appendix shows the proof for the l_2 convergence from Theorem 2.1.

Solution. Let us look at a particular step where we have already selected $\mathcal{S}_m$ vertices. The solutions to the following optimization problems are equivalent.

$$\boldsymbol{\psi}_k^* = \arg\min_{\boldsymbol{\psi}} \frac{\boldsymbol{\psi}^T \mathbf{L}^k \boldsymbol{\psi}}{\boldsymbol{\psi}^T \boldsymbol{\psi}} = \arg\min_{\boldsymbol{\psi}, \|\boldsymbol{\psi}\|=1} \boldsymbol{\psi}^T \mathbf{L}^k \boldsymbol{\psi}.$$

Therefore, we will consider solutions with $\|\boldsymbol{\psi}\| = 1$.

Let us consider the space of our signals. $\boldsymbol{\phi}(\mathcal{S}_m) = \mathbf{0}, \boldsymbol{\phi} \in \mathcal{R}^n$ is a vector space. Dimension of this vector space is $n - m$.

For any k, let us represent our solution for k as $\boldsymbol{\psi} = \alpha_1 \boldsymbol{\psi}^* + \alpha_2 \boldsymbol{\psi}^\perp$. Here $\boldsymbol{\psi}^\perp$ is a vector in the orthogonal subspace to our vector $\boldsymbol{\psi}^*$. We can do this because we have a vector space and it has finite dimensions. One condition on our signal is that $\alpha_1^2 + \alpha_2^2 = 1$, $\|\boldsymbol{\psi}^*\| = 1$, $\left\|\boldsymbol{\psi}^\perp\right\| = 1$. Furthermore, we know the Fourier transform of our two signal components.

$$\begin{aligned}
\boldsymbol{\psi}^* \xrightarrow{\mathcal{F}} \mathbf{U}^T \boldsymbol{\psi}^* &= \begin{bmatrix} \gamma_1 & \cdots & \gamma_{m+1} & 0 & \cdots & 0 \end{bmatrix}^T = \boldsymbol{\gamma}, \\
\boldsymbol{\psi}^\perp \xrightarrow{\mathcal{F}} \mathbf{U}^T \boldsymbol{\psi}^\perp &= \begin{bmatrix} \beta_1 & \vdots & \beta_n \end{bmatrix}^T = \boldsymbol{\beta}.
\end{aligned}$$

Our signal can be written as

$$\begin{bmatrix} \boldsymbol{\psi}^* & \boldsymbol{\psi}^\perp \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\psi}^* & \boldsymbol{\psi}^\perp \end{bmatrix} \boldsymbol{\alpha}.$$

Our objective function becomes the following:

$$\begin{aligned}
\boldsymbol{\alpha}^T \begin{bmatrix} \boldsymbol{\psi}^{*T} \\ \boldsymbol{\psi}^{\perp T} \end{bmatrix} \mathbf{L}^k \begin{bmatrix} \boldsymbol{\psi}^* & \boldsymbol{\psi}^\perp \end{bmatrix} \boldsymbol{\alpha} &= \boldsymbol{\alpha}^T \begin{bmatrix} \boldsymbol{\gamma}^T \\ \boldsymbol{\beta}^T \end{bmatrix} \boldsymbol{\Sigma}^k \begin{bmatrix} \boldsymbol{\gamma} & \boldsymbol{\beta} \end{bmatrix} \boldsymbol{\alpha} \\
&= \boldsymbol{\alpha}^T \begin{bmatrix} \sum_{i=1}^{m+1} \gamma_i^2 \sigma_i^k & \sum_{i=1}^{m+1} \gamma_i \beta_i \sigma_i^k \\ \sum_{i=1}^{m+1} \gamma_i \beta_i \sigma_i^k & \sum_{i=i}^{n} \beta_i^2 \sigma_i^k \end{bmatrix} \boldsymbol{\alpha} = \boldsymbol{\alpha}^T \begin{bmatrix} a & b \\ b & d \end{bmatrix} \boldsymbol{\alpha}.
\end{aligned}$$

In the last equation a, b, c, d are just convenient notations for the scalar values in the 2×2 matrix. Note that $a, d > 0$ because $\sigma_i > 0$ and the expression is then a sum of positive quantities. Note that we want to minimize the objective function subject to the constraint $\|\alpha\| = 1$. We solve this optimization problem by a standard application of the KKT conditions [11].

The Lagrangian function corresponding to our constrained minimization problem is as follows:

$$L(\boldsymbol{\alpha}, \lambda) = \boldsymbol{\alpha}^T \begin{bmatrix} a & b \\ b & d \end{bmatrix} \boldsymbol{\alpha} + \lambda(\boldsymbol{\alpha}^T \boldsymbol{\alpha} - 1).$$

The solution which minimizes this objective function is the eigenvector of the matrix $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ with the minimum eigenvalue. To prove that we take the gradient of the equation with respect to $\boldsymbol{\alpha}$ and put it to $\mathbf{0}$.

This gives us two first order equations.

$$a\alpha_1 + b\alpha_2 + \lambda\alpha_1 = 0, \qquad b\alpha_1 + d\alpha_2 + \lambda\alpha_2 = 0.$$

$\alpha_1 = 0$ implies $\alpha_2 = 0$ unless $b = 0$ and vice versa. Both α_1 and α_2 cannot be zero at the same time otherwise our solution does not lie in our domain of unit length vectors. However either of α_1 or α_2 can be 0 only if $b = 0$. If $b = 0$, for large k the solution is given by $\alpha_2 = 0, \alpha_1 = 1$ because we show next that a/d can be made less than $1/2$ for $k > k_0$. We now analyze the case where $b \neq 0$ and so $a + \lambda \neq 0$ and $d + \lambda \neq 0$. Writing α_2 in terms of α_1 for both the equations we get:

$$\alpha_2 = \frac{-(a+\lambda)}{b}\alpha_1, \quad \alpha_2 = \frac{-b}{d+\lambda}\alpha_1.$$

Equating both the expressions for α_2 (and assuming $\alpha_1 \neq 0$)gives us a quadratic with two solutions. Since $a, d > 0$ the positive sign gives us the λ with lower magnitude.

$$\lambda = \frac{-(a+d) + \sqrt{(a-d)^2 + 4b^2}}{2}.$$

We now use the condition that the solution has norm one. Solution of this gives us a value for α_2.

$$\alpha_2 = \mp \frac{a - d + \sqrt{(a-d)^2 + 4b^2}}{\sqrt{\left(a - d + \sqrt{(a-d)^2 + 4b^2}\right)^2 + 4b^2}}.$$

We look at the absolute value of the α_2.

$$|\alpha_2| = \frac{2|b|}{\sqrt{(-(a-d) + \sqrt{(a-d)^2 + 4b^2})^2 + (2b)^2}}$$

Let us find a k_0 such that $|b|/d < \epsilon/2 (\epsilon > 0)$ for all $k > k_0$. This will also make $a/d < 1/2$. This will help us make the entire expression less than ϵ for $k > k_0$ (A.3). We upper bound b in the following way.

$$\begin{aligned} |b| &= |\sum_{i=1}^{m+1} \beta_i \gamma_i \sigma^k| \\ &\leq \sqrt{\sum_{i=1}^{m+1} \beta_i^2 \gamma_i^2} \sqrt{\sum_{i=1}^{m+1} \sigma_i^{2k}} \\ &\leq 1.\sqrt{m+1}\sigma_{m+1}^k = b_1. \end{aligned}$$

Now we know that the least possible value of d is $c_0 \sigma_{m+2}^k$. So when $k > k_0$ we get the following upper bound for $|b|/d$ in terms of ϵ:

$$\frac{|b|}{d} \leq \frac{\sqrt{m+1}\sigma_{m+1}^k}{c_0 \sigma_{m+2}^k}.$$

We want this to be less than $\epsilon/2$, which gives us our condition on k.

$$\frac{\sqrt{m+1}\sigma_{m+1}^k}{c_0\sigma_{m+2}^k} < \frac{\epsilon}{2}$$
$$k > \left\lceil \frac{\log(m+1)/2 + \log 1/\epsilon + \log(2/c_0)}{\log \frac{\sigma_{m+2}}{\sigma_{m+1}}} \right\rceil. \tag{A.2}$$

a/d also admits a similar analysis.

$$\frac{a}{d} \leq \frac{\sigma_{m+1}^k}{c_0\sigma_{m+2}^k}$$
$$k > \left\lceil \frac{\log(2/c_0)}{\log \frac{\sigma_{m+2}}{\sigma_{m+1}}} \right\rceil. \tag{A.3}$$

Since this value of k is equal or lesser than the value of k required for $|b|/d < \epsilon/2$, for our theorem we will take the value (A.2). When d divides both the numerator and denominator of the equation it gives us the expressions we need in terms of a/d and $|b|/d$.

$$\begin{aligned}
|\alpha_2| &= \frac{|2b/d|}{\sqrt{\left(1 - a/d + \sqrt{(a/d-1)^2 + 4(b/d)^2}\right)^2 + (2b/d)^2}} \\
&< \frac{\epsilon}{|1 - a/d + \sqrt{(1-a/d)^2 + \epsilon^2}|} \\
&< \frac{\epsilon}{|2(1-a/d)|} < \epsilon.
\end{aligned}$$

This implies that as k increases the coefficient of out-of-bandwidth component goes to zero. Because the out-of bandwidth signal has finite energy, the signal energy goes to zero as $\alpha_2 \to 0$. Whether $\boldsymbol{\psi}_k^*$ converges to $\boldsymbol{\psi}^*$ or $-\boldsymbol{\psi}^*$ is a matter of convention. Hence as $k \to \infty$, $\boldsymbol{\psi}_k^* \to \boldsymbol{\psi}^*$. □

B Justification for ignoring target bandwidth while sampling

We know that selecting the right $\mathbf{U}_{\mathcal{SF}}$ matrix is essential to prevent a blow-up of the error while reconstructing using (1.1). In practice for reconstruction the bandwidth is $f \leq s$. However, for our AVM sampling algorithm we chose the bandwidth to be $|\mathcal{R}| = s$ instead. We next address why that is a logical choice with respect to D-optimality.

The matrix $\mathbf{U}_{\mathcal{SS}}^T\mathbf{U}_{\mathcal{SS}}$ is positive definite following from our initial condition of the set $\mathcal{S}$ being a uniqueness set. This provides us with the needed relations between determinants. We can see that $\mathbf{U}_{\mathcal{SF}}^T\mathbf{U}_{\mathcal{SF}}$ is a submatrix of $\mathbf{U}_{\mathcal{SS}}^T\mathbf{U}_{\mathcal{SS}}$.

$$\mathbf{U}_{\mathcal{SS}}^T\mathbf{U}_{\mathcal{SS}} = \begin{bmatrix} \mathbf{U}_{\mathcal{SF}}^T\mathbf{U}_{\mathcal{SF}} & \mathbf{U}_{\mathcal{SF}}^T\mathbf{U}_{\mathcal{S},f+1:s} \\ \mathbf{U}_{\mathcal{S},f+1:s}^T\mathbf{U}_{\mathcal{SF}} & \mathbf{U}_{\mathcal{S},f+1:s}^T\mathbf{U}_{\mathcal{S},f+1:s} \end{bmatrix}.$$

This helps us to relate the matrix and its submatrix determinants using Fischer's inequality from Theorem 7.8.5 in [31].

$$\det(\mathbf{U}_{\mathcal{SS}}^T\mathbf{U}_{\mathcal{SS}}) \leq \det(\mathbf{U}_{\mathcal{SF}}^T\mathbf{U}_{\mathcal{SF}}) \det(\mathbf{U}_{\mathcal{S},f+1:s}^T\mathbf{U}_{\mathcal{S},f+1:s}) \tag{B.1}$$

The determinant of the matrix $\mathbf{U}_{\mathcal{S},f+1:s}^T\mathbf{U}_{\mathcal{S},f+1:s}$ can be bounded above. The eigenvalues of $\mathbf{U}_{\mathcal{S},f+1:s}^T\mathbf{U}_{\mathcal{S},f+1:s}$ are the same as the non-zero eigenvalues of $\mathbf{U}_{\mathcal{S},f+1:s}\mathbf{U}_{\mathcal{S},f+1:s}^T$ by Theorem 1.2.22 in [31]. Using eigenvalue interlacing Theorem 8.1.7 from [28], the eigenvalues of the matrix $\mathbf{U}_{\mathcal{S},f+1:s}\mathbf{U}_{\mathcal{S},f+1:s}^T$ are less than or equal to 1 because it is submatrix of $\mathbf{U}_{\mathcal{V},f+1:s}\mathbf{U}_{\mathcal{V},f+1:s}^T$ whose non-zero eigenvalues are all 1. As the determinant of a matrix is the product of its eigenvalues, the following bound applies:

$$\det(\mathbf{U}_{\mathcal{S},f+1:s}^T\mathbf{U}_{\mathcal{S},f+1:s}) \leq 1. \tag{B.2}$$

Using (B.1) and (B.2) and positive definiteness of the matrices, we now have a simple lower bound for our criteria under consideration:

$$|\det(\mathbf{U}_{\mathcal{SS}}^T\mathbf{U}_{\mathcal{SS}})| \leq |\det(\mathbf{U}_{\mathcal{SF}}^T\mathbf{U}_{\mathcal{SF}})|. \tag{B.3}$$

Thus, for example it is impossible for $|\det(\mathbf{U}_{\mathcal{SS}}^T\mathbf{U}_{\mathcal{SS}})|$ to be equal to some positive value while $|\det(\mathbf{U}_{\mathcal{SF}}^T\mathbf{U}_{\mathcal{SF}})|$ being half of that positive value.

To summarize, instead of aiming to maximize $|\det(\mathbf{U}_{\mathcal{SF}}^T\mathbf{U}_{\mathcal{SF}})|$, we aimed to maximize $|\det(\mathbf{U}_{\mathcal{SS}}^T\mathbf{U}_{\mathcal{SS}})|$. This intuitively worked because optimizing for a D-optimal matrix indirectly ensured a controlled performance of the subset of that matrix. In this way due to the relation (B.3), we avoided knowing the precise bandwidth f and still managed to sample using the AVM algorithm.

C Approximating Gram matrix by a diagonal matrix

Here we try to estimate how close our approximation of $\mathbf{D}_m^T\mathbf{D}_m$ to a diagonal matrix is. Towards this goal we define a simple metric for a general matrix $\mathbf{A}$.

$$\text{Fraction of energy in diagonal} = \frac{\sum_i \mathbf{A}_{ii}^2}{\sum_i \sum_j \mathbf{A}_{ij}^2}. \tag{C.1}$$

Since this can be a property dependent on the graph topology, we take 5 different types of graphs with 1000 vertices — Scale-free, WRS sensor nearest neighbors, Erdős Rényi, Grid, Line. Using AVM we select a varying number of samples ranging from 1 to 50. With the bandwidth f taken to be 50, we average the fraction of the energy (C.1) over 10 instances of each graph and represent it in Fig. 6.2.

We observe more than 0.75 fraction of energy in the diagonal of the matrix $\mathbf{D}_m^T\mathbf{D}_m$, which justifies this approximation. According to our experiments, which are not presented here, the inverse of the matrix $(\mathbf{D}_m^T\mathbf{D}_m)^{-1}$ is not as close to a diagonal matrix as $\mathbf{D}_m^T\mathbf{D}_m$ is to a diagonal

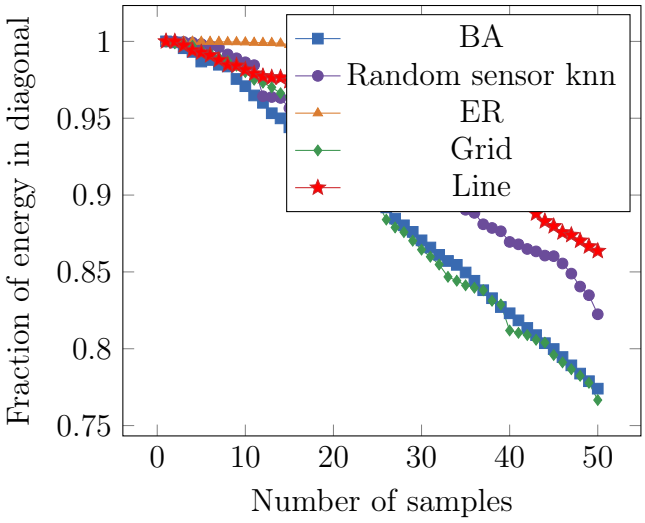


Figure 6.2: Closeness to diagonal at each iteration.

matrix. Nevertheless, in place of $(\mathbf{D}_m^T\mathbf{D}_m)^{-1}$ we still use diag $\left(1/\|\mathbf{d}_1\|^2, \cdots, 1/\|\mathbf{d}_m\|^2\right)$ for what it is, an approximation.

Note however that the approximation does not hold in general for any samples. It holds when the samples are selected in a determinant maximizing conscious way by Algorithm 2. This approximation is suited to AVM because of its choice of sampling bandwidth, $\mathcal{R}$. As the number of samples requested increases, so does the sampling bandwidth. The higher bandwidth causes the filtered delta signals to become more localized causing energy concentration in the diagonal and keeping the diagonal approximation reasonable and applicable.

D D-optimal sampling for generic kernels

Another graph signal model is a probabilistic distribution instead of a bandlimited model [25], [73]. In such cases, the covariance matrix is our kernel. The subset selection problem is defined as a submatrix selection of the covariance matrix. Framing the problem as entropy maximization naturally leads to a determinant maximization approach [61].

To define our problem more formally, let us restrict space of all possible kernels to the space of kernels which can be defined as $\mathbf{K} = g(\mathbf{L})$ with g defined on matrices but induced

from a function from non-negative reals to positive reals $g : \mathbb{R}_{\geq 0} \to \mathbb{R}_{>0}$. An example of such a function of $\mathbf{L}$ would be $(L + \delta I)^{-1}$. Such a function can be written as a function on the eigenvalues of the Laplacian $\mathbf{K} = \mathbf{U}g(\mathbf{\Sigma})\mathbf{U}^T$. Motivated by entropy maximization in the case of probabilistic graph signal model, suppose we wish to select a set $\mathcal{S}$ so that we maximize the determinant magnitude $|\det(\mathbf{K}_{\mathcal{S}\mathcal{S}})|$.

There are a few differences for solving the new problem, although most of Algorithm 2 translates well. We now wish to maximize $|\det(\mathbf{U}_{\mathcal{S}}g(\mathbf{\Sigma})\mathbf{U}_{\mathcal{S}}^T)|$. The expression for the determinant update remains the same as before.

$$\det\left(\begin{bmatrix}\mathbf{D}_m^T\mathbf{D}_m & \mathbf{D}_m^T\mathbf{d}_v \\ \mathbf{d}_v^T\mathbf{D}_m & \mathbf{d}_v^T\mathbf{d}_v\end{bmatrix}\right) \approx \det(\mathbf{D}_m^T\mathbf{D}_m)\det(\mathbf{d}_v^T\mathbf{d}_v - \mathbf{d}_v^T\mathbf{D}_m(\mathbf{D}_m^T\mathbf{D}_m)^{-1}\mathbf{D}_m^T\mathbf{d}_v)$$

Only now we have to maximize the volume of the parallelelpiped formed by the vectors $\mathbf{d}_v = \mathbf{U}g^{1/2}(\mathbf{\Sigma})\mathbf{U}^T\boldsymbol{\delta}_v$ for $v \in \mathcal{S}$. The squared coherences with respect to our new kernel $\mathbf{d}_v^T\mathbf{d}_v$ are computed in the same way as before by random projections. The diagonal of our new kernel matrix now approximates the matrix $\mathbf{D}_m^T\mathbf{D}_m$.

$$\mathbf{D}_m^T\mathbf{D}_m \approx \operatorname{diag}((\mathbf{U}g(\mathbf{\Sigma})\mathbf{U}^T)_{11}, \cdots, (\mathbf{U}g(\mathbf{\Sigma})\mathbf{U}^T)_{nn})$$

The other difference is that the approximate update stage is given by

$$v* \leftarrow \arg\max_{v \in \mathcal{S}^c} \|\mathbf{d}_v\|^2 - \sum_{w \in \mathcal{S}} \frac{(\mathbf{U}g^{1/2}(\mathbf{\Sigma})\mathbf{U}^T\mathbf{d}_w)^2(v)}{\|\mathbf{d}_w\|^2}$$

with the difference resulting from the kernel not being a projection operator. So for a generic kernel with a determinant maximization objective, Algorithm 2 works the same way with minor modifications discussed here.