Distributed Implementation of Linear Network Operators using Graph Filters

Santiago Segarra, Antonio G. Marques, and Alejandro Ribeiro

Abstract

A signal in a network (graph) can be defined as a vector whose entries represent the value of a given magnitude at the different nodes. A linear network (graph) operator is then a linear transformation whose input and output are graph signals. This paper investigates how to implement generic network linear operators in a distributed manner, so that nodes only need to exchange a *finite* number of *messages* with their *neighbors*. The schemes are designed within the framework of shift-invariant graph filters, which are polynomials of the so-called graph-shift operator. The graph-shift operator is a matrix that accounts for the topology of the network, and the filter coefficients – which are the same across nodes – are the coefficients of that polynomial. First, we identify conditions under which the linear operator can be computed exactly. Then, we provide approximate designs for cases where perfect implementation is unfeasible. Setups where each node is allowed to use a different set of filter coefficients are also discussed. Finally, we apply this framework to the problem of finite-time consensus and analyze the graph-filter approximation performance for general linear graph operators.

Index Terms

Graph signal processing, Graph filter, Distributed linear transformations, Finite-time consensus

I. Introduction

In a very simple way, networks and graphs can be viewed as structures that encode pairwise relationships between elements of a set. Often, networks have intrinsic value and are themselves the object of study. In other occasions, the network defines an underlying notion of proximity or dependence, but the object of interest is a signal defined on top of the graph, i.e., data associated with the nodes of the network. This is the matter addressed in the field of Graph Signal Processing (GSP), where the notions of, e.g., frequency and linear filtering are extended to signals supported on graphs. A plethora of graph-supported signals exist in different fields, including gene-expression patterns defined on top of gene networks, the spread of epidemics over a social network, and the congestion level at the nodes of a telecommunication network, to name a few. Transversal to the particular application, we must address the question

Work in this paper is supported by USA NSF CCF-1217963 and Spanish MINECO grant No TEC2013- 41604-R. S. Segarra and A. Ribeiro are with the Dept. of Electrical and Systems Eng., Univ. of Pennsylvania. A. G. Marques is with the Dept. of Signal Theory and Comms., King Juan Carlos Univ. Emails: ssegarra@seas.upenn.edu, antonio.garcia.marques@urjc.es, and aribeiro@seas.upenn.edu.

of how to redesign traditional tools originally conceived to study and *process* signals defined on regular domains (such as time-varying signals) and extend them to analyze signals on the more complex graph domain.

The problem that we investigate in this paper is how to design graph filters – which are a generalization of classical time-invariant systems – to implement a pre-specified linear transformation in a distributed manner. During the last decade, development of distributed linear schemes has attracted a lot of attention. The field has evolved quickly, shifting from the estimation of simple network parameters such as the mean using consensus algorithms [1]–[4], to advanced inference problems such as sparse linear regression [5]; see also [6] for a review. In the context of GSP but without leveraging the particular structure of a graph filter, several works have addressed the problem of *distributed* implementation of different linear problems. Relevant examples are the projection of graph-signals onto low-dimensional spaces [7]; the approximation of a specific class of linear transformations (the so-called graph Fourier multipliers) using Chebyshev polynomials [8]; or graph-signal inpainting [9], [10]. Graph filters have been used to implement distributedly *specific* linear transformations such as fast consensus [11], projections onto the low-rank space of graph-bandlimited signals [12], or interpolation of graph-bandlimited signals [13], but not for general linear transformations. Design of distributed linear network operators under a graph-filter framework is useful not only because general conditions for perfect and approximate implementation can be identified, but also because it opens the door to leverage many of the insights and results existing for classical time-invariant filters.

To describe our contribution more clearly, we consider that each node has a certain value, and these values are collected across nodes to form a graph signal. With this definition, graph filters are a specific class of operators whose input and output are graph signals. Mathematically, graph filters are linear transformations that can be expressed as polynomials of the graph-shift operator [14]. Although most existing works use either the Laplacian or the adjacency matrix as shift operators, any sparse matrix that encodes the structure of the underlying network can serve as a shift. For any given shift, the polynomial coefficients determine completely the linear transformation and are referred to as *filter coefficients*. The goal of this paper is then to design the filter coefficients that, given a graph-shift operator, yield the best approximation to a pre-specified linear transformation.

The investigated problem is relevant not only from a theoretical point of view, but also from an application perspective. Since graph filters act on graph signals through the successive application of local operators, the output of a graph filter can be viewed as the outcome of a diffusion or percolation process, with the filter coefficients corresponding to the rate of diffusion. Hence, a problem specially suited for this approach is that of network processes evolving according to a dynamics given by a (weighted) connectivity graph. The input of the graph filter is the initial network state, the output is the final network state, and the shift operator is given by the connectivity graph. The objective is then to design the filter coefficients, which amounts to tuning the parameters that control the dynamics, to approximate a given linear transformation. Another relevant example is distributed state estimation in power or communication networks, such as wireless sensor networks, where due to e.g. lack of infrastructure or privacy issues, a central controller cannot be used and the processing and exchange of information must be implemented locally. In this case, the entries of the shift correspond to the weights that each node gives to the information received from its neighbors, and the filter coefficients correspond to the weights that nodes give to the signals exchanged at different time instants.

The contributions and organization of the paper are as follows. After some modeling preliminaries, we first cast the problem of approximating linear network transformations as a graph-filter design. We then identify the conditions under which a specific linear transformation can be implemented perfectly. Those depend on the spectrum of the linear transformation, the spectrum of the graph-shift operator, and the order of the filter (number of coefficients). The design of optimal approximations when the previous conditions are not met is also addressed. Different approximation (error) metrics are considered, each of them leading to a different design. The results are then complemented in two ways: i) we introduce a more general definition of graph filters, that allows for perfect implementation of a larger class of linear transformations and ii) we particularize our framework to relevant applications, and validate the optimal solution to those problems using numerical simulations.

Notation: Generically, the entries of a matrix $\mathbf{X}$ and a vector $\mathbf{x}$ will be denoted as X_{ij} and x_i; however, when contributing to avoid confusion, the alternative notation $[\mathbf{X}]_{ij}$ and $[\mathbf{x}]_i$ will be used. The notation ^{T} and ^{H} stands for transpose and transpose conjugate, respectively; $\mathrm{Trace}(\mathbf{X}) := \sum_i X_{ii}$ is the trace of the square matrix $\mathbf{X}$; $\lambda_{\max}(\mathbf{X})$ is the largest eigenvalue of the symmetric matrix $\mathbf{X}$; $\mathrm{diag}(\mathbf{x})$ is a diagonal matrix satisfying $[\mathrm{diag}(\mathbf{x})]_{ii} = [\mathbf{x}]_i$; $\mathbf{e}_i$ is the i-th $N \times 1$ canonical basis vector (all entries of $\mathbf{e}_i$ are zero except the ith one, which is one); and $\mathbf{1}$ is the all-ones vector.

II. Graph signals and graph filters

Let $\mathcal{G}$ denote a directed graph with a set of N nodes or vertices $\mathcal{N}$ and a set of links $\mathcal{E}$, such that if node i is connected to j, then $(i, j) \in \mathcal{E}$. The (incoming) neighborhood of i is defined as the set of nodes $\mathcal{N}_i = \{j \,|\, (j, i) \in \mathcal{E}\}$ connected to i. For any given graph we define the adjacency matrix $\mathbf{A}$ as a sparse $N \times N$ matrix with nonzero elements A_{ji} if and only if $(i, j) \in \mathcal{E}$. The value of A_{ji} captures the strength of the connection from i to j. The focus of this paper is not on analyzing $\mathcal{G}$, but a graph signal defined on the set of nodes $\mathcal{N}$. Formally, such a signal can be represented as a vector $\mathbf{x} = [x_1, ..., x_N]^T \in \mathbb{R}^N$ where the i-th component represents the value of the signal at node i or, alternatively, as a function $f : \mathcal{N} \to \mathbb{R}$, defined on the vertices of the graph.

The graph $\mathcal{G}$ is endowed with a *graph-shift operator* $\mathbf{S}$ [14], [15]. The operator $\mathbf{S}$ is a $N \times N$ matrix whose entry S_{ji} can be nonzero only if $i = j$ or if $(i, j) \in \mathcal{E}$. The sparsity pattern of the matrix $\mathbf{S}$ captures the local structure of $\mathcal{G}$, but we make no specific assumptions on the values of the nonzero entries of $\mathbf{S}$. Choices for $\mathbf{S}$ are the adjacency matrix of the graph [14], [15], its Laplacian [16], and their respective generalizations [17]. The intuition behind $\mathbf{S}$ is to represent a linear transformation that can be computed locally at the nodes of the graph. More rigorously, if $\mathbf{y}$ is defined as $\mathbf{y} = \mathbf{S}\mathbf{x}$, then node i can compute y_i provided that it has access to the value of x_j at $j \in \mathcal{N}_i$. We assume henceforth that $\mathbf{S}$ is diagonalizable, so that there exists a $N \times N$ matrix $\mathbf{V}$ and a $N \times N$ diagonal matrix $\mathbf{\Lambda}$ that can be used to decompose $\mathbf{S}$ as $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}$. In particular, $\mathbf{S}$ is diagonalizable when it is normal, i.e., it satisfies $\mathbf{S}\mathbf{S}^H = \mathbf{S}^H\mathbf{S}$ where $\mathbf{S}^H$ denotes the conjugate transpose of $\mathbf{S}$. In that case we have that $\mathbf{V}$ is unitary, which implies $\mathbf{V}^{-1} = \mathbf{V}^H$, and leads to the decomposition $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^H$. Given a graph signal $\mathbf{x}$, we refer to $\widehat{\mathbf{x}} := \mathbf{V}^{-1}\mathbf{x}$ as the frequency representation of $\mathbf{x}$ [14]. Whenever $\widehat{\mathbf{x}}$ is sparse, we say that $\mathbf{x}$ is bandlimited.

A. Graph filters

Graph filters $\mathbf{H}: \mathbb{R}^N \to \mathbb{R}^N$ are linear graph-signal operators of the form

$$\mathbf{H} := \sum_{l=0}^{L-1} h_l \mathbf{S}^l, \tag{1}$$

i.e., polynomials (of degree $L-1$) of the graph-shift operator [15]. The graph filter $\mathbf{H}$ can also be written as $\mathbf{H} = \mathbf{V}\big(\sum_{l=0}^{L-1} h_l \boldsymbol{\Lambda}^l\big)\mathbf{V}^{-1}$. The diagonal matrix $\widehat{\mathbf{H}} := \sum_{l=0}^{L-1} h_l \boldsymbol{\Lambda}^l$ can then be viewed as the frequency response of $\mathbf{H}$ and it can be alternatively written as $\widehat{\mathbf{H}} = \text{diag}(\widehat{\mathbf{h}})$, where vector $\widehat{\mathbf{h}}$ is a vector that contains the N frequency responses of the filter. Let λ_k denote the k-th eigenvalue of $\mathbf{S}$ and define the $N \times L$ Vandermonde matrix

$$\boldsymbol{\Psi} := \begin{pmatrix} 1 & \lambda_1 & \ldots & \lambda_1^{L-1} \\ \vdots & \vdots & & \vdots \\ 1 & \lambda_N & \ldots & \lambda_N^{L-1} \end{pmatrix}. \tag{2}$$

Upon defining the vector containing the coefficients of the filter as $\mathbf{h} := [h_0, \ldots, h_{L-1}]^T$, it holds that $\widehat{\mathbf{h}} = \boldsymbol{\Psi}\mathbf{h}$ and therefore

$$\mathbf{H} = \textstyle\sum_{l=0}^{L-1} h_l \mathbf{S}^l = \mathbf{V}\text{diag}\big(\boldsymbol{\Psi}\mathbf{h}\big)\mathbf{V}^{-1} = \mathbf{V}\text{diag}\big(\widehat{\mathbf{h}}\big)\mathbf{V}^{-1}. \tag{3}$$

This implies that if $\mathbf{y}$ is defined as $\mathbf{y} = \mathbf{H}\mathbf{x}$, its frequency representation $\widehat{\mathbf{y}} = \mathbf{V}^{-1}\mathbf{y}$ satisfies

$$\widehat{\mathbf{y}} = \text{diag}\big(\boldsymbol{\Psi}\mathbf{h}\big)\widehat{\mathbf{x}}. \tag{4}$$

A particularity of graph filters is that they can be implemented locally, e.g., with $L-1$ exchanges of information among neighbors. To see this, it is convenient to define the l-th shifted input signal as $\mathbf{z}^{(l)} := \mathbf{S}^l\mathbf{x}$. From an implementation perspective, two important properties of $\mathbf{z}^{(l)}$ are: i) it can be computed recursively (sequentially) as $\mathbf{z}^{(l)} = \mathbf{S}\mathbf{z}^{(l-1)}$, with $\mathbf{z}^{(0)} = \mathbf{x}$; and ii) node i can compute $[\mathbf{z}^{(l)}]_i$ locally based on the values of $[\mathbf{z}^{(l-1)}]_j$ at $j \in \mathcal{N}_i$. To emphasize this local property, we define $\mathbf{z}_i$ as an $L \times 1$ vector collecting the entries of $\{\mathbf{z}^{(l)}\}_{l=0}^{L-1}$ that are known by node i, so that $[\mathbf{z}_i]_l := [\mathbf{z}^{(l)}]_i$. With $\mathbf{y}$ denoting the output of a graph filter for the input signal $\mathbf{x}$, it follows from (1) that

$$\mathbf{y} = \mathbf{H}\mathbf{x} = \sum_{l=0}^{L-1} h_l \mathbf{S}^l \mathbf{x} = \sum_{l=0}^{L-1} h_l \mathbf{z}^{(l)}. \tag{5}$$

Hence, the i-th entry of vector $\mathbf{y}$ can be computed as $y_i = \sum_{l=0}^{L-1} h_l [\mathbf{z}^{(l)}]_i = \mathbf{h}^T\mathbf{z}_i$, showing that if the nodes know the value of the filter coefficients, y_i can be computed using solely information available at node i. This also suggests a two-step distributed implementation of graph filters. First, $L-1$ sequential shifts are applied to the input signal $\mathbf{x}$ using only local information. At each shift, every node stores its own local value of the shifted signal. In the second step, each node linearly combines weighted versions of the values obtained in the first step. The weights, that are the same across nodes, are given by the coefficients $\mathbf{h}$.

An alternative expression to define a graph filter is [12]

$$\mathbf{H} = a_0 \prod_{l=1}^{L-1} (\mathbf{S} - a_l\mathbf{I}), \tag{6}$$

which also gives rise to a polynomial on $\mathbf{S}$ of degree $L-1$. The above expression can be leveraged to obtain an alternative distributed implementation of graph filters. To show this, we define the intermediate graph signal $\mathbf{w}^{(l)} = \mathbf{S}\mathbf{w}^{(l-1)} - a_l \mathbf{w}^{(l-1)}$, with $\mathbf{w}^{(0)} = \mathbf{x}$. Notice that: i) the computation of $\mathbf{w}^{(l)}$ based on $\mathbf{w}^{(l-1)}$ can be performed by each node locally; and ii) the output of applying $\mathbf{H}$ to input $\mathbf{x}$ can be obtained after $L-1$ shifts just by setting $\mathbf{y} = \mathbf{H}\mathbf{x} = a_0 \mathbf{w}^{(L-1)}$.

A specific advantage of the representation in (6) is that it provides a straightforward way to design frequency annihilating filters. In particular, upon setting $a_l = \lambda_k$ for the k-th eigenvalue of $\mathbf{S}$, then $\mathbf{H}$ will eliminate the frequency basis $\mathbf{v}_k$, i.e., the eigenvector associated with λ_k. It is also useful for shift operators with a large condition number. In such a case, the sequential implementation that first computes all $\{\mathbf{z}^{(l)}\}_{l=0}^{L-1}$ and then combines them linearly can give rise to numerical problems for high powers of $\mathbf{S}$. The implementation based on $\{\mathbf{w}^{(l)}\}_{l=0}^{L-1}$ is more robust to this type of problems by performing the linear combination and the application of the shift jointly.

III. Implementation of linear operators using graph filters

The goal in this paper is to use graph filters [cf. (1)] to implement a pre-specified *linear* graph signal transformation $\mathbf{B}: \mathbb{R}^N \to \mathbb{R}^N$. Since the transformation is linear and the input and output space are the same, $\mathbf{B}$ can be represented by a square $N \times N$ matrix. To be more concrete, we want to find the filter coefficients $\mathbf{h}$ such that

$$\mathbf{B} = \sum_{l=0}^{L-1} h_l \mathbf{S}^l. \tag{7}$$

Conditions under which this equality can be achieved are discussed in Section III-A. In Section III-B we present results on imperfect reconstruction where the goal is to design the filter coefficients such that either the linear transformation obtained is close to $\mathbf{B}$ or the filtered signal is close to $\mathbf{B}\mathbf{x}$ for some statistical model of the input $\mathbf{x}$.

A. Conditions for perfect implementation of linear operators

With $\{\gamma_k\}_{k=1}^N$ denoting the set of eigenvalues of the linear transformation $\mathbf{B}$ and D denoting the number of distinct eigenvalues in $\{\lambda_k\}_{k=1}^N$, the conditions for perfect implementation are given in the next proposition.

Proposition 1: *The linear transformation $\mathbf{B}$ can be implemented using the graph filter $\mathbf{H}$ in* (1) *if the three following conditions hold true:*

i) Matrices $\mathbf{B}$ and $\mathbf{S}$ are simultaneously diagonalizable; i.e., all the eigenvectors of $\mathbf{B}$ and $\mathbf{S}$ coincide.

ii) For all (k_1, k_2) such that $\lambda_{k_1} = \lambda_{k_2}$, it holds that $\gamma_{k_1} = \gamma_{k_2}$.

iii) The degree of $\mathbf{H}$ satisfies $L \geq D$.

If the conditions in Proposition 1 are satisfied, the filter $\mathbf{h}$ that satisfies (7) is the solution to

$$\boldsymbol{\gamma} := [\gamma_1, \ldots, \gamma_N]^T = \widehat{\mathbf{h}} = \boldsymbol{\Psi}\mathbf{h}. \tag{8}$$

If all the eigenvalues of $\mathbf{S}$ are distinct, then condition *ii)* is automatically satisfied and $\boldsymbol{\Psi}$ is a full-rank Vandermonde matrix. Thus, if we require a filter with degree $L = N$, the system of equations in (8) is square and $\boldsymbol{\Psi}$ is invertible, so that $\mathbf{h} = \boldsymbol{\Psi}^{-1}\boldsymbol{\gamma}$. By contrast, if some of the eigenvalues of $\mathbf{S}$ are repeated, $\boldsymbol{\Psi}$ is not full-rank. However, condition *ii)* allows us to remove the rows of $\mathbf{\Psi}$ associated with repeated eigenvalues and invert the corresponding submatrix which, after the removal, is full-rank.

When the linear transformation of interest takes the form $\mathbf{B}_{\text{con}} = \mathbf{1}\mathbf{1}^T/N$, its application to a signal $\mathbf{x}$ yields the consensus signal $\mathbf{B}_{\text{con}}\mathbf{x}$, i.e., a constant signal whose value is equal to $\bar{x} = N^{-1}\sum_{i=1}^{N} x_i$. The fact that $N-1$ eigenvalues of $\mathbf{B}_{\text{con}}$ are equal to zero can be leveraged to show the following corollary.

Corollary 1: *The consensus transformation $\mathbf{B}_{con}$ can be written as a filter $\sum_{l=0}^{N-1} h_l \mathbf{S}^l$ for some $\mathbf{S}$ associated with an undirected graph $\mathcal{G}$ if and only if $\mathcal{G}$ is connected.*

Notice that an immediate consequence of Corollary 1 is that consensus can be achieved in finite time for every connected undirected graph, where $N-1$ is an upper bound on the number of local interactions needed for convergence. One obvious choice for the shift is to set $\mathbf{S} = \mathbf{L}$, but any $\mathbf{S}$ having $\mathbf{1}$ as an eigenvector with an associated simple (non-repeated) eigenvalue can be used. Compared to classical consensus algorithms that require infinite number of iterations, the price to pay here is that the values of $\{\lambda_k\}_{k=1}^{N}$ need to be known in order to compute $\mathbf{\Psi}$ in (8).

In Corollary 1, the leniency of the sufficient requirement on $\mathcal{G}$ for perfect approximation (connectedness) is a consequence of the low-rank of $\mathbf{B}_{\text{con}}$. This hints that rank-1 linear operators are well-suited for graph-filter approximations. For general linear operators, as the rank increases, finding shifts $\mathbf{S}$ that preserve the local connectivity of the network while simultaneously sharing the eigenvectors of $\mathbf{B}$ becomes less likely.

B. Imperfect implementation of linear operators

In general, if the conditions in Proposition 1 are not satisfied, perfect implementation of $\mathbf{B}$ is not feasible. In such cases, the filter design can be carried out to minimize a pre-specified error metric. First, we consider the case where no information about $\mathbf{x}$ is available and the goal is to design $\mathbf{H}$ to resemble $\mathbf{B}$ as much as possible. Defining the error *matrix* as $\mathbf{E} := \mathbf{H} - \mathbf{B}$, our objective is to design the filter coefficients $\mathbf{h}$ so that the Frobenius norm $\|\mathbf{E}\|_{\text{F}}$ is minimized.

If we define the $N^2 \times L$ matrix $\mathbf{\Theta} := [\text{vec}(\mathbf{I}), \text{vec}(\mathbf{S}), \ldots, \text{vec}(\mathbf{S}^{L-1})]$ and the $N^2 \times 1$ vector $\mathbf{b} := \text{vec}(\mathbf{B})$, the optimal filter coefficients are given in the following proposition.

Proposition 2: *The optimal filter coefficients $\mathbf{h}^*$ defined as $\mathbf{h}^* := \text{argmin}_{\mathbf{h}} \|\mathbf{H} - \mathbf{B}\|_F$ are given by*

$$\mathbf{h}^* = (\mathbf{\Theta}^H \mathbf{\Theta})^{-1} \mathbf{\Theta}^H \mathbf{b}. \tag{9}$$

The above proposition specifies the optimal filter whenever the spectrum of $\mathbf{B}$ is incompatible with that of $\mathbf{S}$ – violations on conditions *i)* or *ii)* in Proposition 1 – or when the filter degree is lower than that needed for perfect reconstruction. Optimal filter designs can be developed for alternative ways of quantifying the error, e.g., $\mathbf{h}$ can be chosen so that $\|\mathbf{E}\|_2$ is minimized. This optimization will be carried out in the full version of this paper.

Whenever prior knowledge of the input signal $\mathbf{x}$ is available, it can be incorporated into the design of the filter coefficients. In such a case, the goal is to minimize a metric of the error *vector* $\mathbf{e} := \mathbf{H}\mathbf{x} - \mathbf{B}\mathbf{x}$. A case of particular interest is when $\mathbf{x}$ is drawn from a zero-mean distribution with known covariance $\mathbf{R}_{\mathbf{x}} := \mathbb{E}[\mathbf{x}\mathbf{x}^H]$. In this case, the error covariance is given by $\mathbf{R_e} := \mathbb{E}[\mathbf{e}\mathbf{e}^H] = (\mathbf{H}-\mathbf{B})\mathbf{R_x}(\mathbf{H}-\mathbf{B})^H$. Our objective is to pick $\mathbf{h}$ to minimize some metric of the error covariance matrix $\mathbf{R_e}$. Analogously to the notation introduced before Proposition 2, define the $N^2 \times L$ matrix $\mathbf{\Theta}_{\mathbf{R_e}} := [\mathrm{vec}(\mathbf{I}\mathbf{R}_\mathbf{e}^{1/2}),\ \mathrm{vec}(\mathbf{S}\mathbf{R}_\mathbf{e}^{1/2}), \ldots, \mathrm{vec}(\mathbf{S}^{L-1}\mathbf{R}_\mathbf{e}^{1/2})]$ and the $N^2 \times 1$ vector $\mathbf{b}_{\mathbf{R_e}} := \mathrm{vec}(\mathbf{B}\mathbf{R}_\mathbf{e}^{1/2})$. If we focus on minimizing $\mathrm{Trace}(\mathbf{R_e})$, which is equivalent to minimizing the mean squared error of $\mathbf{e}$, the following result follows.

Proposition 3: *The optimal filter coefficients* $\mathbf{h}^*$ *defined as* $\mathbf{h}^* := \mathrm{argmin}_\mathbf{h}\, \mathrm{Trace}(\mathbf{R_e})$ *are given by*

$$\mathbf{h}^* = (\mathbf{\Theta}_{\mathbf{R_e}}^H \mathbf{\Theta}_{\mathbf{R_e}})^{-1} \mathbf{\Theta}_{\mathbf{R_e}}^H \mathbf{b}_{\mathbf{R_e}}. \tag{10}$$

Expression (10) specifies how the statistical knowledge of $\mathbf{x}$ can be incorporated into the design of $\mathbf{h}$. Moreover, by comparing (9) and (10), it is immediate that the first scenario, where no prior knowledge of $\mathbf{x}$ was assumed, is equivalent to the second scenario for the particular case of $\mathbf{R_x} = \mathbf{I}$, i.e., the case where the components of $\mathbf{x}$ are uncorrelated. The filter coefficients could be chosen to minimize a different error metric, e.g., $\lambda_{\max}(\mathbf{R_e})$. In this case, instead of minimizing the mean squared error of $\mathbf{e}$ across realizations of $\mathbf{x}$ we would be minimizing the worst-case error achievable by all possible realizations of $\mathbf{x}$. Additional assumptions could be incorporated into the designs including, but not limited to, additional statistical knowledge of $\mathbf{x}$ and structural properties such as sparsity or bandlimitedness [14].

C. Extensions to generalized graph filters

The goal in this section is to modify the definition in (1) so that a larger class of linear operators can be implemented. In particular, we propose a node-varying shift-invariant graph filter, which is defined as a graph signal operator $\mathbf{H}_{\mathrm{nv}} : \mathbb{R}^N \to \mathbb{R}^N$ of the form [cf. (1)]

$$\mathbf{H}_{\mathrm{nv}} := \sum_{l=0}^{L-1} \mathrm{diag}(\mathbf{c}^{(l)})\mathbf{S}^l. \tag{11}$$

In the above definition, $\mathbf{c}^{(l)}$ is a $N \times 1$ vector whose i-th element represents the weight that the filter applies to the i-th entry of $\mathbf{S}^l\mathbf{x}$; i.e., the value of the shifted signal $\mathbf{S}^l\mathbf{x}$ at node i. In expression (1), the weight was the same for all the entries of $\mathbf{S}^l\mathbf{x}$, so that the expression in (11) provides additional degrees of freedom. For notational convenience, the filter coefficients associated with node i are collected into the $L \times 1$ vector $\mathbf{c}_i$, such that $[\mathbf{c}_i]_l = [\mathbf{c}^{(l)}]_i$.

Since $\mathbf{S}^l$ and $\mathrm{diag}(\mathbf{c}^{(l)})$ are not simultaneously diagonalizable (i.e., their eigenvectors are not the same), the frequency interpretation of the filter in (1) does not hold true for (11). However, the spectral decomposition of $\mathbf{S}$ can still be used to understand how the output of the filter at a given node i depends on frequency components of the input. To facilitate exposition, the result is given in the form of a proposition.

Proposition 4: *Define vectors* $\mathbf{u}_i := \mathbf{V}^T\mathbf{e}_i$, $\widehat{\mathbf{c}}_i := \mathbf{\Psi}\mathbf{c}_i$, *and* $\mathbf{h}_i := \mathbf{H}_{\mathrm{nv}}^T\mathbf{e}_i$. *Then, the entries of the i-th row of the filter in* (11) *can be written as*

$$\mathbf{h}_i^T = \mathbf{u}_i^T \mathrm{diag}(\widehat{\mathbf{c}}_i)\mathbf{V}^{-1}, \tag{12}$$

or, alternatively

$$\mathbf{h}_i = (\mathbf{V}^{-1})^T \mathrm{diag}(\mathbf{u}_i)\mathbf{\Psi}\mathbf{c}_i. \tag{13}$$

The expression in (12) reveals that the output of the filter at node i, which can be written as $\mathbf{h}_i^T\mathbf{x}$, can be viewed as an inner product of $\mathbf{V}^{-1}\mathbf{x}$ (the input in the frequency domain) and $\mathbf{u}_i$ (that represents how strongly node i expresses the different frequencies), modulated by $\widehat{\mathbf{c}}_i$ (that is the frequency response associated with the coefficients used by node i). On the other hand, the expression in (13) can be used to design filter coefficients $\{\mathbf{c}_i\}_{i=1}^N$ that implement a given linear transformation $\mathbf{B}$.

IV. Numerical experiments

We first demonstrate that our framework of approximating network operators via graph filters includes the well-studied problem of reaching consensus in finite time (Section IV-A). We then illustrate the utility of this framework by analyzing the error performance when approximating more general linear network operators (Section IV-B).

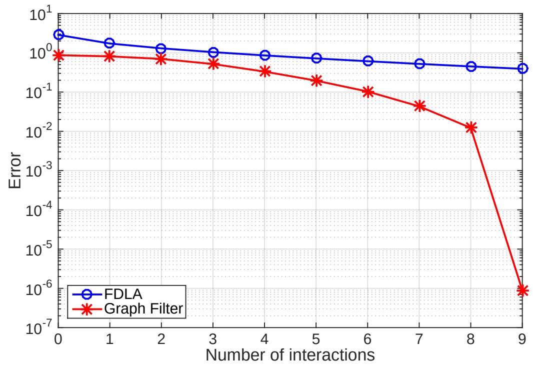
A. *Finite-time consensus*

Consider the particular case of approximating the consensus operator $\mathbf{B}_{\text{con}} = \mathbf{1}\mathbf{1}^T/N$ via local interactions. We compare the performance of two different methods: our finite-time graph-filter approximation and the asymptotic fastest distributed linear averaging (FDLA) in [2]. To asses the performance of the approximation algorithms, we generate 100 undirected and connected Erdös-Rényi graphs with 10 nodes and edge probability 0.2. For each graph, we define the graph-shift operator $\mathbf{S} = \mathbf{W}$ where $\mathbf{W}$ is the solution of the FDLA problem, i.e, $\lim_{k\to\infty}\mathbf{W}^k = \mathbf{B}_{\text{con}}$ with fastest convergence. Moreover, on each graph we define a signal $\mathbf{x}$ drawn from a standard multivariate Gaussian distribution. For a fixed number K of local interactions, we define the FDLA approximation as $\mathbf{x}_{\text{FDLA}}^{(K)} = \mathbf{W}^K\mathbf{x}$ and the graph-filter approximation as $\mathbf{x}_{\text{GF}}^{(K)} = \sum_{l=0}^K h_l^*\mathbf{S}^l\mathbf{x}$ where the filter coefficients $\mathbf{h}^*$ are obtained from (9). Further, we define the error $e_{\text{GF}}^{(K)} = \|\mathbf{x}_{\text{GF}}^{(K)} - \mathbf{B}_{\text{con}}\mathbf{x}\|_2$ and similarly for $e_{\text{FDLA}}^{(K)}$. In Figure 1a we plot the average errors across the 100 graphs as a function of the number of local interactions. The error attained by the graph-filter approach is around one order of magnitude lower than that of the asymptotic approach for intermediate number of interactions and, when $K = N-1 = 9$, perfect recovery is achieved using graph filters (cf. Corollary 1). The performance improvement attained by the graph filters is due to the fact that, for a fixed K, FDLA returns the value of $\mathbf{W}^K\mathbf{x}$ whereas the graph filter returns an optimal linear combination of all $\mathbf{W}^k\mathbf{x}$ for $0 \leq k \leq K$.

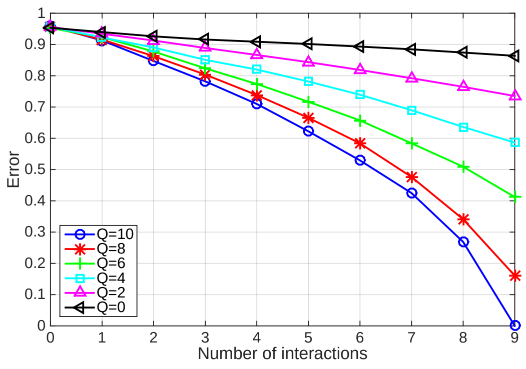
As mentioned in Section I, other finite-time consensus algorithms exist. However, the objective of the current numerical experiment is *not* to show that the introduced method outperforms existing finite-time consensus algorithms but rather to demonstrate that it shares the benefits of finite-time algorithms while being a more general framework, as illustrated in the next section.

B. *General linear network operators*

Our objective is to analyze the quality of the graph-filter approximations for arbitrary linear operators $\mathbf{B}$. To do this, we generate again an undirected and connected Erdös-Rényi graph with 10 nodes and edge probability 0.2, and set the graph-shift operator equal to its adjacency matrix $\mathbf{S} = \mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^H$. For an arbitrary normal operator $\mathbf{B} = \mathbf{V}_\mathbf{B}\mathbf{\Lambda}_\mathbf{B}\mathbf{V}_\mathbf{B}^H$, we denote by Q the number of eigenvectors shared by the bases $\mathbf{V}$ and $\mathbf{V}_\mathbf{B}$. Notice



(a)



(b)

Fig. 1: (a) Error (log-scale) in approximating the consensus operator using graph filters (red) and the fastest asymptotic approach (blue) in [2]. For a given number of interactions, the graph-filter approximation is a linear combination of all the intermediate signals observed, enabling perfect recovery for $N-1$ interactions. (b) Error in approximating general linear operators using graph filters as a function of the filter degree. Parameter Q quantifies the overlap between the eigenvectors of $\mathbf{S}$ and $\mathbf{B}$. As the number of shared eigenvectors decreases, the error for a given number of interactions grows.

that whenever $Q = 10$, $\mathbf{B}$ and $\mathbf{S}$ are simultaneously diagonalizable [cf. condition *i)* in Proposition 1]. We generate 1,000 random linear operators $\mathbf{B}$ for each value of $Q \in \{0, 2, \dots, 10\}$. For a given operator $\mathbf{B}$ and a random graph signal $\mathbf{x}$ drawn from a standard multivariate Gaussian distribution, we quantify the error $e_{\text{GF}}^{(K)}$ for a graph-filter approximation of degree K as explained in the previous section. In Figure 1b we present the average error across the 1,000 approximations as a function of the number of local interactions (filter degree) and parametrized by the value of Q. As expected, whenever $Q = 10$, perfect recovery is achieved for filters of degree $N-1$. Moreover, notice that as the spectral compositions of $\mathbf{B}$ and $\mathbf{S}$ are more different (decreasing values of Q), the error increases, entailing poor approximations even for large number of interactions. This is not surprising since the eigenvectors of a graph filter coincide with those of $\mathbf{S}$ and, hence, differ from those in $\mathbf{B}$, independently of the filter coefficients chosen. This further motivates the development of generalized graph filters (cf. Section III-C) where the eigenvectors of $\mathbf{H}_{\text{nv}}$ and $\mathbf{S}$ need not coincide.

References

[1] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, Inc., 1989.

[2] L. Xiao and S. Boyd, “Fast linear iterations for distributed averaging,” *Systems & Control Letters*, vol. 53, no. 1, pp. 65 – 78, 2004.

[3] S. Kar, J. Moura, and H. Poor, “Distributed linear parameter estimation: asymptotically efficient adaptive strategies,” *SIAM Journal on Control and Optimization*, vol. 51, no. 3, pp. 2200–2229, May 2013.

[4] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, “Consensus in ad hoc WSNs with noisy links - part I: Distributed estimation of deterministic signals,” *IEEE Trans. Signal Process.*, vol. 56, no. 1, pp. 350–364, Jan. 2008.

[5] G. Mateos, J.-A. Bazerque, and G. B. Giannakis, “Distributed sparse linear regression,” *IEEE Trans. Signal Process.*, vol. 58, no. 10, pp. 5262–5276, Oct. 2010.

[6] A. Sayed, S. Barbarossa, S. Theodoridis, and I. Yamada, “Adaptation and learning over complex networks [from the guest editors],” *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 14–15, May 2013.

[7] S. Barbarossa, G. Scutari, and T. Battisti, “Distributed signal subspace projection algorithms with maximum convergence rate for sensor networks with topological constraints,” in *IEEE Intl. Conf. Acoust., Speech and Signal Process. (ICASSP)*, Apr. 2009, pp. 2893–2896.

[8] D. I. Shuman, P. Vandergheynst, and P. Frossard, “Distributed signal processing via Chebyshev polynomial approximation,” *CoRR*, vol. abs/1111.5239, 2011.

[9] X. Wang, P. Liu, and Y. Gu, “Local-set-based graph signal reconstruction,” *IEEE Trans. Signal Process.*, vol. 63, no. 9, pp. 2432–2444, Sept. 2015.

[10] S. Chen, A. Sandryhaila, and J. Kovacevic, “Distributed algorithm for graph signal inpainting,” in *IEEE Intl. Conf. Acoust., Speech and Signal Process. (ICASSP)*, Brisbane, Australia, Apr. 19-24, 2015.

[11] A. Sandryhaila, S. Kar, and J. M. Moura, “Finite-time distributed consensus through graph filters,” in *IEEE Intl. Conf. Acoust., Speech and Signal Process. (ICASSP)*, May 2014, pp. 1080–1084.

[12] S. Safavi and U. Khan, “Revisiting finite-time distributed algorithms via successive nulling of eigenvalues,” *IEEE Signal Process. Lett.*, vol. 22, no. 1, pp. 54–57, Jan. 2015.

[13] S. Segarra, A. G. Marques, G. Leus, and A. Ribeiro, “Interpolation of graph signals using shift-invariant graph filters,” in *European Signal Process. Conf. (EUSIPCO)*, Nice, France, Aug. 31 - Sept. 4, 2015.

[14] A. Sandryhaila and J. Moura, “Discrete signal processing on graphs: Frequency analysis,” *IEEE Trans. Signal Process.*, vol. 62, no. 12, pp. 3042–3054, June 2014.

[15] ——, “Discrete signal processing on graphs,” *IEEE Trans. Signal Process.*, vol. 61, no. 7, pp. 1644–1656, Apr. 2013.

[16] D. Shuman, S. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, “The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains,” *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, Mar. 2013.

[17] C. Godsil and G. Royle, *Algebraic Graph Theory*. Springer-Verlag, Graduate Texts in Mathematics, 2001, vol. 207.