Practical graph signal sampling with log-linear size scaling

Ajinkya Jayawant* jayawant@usc.edu, Antonio Ortega† ortega@sipi.usc.edu
Ming Hsieh Department of Electrical and Computer Engineering, University of Southern Calfornia
3740 McClintock Ave, Los Angeles, CA 90089, US

Abstract

Graph signal sampling is the problem of selecting a subset of representative graph vertices whose values can be used to interpolate missing values on the remaining graph vertices. Optimizing the choice of sampling set using concepts from experiment design can help minimize the effect of noise in the input signal. While many existing sampling set selection methods are computationally intensive because they require an eigendecomposition, existing eigendecompostion-free methods are still much slower than random sampling algorithms for large graphs. In this paper, through optimizing sampling sets towards the D-optimal objective from experiment design, we propose a sampling algorithm that has complexity comparable to random sampling algorithms, while reaching accuracy similar to existing eigendecomposition-free methods for a broad range of graph types.

Keywords

Graph, signal, sampling, D-optimal, volume, coherence.

I. Introduction

Graphs are a convenient way to represent and analyze data having irregular relationships between data points [1] and can be useful in a variety of different scenarios, such as characterizing the Web [2], semi-supervised learning [3], community detection [4], or traffic analysis [5]. We call *graph signal* the data associated with the nodes of a graph. Similar to traditional signals, a smooth graph signal can be sampled by making observations on a few nodes of the graph, so that the signal at the remaining (non-observed) nodes can be estimated [6], [7], [8]. For this we need to choose a set of vertices, $\mathcal{S}$, called the sampling set, on which we observe the signal values in order to predict 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. 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. The concept of reconstructing sampled graph signals with some accuracy

*Corresponding author
†EURASIP member

usually relies on the assumption that the underlying signal is smooth. Intuitively this means that signal values for neighboring vertices aren't drastically different. This is a reasonable assumption in a variety of scenarios such as sensor networks modelling temperature distribution, graph signal representing labels in semi-supervised learning, or preferences in social networks. This makes it possible for us to reconstruct them by knowing a few signal values [9].

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 operator [1]. 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 [7]. Specifically, the problem reduces to a row/column subset selection similar to linear measurement sensor selection problem [10]. In the graph signal sampling context, several papers leverage this knowledge to propose novel algorithms — [11], [6], [12], [13], [14], [15]. We refer the reader to [8] for a recent comprehensive review of the literature on this topic.

However, to solve the graph sampling set selection problem, row/column selection needs to be applied on 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.

We can classify sampling set selection methods into two main types of approaches, based on whether they require eigendecomposition or not. Some methods compute the full eigendecomposition [11], [6], [12], or instead require a sequential eigendecomposition, where one eigenvector is computed at each step [7]. Alternatively, *eigendecomposition-free* methods do not make use of an eigendecomposition of the Laplacian matrix [16], [15], [13], [17] and are usually faster. Weighted Random Sampling (WRS) [16] 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 eigendecompostion-free methods. Among eigendecomposition-free methods discussed in [8], Neumann series based sampling [15] has a higher computational complexity, Binary Search with Gershgorin Disc Alignment (BS-GDA) [18] has low computational complexity for smaller graphs, but cannot compete with WRS for large graphs, and Localization operator based Sampling Set Selection (LSSS) [17] achieves good performance but requires some parameter tuning to achieve optimal performance. Our proposed method can overcome these limitations: similar to [15], [18], [17] 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 paper. For example, the error diffusion sampling algorithm (Algorithm 5 from [19]) achieves

[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). For simplicity, we describe these as full decomposition methods, even though in practice only a subset of eigenvectors is needed.

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 [19] 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 [20] do not require vertex numbering, they involve distance computations on the graph similar to DC in [21]. 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 [22] and [23] are designed for sampling clustered piecewise constant graph signals. However, in this paper, we focus on a bandlimited smoothness model for graph signals, with graph topologies not limited to clustered graphs.

To motivate our methods consider first WRS, where vertices are sampled with a probability proportional to their *squared local coherence* [16]. 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 [17] improve performance by selecting vertices based on importance but avoid the redundancy by minimizing a notion of overlapped area between functions centered on the sampled vertices.

In our preliminary work [21], 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 has a negative impact on computation cost, since distance computation is expensive. As an alternative, in this paper we propose a novel Approximate Volume Maximization (AVM) algorithm that replaces the distance computation with a filtering operation. Loosely speaking, our proposed scheme in AVM precomputes squared coherences, as [16], with an additional criterion to maintain separation between selected vertices using a filtering operation. The resulting complexity (see Section III-D) 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 algorithm in literature that uses spectral information, second only to unweighted random sampling from [16]. AVM can also be viewed as an efficient approximation to the D-optimality criterion [24]. In this paper we review the main concepts in DC and introduce AVM, showing that these methods can improve upon existing algorithms in various ways. Our main contributions are:

1) We describe our distance-based sampling DC algorithm (Section III) 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 few parameters to tune. Moreover, the accuracy of the reconstruction is a monotonic function of those parameters. This eliminates the need to search for the right parameter, as we only evolve a parameter unidirectionally for a better reconstruction.
3) Using the framework of volume based sampling (Section III), we interpret a series of algorithms — exact greedy [12], WRS, Spectral Proxies (SP) [7], LSSS, DC, and our proposed AVM as variations of the volume

TABLE I

LINEAR ALGEBRA NOTATION IN THIS PAPER

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 $\mathbf{x}$

maximization problem formulation (Section IV), 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 V. The practicality of AVM is apparent for larger graph sizes (e.g., of the order of a hundred thousand nodes)—: with the limits placed by the system used in our experiments(see Section V-A4), other state-of-the-art algorithms such as SP, LSSS and BS-GDA often fail at these graph sizes, 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 VI. We explain these advantages in terms of complexity towards the end of Section III 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 any prior knowledge of the signal bandwidth, and can be used for different graphs while requiring a few easy-to-tune parameters.

II. PROBLEM SETUP

A. Notation

In this paper, 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 I lists additional notations.

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 [25]. 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 $w_{ii} = 0$, where n is the number of vertices in the graph, which we also call as the graph size. The degree matrix $\mathbf{D}$ of a graph is a diagonal matrix with diagonal entries $d_{ii} = \sum_j w_{ij}$. In this paper we consider weighted undirected graphs, without self loops and with non-negative edge weights. Throughout the paper, $\mathbf{I}$ is $n \times n$ identity matrix.

The combinatorial Laplacian for the graph is given by $\mathbf{L} = \mathbf{D} - \mathbf{A}$, with its corresponding eigendecomposition defined as $\mathbf{L} = \mathbf{U}\Sigma\mathbf{U}^{\mathsf{T}}$ 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 $\lambda_1 \leq \cdots \leq \lambda_n$ representing the frequencies. The column vectors of $\mathbf{U}$ provide a frequency representation for graph signals, so that the operator $\mathbf{U}^{\mathsf{T}}$ 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 [1].

The sampling set $\mathcal{S}$ is defined as a subset of $\mathcal{V}$ where the values of the graph signal $\mathbf{f}$ are known, leading to a vector of known values $\mathbf{f}_{\mathcal{S}}$. The problem we consider here is that of finding the set $\mathcal{S}$ such that the error in interpolating $\mathbf{f}_{\mathcal{S}^c}$ from $\mathbf{f}_{\mathcal{S}}$ is minimized. Here, different error metrics are possible and the actual error depends on assumptions made about the signal. When comparing algorithms we assume they all operate with the same sampling set size: s. 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 they were chosen, $\mathcal{S} = \{1, 2, \cdots\}$.

For reconstruction, we will often work with sub-matrices of $\mathbf{U}$ corresponding to different frequencies or vertex localizations. The cardinality of the set of frequencies, $|\mathcal{F}|$, is the bandwidth of the signal, whereas the set $\mathcal{F}$ is the bandwidth support. Letting $\mathcal{F}$ be the set $\{1, \cdots, f\}$, where $f = |\mathcal{F}|$, the matrix constructed by selecting the first f columns of $\mathbf{U}$ will be denoted by $\mathbf{U}_{\mathcal{V}\mathcal{F}}$ or simply $\mathbf{U}_{\mathcal{F}}$. The matrix constructed by further selecting rows of $\mathbf{U}_{\mathcal{F}}$ indexed by $\mathcal{S}$ (corresponding to selected nodes) will be written as $\mathbf{U}_{\mathcal{S}\mathcal{F}}$.

B. Problem formulation

For sampling bandlimited signals $\mathbf{x}$, which can be written as

$$\mathbf{x} = \mathbf{U}_{\mathcal{F}}\tilde{\mathbf{x}}_{\mathcal{F}},$$

a sampling set that satisfies the following two conditions: i) the number of samples requested is larger than the bandwidth, that is $|\mathcal{S}| \geq f$, and ii) the sampling set $\mathcal{S}$ is a uniqueness set [9] corresponding to the bandwidth support $\mathcal{F}$, will allow us to recover $\mathbf{x}$ exactly. Given the observed samples, $\mathbf{x}_{\mathcal{S}}$, the reconstruction is given by the least squares solution:

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

In this paper 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[2]. While Condition ii) is difficult to

[2]We do not consider cases where signals are not bandlimited but can be sampled and reconstructed (refer to [8] and references therein). Exploring more general models for signal sampling is left for future work.

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 [7], [16], [17], [18], our sampling algorithms are not designed to return uniqueness sets satisfying Condition ii) thus providing exact recovery, and instead we assume that Condition i) is sufficient to guarantee exact recovery.

In practice signals are never exactly bandlimited and it is common to consider the signal model $\mathbf{f} = \mathbf{x} + \mathbf{n}$, where $\mathbf{x}$ is bandlimited and $\mathbf{n}$ is a noise vector. The reconstruction from the sampled signal $\mathbf{f}_{\mathcal{S}} = \mathbf{x}_{\mathcal{S}} + \mathbf{n}_{\mathcal{S}}$ is then:

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

Since (1) allows us to reconstruct $\mathbf{x}$ exactly, the error in the reconstructed signal is:

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

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

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

If we assume individual noise entries to be independent with zero mean and equal variance, the expected value, which is the error covariance matrix becomes

$$\mathbb{E}[(\hat{\mathbf{f}} - \mathbf{x})(\hat{\mathbf{f}} - \mathbf{x})^{\mathsf{T}}] = c\mathbf{U}_{\mathcal{F}}(\mathbf{U}_{\mathcal{SF}}^{\mathsf{T}}\mathbf{U}_{\mathcal{SF}})^{-1}\mathbf{U}_{\mathcal{F}}{}^{\mathsf{T}} \tag{2}$$

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 the error covariance matrix, where $M_{n,n}(\mathbb{R})$ is an $n \times n$ matrix of real numbers. Since the error covariance matrix is a function of the sampling set $\mathcal{S}$, we wish to find an $\mathcal{S}$ that maximizes a function $h(.)$ of the error covariance matrix as follows:

$$\mathcal{S} = \underset{\mathcal{S}\subset\mathcal{V},|\mathcal{S}|=s}{\arg\max}\ h\left(\mathbf{U}_{\mathcal{F}}(\mathbf{U}_{\mathcal{SF}}^{\mathsf{T}}\mathbf{U}_{\mathcal{SF}})^{-1}\mathbf{U}_{\mathcal{F}}^{\mathsf{T}}\right). \tag{3}$$

Note that the set $\mathcal{S}$ achieving optimality under general criteria in the form of (3) 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.

A particular choice $h(.)$ of interest to us is $1/\operatorname{pdet}(.)$, where $\operatorname{pdet}(.)$ is the pseudo determinant [26]. Since our error covariance matrix is singular, we used pseudo determinant instead of determinant. Pseudo determinant only differs from determinant in that it is a product of non-zero eigenvalues instead of all eigenvalues of the matrix. With our choice of $h(.)$, (3) is equivalent to the following maximization :

$$\mathcal{S} = \underset{\mathcal{S}\subset\mathcal{V},|\mathcal{S}|=s}{\arg\max}\ \det(\mathbf{U}_{\mathcal{SF}}^{\mathsf{T}}\mathbf{U}_{\mathcal{SF}}). \tag{4}$$

This is also known as the D-optimality criterion. Maximizing the determinant leads to minimizing the confidence interval of the solution $\hat{\mathbf{f}}$ [24] 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 — see Section IV-C.

Moreover, the D-optimal objective is necessary for the approximations we need in order to develop algorithms achieving eigendecomposition-free subset selection.

Sampling algorithms are designed to implicitly or explicitly optimize the sampling set for a particular bandwidth support. In this paper, we denote by $\mathcal{R}$ the bandwidth support assumed by a sampling algorithm, which can be equal to the reconstruction bandwidth support $\mathcal{F}$ for which the objective (4) can be rewritten as:

$$\mathcal{S} = \underset{\mathcal{S}\subset\mathcal{V},|\mathcal{S}|=s}{\arg\max}\ \det(\mathbf{U}_{\mathcal{SR}}^{\mathsf{T}}\mathbf{U}_{\mathcal{SR}}), \quad \text{with} \qquad \mathcal{R}=\mathcal{F}. \tag{5}$$

However, there are advantages to choosing a different $\mathcal{R}$ for optimization than $\mathcal{F}$. For example, if we consider $\mathcal{R} = \{1, \cdots, s\}$ so that $|\mathcal{R}| = |\mathcal{S}|$, we can rewrite the objective function (5) without changing its value, by permuting the order of the matrices:

$$\mathcal{S} = \underset{\mathcal{S}\subset\mathcal{V},|\mathcal{S}|=s}{\arg\max}\ \det(\mathbf{U}_{\mathcal{SR}}\mathbf{U}_{\mathcal{SR}}^{\mathsf{T}}). \tag{6}$$

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 (6) 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.

C. Solving D-optimal objectives

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 [27]. Nearly-optimal methods have been proposed in the literature [28], [29], 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 [12], [14] 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 (4) 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) of [13], 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 [13] upfront. 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 [13]. 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 (4) at every iteration as opposed to WDPP which aims to achieve 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

[27]. 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 [13], we observe that WDPP has worse average performance than WRS. In comparison, in our experiments, for a wide variety of graph topologies and sizes, AVM consistently outperforms WRS [16] in terms of average reconstruction error.

III. Efficient sampling set selection algorithms

In what follows we assume that the conditions for equivalence between the two objective function forms (5) and (6) are verified, so that we focus on solving (6).

A. 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 the low pass filtered signal for the Kronecker delta function $\boldsymbol{\delta}_v$ localized at vertex v:

$$\mathbf{d}_v = \mathbf{U}_{\mathcal{R}} \mathbf{U}_{\mathcal{R}}^{\mathsf{T}} \boldsymbol{\delta}_v. \tag{7}$$

With this definition, the objective in (6) can be written as:

$$\begin{aligned} \det(\mathbf{U}_{\mathcal{SR}} \mathbf{U}_{\mathcal{SR}}^{\mathsf{T}}) &= \det(\mathbf{U}_{\mathcal{SR}} \mathbf{U}_{\mathcal{R}}^{\mathsf{T}} \mathbf{U}_{\mathcal{R}} \mathbf{U}_{\mathcal{SR}}^{\mathsf{T}}) \\ &= \det(\mathbf{I}_{\mathcal{S}}^{\mathsf{T}} \mathbf{U}_{\mathcal{R}} \mathbf{U}_{\mathcal{R}}^{\mathsf{T}} \mathbf{U}_{\mathcal{R}} \mathbf{U}_{\mathcal{R}}^{\mathsf{T}} \mathbf{I}_{\mathcal{S}}) \\ &= \det\left(\begin{bmatrix} \mathbf{d}_1 & \cdots & \mathbf{d}_s \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} \mathbf{d}_1 & \cdots & \mathbf{d}_s \end{bmatrix} \right) \\ &= \mathrm{Vol}^2(\mathbf{d}_1, \cdots, \mathbf{d}_s). \end{aligned} \tag{8}$$

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 $\det(\mathbf{U}_{\mathcal{SR}} \mathbf{U}_{\mathcal{SR}}^{\mathsf{T}})$ is equivalent to maximizing $\mathrm{Vol}(\mathbf{d}_1, \cdots, \mathbf{d}_s)$, and as a consequence the set maximizing (8) also maximizes (6).

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 paper, 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 $\mathcal{D}_m = \mathrm{span}(\mathbf{d}_1, \cdots, \mathbf{d}_m)$. Throughout the rest of the paper, 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:

$$\det\left(\begin{bmatrix} \mathbf{D}_m & \mathbf{d}_v \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} \mathbf{D}_m & \mathbf{d}_v \end{bmatrix} \right) = \det\left(\begin{bmatrix} \mathbf{D}_m^{\mathsf{T}} \mathbf{D}_m & \mathbf{D}_m^{\mathsf{T}} \mathbf{d}_v \\ \mathbf{d}_v^{\mathsf{T}} \mathbf{D}_m & \mathbf{d}_v^{\mathsf{T}} \mathbf{d}_v \end{bmatrix} \right) \tag{9a}$$

$$= \det(\mathbf{D}_m^{\mathsf{T}} \mathbf{D}_m) \det(\mathbf{d}_v^{\mathsf{T}} \mathbf{d}_v - \mathbf{d}_v^{\mathsf{T}} \mathbf{D}_m (\mathbf{D}_m^{\mathsf{T}} \mathbf{D}_m)^{-1} \mathbf{D}_m^{\mathsf{T}} \mathbf{d}_v) \tag{9b}$$

$$= \det(\mathbf{D}_m^{\mathsf{T}} \mathbf{D}_m)(\|\mathbf{d}_v\|^2 - \mathbf{d}_v^{\mathsf{T}} \mathbf{D}_m (\mathbf{D}_m^{\mathsf{T}} \mathbf{D}_m)^{-1} \mathbf{D}_m^{\mathsf{T}} \mathbf{d}_v) \tag{9c}$$

$$= \det(\mathbf{D}_m^{\mathsf{T}} \mathbf{D}_m) \left(\|\mathbf{d}_v\|^2 - \|\mathbf{P}_{\mathcal{D}_m} \mathbf{d}_v\|^2 \right). \tag{9d}$$

The effect on the determinant of adding a column to $\mathbf{D}_m$ can be represented according to a multiplicative update (Section 11.2 [24]) in our D-optimal design. (9b) follows from [30] (Section 0.8.5 in Second Edition), while (9d) follows because $\mathbf{P}_{\mathcal{D}_m} = \mathbf{D}_m(\mathbf{D}_m^\mathsf{T}\mathbf{D}_m)^{-1}\mathbf{D}_m^\mathsf{T}$ is a projection onto the space $\mathcal{D}_m$. Direct greedy determinant maximization requires selecting a vertex that maximizes the update term in (9c):

$$v^* = \underset{v\in\mathcal{S}_m^c}{\arg\max} \left(\|\mathbf{d}_v\|^2 - \mathbf{d}_v^\mathsf{T}\mathbf{D}_m(\mathbf{D}_m^\mathsf{T}\mathbf{D}_m)^{-1}\mathbf{D}_m^\mathsf{T}\mathbf{d}_v \right) \tag{10}$$

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

The first step towards a greedy incremental vertex selection is estimating the two components, $\|\mathbf{d}_v\|^2$ and $\mathbf{d}_v^\mathsf{T}\mathbf{D}_m(\mathbf{D}_m^\mathsf{T}\mathbf{D}_m)^{-1}\mathbf{D}_m^\mathsf{T}\mathbf{d}_v$, of the multiplicative update. The first term $\|\mathbf{d}_v\|^2$ is the *squared coherence* introduced in [16], which is estimated here using the same techniques as in [16], and is defined as

$$\|\mathbf{d}_v\|^2 = \left\|\mathbf{U}_\mathcal{R}\mathbf{U}_\mathcal{R}^\mathsf{T}\boldsymbol{\delta}_v\right\|^2 = \left\|\mathbf{U}_\mathcal{R}^\mathsf{T}\boldsymbol{\delta}_v\right\|^2. \tag{11}$$

For the second term, the projection interpretation of (9d) 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 1. *The space of bandlimited signals* $\operatorname{span}(\mathbf{U}_\mathcal{R})$ *equipped with the dot product is a reproducing kernel Hilbert space (RKHS).*

Proof. Defining the inner product for signals $\mathbf{f}, \mathbf{g} \in \operatorname{span}(\mathbf{U}_\mathcal{R})$ as $\langle \mathbf{f}, \mathbf{g}\rangle = \sum_i f_i g_i$, $\operatorname{span}(\mathbf{U}_\mathcal{R})$ 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 $\phi : \mathbb{R}^n \to \operatorname{span}(\mathbf{U}_\mathcal{R})$ given as:

$$\phi(\mathbf{x}) = \mathbf{U}_\mathcal{R}\mathbf{U}_\mathcal{R}^\mathsf{T}\mathbf{x}.$$

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:

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

Now using Theorem 4 from [31], K is a reproducing kernel for our Hilbert space and using Theorem 1 from [31] we conclude that our bandlimited space of signals is an RKHS. □

Corollary 1. *The dot product of a bandlimited signal* $\mathbf{f} \in \operatorname{span}(\mathbf{U}_\mathcal{R})$ *with a filtered delta* $\mathbf{d}_v$ *is* $\mathbf{f}(v)$*, the entry at node* v *of signal* $\mathbf{f}$*:*

$$\langle \mathbf{f}, \mathbf{d}_v\rangle = \mathbf{f}(v). \tag{12}$$

Proof. 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, the evaluation functional for a signal $\mathbf{f}$ in the bandlimited space at a point $\mathbf{x} \in \mathbb{R}^n$ is (using Section 2 definition and Theorem 1 Property b from [31]) $\langle \mathbf{f}, \phi(\mathbf{x})\rangle$. This definition provides us the required interpretation for $\langle \mathbf{f}, \mathbf{d}_v\rangle$:

$$\langle \mathbf{f}, \mathbf{d}_v\rangle = \langle \mathbf{f}, \phi(\boldsymbol{\delta}_v)\rangle. \tag{13}$$

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

$$\begin{aligned}\langle \mathbf{f}, \phi(\mathbf{x}) \rangle &= \langle \mathbf{f}, \mathbf{U}_{\mathcal{R}} \mathbf{U}_{\mathcal{R}}^{\mathsf{T}} \mathbf{x} \rangle = \langle \mathbf{U}_{\mathcal{R}} \mathbf{U}_{\mathcal{R}}^{\mathsf{T}} \mathbf{f}, \mathbf{x} \rangle \\ &= \langle \mathbf{f}, \mathbf{x} \rangle. \end{aligned} \tag{14}$$

Thus, from (13) and (14):

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

□

As a consequence of (12), if $\mathbf{f} = \mathbf{d}_w$ we have:

$$\langle \mathbf{d}_w, \mathbf{d}_v \rangle = \mathbf{d}_v(w) = \mathbf{d}_w(v). \tag{15}$$

B. Approximation through distances

We start by proposing a distance based algorithm (DC) based on the updates we derived in (9). 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 to us. 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 (9d) we would like to select nodes that have:

1) Large squared local squared graph coherence $\|\mathbf{d}_v\|^2$ with respect to f frequencies (the first term in (9d), 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 (9d).

The squared local graph coherence (11) of a vertex varies between 0 and 1, taking the largest values at vertices that are poorly connected to the rest of the graph [16]. On the other hand, the subspace $\mathcal{D}_m$ is a linear combination of filtered delta signals corresponding to the vertices in $\mathcal{S}_m$. A filtered delta signal at a vertex v is expected to decay as a function of distance from v. Therefore, for a particular energy $\|\mathbf{d}_v\|^2$, a vertex whose overlap is minimum with the filtered delta signals corresponding to vertices in $\mathcal{S}_m$ will have a small $\|\mathbf{P}_{\mathcal{D}_m} \mathbf{d}_v\|^2$. A filtered delta signal $\mathbf{d}_v$ for a vertex which is farther apart from the sampled vertices will have lesser overlap with the filtered delta signals corresponding to the sampled vertices, 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 filtered delta signal $\mathbf{d}_v$ will have a small projection on the space $\mathcal{D}_m$.

Our proposed algorithm (see 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 sufficient large distance from S are defined as follows

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

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

where $\Delta \in [0\ 1]$, $d(\mathcal{S}_m, v) = \min_{u \in \mathcal{S}_m} d(u, v)$ and d is the geodesic distance on the graph. The distance between two adjacent vertices i, j is given by $d(i, j) = 1/w(i, j)$.

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 [16].

The DC (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 work on the second approach here as Algorithm 2 (AVM).

C. 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 bandwidth of the signal 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 bandwidth of the signal. 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 II-B through equations (4) and (5). AVM has the following advantages:

- We can use the optimization framework we defined in Section III.

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

- 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.

AVM successively simplifies the greedy volume maximization step (10) in three stages.

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 [16] 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 [16] to find squared coherences as Function 1. For implementations of APPROXIMATE LARGEST EIGENVALUE, POLYNOMIAL FILTER COEFFICIENTS, and POLYNOMIAL FILTER that Function 1 calls, we refer the reader to GSP toolbox [32].

2) Approximate inner product matrix: We know that the volume of parallelepiped formed by two fixed length vectors is maximized when the vectors are orthogonal to each other. Now, since vectors that optimize (10) also approximately maximize the volume, we expect them to be close to orthogonal. Thus, we approximate $\mathbf{D}_m^\mathsf{T}\mathbf{D}_m$ 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:

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

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

which leads to an approximation of the determinant:

$$\det\left(\begin{bmatrix}\mathbf{D}_m^\mathsf{T}\mathbf{D}_m & \mathbf{D}_m^\mathsf{T}\mathbf{d}_v \\ \mathbf{d}_v^\mathsf{T}\mathbf{D}_m & \mathbf{d}_v^\mathsf{T}\mathbf{d}_v\end{bmatrix}\right) \approx \det(\mathbf{D}_m^\mathsf{T}\mathbf{D}_m)\det(\mathbf{d}_v^\mathsf{T}\mathbf{d}_v - \mathbf{d}_v^\mathsf{T}\hat{\mathbf{D}}_m\hat{\mathbf{D}}_m^\mathsf{T}\mathbf{d}_v), \tag{16}$$

Function 1 Compute coherence [16]

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$(0, \lambda_n, \lambda, d)$
  $\left[\mathbf{r}^1_{\text{filt}}, \cdots, \mathbf{r}^L_{\text{filt}}\right] \leftarrow \left[\text{POLYNOMIAL FILTER}(\mathbf{L}, \text{coeffs}, \mathbf{r}^1), \cdots, \text{POLYNOMIAL FILTER}(\mathbf{L}, \text{coeffs}, \mathbf{r}^L)\right]$
  $SS \leftarrow \sum_{i=1}^{n}\sum_{l=1}^{L}\left(\mathbf{r}^l_{\text{filt}}\right)_i^2$
  while $\text{round}\,(SS) \neq k$ or $|\underline{\lambda} - \bar{\lambda}| > \varepsilon \cdot \bar{\lambda}$ **do**
    if $\text{round}\,(SS) \geq k$ **then**
      $\bar{\lambda} \leftarrow \lambda$.
    else
      $\underline{\lambda} \leftarrow \lambda$.
    end if
    $\lambda \leftarrow (\underline{\lambda} + \bar{\lambda})/2$
    coeffs $\leftarrow$ POLYNOMIAL FILTER COEFFICIENTS$(0, \lambda_n, \lambda, d)$
    $\left[\mathbf{r}^1_{\text{filt}}, \cdots, \mathbf{r}^L_{\text{filt}}\right] \leftarrow \left[\text{POLYNOMIAL FILTER}(\mathbf{L}, \text{coeffs}, \mathbf{r}^1), \cdots, \text{POLYNOMIAL FILTER}(\mathbf{L}, \text{coeffs}, \mathbf{r}^L)\right]$
    $SS \leftarrow \sum_{i=1}^{n}\sum_{l=1}^{L}\left(\mathbf{r}^l_{\text{filt}}\right)_i^2$
  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

where $\hat{\mathbf{D}}_m$ is obtained from $\mathbf{D}_m$ by normalizing the columns: $\hat{\mathbf{D}}_m = \mathbf{D}_m\,\text{diag}\,(1/\|\mathbf{d}_1\|, \cdots, 1/\|\mathbf{d}_m\|)$. The second term in (16) can be written as:

$$\mathbf{d}_v^{\mathsf{T}}\hat{\mathbf{D}}_m\hat{\mathbf{D}}_m^{\mathsf{T}}\mathbf{d}_v = \frac{\langle \mathbf{d}_v, \mathbf{d}_1\rangle^2}{\|\mathbf{d}_1\|^2} + \cdots + \frac{\langle \mathbf{d}_v, \mathbf{d}_m\rangle^2}{\|\mathbf{d}_m\|^2}, \tag{17}$$

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

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 with the ideal low pass filter using (7). However, since we avoid eigendecomposing the Laplacian, $\mathbf{U}$ 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 $\mathbf{d}_v$.

4) Fast inner product computations: Maximization of (16) requires evaluating the inner products $\langle \mathbf{d}_v, \mathbf{d}_i\rangle$ in (17) 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 (15), which allows us to simplify (17) as follows:

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

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:

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

In summary, by virtue of these series of approximations, 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 $cs\log s$ random projections,
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 [16], which we use in Step 1 (Section III-C1) 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 that we 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.

D. 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 finite 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 $O(|\mathcal{E}|dT_1\log|\mathcal{V}| + s|\mathcal{E}|d)$.

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) 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 with respect to an increase in the number of samples. 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 [7], algorithms we compare in Section VI-B grow linearly with respect to the requested number of samples. Additionally, AVM's complexity also scales linearly with respect to an increase in the number of samples as the complexity factor $O(s|\mathcal{E}|d)$ suggests.

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 (9d) were useful to develop DC and AVM[3] 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.

IV. Volume maximization related work

We next study how existing graph signal sampling methods are related to volume maximization. We start by focusing on the SP algorithm from [7] and show how can it be seen as a volume maximization method. This idea is developed in Section IV-A and Section IV-B. Section IV-C then considers other eigendecomposition-free methods and draw parallels with our volume maximization approach.

A. SP algorithm as Gaussian elimination

The SP algorithm is based on the following theorem.

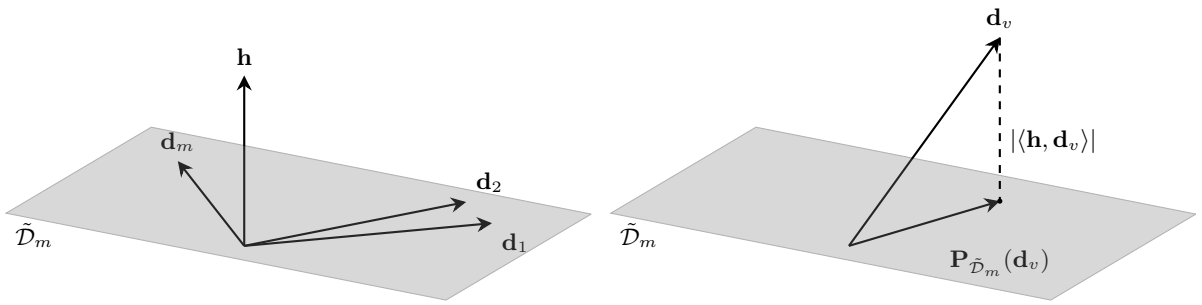
Theorem 1. *[7] Let* $\mathbf{L}$ *be the combinatorial Laplacian of an undirected graph. For a set* $\mathcal{S}_m$ *of size* m*, let* $\mathbf{U}_{\mathcal{S}_m,1:m}$ *be full rank. Let* $\boldsymbol{\psi}_k^*$ *be zero over* $\mathcal{S}_m$ *and a minimizing signal in the Rayleigh quotient of* L^k *for a positive integer* k*.*

$$\boldsymbol{\psi}_k^* = \underset{\boldsymbol{\psi},\boldsymbol{\psi}(\mathcal{S}_m)=\mathbf{0}}{\arg\min} \frac{\boldsymbol{\psi}^\mathsf{T}\mathbf{L}^k\boldsymbol{\psi}}{\boldsymbol{\psi}^\mathsf{T}\boldsymbol{\psi}}. \tag{18}$$

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* $k \to \infty$*.*

Proof. [7], for l_2 convergence see Appendix A. □

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



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

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

Fig. 1. Geometry of SP.

Following (18), the next 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 (18). 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 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.

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 $\mathcal{S}_m$ [7]. $\mathbf{U}'$ can be written as:

$$\mathbf{U}' = \begin{bmatrix} \boldsymbol{u}_1(1) & & & & & & \\ & \boldsymbol{u}_2'(2) & & \mathbf{0} & & & \\ & & \ddots & & \mathbf{u}_{m+2} & \cdots & \mathbf{u}_n \\ & & & \boldsymbol{u}_{m+1}'(m+1) & & & \\ & & & \vdots & & & \\ & \bigstar & & \boldsymbol{u}_{m+1}'(n) & & & \end{bmatrix}, \tag{19}$$

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 next section explains how an iteration in the ideal SP algorithm can be seen as a volume maximization step.

B. 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 $\det(\mathbf{U}_{\mathcal{S}_m \cup v,\mathcal{R}_m} \mathbf{U}_{\mathcal{S}_m \cup v,\mathcal{R}_m}^{\mathsf{T}})$ 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 additional advantage of making the

connection with the Gaussian elimination concept we discussed in Section IV-A. Now, focusing on the selection of the $(m+1)^{\text{th}}$ sample, we can state the following result.

Proposition 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 $|\det(\mathbf{U}_{\mathcal{S}_m \cup v, \mathcal{R}_m})|$.*

Proof. 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 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 $\mathcal{S}_m$ that 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 contained parallelepiped, $\text{Vol}(\mathbf{d}_1, \cdots, \mathbf{d}_m, \mathbf{d}_v)$.

Let $\mathbf{h}$ be a unit vector along the direction of $(m+1)^{\text{th}}$ column of $\mathbf{U}'$ in (19). We are interested in finding the vertex v that maximizes $|\mathbf{h}(v)|$.

Proposition 2. *The signal value $\mathbf{h}(v)$ is the length of projection of $\mathbf{d}_v$ on $\mathbf{h}$.*

Proof. The signal $\mathbf{h}$ belongs to the bandlimited space, $\mathbf{h} \in \text{span}(\mathbf{U}_{\mathcal{R}_m})$. Thus, using (12) 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 $\mathbf{h}$. □

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

Proposition 3. *The signal $\mathbf{h}$ is such that $\mathbf{h} \in \text{span}(\mathbf{d}_1, \cdots, \mathbf{d}_m)^\perp \cap \text{span}(\mathbf{U}_{\mathcal{R}_m})$.*

Proof. All pivots in the Gaussian elimination of $\mathbf{U}_{\mathcal{S}_m \cup v, \mathcal{R}_m}$ are non-zero, as seen in (19), so that the following equivalent statements follow:

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

The second statement follows (0.4.6 (b) [30]) 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 (19) is zero over the set $\mathcal{S}_m$ so that, from Proposition 1:

$$\mathbf{h}(1) = 0 \implies \langle \mathbf{h}, \mathbf{d}_1 \rangle = 0,$$

$$\vdots$$

$$\mathbf{h}(m) = 0 \implies \langle \mathbf{h}, \mathbf{d}_m \rangle = 0,$$

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 $\tilde{\mathcal{D}}_m = \text{span}(\mathbf{d}_1, \cdots, \mathbf{d}_m)$ of dimension m, $\text{span}\,(\mathbf{h})$ is the orthogonal complement subspace to $\tilde{\mathcal{D}}_m$ (see Fig. 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 $\mathcal{R}_m = \{1, \cdots, m+1\}$, 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 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 $\tilde{\mathcal{D}}_m$.*

Proof. Since $\mathbf{d}_v \in \text{span}(\mathbf{U}_{\mathcal{R}_m})$, it can be resolved in to two orthogonal components with respect to the two orthogonal (Prop. 3) spaces $\tilde{\mathcal{D}}_m$ and $\text{span}\,(\mathbf{h})$.

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

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 $\tilde{\mathcal{D}}_m$.

$$\arg\max_v \langle \mathbf{d}_v, \mathbf{h} \rangle^2 = \arg\max_v \|\mathbf{d}_v\|^2 - \left\|\mathbf{P}_{\tilde{\mathcal{D}}_m} \mathbf{d}_v\right\|^2 \tag{20}$$

Fig. 1b shows this orthogonality relation between $\mathbf{d}_v$, $|\langle \mathbf{h}, \mathbf{d}_v \rangle|$, and $\mathbf{P}_{\tilde{\mathcal{D}}_m}(\mathbf{d}_v)$. □

So the v^* chosen is the one that maximizes the volume of the space spanned by the filtered delta signals.

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

The last line follows from using the definition of volume of parallelepiped [33]. Note that, although Proposition 4 could have been derived from the determinant property in Proposition 1 using (8), the approach using the orthogonal vector to the subspace in Proposition 2, Proposition 3 and Proposition 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. For an empirical comparison, in Section V 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**

$\left[\mathbf{d}_1, \cdots, \mathbf{d}_{|\mathcal{S}|}\right] \leftarrow \left[\mathbf{U}_{\mathcal{R}}\mathbf{U}_{\mathcal{R}}^{\mathsf{T}}\boldsymbol{\delta}_1, \cdots, \mathbf{U}_{\mathcal{R}}\mathbf{U}_{\mathcal{R}}^{\mathsf{T}}\mathbf{d}_{|\mathcal{S}|}\right]$

$\tilde{\mathcal{D}} \leftarrow \mathrm{span}_{v\in\mathcal{S}}(\mathbf{d}_v)$

$v* \leftarrow \arg\max_v \|\mathbf{d}_v\|^2 - \|\mathbf{P}_{\tilde{\mathcal{D}}}\mathbf{d}_v\|^2$

$\mathcal{S} \leftarrow \mathcal{S} \cup v^*$

$\mathcal{R} \leftarrow \mathcal{R} \cup |\mathcal{R}| + 1$

end while

return $\mathcal{S}$

end function

C. Eigendecomposition-free methods as volume maximization

So far we have covered existing literature on D-optimality both in general and as it relates to graphs, and proposed two algorithms towards that goal. From (9d), note that the greedy update for approximate volume maximization is

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

Based on this we can revisit some eigendecomposition-free algorithms for graph signal sampling from the perspective of volume maximization. For each of these 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 and sampling based only on $\|\mathbf{d}_v\|^2$. Alternatively, the SP algorithm approximates this by

$$v^* = \underset{v\in\mathcal{S}_m^c}{\arg\max} \|\mathbf{d}_v\|^2 - \left\|\mathbf{P}_{\tilde{\mathcal{D}}_m}\mathbf{d}_v\right\|^2 \tag{21}$$

for a finite value of parameter k and a varying $\tilde{\mathcal{D}}_m$ in place of $\mathcal{D}_m$ — see Section IV-B. The generalization to the eigendecomposition-free approach of [34](V2) proposes to maximize (using the greedy selection in Equation (31)):

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

but in practice maximizes a different expression — see (32) in [17].

The crucial difference between our proposed method and [17] is that we obtain a specific expression to be maximized through D-optimality. Whereas [17] clearly shows the relation between various experiment design objective functions and their corresponding localization operators, the relation between the algorithm proposed in [17], LSSS, and the experiment design objective functions is unclear.

We do not attempt to explain methods such as [35] under the volume maximization framework as they define the signal smoothness through 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

TABLE II

APPROXIMATION TO GREEDY MAXIMIZATION OF DETERMINANT. LSSS - FOR IMPLEMENTATION DETAILS REFER TO SECTION IV-C

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	$p(v) \propto \|\mathbf{d}_v\|^2$	No projection.
SP	$\arg\max_{v\in\mathcal{S}_m^c} \|\mathbf{d}_v\|^2 - \left\|\mathbf{P}_{\tilde{\mathcal{D}}_m}\mathbf{d}_v\right\|^2$	Projection space approximate and increasing in size.
LSSS	$\arg\max_{v\in\mathcal{S}_m^c} \|\mathbf{d}_v\|^2 - 2\sum_{w\in\mathcal{S}_m}\langle\|\mathbf{d}_w\|,\|\mathbf{d}_v\|\rangle$	Inner product approximation for projection.
AVM	$\arg\max_{v\in\mathcal{S}_m^c} \|\mathbf{d}_v\|^2 - \sum_{w\in\mathcal{S}_m}\frac{\mathbf{d}_w^2(v)}{\|\mathbf{d}_w\|^2}$	Assumption of orthogonality.

function for various eigendecomposition-free sampling methods that we studied are summarized in Table II. The differences between various sampling methods will be apparent when we compare their performance for various sampling and reconstruction settings.

V. EXPERIMENTAL SETTINGS

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

A. Signal, Graph Models and Sampling setups

With a perfectly bandlimited signal, most sampling schemes achieve similar performance in terms of reconstruction error. 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.

1) Signal smoothness and graph topologies: : Consider first a synthetic noisy signal model. The signal $\mathbf{f}$ is bandlimited with added noise $\mathbf{n}$. The resulting signal is $\mathbf{f} = \mathbf{x} + \mathbf{n}$ which can be expressed as $\mathbf{U}_{\mathcal{F}}\tilde{\mathbf{f}}_{\mathcal{F}} + \mathbf{n}$ with the frequency components of $\mathbf{x}$, and noise being random variables distributed as multivariate normal distributions: $\tilde{\mathbf{f}}_{\mathcal{F}} \sim \mathcal{N}(\mathbf{0}, c_1\mathbf{I}_{\mathcal{F}\mathcal{F}})$, $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, c_2\mathbf{I}_{\mathcal{V}\mathcal{V}})$. 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 number of samples, graph topologies, and the graph size on DC and AVM, we fix the bandwidth of the signal 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 [16]).

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) Sampling set sizes: We use two graph sizes 500 and 1000. Except for the Erdős Rényi graph model, we use the Grasp [36] and GSPBox [32] MATLAB toolboxes to generate the graph instances — see Table III.

We use sampling set sizes from 60 samples to 200 samples to compare the variation of reconstruction error. For comparing algorithms in this setting, we do not show the full range of reconstruction SNRs from WRS because

TABLE III

TYPES OF GRAPHS IN THE EXPERIMENTS

Graph model	Instance	Construction comments
Random sensor knn	`grasp_plane_knn(n, k),k=8 or 15`	Uniformly sampled vertices on 2d plane with k nearest neighbors
	`gsp_sensor(n, 20)`	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`	`param.Nc` communities
	`gsp_community(n, param)`	
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

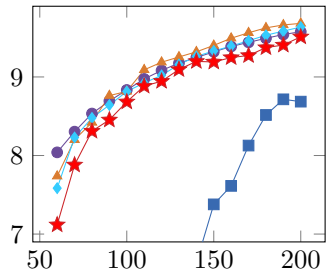
its SNR is usually 5-10 dBs lower than other methods at starting sampling rate of 60 (see Tables V and VII for performance at higher sampling rates).

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 [7]. The graph sampling and reconstruction algorithms then select number of samples ranging from 60 to 200. Using 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.

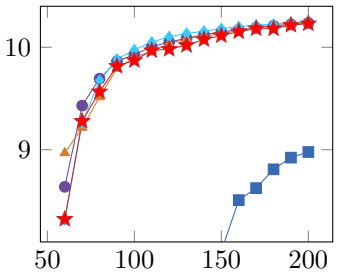
4) Effect of scaling graph sizes: One of our primary goals is to develop fast and scalable algorithms. To evaluate these properties, we use a random sensor nearest neighbors graph with 20 nearest neighbors, and community graph with 10 communities with different graphs sizes of 500, 1000, 2000, 4000, 8000. For each graph size we sample 150 vertices, and the signal model remains the same as in Section V-A1. We report the SNR of the reconstruction 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 orders of magnitude larger than thousands of vertices is an indicator of scalability, so we also test the sampling algorithms on relatively larger graph sizes of 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 or require more than 64GB of the available 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 [37], 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, but due to fundamental difference in the reconstruction method we do not plot the SNRs together with those of smaller graphs. The execution times measured in seconds are

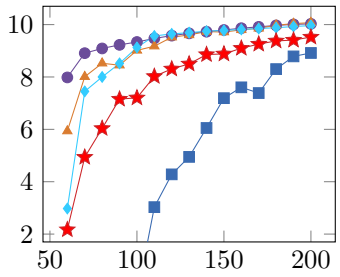
rounded to one decimal precision for display.



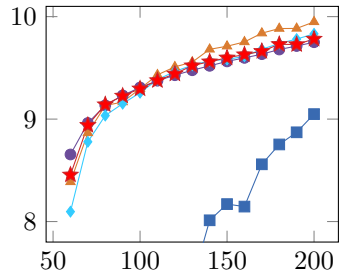
(a) Random sensor knn graph, size 1000



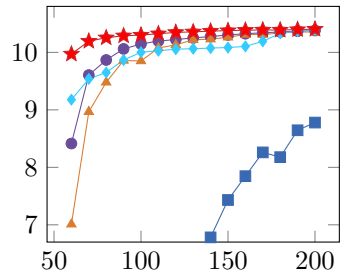
(b) Scale-free graph, size 1000



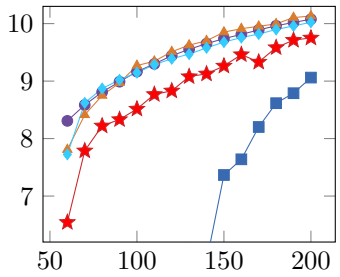
(c) Community graph, size 1000, 10 communities



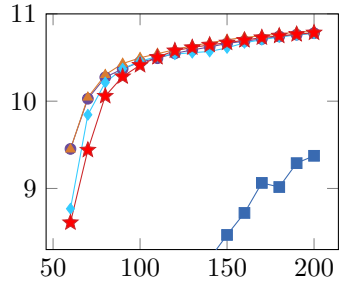
(d) WS graph, size 1000



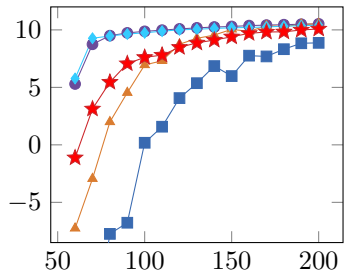
(e) Erdős Rényi graph, size 1000



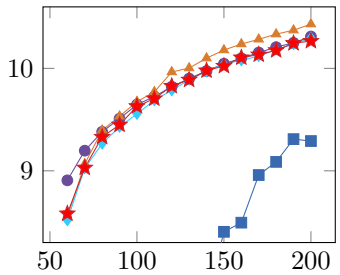
(f) Random sensor knn graph, size 500



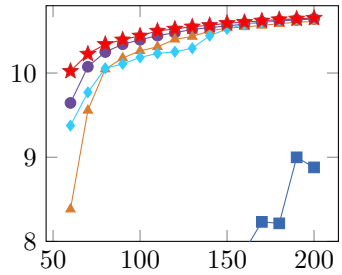
(g) Scale-free graph, size 500



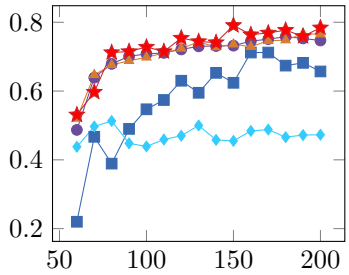
(h) Community graph, size 500, 5 communities



(i) WS graph, size 500



(j) Erdős Rényi graph, size 500



(k) Classification accuracies

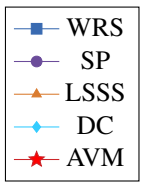


Fig. 2. Comparison of eigendecomposition-free methods in the literature. x-axis: number of samples selected. y-axis: average SNR of the reconstructed signals. We do not include the entire range of SNR from WRS based reconstruction because of its comparatively wider range.

We will now describe how algorithms are initialized.

B. Initialization details

We wish to evaluate all the algorithms on an equal footing. So for evaluating graph 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 balance between the approximation accuracy and the amount of computations. The degree of the polynomial is selected so as to be larger than the diameter of most graphs we consider.

The various algorithms we consider have some hard-coded parameter values. SP has just one parameter k to which we assign $k = 4$. LSSS has a few more parameters to tune like ν, η. In [17] the parameter ν is chosen experimentally to be 75, 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 cases where optimal reconstruction SNR was achieved at ν values differing from the proposed 75 by several orders of magnitude when we considered different topologies, graph sizes and Laplacians. As for the sampling times, we choose the sampling time corresponding to the ν chosen as per the maximum SNR. We experimentally determine η the same way as in the original implementation. For the DC algorithm, we choose $\Delta = 0.9$.

C. Reconstruction techniques

We denote the sampled signal $\mathbf{f}_S$ and the lowpass frequencies of the original signal by $\tilde{\mathbf{f}}_{\mathcal{F}} = \mathbf{U}_{\mathcal{F}}^{\mathsf{T}}\mathbf{f}$. The ideal reconstruction which minimizes the mean square error using the sampled signal is given by the least squares solution to $\left\|\mathbf{U}_{\mathcal{SF}}\tilde{\mathbf{f}}_{\mathcal{F}} - \mathbf{f}_{\mathcal{S}}\right\|_2$. Other existing methods of reconstruction are — using a linear combination of tailored kernels as seen in LSSS, or solving a regularized least squares as seen 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 best results for WRS, instead of least squares solution we use the recommended weighted least squares [16], although it is slightly different from what we use for all other algorithms. We use Moore-Penrose pseudo inverse for all our least squares solutions.

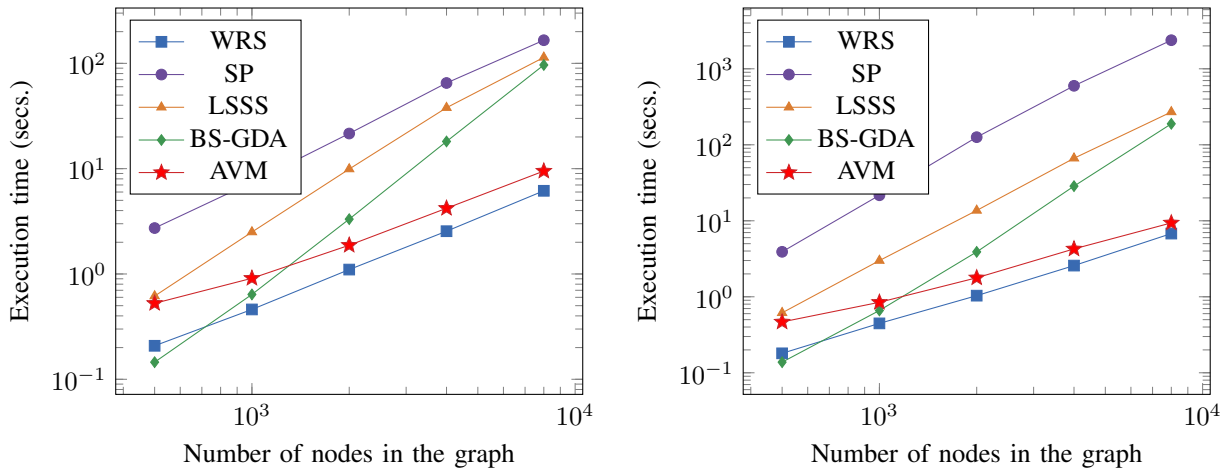
VI. Results

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

A. Reconstruction error

In this paper we look at the mean squared error in reconstructing the signal $\mathbf{f}$. So we measure the error $\left\|\hat{\mathbf{f}} - \mathbf{f}\right\|^2$ in the reconstructed signal with respect to 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 model of graph, with a new signal generated for each graph instance Fig. 2. The two graph models where we observe a lesser SNR for AVM



(a) Random sensor graphs with 20 nearest neighbour connections

(b) Community graphs with 10 communities

Fig. 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.

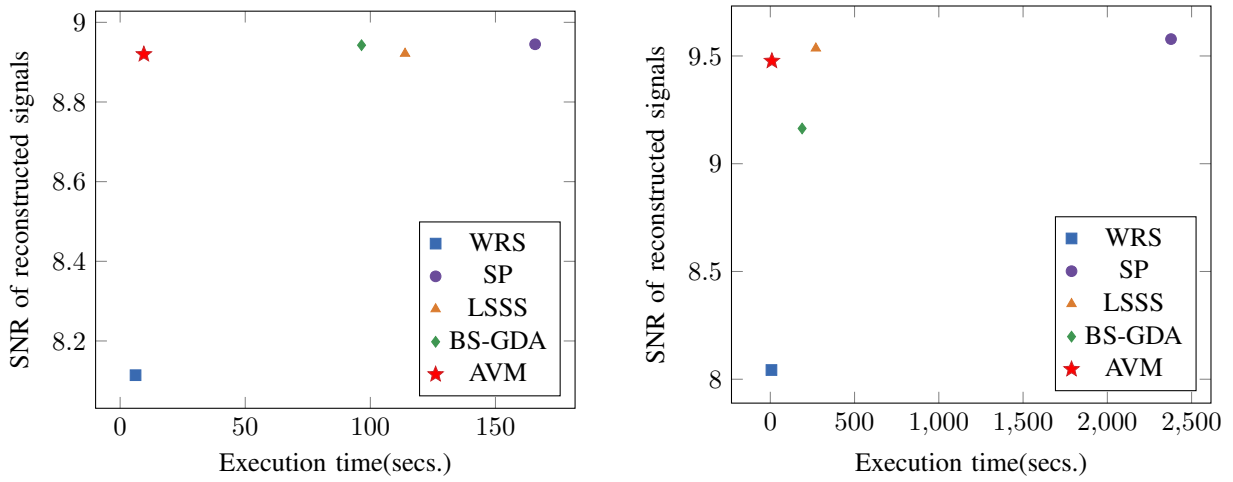
algorithm are the Random sensor nearest neighbors and the Community graph models which we discuss next. For the remaining graph models the reconstruction SNR from DC and AVM sampling is comparable to other algorithms, such as SP and LSSS. In fact, we find the sampling from AVM to be comparatively satisfactory considering that we are reporting the maximum SNRs for LSSS over 5 different parameter values.

In Fig. 2f we notice that for Random sensor nearest neighbors 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 we can see in Figures 2c and 2h.

The issue, however, is no longer critical for larger graphs as we see when we increase the number of samples to 150 — (Table VII), 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 [38].

For the USPS dataset classification, we do 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.



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

(b) Community graphs with 10 communities

Fig. 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.

TABLE IV

EXECUTION TIME(SECS.) FOR SAMPLING, RANDOM SENOR 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 V

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

B. Speed

Using the setup from Section V-A, 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 [32] 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 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.

TABLE VI

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 VII

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

To minimize the effect of other processes running at different times, we run the sampling algorithms in a round robin fashion. 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 affects our timing. It is impossible to stop virtually all background processes, so we try to reduce their impact in two ways. We iterate over each sampling scheme 50 times and report the averages. Secondly, 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 size of the graph increases, AVM is only slightly slower compared to WRS. It is orders of magnitude faster than SP, LSSS and the BS-GDA algorithm — see Tables IV and VI, while having a very small impact on the SNR of the reconstructed signal — Tables V and VII. The execution time also scales very well with respect to the graph size Fig. 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 existing graph sampling algorithms consider WRS as the fastest sampling algorithm and benchmark against it. By specifying our overhead rates with respect to WRS we can indirectly compare our algorithm with myriad others without doing so one by one. We report these factors for various graph sizes in Tables IV and VI.

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 Fig. 4a, 4 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 the size of 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.

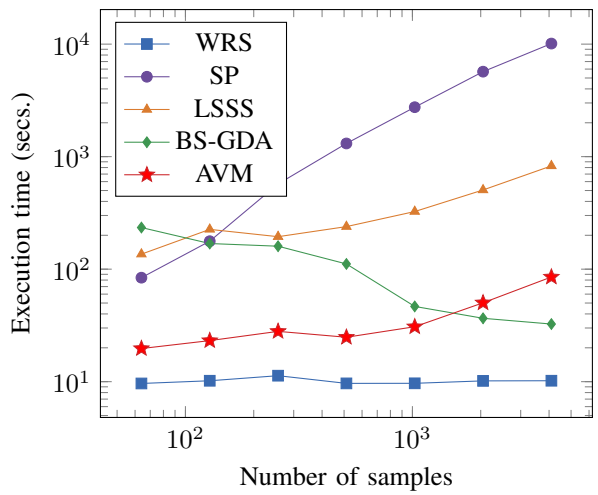


Fig. 5. Random sensor graphs with 8192 vertices and 20 nearest neighbour connections.

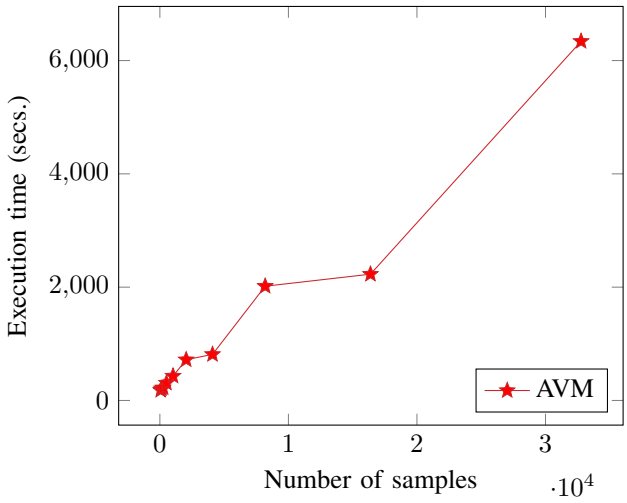


Fig. 6. Random sensor graphs with 50,000 vertices and 20 nearest neighbour connections.

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 IV, V, VI, and VII.

In those tables of algorithm comparisons, we look for scalable algorithms that have low execution times and high SNRs, or which at least finish execution within our limits as mentioned in Section V-A4. 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 the SNR from reconstructed signal over that of WRS, and at most 0.46% decrease compared to other algorithms, although the SNRs are low compared to 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.

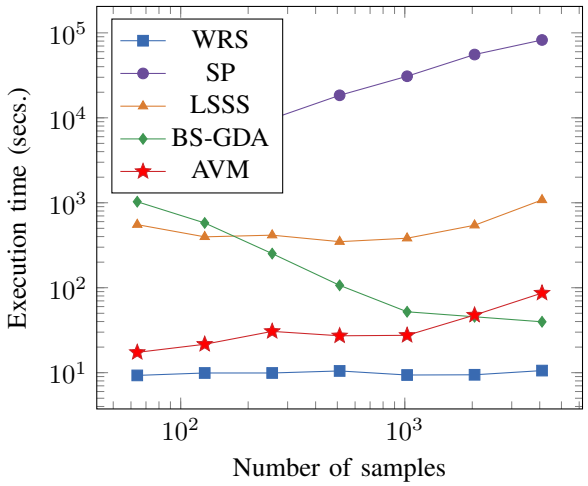


Fig. 7. Community graphs with 8192 vertices and 10 communities.

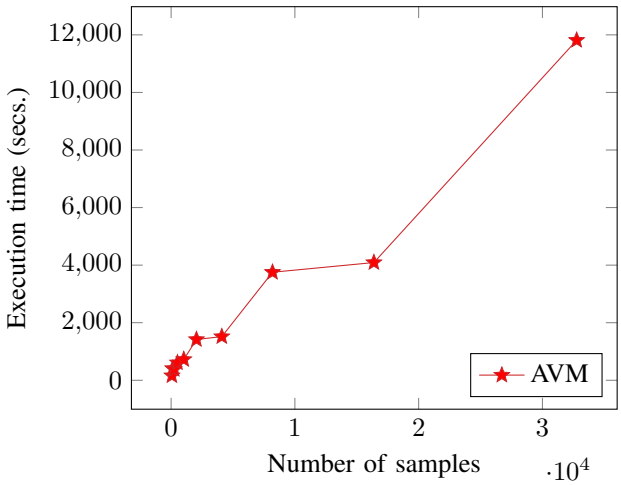


Fig. 8. Community graphs with 50,000 vertices and 10 communities.

C. *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 of the graph types, 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 5 and 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 are 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 an increase in the number of requested samples. Except for sample set sizes 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 further assess 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 range 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 6 and 8, apart from minor fluctuations, we see that the execution times vs the number of samples data points lie on a straight line for an order of 10^4 range in the number of sampled vertices. This observation agrees with our theoretical analysis of AVM complexity in Section III-D2 explaining an additional $O(s|\mathcal{E}|d)$ dependence on the number of samples compared to WRS.

VII. 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 paper 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 improvements in speed.

VIII. Acknowledgements

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

Appendix A
Proof of eigenvector convergence

Lemma 2. *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 $c_0 \neq 0$.*

Proof. The set of signals $\{\boldsymbol{\phi} : \boldsymbol{\phi}(\mathcal{S}_m) = \mathbf{0}, \|\boldsymbol{\phi}\| = 1\}$ is a closed set. Let $\begin{pmatrix} x_1 & \cdots & x_n \end{pmatrix}^\mathsf{T}$ be in the set for any ϵ. Then $\begin{pmatrix} x_1 + \epsilon/2 & x_2 & \cdots & x_n \end{pmatrix}^\mathsf{T}$ 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 $(\alpha_1, \cdots, \alpha_n)^\mathsf{T}$ and $(\beta_1, \cdots, \beta_n)^\mathsf{T}$. 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 1.

Proof. 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.

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

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.

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

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:

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

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 [39].

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

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

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.

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

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 $k \to \infty$, $\boldsymbol{\psi}_k^* \to \boldsymbol{\psi}^*$. □

APPENDIX 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). 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 $\mathbf{U}_{\mathcal{SS}}^\mathsf{T}\mathbf{U}_{\mathcal{SS}}$ 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 $\mathbf{U}_{\mathcal{SF}}^\mathsf{T}\mathbf{U}_{\mathcal{SF}}$ is a submatrix of $\mathbf{U}_{\mathcal{SS}}^\mathsf{T}\mathbf{U}_{\mathcal{SS}}$.

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

This helps us to relate the matrix and its submatrix determinants using Fischer's inequality from Theorem 7.8.5 in [30].

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

The determinant of the matrix $\mathbf{U}_{\mathcal{S},f+1:s}^\mathsf{T}\mathbf{U}_{\mathcal{S},f+1:s}$ can be bounded above. The eigenvalues of $\mathbf{U}_{\mathcal{S},f+1:s}^\mathsf{T}\mathbf{U}_{\mathcal{S},f+1:s}$ are the same as the non-zero eigenvalues of $\mathbf{U}_{\mathcal{S},f+1:s}\mathbf{U}_{\mathcal{S},f+1:s}^\mathsf{T}$ by Theorem 1.2.22 in [30]. Using eigenvalue interlacing Theorem 8.1.7 from [40], the eigenvalues of the matrix $\mathbf{U}_{\mathcal{S},f+1:s}\mathbf{U}_{\mathcal{S},f+1:s}^\mathsf{T}$ are less than or equal to 1 because it is submatrix of $\mathbf{U}_{\mathcal{V},f+1:s}\mathbf{U}_{\mathcal{V},f+1:s}^\mathsf{T}$ whose non-zero eigenvalues are all 1. As the determinant of a matrix is the product of its eigenvalues, the following bound applies:

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

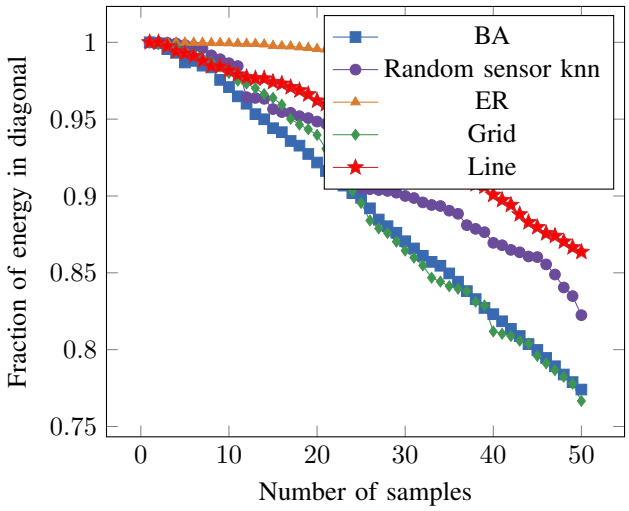


Fig. C.1. Closeness to diagonal at each iteration.

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

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

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

To summarize, instead of aiming to maximize $|\det(\mathbf{U}_{\mathcal{SF}}^{\mathsf{T}}\mathbf{U}_{\mathcal{SF}})|$, we aimed to maximize $|\det(\mathbf{U}_{\mathcal{SS}}^{\mathsf{T}}\mathbf{U}_{\mathcal{SS}})|$. 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.

APPENDIX C
APPROXIMATING GRAM MATRIX BY A DIAGONAL MATRIX

Here we try to estimate how close our approximation of $\mathbf{D}_m^{\mathsf{T}}\mathbf{D}_m$ 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. C.1.

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

is not as close to a diagonal matrix as $\mathbf{D}_m^\mathsf{T}\mathbf{D}_m$ is to a diagonal matrix. Nevertheless, in place of $(\mathbf{D}_m^\mathsf{T}\mathbf{D}_m)^{-1}$ we still use $\operatorname{diag}\left(1/\|\mathbf{d}_1\|^2, \cdots, 1/\|\mathbf{d}_m\|^2\right)$ 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.

APPENDIX D
D-OPTIMAL SAMPLING FOR GENERIC KERNELS

Another graph signal model is a probabilistic distribution instead of a bandlimited model [41], [3]. 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 [42].

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 $\mathbf{K} = \mathbf{U}g(\mathbf{\Sigma})\mathbf{U}^\mathsf{T}$. 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 $|\det(\mathbf{U}_{\mathcal{S}}g(\mathbf{\Sigma})\mathbf{U}_{\mathcal{S}}^\mathsf{T})|$. The expression for the determinant update remains the same as before.

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

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

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

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

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

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.

REFERENCES

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

[2] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tompkins, and E. Upfal, "The web as a graph," in Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. ACM, 2000, pp. 1–10.

[3] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in Proceedings of the 20th International conference on Machine learning (ICML-03), 2003, pp. 912–919.

[4] S. Fortunato, "Community detection in graphs," Physics reports, vol. 486, no. 3, pp. 75–174, 2010.

[5] M. Crovella and E. Kolaczyk, "Graph wavelets for spatial traffic analysis," in INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, vol. 3. IEEE, 2003, pp. 1848–1857.

[6] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević, "Discrete signal processing on graphs: Sampling theory," IEEE Transactions on Signal Processing, vol. 63, no. 24, pp. 6510–6523, 2015.

[7] A. Anis, A. Gadde, and A. Ortega, "Efficient sampling set selection for bandlimited graph signals using graph spectral proxies," IEEE Transactions on Signal Processing, vol. 64, no. 14, pp. 3775–3789, 2015.

[8] Y. Tanaka, Y. C. Eldar, A. Ortega, and G. Cheung, "Sampling signals on graphs: From theory to applications," IEEE Signal Processing Magazine, vol. 37, no. 6, pp. 14–30, 2020.

[9] I. Pesenson, "Sampling in paley-wiener spaces on combinatorial graphs," Transactions of the American Mathematical Society, vol. 360, no. 10, pp. 5603–5627, 2008.

[10] S. Joshi and S. Boyd, "Sensor selection via convex optimization," IEEE Transactions on Signal Processing, vol. 57, no. 2, pp. 451–462, 2008.

[11] H. Shomorony and A. S. Avestimehr, "Sampling large data on graphs," in Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conference on. IEEE, 2014, pp. 933–936.

[12] M. Tsitsvero, S. Barbarossa, and P. Di Lorenzo, "Signals on graphs: Uncertainty principle and sampling," IEEE Transactions on Signal Processing, vol. 64, no. 18, pp. 4845–4860, 2016.

[13] N. Tremblay, P.-O. Amblard, and S. Barthelme, "Graph sampling with determinantal processes," arXiv preprint arXiv:1703.01594, 2017.

[14] L. F. Chamon and A. Ribeiro, "Greedy sampling of graph signals," arXiv preprint arXiv:1704.01223, 2017.

[15] F. Wang, Y. Wang, and G. Cheung, "A-optimal sampling and robust reconstruction for graph signals via truncated neumann series," IEEE Signal Processing Letters, vol. 25, no. 5, pp. 680–684, 2018.

[16] G. Puy, N. Tremblay, R. Gribonval, and P. Vandergheynst, "Random sampling of bandlimited signals on graphs," Applied and Computational Harmonic Analysis, 2016.

[17] A. Sakiyama, Y. Tanaka, T. Tanaka, and A. Ortega, "Eigendecomposition-free sampling set selection for graph signals," IEEE Transactions on Signal Processing, vol. 67, no. 10, pp. 2679–2692, 2019.

[18] Y. Bai, F. Wang, G. Cheung, Y. Nakatsukasa, and W. Gao, "Fast graph sampling set selection using gershgorin disc alignment," IEEE Transactions on Signal Processing, vol. 68, pp. 2419–2434, 2020.

[19] A. Parada-Mayorga, "Blue noise and optimal sampling on graphs," Ph.D. dissertation, University of Delaware, 2019.

[20] A. Parada-Mayorga, D. L. Lau, J. H. Giraldo, and G. R. Arce, "Blue-noise sampling on graphs," IEEE Transactions on Signal and Information Processing over Networks, vol. 5, no. 3, pp. 554–569, 2019.

[21] A. Jayawant and A. Ortega, "A distance-based formulation for sampling signals on graphs," in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2018, pp. 6318–6322.

[22] S. Basirian and A. Jung, "Random walk sampling for big data over networks," in 2017 International Conference on Sampling Theory and Applications (SampTA). IEEE, 2017, pp. 427–431.

[23] O. Abramenko and A. Jung, "Graph signal sampling via reinforcement learning," in ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019, pp. 3077–3081.

[24] A. Atkinson, A. Donev, and R. Tobias, Optimum experimental designs, with SAS. Oxford University Press, 2007, vol. 34.

[25] B. Bollobás, Modern graph theory. Springer Science & Business Media, 2013, vol. 184.

[26] T. Minka, "Inferring a gaussian distribution," Media Lab Note, 1998.

[27] A. Çivril and M. Magdon-Ismail, "On selecting a maximum volume sub-matrix of a matrix and related problems," Theoretical Computer Science, vol. 410, no. 47-49, pp. 4801–4811, 2009.

[28] S. A. Goreinov, I. V. Oseledets, D. V. Savostyanov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, "How to find a good submatrix," in Matrix Methods: Theory, Algorithms And Applications: Dedicated to the Memory of Gene Golub. World Scientific, 2010, pp. 247–256.

[29] A. Deshpande and L. Rademacher, "Efficient volume sampling for row/column subset selection," in 2010 IEEE 51st Annual Symposium on Foundations of Computer Science. IEEE, 2010, pp. 329–338.

[30] R. A. Horn and C. R. Johnson, Matrix analysis. Cambridge university press, 2012.

[31] A. Berlinet and C. Thomas-Agnan, Reproducing kernel Hilbert spaces in probability and statistics. Springer Science & Business Media, 2011.

[32] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond, "Gspbox: A toolbox for signal processing on graphs," 2016.

[33] B. Peng, "The determinant: A means to calculate volume," Recall, vol. 21, p. a22, 2007.

[34] A. Sakiyama, Y. Tanaka, T. Tanaka, and A. Ortega, "Eigendecomposition-free sampling set selection for graph signals," arXiv preprint arXiv:1809.01827, 2018.

[35] A. Jung and N. Tran, "Localized linear regression in networked data," IEEE Signal Processing Letters, vol. 26, no. 7, pp. 1090–1094, 2019.

[36] B. Girault, S. Narayanan, P. Gonçalves, A. Ortega, and E. Fleury, "Grasp: A matlab toolbox for graph signal processing," in 42nd IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2017), 2017.

[37] S. K. Narang, A. Gadde, E. Sanou, and A. Ortega, "Localized iterative methods for interpolation in graph structured data," in 2013 IEEE Global Conference on Signal and Information Processing. IEEE, 2013, pp. 491–494.

[38] P. Erdős and A. Rényi, "On the evolution of random graphs," Publ. Math. Inst. Hung. Acad. Sci, vol. 5, no. 1, pp. 17–60, 1960.

[39] S. Boyd, S. P. Boyd, and L. Vandenberghe, Convex optimization. Cambridge university press, 2004.

[40] G. H. Golub and C. F. Van Loan, Matrix computations. JHU Press, 2012, vol. 3.

[41] A. Gadde and A. Ortega, "A probabilistic interpretation of sampling theory of graph signals," in 2015 IEEE international conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2015, pp. 3257–3261.

[42] M. C. Shewry and H. P. Wynn, "Maximum entropy sampling," Journal of applied statistics, vol. 14, no. 2, pp. 165–170, 1987.