IIR Filtering on Graphs With Random Node-Asynchronous Updates

Oguzhan Teke, *Student Member, IEEE*, and Palghat P. Vaidyanathan, *Life Fellow, IEEE*

***Abstract*—Graph filters play an important role in graph signal processing, in which the data is analyzed with respect to the underlying network (graph) structure. As an extension to classical signal processing, graph filters are generally constructed as a polynomial (FIR), or a rational (IIR) function of the underlying graph operator, which can be implemented via successive shifts on the graph. Although the graph shift is a localized operation, it requires all nodes to communicate synchronously, which can be a limitation for large scale networks. To overcome this limitation, this study proposes a node-asynchronous implementation of rational filters on arbitrary graphs. In the proposed algorithm nodes follow a randomized collect-compute-broadcast scheme: if a node is in the passive stage it collects the data sent by its incoming neighbors and stores only the most recent data. When a node gets into the active stage at a random time instance, it does the necessary filtering computations locally, and broadcasts a state vector to its outgoing neighbors. For the analysis of the algorithm, this study first considers a general case of randomized asynchronous state recursions and presents a sufficiency condition for its convergence. Based on this result, the proposed algorithm is proven to converge to the filter output in the mean-squared sense when the filter, the graph operator and the update rate of the nodes satisfy a certain condition. The proposed algorithm is simulated using rational and polynomial filters, and its convergence is demonstrated for various different cases, which also shows the robustness of the algorithm to random communication failures.**

***Index Terms*—Graph signal processing, graph filters, fixed point iteration, randomized iterations, node asynchronicity.**

I. Introduction

In THE recent area of graph signal processing [1]–[4], the data at hand is modeled with respect to a network structure, in which the underlying graph is assumed to represent the dependency between the data points. In order to analyze such network structured models, classical signal processing techniques have been extended to the case of graphs. In particular, the analysis is based on the "graph operator," whose eigenvectors serve as the graph Fourier basis (GFB). With the use of GFB, sampling, reconstruction, multirate processing of graph signals and some uncertainty results have been extended to the case of graphs in [5]–[14].

One important aspect of graph signal processing is the use of graph filters, which can be utilized in order to smooth out graph signals (low-pass filters), or detect anomalies (high-pass filters) [2]. Similar to the classical signal processing, graph filters can be constructed in two different forms: finite impulse response (FIR), or infinite impulse response (IIR). The FIR case corresponds to a matrix polynomial of the given graph operator [3]–[5]. It is well-known that a polynomial graph filter of order L is localized on the graph, that is, nodes are required to communicate only with its L-hop neighbors in order to implement the filter. For this reason it is very natural to think of polynomial graph filtering as a way of distributed signal processing, in which the low-order polynomials are favored to keep the communications localized. The papers [15]–[18] (and references therein) made explicit connections between polynomial graph filters and distributed computation, and studied various problems including smoothing, regularization, and consensus.

In the IIR case, the graph filter is constructed with respect to a rational function rather than a polynomial. It should be noted that an IIR graph filter can be equivalently represented as an FIR graph filter (possibly with a very high order) due to the finite spectrum of the graph operator. Nevertheless, IIR filters are still useful to consider since they can provide better approximations for a given filter specifications. When extended to the case of graphs, an IIR filter of order L can be implemented via iterative procedures that preserve the locality of the communications. The studies in [19]–[23] analyzed the convergence behavior of such filters and showed successful applications on graph signals with distributed processing.

Although both polynomial and rational graph filters can be implemented in a distributed fashion, aforementioned implementations are based on successive graph shifts (multiplication with the graph operator). Although the graph shift can be implemented via data exchange with the neighboring nodes, it requires all the nodes to communicate simultaneously. That is, all the nodes should send and receive data at the same time instance, or nodes should wait until all the communications are terminated before proceeding to the next iteration (shift). Synchronization becomes an important limitation when the size of the network, N, is large, e.g. distributed large-scale graph processing frameworks [24]–[27], or the network has autonomous behavior without a centralized control.

In order to eliminate the need for synchronization, this study proposes a node-asynchronous implementation of an arbitrary rational filter (including FIR) on an *arbitrary* graph. In the proposed algorithm neighboring nodes send and receive a vector variable (state vector) whose size is determined by the order of the filter, and the nodes follow a collect-compute-broadcast

Manuscript received June 23, 2019; revised December 30, 2019 and April 22, 2020; accepted June 22, 2020. Date of publication June 26, 2020; date of current version July 10, 2020. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Pierre Borgnat. This work was supported in part by the Office of Naval Research under Grant N00014-18-1-2390, in part by the National Science Foundation under Grant CCF-1712633, and in part by the Carver Mead Research Seed Fund of the California Institute of Technology. *(Corresponding author: Oguzhan Teke.)*

The authors are with the Department of Electrical Engineering, California Institute of Technology, Pasadena, CA 91125 USA (e-mail: oteke@caltech.edu; ppvnath@systems.caltech.edu).

Digital Object Identifier 10.1109/TSP.2020.3004912

1053-587X © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on July 02,2021 at 16:04:36 UTC from IEEE Xplore. Restrictions apply.

framework. More precisely, the algorithm consists of two main stages: passive and active. In the passive stage, a node receives and stores the data (local state vectors) sent by its incoming neighbors. When a node gets into the active stage at a *random* time instance, it completes the necessary filtering calculations (local state recursions), and then broadcasts its most recent state vector to its outgoing neighbors. Thus, nodes behave asynchronously on the network. By carefully designing the computation scheme, the proposed algorithm is proven to converge to the desired filtered signal in the mean-squared sense under mild stability conditions.

A. Relations With Asynchronous Fixed Point Iterations

In this study, the analysis of the algorithm will be based on the convergence properties of randomized asynchronous linear fixed point iterations (state recursions). We note that non-random asynchronous fixed point iterations are well studied problems in the literature [28]–[31], which considered more general nonlinear update models. For the linear model (which is the case in this study), the earliest analysis can be traced back to the study in [28] that provided the necessary and sufficient condition under which the asynchronous iterations are guaranteed to converge for any index sequence. More recently, studies in [32], [33] (and references therein) studied the randomized variations of asynchronous iterations, in which indices are assumed to be selected with equal probabilities, and they provided sufficiency conditions for the convergence. Asynchronous iterations are considered also in the context of semi-supervised learning on graphs [34], [35].

In the case considered in this study, the indices are allowed to be selected with non-equal probabilities during the asynchronous recursions. More importantly, the possibility of updating different number of indices in each iteration (which can be considered as partial synchrony) is also not ruled out. In fact, convergence analysis of a similar setting is studied in [36], [37] for the case of zero-input, in which the system is assumed to have a unit eigenvalue, and the iterand is proven to converge to a point in the eigenspace of the unit eigenvalue when all the indices are updated with equal probabilities. On the contrary, the model considered here starts with the assumption that the system does *not* have a unit eigenvalue, and it further assumes that the input is a nonzero constant, so there is a unique nonzero fixed point. This study also considers the effect of the input noise. For this setting, we prove that the Schur diagonal stability of the system matrix (which is more relaxed than the condition given in [28], [33]) is sufficient for the convergence of the randomized asynchronous iterations in the mean-squared sense.

B. Outline and Contributions

This study consists of two main parts. The first part (Section II) considers the analysis of the randomized asynchronous state recursions in arbitrary linear systems, of which synchronous non-random recursions are special-cases and all results continue to be applicable. The second part (Sections III and IV) focuses on the specific case of graphs and considers a node-asynchronous implementation of rational graph filters. More precisely, in Section II, we introduce the randomized asynchronous model for the state recursions and present the first main result (Theorem 1) that provides upper and lower bounds for the mean-squared error of the randomized iterations. Based on this result, we provide a sufficient condition (Corollary 1) that ensures the convergence of the iterations. Then, we prove that the presented condition is more relaxed than the well-known necessary condition for the convergence of the non-random asynchronous iterations (Lemma 1). The special case of uniform index-selection probabilities is also considered (Corollary 2). In Section III, we propose a node-asynchronous implementation of a graph filter (Algorithm 1) and describe its behavior. Then, in Section IV we prove the convergence of the proposed algorithm to the desired filtered signal in the mean-squared sense for both synchronous and asynchronous cases (Theorems 2 and 3). Finally in Section V, we simulate the proposed algorithm for various different graph filters (including rational and polynomial) and demonstrate the convergence behavior of the algorithm numerically. A preliminary version of this study was presented in [38]. Some extensions of these results are presented in [39].

We note that results presented in this study allows the graph to have directed edges possibly with a non-diagonalizable operator. It is also important to point out that this study does not consider the design of graph filters. The main focus here is a node-asynchronous implementation of a *given* graph filter.

C. Notation

We will use $\mathbb{E}[\cdot]$ to denote the expectation. We will use $\succ$ and $\succeq$ to denote the positive definite (PD) and the positive semi-definite (PSD) ordering, respectively. For a matrix $\mathbf{X}$ possibly with complex entries, we will use $\mathbf{X}^{\mathrm{T}}$ and $\mathbf{X}^{\mathrm{H}}$ to denote its transpose and conjugate transpose, respectively; $\mathrm{tr}(\mathbf{X})$ to denote its trace; $\sigma_{\min}(\mathbf{X})$ and $\sigma_{\max}(\mathbf{X}) = \|\mathbf{X}\|_2$ to denote its the smallest and the largest singular values, respectively; $\rho(\mathbf{X})$ to denote the spectral radius (the largest eigenvalue in magnitude). When $\mathbf{X}$ is a Hermitian matrix, we will use $\lambda_{\min}(\mathbf{X})$ and $\lambda_{\max}(\mathbf{X})$ to denote its the smallest and the largest eigenvalues, respectively. We will use $|\mathbf{X}|$ to denote the matrix obtained by replacing the elements of $\mathbf{X}$ by their absolute values. For a matrix $\mathbf{X} \in \mathbb{C}^{M\times N}$, we will use $\mathrm{vec}(\mathbf{X}) \in \mathbb{C}^{MN}$ to denote the vector obtained by stacking the columns of $\mathbf{X}$. We will use $\otimes$ to denote the Kronecker product, which has the following mixed-product property:

$$(\mathbf{A}\otimes\mathbf{B})(\mathbf{X}\otimes\mathbf{Y}) = (\mathbf{A}\mathbf{X})\otimes(\mathbf{B}\mathbf{Y}) \tag{1}$$

for matrices $\mathbf{A}, \mathbf{B}, \mathbf{X}, \mathbf{Y}$ with conforming sizes.

We will use $\mathcal{T}$ to denote a subset of $\{1,\ldots,N\}$. Given a subset $\mathcal{T}$, its corresponding index-selection matrix will be denoted as $\mathbf{P}_{\mathcal{T}} \in \mathbb{R}^{N\times N}$, which is a diagonal matrix that has value 1 only at the indices specified by the set $\mathcal{T}$. That is,

$$\mathbf{P}_{\mathcal{T}} = \sum_{i\in\mathcal{T}} \mathbf{e}_i\,\mathbf{e}_i^{\mathrm{H}}, \qquad \text{and} \qquad \mathrm{tr}(\mathbf{P}_{\mathcal{T}}) = |\mathcal{T}|, \tag{2}$$

where $\mathbf{e}_i \in \mathbb{R}^N$ is the i^{th} standard vector that has 1 at the i^{th} index and 0 elsewhere, and $|\mathcal{T}|$ denotes the size of $\mathcal{T}$.

II. Asynchronous State Recursions

Given a matrix $\mathbf{S} \in \mathbb{C}^{N\times N}$ and a constant input signal $\mathbf{u} \in \mathbb{C}^N$, we will consider the following type of recursion on the state vector $\mathbf{x}_k \in \mathbb{C}^N$:

$$\mathbf{x}_k = \mathbf{S}\,\mathbf{x}_{k-1} + \mathbf{u}_{k-1}, \tag{3}$$

where $\mathbf{x}_0$ denotes the initial value of the state vector, and $\mathbf{u}_k$ denotes the noisy input signal. That is,

$$\mathbf{u}_k = \mathbf{u} + \mathbf{w}_k, \tag{4}$$

Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on July 02,2021 at 16:04:36 UTC from IEEE Xplore. Restrictions apply.

where $\mathbf{w}_k$ is the noise term with the following statistics:

$$\mathbb{E}[\mathbf{w}_k] = \mathbf{0}, \qquad \mathbb{E}\left[\mathbf{w}_k\ \mathbf{w}_s^{\mathrm{H}}\right] = \delta(k-s)\ \mathbf{\Gamma}, \tag{5}$$

where $\delta(\cdot)$ denotes the discrete Dirac delta function, and $\mathbf{\Gamma}$ is allowed to be non-diagonal.

In the noise-free case, i.e., $\mathbf{\Gamma} = \mathbf{0}$, the fixed point of the recursion in (3) is given as follows:

$$\mathbf{x}^\star = (\mathbf{I} - \mathbf{S})^{-1}\ \mathbf{u}, \tag{6}$$

which requires $\mathbf{S}$ *not* to have eigenvalue 1 so that $\mathbf{I} - \mathbf{S}$ is invertible. In order to analyze the convergence behavior, we first define the residual (error) vector as follows:

$$\mathbf{r}_k = \mathbf{x}_k - \mathbf{x}^\star. \tag{7}$$

By substituting (7) into the state recursion in (3), the residual $\mathbf{r}_k$ can be written explicitly as follows:

$$\mathbf{r}_k = \mathbf{S}^k\ \mathbf{r}_0 + \sum_{n=0}^{k-1} \mathbf{S}^n\ \mathbf{w}_{k-1-n}. \tag{8}$$

Due to the fact that $\mathbf{w}_k$'s are uncorrelated in different iterations and have zero-mean, the expected squared ℓ_2-norm of the residual $\mathbf{r}_k$ can be written as follows:

$$\mathbb{E}\left[\|\mathbf{r}_k\|_2^2\right] = \left\|\mathbf{S}^k\ \mathbf{r}_0\right\|_2^2 + \mathrm{tr}\left(\sum_{n=0}^{k-1} \mathbf{S}^n\ \mathbf{\Gamma}\ (\mathbf{S}^n)^{\mathrm{H}}\right). \tag{9}$$

It is clear from (9) that when $\mathbf{S}$ is a stable matrix, i.e., when the following holds true:

$$\rho(\mathbf{S}) < 1, \tag{10}$$

the error term in (9) approaches an error floor. More precisely,

$$\lim_{k\to\infty} \mathbb{E}[\mathbf{r}_k] = \mathbf{0}, \quad \text{and} \quad \lim_{k\to\infty} \mathbb{E}\left[\|\mathbf{r}_k\|_2^2\right] = \mathrm{tr}(\mathbf{\Upsilon}\,\mathbf{\Gamma}), \tag{11}$$

where $\mathbf{\Upsilon}$ is given as follows:

$$\mathbf{\Upsilon} = \mathbf{I} + \sum_{n=1}^{\infty} (\mathbf{S}^n)^{\mathrm{H}}\ \mathbf{S}^n, \tag{12}$$

which converges due to (10). (See [40, Appendix D].)

In the noise-free case ($\mathbf{\Gamma} = \mathbf{0}$), the limit in (11) implies the convergence of $\mathbf{x}_k$ to $\mathbf{x}^\star$. On the other hand, in the case of an unstable transition matrix $\mathbf{S}$, i.e., $\rho(\mathbf{S}) \geq 1$, the mean-squared error is bounded away from zero even in the noise-free case. Therefore, the condition in (10) is both sufficient and necessary for the convergence of the state recursions in (3). This is, in fact, a well-known result from the linear system theory [40].

In the context of graph signal processing [1]–[4], the matrix $\mathbf{S}$ is assumed to be a local graph operator (shift matrix) on the graph of interest. Thus, an iteration in the form of (3) can be implemented on the graph as a data exchange between the neighboring nodes. That is, (3) can be written as follows:

$$(\mathbf{x}_k)_i = \sum_j S_{i,j}\ (\mathbf{x}_{k-1})_j + (\mathbf{u}_{k-1})_i, \tag{13}$$

for all nodes i in $1 \leq i \leq N$. In this setting, $\mathbf{u}$ is considered as a signal defined on the graph, where the nodes will be the "domain" analogous to time. The index k will denote the round of communication, so the graph signal $\mathbf{u}$ does not depend on the iteration index k. Note that the noisy measurement $\mathbf{u}_k = \mathbf{u} + \mathbf{w}_k$ depends on k.

Although the individual nodes can perform the updates of (13) locally, such an implementation requires a synchronization mechanism among the nodes. That is, all the nodes should send and receive data at the same time instance, or nodes should wait until all the communications are terminated before proceeding to the next iteration. Synchronization becomes an important limitation when the size of the network, N, is large, or the network has autonomous behavior, in which case there is no centralized control over the network.

In order to overcome the need for synchronization, in this study we will consider a randomized asynchronous variation of the state recursion in (3), in which only a random subset of indices are updated simultaneously and the remaining ones stay unchanged. More precisely, we consider the following update model:

$$(\mathbf{x}_k)_i = \begin{cases} (\mathbf{S}\,\mathbf{x}_{k-1})_i + (\mathbf{u}_{k-1})_i, & i \in \mathcal{T}_k, \\ (\mathbf{x}_{k-1})_i, & i \notin \mathcal{T}_k, \end{cases} \tag{14}$$

where $\mathcal{T}_k$ denotes the set of indices updated at the k^{th} iteration.

For non-random variants of the model in (14), the study [28] assumed that only one index is updated per iteration and allowed the use of the past values of the iterant, that is, $\mathbf{x}_k$ may depend on $\{\mathbf{x}_{k-1}, \ldots, \mathbf{x}_{k-s}\}$ for some fixed s. Thus, a noise-free and non-random version of (14) with $|\mathcal{T}_k| = 1$ corresponds to the model considered in [28] with $s = 1$, for which the following condition is shown to be both necessary and sufficient for the convergence of the iterations (see [28, Section 5] and [29, Section 3.2]):

$$\rho(|\mathbf{S}|) < 1. \tag{15}$$

In words, if (15) is satisfied, the iterations converge for any index sequence in which no index is left out. On the contrary, if (15) is violated, then *there exists* an index sequence for which the iterations do not convergence.

The case of randomized index-selection was also studied more recently in [32], [33] for the solution of N linear equations with N unknowns. These studies focused also on the case of $|\mathcal{T}_k| = 1$ (updating only one index per iteration) and showed that the following condition:

$$\|\mathbf{S}\|_2 < 1, \tag{16}$$

is sufficient (with some additional assumptions on $\mathbf{S}$) to ensure the convergence of the iterations. We refer to [32, Lemma 3.1] and [33, Section 2.1] for the precise details.

In the randomized asynchronous model considered in this study, we allow the case of updating more than one index per iteration possibly with indices having non-uniform selection probabilities. In the next subsection, we will elaborate on the statistical properties of the index-selection model and define the average index-selection matrix, which will play an important role in the convergence analysis of the iterations.

A. Random Selection of the Update Sets

In the asynchronous model we consider in (14), the update set $\mathcal{T}_k$ is assumed to be selected randomly and independently among all possible 2^N different subsets of $\{1, \ldots, N\}$ in every iteration of (14). However, we would like to emphasize that the independent selection of the update sets *do not* necessarily imply independent selection of the indices. Thus, the model considered here allows correlated index-selection schemes. We also note that both the content and the size of $\mathcal{T}_k$ are random variables. We do not assume that $\mathcal{T}_k$'s have identical distributions at every

Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on July 02,2021 at 16:04:36 UTC from IEEE Xplore. Restrictions apply.

iteration k. Nevertheless, we do assume that the distribution of $\mathcal{T}_k$ is first-order stationary in the following sense: expectation of the index-selection matrix $\mathbf{P}_{\mathcal{T}_k}$ does not depend on k. More precisely,

$$\mathbb{E}\left[\mathbf{P}_{\mathcal{T}_k}\right] = \mathbf{P} \qquad \forall\, k. \tag{17}$$

In the rest of the paper, the matrix $\mathbf{P} \in \mathbb{R}^{N\times N}$ will be referred to as the *average index (node) selection matrix*, which is a deterministic and diagonal matrix satisfying the following:

$$\mathbf{0} \prec \mathbf{P} \preceq \mathbf{I}, \tag{18}$$

where the positive definiteness follows from the fact that no index is left out (on average) in the update scheme of (14). We also note that $\mathrm{tr}(\mathbf{P}) = \mathbb{E}[\,|\mathcal{T}_k|\,]$ corresponds to the average number of indices updated per iteration.

B. Convergence in the Mean-Squared Sense

It is easily verified that the fixed point of the randomized model (14) continues to be $\mathbf{x}^\star$ given in (6). Therefore, the vector $\mathbf{r}_k$ defined in (7) represents the residual for the randomized asynchronous model as well. Thus, the convergence of $\mathbf{r}_k$ to zero implies the convergence of $\mathbf{x}_k$ to the fixed point $\mathbf{x}^\star$. However, $\mathbf{r}_k$ is a random variable in the asynchronous case due to the random selection of the indices. The following theorem, whose proof is presented in Appendix A, provides bounds on the mean-squared error as follows:

Theorem 1: In the randomized asynchronous model (14), the mean-squared error can be bounded as follows:

$$\mathbb{E}\left[\|\mathbf{r}_k\|_2^2\right] \le \Psi^k \|\mathbf{r}_0\|_2^2 + \frac{1-\Psi^k}{1-\Psi}\,\mathrm{tr}(\mathbf{P}\,\mathbf{\Gamma}), \tag{19}$$

$$\mathbb{E}\left[\|\mathbf{r}_k\|_2^2\right] \ge \psi^k \|\mathbf{r}_0\|_2^2 + \frac{1-\psi^k}{1-\psi}\,\mathrm{tr}(\mathbf{P}\,\mathbf{\Gamma}), \tag{20}$$

where

$$\psi = \lambda_{\min}(\mathbf{I} + \mathbf{S}^{\mathrm{H}}\,\mathbf{P}\,\mathbf{S} - \mathbf{P}), \quad \Psi = \lambda_{\max}(\mathbf{I} + \mathbf{S}^{\mathrm{H}}\,\mathbf{P}\,\mathbf{S} - \mathbf{P}). \tag{21}$$

Regarding the bounds in (19) and (20) we first note that the inequality in (18) implies $\psi \ge 0$, hence $\Psi \ge 0$, irrespective of the values of $\mathbf{S}$ and $\mathbf{P}$. As a result the expressions on the right-hand-side of (19), (20) are positive and finite. However, Theorem 1 by itself does not ensure the convergence of the iterations as the values of ψ and Ψ can be larger than or equal to 1 for some values of $\mathbf{S}$ and $\mathbf{P}$. The following corollary presents a *sufficiency* condition that ensures the convergence of the randomized iterations of (14) in the mean-squared sense up to an error floor depending on the amount of input noise:

Corollary 1: If the state transition matrix $\mathbf{S}$ and the average index-selection matrix $\mathbf{P}$ satisfy the following:

$$\mathbf{S}^{\mathrm{H}}\,\mathbf{P}\,\mathbf{S} \prec \mathbf{P}, \tag{22}$$

then, the limit of the mean squared error of the asynchronous model in (14) is bounded as follows:

$$\frac{\mathrm{tr}(\mathbf{P}\,\mathbf{\Gamma})}{\lambda_{\max}\left(\mathbf{P} - \mathbf{S}^{\mathrm{H}}\,\mathbf{P}\,\mathbf{S}\right)} \le \lim_{k\to\infty} \mathbb{E}\left[\|\mathbf{r}_k\|_2^2\right] \le \frac{\mathrm{tr}(\mathbf{P}\,\mathbf{\Gamma})}{\lambda_{\min}\left(\mathbf{P} - \mathbf{S}^{\mathrm{H}}\,\mathbf{P}\,\mathbf{S}\right)}. \tag{23}$$

Proof: The assumption (22) implies that ψ and Ψ defined in (21) satisfy the inequality $0 \le \psi \le \Psi < 1$. Then, the bounds in (23) follow directly from Theorem 1.

A number of remarks are in order:

1) *Update Probabilities and Convergence:* The convergence of the iterations depends on the matrix $\mathbf{S}$ as well as the average index-selection matrix $\mathbf{P}$. Thus, the random asynchronous iterations running on a given matrix $\mathbf{S}$ may not converge for an arbitrary set of update probabilities, yet the convergence can still be achieved for specific sets of probabilities. The question of whether *there exists* a $\mathbf{P}$ satisfying (22) or not for a given $\mathbf{S}$ will be discussed in the next section.
2) *Error Floor:* The lower bound in (23) reveals an error floor: no matter how many iterations are used, the expected residual error is always bounded away from zero in the presence of noise ($\mathbf{\Gamma} \neq \mathbf{0}$), which is also the case in synchronous iterations as seen in (11). Nevertheless, (23) shows that the error floor is bounded linearly by the noise covariance matrix.
3) *Convergence Rate and Index Selection:* It should be noted from Theorem 1 that the rate of convergence as well as the error floor depend on the average index-selection matrix $\mathbf{P}$. That is to say, some set of index-selection probabilities may yield a faster rate of convergence or a lower error floor, for which we provide a numerical evidence in Section V (See Figs. 5, 8). However, their theoretical analysis will be considered in a later study.
4) *Sufficiency:* It is important to emphasize that the condition (22) is only sufficient but not necessary to ensure the convergence of the randomized asynchronous iterations. When (22) does not hold true, it merely means that the upper bound dictated by Theorem 1 diverges in the limit, which makes the theorem inconclusive regarding the convergence. The non-necessity of the condition (22) will be numerically verified later in Section V-C. Nevertheless, the importance of the sufficient condition (22) follows from the fact that it does not have any additional assumption on the matrix $\mathbf{S}$: it may have complex values, may be non-Hermitian, and it may even be non-diagonalizable. When the graph operators are considered in Sections III and IV, this will be very important to ensure the convergence of filters on *an arbitrary directed graph*.

In the following Sections II-C and II-D, we will elaborate on the condition (22) as well as the implications of Corollary 1. If desired, the reader can skip these two subsections and jump to Section III directly, where we present an asynchronous implementation of IIR graph filters.

C. On the Schur Diagonal Stability

In addition to the convergence results presented here for the randomized asynchronous state recursions, the mathematical condition (22) appears in various different contexts. For example, an implementation of a digital filter is guaranteed to be free from limit cycles (overflow oscillation) when the transition matrix $\mathbf{S}$ of its realization satisfies (22) for some $\mathbf{P}$ [41], [42]. (In fact, [42] requires $\mathbf{S}^{\mathrm{H}}\mathbf{P}\mathbf{S} \preceq \mathbf{P}$ only.) Moreover, the study in [43] showed that a condition in the form of (22) is sufficient to ensure the convergence of time-varying block asynchronous iterations.

Due to its importance in various different application, the condition (22) and its variations have been studied extensively in the literature. In fact, the condition was first referred to as diagonal

Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on July 02,2021 at 16:04:36 UTC from IEEE Xplore. Restrictions apply.

stability in [44]. Later in [45], the term was revised as Schur diagonal stability in order to distinguish the discrete and the continuous counterparts. (See [45, Definitions 2.1.3 and 2.5.2].) More precisely:

Definition 1: A matrix $\mathbf{S} \in \mathbb{C}^{N\times N}$ is said to be Schur diagonally stable (alternatively, $\mathbf{S} \in \mathcal{D}_{\mathrm{d}}$) if and only if there exists a positive diagonal matrix $\mathbf{P}$ such that $\mathbf{S}^{\mathrm{H}}\,\mathbf{P}\,\mathbf{S} - \mathbf{P} \prec \mathbf{0}$.

Unlike the stability condition (10) that depends only on the eigenvalues of a matrix, the Schur diagonal stability of a matrix cannot be decided just by its eigenvalues in the sense that among two *similar* matrices one may be Schur diagonally stable and the other may not [42], [45]. Furthermore, Schur diagonal stability is more restrictive than stability, but more relaxed than (15) as shown by the following lemma (whose proof is provided in Appendix B):

Lemma 1: The following hold true for any $\mathbf{S} \in \mathbb{C}^{N\times N}$:

$$\rho(|\mathbf{S}|) < 1 \implies \mathbf{S} \in \mathcal{D}_{\mathrm{d}} \implies \rho(\mathbf{S}) < 1. \tag{24}$$

Furthermore,

$$\|\mathbf{S}\|_2 < 1 \implies \mathbf{S} \in \mathcal{D}_{\mathrm{d}}. \tag{25}$$

We also note that the converse of the implications in (24) and (25) do not hold in general. We refer to [45, Section 2] (and references therein) for an elaborate compilation of properties of the diagonal stability.

Two remarks are in order:

1) *Random vs Non-Random Iterations:* We would like to point out that Corollary 1 together with Lemma 1 does not contradict the well-known result of [28] that showed the necessity of the condition $\rho(|\mathbf{S}|) < 1$ for the convergence of non-random asynchronous iterations. The key difference between Corollary 1 and [28] is the notion of convergence. The study [28] ensures the convergence of the iterations for *any index sequence*, whereas Corollary 1 considers the convergence in the mean-squared sense. When $\rho(|\mathbf{S}|) \geq 1$, there exists an index sequence for which iterations do not converge, yet the iterations do converge in the mean-squared sense if the indices can be updated with appropriate probabilities, i.e., the condition (22) of Corollary 1 is satisfied.
2) *Numerical Search:* Schur diagonal stability of a given matrix can be verified via the following semi-definite program:

$$\min_{c,\,\mathbf{p}} \quad c \qquad \text{s.t.} \qquad \begin{matrix} c\,\mathbf{I} \succeq \mathbf{S}^{\mathrm{H}}\,\mathrm{diag}(\mathbf{p})\,\mathbf{S} - \mathrm{diag}(\mathbf{p}), \\ \mathbb{1} \geq \mathbf{p} \geq \mathbf{0}, \end{matrix} \tag{26}$$

where $\mathbb{1}$ denotes the vector with all ones. More precisely, it can be shown that the optimal value of (26) satisfies $c^\star < 0$ if and only if the matrix $\mathbf{S}$ is Schur diagonally stable. Thus, the strict negativity of the numerical solution of (26) for a given matrix $\mathbf{S}$ determines the Schur diagonal stability of $\mathbf{S}$.

D. The Case of Uniform Probabilities

The sufficiency condition given by Corollary 1 involves both the matrix $\mathbf{S}$ and the average index-selection matrix $\mathbf{P}$. In many practical scenarios, the indices (or the update sets) are selected with uniform probabilities, in which case implications of Theorem 1 can be simplified further as we discuss next.

When the indices are equally likely to be updated in each iteration of (14), the average index-selection matrix becomes a scaled identity matrix. More precisely,

$$\mathbf{P} = p\,\mathbf{I}, \qquad \text{where} \qquad 0 < p \leq 1. \tag{27}$$

In general, it is possible to use different stochastic models for the selection of the update sets whose average index-selection matrix is in the form of (27). For example, when a subset of size T is selected uniformly randomly among all possible $\binom{N}{T}$ different subsets, the average index-selection matrix becomes $\mathbf{P} = (T/N)\,\mathbf{I}$. Notice that the case of $T = 1$ corresponds to the selection of only one index uniformly randomly per iteration. It is also possible to select subsets of different sizes, which is considered in [36], [37], [46], [47].

When the matrix $\mathbf{P}$ has the form in (27), the rate parameters in (21) given by Theorem 1 reduce to the following form:

$$\psi = p\,\sigma^2_{\min}(\mathbf{S}) + 1 - p, \qquad \Psi = p\,\sigma^2_{\max}(\mathbf{S}) + 1 - p, \tag{28}$$

which shows that the singular values of the matrix $\mathbf{S}$ bound the rate of convergence of the iterations of (14). As a result, the matrix $\mathbf{S}$ having a bounded spectral norm is sufficient to ensure the convergence of the randomized asynchronous iterations, which is formally presented in the following corollary:

Corollary 2: If $\|\mathbf{S}\|_2 < 1$ and the indices are updated with equal probabilities in the random asynchronous model of (14), then the limit of the mean squared error is bounded as follows:

$$\frac{\mathrm{tr}(\mathbf{\Gamma})}{1-\sigma^2_{\min}(\mathbf{S})} \leq \lim_{k\to\infty} \mathbb{E}\left[\|\mathbf{r}_k\|_2^2\right] \leq \frac{\mathrm{tr}(\mathbf{\Gamma})}{1-\sigma^2_{\max}(\mathbf{S})}. \tag{29}$$

Proof: If the indices are updated with equal probabilities, the average index-selection matrix $\mathbf{P}$ is in the form of (27), thus the condition (22) of Corollary 1 reduces to $\mathbf{S}^{\mathrm{H}}\mathbf{S} \prec \mathbf{I}$, which is readily satisfied due to the assumption $\|\mathbf{S}\|_2 < 1$. Then, we can apply Corollary 1. The use of (28) in Theorem 1 gives the bounds in (29).

Some remarks are in order:

1) *Convergence Irrespective of the Update Probability:* Unlike the condition presented by Corollary 1, when the indices are updated with equal probabilities, the sufficiency condition given by Corollary 2 involves only the matrix $\mathbf{S}$. Therefore, if the condition (bounded spectral norm) is met, the convergence is ensured irrespective of the actual value of the average index-selection matrix $\mathbf{P}$. However, the rate of convergence does depend on $\mathbf{P}$ in general as suggested by (28), which will be verified numerically as well in Section V.
2) *Noise Amplification:* When the indices are equally likely to be selected in the random asynchronous iterations, there is an amplification to the input noise. This observation follows simply from the assumption $\|\mathbf{S}\|_2 < 1$ that implies $1/(1-\sigma^2_{\min}(\mathbf{S})) \geq 1$. Thus, the lower bound in (29) can be further lower bounded with $\mathrm{tr}(\mathbf{\Gamma}) = \mathbb{E}[\|\mathbf{w}_k\|_2^2]$, which shows that the error floor is always larger than the amount of input noise. This behavior of the random asynchronous iterations is consistent with the synchronous counterpart. The error floor of the synchronous iterations given in (11) can be lower bounded as $\mathrm{tr}(\mathbf{\Upsilon}\,\mathbf{\Gamma}) \geq \mathrm{tr}(\mathbf{\Gamma})$ since the matrix $\mathbf{\Upsilon}$ in (12) satisfies $\mathbf{\Upsilon} \succeq \mathbf{I}$.
3) *Nonstationary Noise Covariance:* We note that the input noise need not have a stationary distribution for Theorem 1, Corollary 1, and Corollary 2 to be valid. As long as the noise covariance matrix is upper bounded as $\mathbb{E}[\mathbf{w}_k\,\mathbf{w}_k^{\mathrm{H}}] \preceq \mathbf{\Gamma}$ for all k, the corresponding upper bounds

Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on July 02,2021 at 16:04:36 UTC from IEEE Xplore. Restrictions apply.

remain valid. Similarly, the corresponding lower bounds are valid as long as the covariance matrix is lower bounded as $\mathbb{E}[\mathbf{w}_k\,\mathbf{w}_k^{\mathrm{H}}] \succeq \mathbf{\Gamma}$ for all k.

III. Asynchronous Rational Filters on Graphs

In this section, we will consider a node-asynchronous implementation of a rational graph filter that is specified as follows:

$$h(x) = p(x)\,/\,q(x), \tag{30}$$

where the polynomials $p(x)$ and $q(x)$ are of degree (at most) L, and they are assumed to be in the following form:

$$p(x) = \sum_{n=0}^{L} p_n\, x^n, \qquad q(x) = 1 + \sum_{n=1}^{L} q_n\, x^n. \tag{31}$$

The coefficients are allowed to be complex in general, i.e., $p_n, q_n \in \mathbb{C}$. In particular, polynomial graph filters, which corresponds to the case of $q_1 = \cdots = q_L = 0$, are not excluded.

A. Rational Graph Filters

In the following we will use $\mathbf{G} \in \mathbb{C}^{N\times N}$ to denote a graph operator for the graph with N nodes. Here $G_{i,j}$ denotes the weight of the edge from node j to node i. In particular, $G_{i,j} = 0$ when nodes i and j are not neighbors. Examples of such local graph operators include the adjacency matrix, the graph Laplacian, etc. *The graph is allowed to be directed possibly with a non-diagonalizable adjacency matrix.* We will use $\mathcal{N}_{\text{in}}(i)$ and $\mathcal{N}_{\text{out}}(i)$ to denote the incoming and outgoing neighbors of the node i. More precisely we have:

$$\mathcal{N}_{\text{in}}(i) = \{j \mid G_{i,j} \neq 0\}, \qquad \mathcal{N}_{\text{out}}(i) = \{j \mid G_{j,i} \neq 0\}. \tag{32}$$

For a given graph operator $\mathbf{G} \in \mathbb{C}^{N\times N}$, the rational graph filter corresponding to (30) has the following form:

$$h(\mathbf{G}) = p(\mathbf{G})\; q(\mathbf{G})^{-1}, \tag{33}$$

where we implicitly assume that $q(\mathbf{G})$ is an invertible matrix.

When $\mathbf{u} \in \mathbb{C}^N$ is a signal on the graph, we will use $\widetilde{\mathbf{u}}$ to denote the filtered version of $\mathbf{u}$ with the filter $h(\mathbf{G})$. That is,

$$\widetilde{\mathbf{u}} = h(\mathbf{G})\,\mathbf{u}, \tag{34}$$

where $\mathbf{u}$ is the given signal on the graph.

A special case of rational graph filtering corresponds to Laplacian smoothing [15], [48], [49]. More precisely, given an undirected graph with the Laplacian matrix $\mathbf{L}$ and a signal $\mathbf{u}$ on the graph, the Laplacian smoothing is obtained as the solution of the following regularized least-squares problem:

$$\widetilde{\mathbf{u}} = \arg\min_{\boldsymbol{\xi}} \|\mathbf{u} - \boldsymbol{\xi}\|_2^2 + \gamma\, \boldsymbol{\xi}^{\mathrm{H}}\, \mathbf{L}\, \boldsymbol{\xi}, \qquad \gamma \geq 0, \tag{35}$$

whose closed form solution can be obtained as follows:

$$\widetilde{\mathbf{u}} = h(\mathbf{L})\,\mathbf{u}, \qquad \text{where} \qquad h(x) = 1\,/\,(1+\gamma\, x). \tag{36}$$

Thus, a rational graph filter can be considered as an extension to the Laplacian smoothing, in which the filter can have an arbitrary response in the graph frequency rather than (36) [15].

B. Node-Asynchronous Implementation

Unlike the classical digital filters, a rational graph filter can be represented as an FIR (polynomial) graph filter of order at most $N-1$. (See [50, Theorem 6.2.9].) Thus, one way to implement (34) is to compute $N-1$ graph shifts and take an appropriate linear combination. However, for large graphs (N is large) this is not practical because of its complexity. Furthermore, as discussed in [36], the graph shift (multiplication with $\mathbf{G}$) forces all nodes to communicate at the same time, which requires a synchronization among the nodes of the network. In a large network synchronization introduces delays, or it may not be even possible in the case of autonomous networks. In order to overcome this limitation, this section will introduce a randomized node-asynchronous implementation of the rational graph filtering in (34).

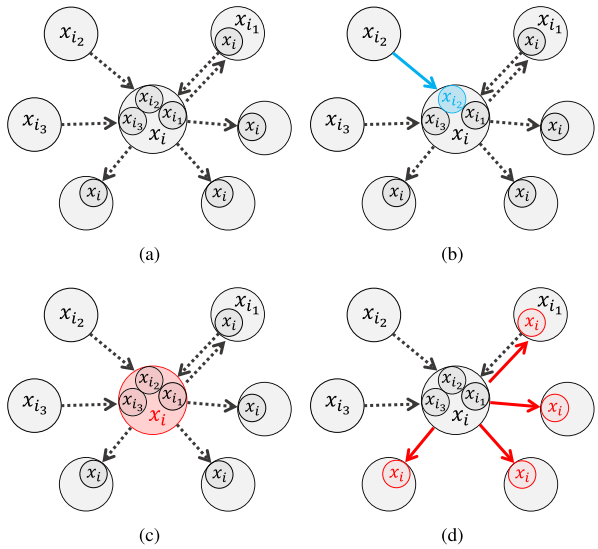


Fig. 1. Visual illustration of the proposed asynchronous implementation of a given graph filter. Edges can be directed in the network. (a) The node i waits and listens in the passive stage. (b) When the node i receives a message, it updates its buffer. (c) When the node i gets into the active stage at a random time instance, it first updates its state vector. (d) After the update, the node i broadcasts its state vector to its outgoing neighbors.

In the proposed implementation, the i^{th} node is assumed to have the following four local variables:

- an input signal: $u_i \in \mathbb{C}$,
- a state-vector: $\mathbf{x}_i \in \mathbb{C}^L$,
- an output variable: $y_i \in \mathbb{C}$,
- a buffer of size $L\,|\mathcal{N}_{\text{in}}(i)|$,

where only the input signal u_i is constant, and the value of the remaining quantities are changing over time in a random manner. In fact, the output variable y_i will be proven to converge to the corresponding element of the filtered signal $\widetilde{u}_i$ in the mean square sense in the proposed approach under some (realistic and practical) conditions on the filter, the graph operator, and the update statistics. (See Theorem 3.)

An overview of our approach (which is illustrated in Fig. 1) is as follows: while a node is not doing updates, it stays in the "passive" stage in which it only receives and stores the state vectors in its buffer sent by its incoming neighbors. See Fig. 1(b). When the node i "wakes up" at a random time instance (asynchronously with respect to other nodes), it follows a two-step update procedure (see Fig. 1(c)):

1) Graph shift step: using the values held in its buffer, the state vector $\mathbf{x}_i$ is updated based on its neighbors according to the graph operator $\mathbf{G}$.
2) Filtering step: the state vector $\mathbf{x}_i$ is updated once more using the input signal and state recursions imposed by the underlying graph filter in (33).

Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on July 02,2021 at 16:04:36 UTC from IEEE Xplore. Restrictions apply.

Once the graph filtering stage is completed, the node i broadcasts its most recent state vector $\mathbf{x}_i$ to its outgoing neighbors, who can use its value to update themselves at random asynchronous times in future, in a similar manner. See Fig. 1(d). In the mean time, the local output variable y_i also gets updated using the state vector $\mathbf{x}_i$ and the input signal u_i.

C. Implementation Details

In this section, we will present the precise details of the proposed asynchronous update mechanism, which was outlined in the previously section. Then, we will present the proposed method formally in Algorithm 1.

We first consider the graph shift step. In order to incorporate the underlying graph structure into the filtering operation, the graph shift step updates the local state vector as *the linear combination of the state vectors of the incoming neighbors*. More precisely, when the i^{th} node is updating its state vector, the node does the following computation first:

$$\mathbf{x}'_i \leftarrow \sum_{j \in \mathcal{N}_{\text{in}}(i)} G_{i,j}\, \mathbf{x}_j, \tag{37}$$

where $\mathbf{x}'_i \in \mathbb{C}^L$ denotes the "graph shifted version" of the state vector $\mathbf{x}_i$. It is important to note that the computation in (37) can be done locally and *asynchronously* by the i^{th} node, as the node is assumed to have all the state vectors of its incoming neighbors already available in its buffer.

In the filtering step, we use the graph shifted state vector $\mathbf{x}'_i$ to carry out a state recursion corresponding to the underlying filter. In this regard, consider the scalar IIR digital filter, $h_{\text{d}}(z)$, whose transfer function is given as follows:

$$h_{\text{d}}(z) = p(z^{-1}) \,/\, q(z^{-1}) = \sum_{n=0}^{\infty} h_n\, z^{-n}, \tag{38}$$

where $p(z)$ and $q(z)$ are as in (31), and h_n's correspond to the coefficients of the impulse response of the digital filter. Furthermore, we assume that the digital filter (38) has the following state-space description:

$$\mathbf{A} = \mathbf{T}^{-1}\, \widehat{\mathbf{A}}\, \mathbf{T}, \quad \mathbf{b} = \mathbf{T}^{-1}\, \widehat{\mathbf{b}}, \quad \mathbf{c} = \widehat{\mathbf{c}}\, \mathbf{T}, \quad d = \widehat{d}, \tag{39}$$

where the quadruple $(\widehat{\mathbf{A}},\, \widehat{\mathbf{b}},\, \widehat{\mathbf{c}},\, \widehat{d})$ corresponds to the direct form description of the filter in (38) (see [51, Section 13.4]):

$$\widehat{\mathbf{A}} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -q_L & -q_{L-1} & \cdots & \cdots & -q_1 \end{bmatrix}, \quad \widehat{\mathbf{b}} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, \quad \widehat{d} = p_0,$$

$$\widehat{\mathbf{c}} = \begin{bmatrix} p_L - p_0\, q_L & p_{L-1} - p_0\, q_{L-1} & \cdots & p_1 - p_0\, q_1 \end{bmatrix}, \tag{40}$$

and $\mathbf{T} \in \mathbb{C}^{L \times L}$ is an arbitrary *invertible* matrix.

Although the response of the filter in (38) does not depend on the particular selection of the matrix $\mathbf{T}$, the convergence properties of the node-asynchronous implementation of the filter on the graph does depend on the similarity transformation. We will elaborate on this in Section IV-A, but the optimal choice of $\mathbf{T}$ is not known at this time.

Using the state-space description of the underlying filter in (39), the i^{th} node executes the following updates locally in the filtering step:

$$\begin{aligned} y_i &\leftarrow \mathbf{c}\, \mathbf{x}'_i + d\, (u_i + w_i), \\ \mathbf{x}_i &\leftarrow \mathbf{A}\, \mathbf{x}'_i + \mathbf{b}\, (u_i + w_i), \end{aligned} \tag{41}$$

where w_i denotes the additive input noise measured by the node i during an update. We note that the value of u_i remains the same, but the value of w_i is different in each update due to it being random. If the nodes do not take measurements, one can easily assume that the measurements are noise-free and set $w_i = 0$ so that the noise covariance is $\mathbf{\Gamma} = \mathbf{0}$.

Algorithm 1: Node-Asynchronous Rational Graph Filtering.

1: **procedure** INITIALIZATION(i)
2: Initialize the state vector $\mathbf{x}_i \in \mathbb{C}^L$ as $\mathbf{x}_i = \mathbf{0}$.
3: **procedure** PASSIVE STAGE(i)
4: **if** $\mathbf{x}_j$ is received from the node $j \in \mathcal{N}_{\text{in}}(i)$ **then**
5: Store the most recent value of $\mathbf{x}_j$.
6: **procedure** ACTIVE STAGE(i)
7: $\mathbf{x}'_i \leftarrow \sum_{j \in \mathcal{N}_{\text{in}}(i)} G_{i,j}\, \mathbf{x}_j$. ▷ graph shift
8: $v_i \leftarrow u_i + w_i$. ▷ noisy sample
9: $y_i \leftarrow \mathbf{c}\, \mathbf{x}'_i + d\, v_i$. ▷ filtering
10: $\mathbf{x}_i \leftarrow \mathbf{A}\, \mathbf{x}'_i + \mathbf{b}\, v_i$. ▷ filtering
11: Broadcast $\mathbf{x}_i$ to all $j \in \mathcal{N}_{\text{out}}(i)$.

The random node-asynchronous implementation of the IIR graph filter (33) is summarized in Algorithm 1. In the next section we will prove that this algorithm is indeed a valid implementation of (33) under some conditions to be stated.

Except the initialization stage, which is executed only once, Algorithm 1 consists of two main states: passive and active, both of which are triggered asynchronously. More precisely, the active stage is triggered randomly by a node-specific timer, or condition, and the passive stage is triggered when a node receives a state vector from its incoming neighbors. It is also assumed that a node stores only the most recent received data.

When the node i gets into the active stage at a random time instance, the node first computes its graph shifted state vector $\mathbf{x}'_i$ in Line 7 using the values that are available in its buffer. Then, the node takes a noisy measurement of the underlying graph signal u_i. When the filtering recursions in Lines 9 and 10 are completed, the node broadcasts its own local state vector to its outgoing neighbors, and gets back into the passive stage.

In the presented algorithm we emphasize that a node getting into the active stage is independent of the values in its buffer. In general, a node does *not* wait until it receives state vectors from all of its neighbors. In between two activations (i.e., in the passive stage), some values in the buffer may be updated more than once, and some may not be updated at all. Nodes use the most recent update only.

Since nodes are assumed to store the most recent data of its incoming neighbors in the presented form of the algorithm, the node i requires a buffer of size $L \cdot |\mathcal{N}_{\text{in}}(i)|$. In fact, Algorithm 1 can be implemented in such a way that each node uses a buffer of size $2L$ only. One can show that this is achieved when each node broadcasts the difference in its state vector rather than the state vector itself, and the variable $\mathbf{x}'_i$ accumulates all the received differences in the passive stage. However, such an implementation may not be robust under communication failures, whereas the current form of the algorithm is shown to be

Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on July 02,2021 at 16:04:36 UTC from IEEE Xplore. Restrictions apply.

robust to communication failures. (See Section V-B.) Moreover, the current form of the algorithm is easier to model and analyze mathematically as we shall elaborate in the next section.

Due to its random asynchronous nature, Algorithm 1 appears similar to filtering over time varying graphs, which is studied extensively in [21]. However, random asynchronous communications differ from randomly varying graph topologies in two ways: 1) Expected value of the signal depends on the "expected graph" in randomly varying graph topologies [21, Theorem 1], whereas the fixed point does not depend on the update probabilities in the case of asynchronous communications. 2) The graph signal converges in the mean-squared sense in the case of asynchronous communications (see Section IV), whereas the signal has a nonzero variance in the case of randomly varying graph topologies [21, Theorem 3].

We also note that the study in [52] proposed a similar algorithm, in which nodes retrieve and aggregate information from a subset of neighbors of fixed size selected uniformly randomly. However, the computational stage of [52] consists of a linear mapping followed by a sigmoidal function, whereas Algorithm 1 uses a linear update model. More importantly, aggregations are done synchronously in [52], that is, all nodes are required to complete the necessary computations before proceeding to the next level of aggregation. On the contrary, nodes aggregate information repetitively and *asynchronously* without waiting for each other in Algorithm 1.

IV. Convergence of the Proposed Algorithm

For convenience of analysis we define

$$\mathbf{X}_{(k)} = [\mathbf{x}_1 \quad \mathbf{x}_2 \quad \cdots \quad \mathbf{x}_{N-1} \quad \mathbf{x}_N] \in \mathbb{C}^{L\times N},$$
$$\mathbf{y}_{(k)} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, \qquad \mathbf{w}_{(k)} = \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix}, \qquad \mathbf{u} = \begin{bmatrix} u_1 \\ \vdots \\ u_N \end{bmatrix}. \quad (42)$$

Here, $\mathbf{X}_{(k)}$ will be called the augmented state variable matrix, and $\mathbf{y}_{(k)} \in \mathbb{C}^N$ is the output vector after k iterations of the algorithm. Also $\mathbf{w}_{(k)} \in \mathbb{C}^N$ is the noise vector at the k^{th} iteration, and $\mathbf{u} \in \mathbb{C}^N$ is the graph input signal as before.

We note that the index k is a global counter that we use to enumerate the iterations. In general, nodes are unaware of the value of k, which is why the augmented variables in (42) are indexed with k in parenthesis, but the variables corresponding to individual nodes are not indexed by k at all. Whenever a node completes the execution of the active stage of Algorithm 1, we assume that an iteration has passed. Thus, *(42) denotes the variables at the end of the k^{th} iteration.* Furthermore, we will use $\mathcal{T}_k$ to denote the set of nodes that get into the active stage simultaneously at the k^{th} iteration.

Algorithm 1 allows the nodes to update their values with different frequencies. Similar to (17), we will use the diagonal matrix $\mathbf{P} \in \mathbb{R}^{N\times N}$ to denote the average node (index) selection matrix in the algorithm. In particular, $\mathbf{P} = p\,\mathbf{I}$ corresponds to the case of all the nodes having the same rate of getting into the active stage.

In order to analyze the evolution of the state variables in the algorithm, we first note that the state vector of a node i at the *beginning* of the k^{th} iteration can be written as follows:

$$\mathbf{x}_i = \mathbf{X}_{(k-1)}\,\mathbf{e}_i, \qquad 1 \le i \le N. \quad (43)$$

Thus, if the node i gets into the active stage at the k^{th} iteration, i.e., $i \in \mathcal{T}_k$, then its graph shifted state vector (computed in Line 7 of the algorithm) can be written as follows:

$$\mathbf{x}'_i = \sum_{j\in\mathcal{N}_{\text{in}}(i)} G_{i,j}\,\mathbf{x}_j = \sum_j \mathbf{X}_{(k-1)}\,\mathbf{e}_j\,\mathbf{e}_j^{\mathrm{T}}\,\mathbf{G}^{\mathrm{T}}\,\mathbf{e}_i$$
$$= \mathbf{X}_{(k-1)}\,\mathbf{G}^{\mathrm{T}}\,\mathbf{e}_i. \quad (44)$$

Therefore, the next value for its state vector is given as follows:

$$\mathbf{X}_{(k)}\,\mathbf{e}_i = \left(\mathbf{A}\,\mathbf{X}_{(k-1)}\mathbf{G}^{\mathrm{T}} + \mathbf{b}\,(\mathbf{u} + \mathbf{w}_{(k-1)})^{\mathrm{T}}\right)\,\mathbf{e}_i, \quad i \in \mathcal{T}_k. \quad (45)$$

On the other hand, if the node i does not get into the active stage at the k^{th} iteration, i.e., $i \notin \mathcal{T}_k$, its state vector remains unchanged. Thus, we can write following:

$$\mathbf{X}_{(k)}\,\mathbf{e}_i = \mathbf{X}_{(k-1)}\,\mathbf{e}_i, \qquad i \notin \mathcal{T}_k. \quad (46)$$

Since both (45) and (46) are linear in the augmented state variable matrix $\mathbf{X}_{(k)}$, we can transpose, and then vectorize both equations and represent them as follows:

$$(\overline{\mathbf{x}}_k)_i = \begin{cases} (\overline{\mathbf{A}}\,\overline{\mathbf{x}}_{k-1})_i + (\overline{\mathbf{u}}_{k-1})_i, & i \in \overline{\mathcal{T}}_k, \\ (\overline{\mathbf{x}}_{k-1})_i, & i \notin \overline{\mathcal{T}}_k, \end{cases} \quad (47)$$

where the variables of the vectorized model are as follows:

$$\overline{\mathbf{x}}_k = \mathrm{vec}\left(\mathbf{X}_{(k)}^{\mathrm{T}}\right), \qquad \overline{\mathbf{A}} = \mathbf{A} \otimes \mathbf{G},$$
$$\overline{\mathbf{u}} = \mathbf{b} \otimes \mathbf{u}, \qquad \overline{\mathbf{w}}_k = \mathbf{b} \otimes \mathbf{w}_{(k)}, \quad (48)$$

and $\overline{\mathbf{u}}_k$ is defined similar to (4) as $\overline{\mathbf{u}}_k = \overline{\mathbf{u}} + \overline{\mathbf{w}}_k$. Furthermore, the update set $\overline{\mathcal{T}}_k$ of the vectorized model is defined as follows:

$$\overline{\mathcal{T}}_k = \{i + jN | i \in \mathcal{T}_k, 0 \le j < L\}, \quad (49)$$

which follows from the fact that when a node gets into the active stage, it updates all elements of its own state vector simultaneously according to Line 10 of the algorithm.

We note that the mathematical model in (47) appears as a pull-like algorithm, in which nodes retrieve data from their incoming neighbors. However, with the use of a buffer, the model (47) can be implemented in a collect-compute-broadcast scheme as proposed in Algorithm 1. See also Fig. 1.

When the algorithm is implemented in a synchronous manner, the state recursions of (47) reduce to the following form:

$$\overline{\mathbf{x}}_k = \overline{\mathbf{A}}\,\overline{\mathbf{x}}_{k-1} + \overline{\mathbf{u}}_{k-1}, \quad (50)$$

and the following theorem (whose proof is provided in Appendix C) presents the mean-squared error of the algorithm:

Theorem 2: In Algorithm 1, assume that all the nodes on the graph get into the active stage synchronously, and the matrix $\overline{\mathbf{A}}$ does not have an eigenvalue equal to 1. Then,

$$\mathbb{E}\left[\left\|\mathbf{y}_{(k)} - \widetilde{\mathbf{u}}\right\|_2^2\right] = \left\|(\mathbf{c}\otimes\mathbf{G})\;\overline{\mathbf{A}}^{k-1}\;(\overline{\mathbf{x}}_0 - \overline{\mathbf{x}}^\star)\right\|_2^2$$
$$+ \sum_{n=0}^{k-1} |h_n|^2\;\mathrm{tr}\left(\mathbf{G}^n\,\mathbf{\Gamma}\,(\mathbf{G}^n)^{\mathrm{H}}\right), \quad (51)$$

where $\overline{\mathbf{x}}^\star$ is the fixed point of (50), and h_n's are the coefficients of the impulse response of the digital filter as in (38).

In (51) it is clear that as long as

$$\rho(\overline{\mathbf{A}}) < 1, \quad (52)$$

Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on July 02,2021 at 16:04:36 UTC from IEEE Xplore. Restrictions apply.

the first term of (51) converges to zero irrespective of the initial vector $\overline{\mathbf{x}}_0$, as the iteration progresses. So, from Theorem 2 the residual error approaches an error floor:

$$\lim_{k\to\infty} \mathbb{E}\left[\left\|\mathbf{y}_{(k)} - \widetilde{\mathbf{u}}\right\|_2^2\right] = \mathrm{tr}(\mathbf{H}\,\mathbf{\Gamma}), \tag{53}$$

where

$$\mathbf{H} = \sum_{n=0}^{\infty} |h_n|^2 \,(\mathbf{G}^n)^{\mathrm{H}}\,\mathbf{G}^n. \tag{54}$$

Thus, the error floor in the synchronous case depends on the *impulse response* of the underlying digital filter as well as the graph operator, but the similarity transform $\mathbf{T}$ does not affect the error floor. In short, the similarity transform does not affect either the convergence or the error floor in the synchronous case. Note that the stability condition in (52) ensures the convergence of (54). Note also that $\rho(\overline{\mathbf{A}}) = \rho(\mathbf{A})\,\rho(\mathbf{G})$ in view of (48).

Next consider the asynchronous case. The equivalent model of the algorithm in (47) is in the form of (14), thus the results presented in Section II (Corollary 1 in particular) can be used to study the convergence of the algorithm. In this regard, we present the following theorem, whose complete proof is given in Appendix D:

Theorem 3: In Algorithm 1, let $\mathbf{P}$ denote the average node selection matrix and $\mathbf{\Gamma}$ the covariance matrix of the measurement noise. If the state transition matrix $\mathbf{A}$ of the filter, and the operator $\mathbf{G}$ of the graph satisfy the following:

$$\|\mathbf{A}\|_2^2\;\mathbf{G}^{\mathrm{H}}\,\mathbf{P}\,\mathbf{G} \prec \mathbf{P}, \tag{55}$$

then

$$\lim_{k\to\infty} \mathbb{E}\left[\left\|\mathbf{y}_{(k)} - \widetilde{\mathbf{u}}\right\|_2^2\right] \leq \mathrm{tr}(\mathbf{R}\,\mathbf{\Gamma}), \tag{56}$$

where

$$\mathbf{R} = \frac{\|\mathbf{b}\|_2^2\;\|\mathbf{c}\|_2^2\;\|\mathbf{G}\|_2^2}{\lambda_{\min}\left(\mathbf{P} - \|\mathbf{A}\|_2^2\,\mathbf{G}^{\mathrm{H}}\,\mathbf{P}\,\mathbf{G}\right)}\,\mathbf{P} + |d|^2\,\mathbf{I}. \tag{57}$$

Theorem 3 presents an upper bound on the mean-squared error. In the noise-free case ($\mathbf{\Gamma} = \mathbf{0}$), the right-hand-side of (56) becomes zero, and the condition (55) ensures the convergence of the output signal to the desired filtered signal in the mean-squared sense. We note also that the right-hand-side of (56) is linear in the noise covariance matrix, which implies that the error floor of the algorithm increases at most linearly with the input noise. This will be numerically verified later in Section V-A. (See Fig. 4(b).) In fact, it is possible to integrate stochastic averaging techniques studied in [34], [35] into Algorithm 1 in order to overcome the error due to noise at expense of a reduced convergence rate.

We conclude by noting that graph filtering implementations considered in [15]–[23] are likely to tolerate asynchronicity up to a certain degree. In fact, [23] presented numerical evidences in this regard. This is not surprising because linear asynchronous fixed-point iterations are known to converge under some conditions [28], [29]. The main difference of Algorithm 1 studied in this paper is due to its *proven convergence* under some mild and interpretable conditions with the assumed random asynchronous model (Theorem 3).

A. Selection of the Similarity Transform

In addition to the dependency on the graph operator and the average node selection matrix, the sufficiency condition (55) depends also on the realization of the filter of interest. Thus, in the asynchronous case, both the condition for convergence and the error bound depend on the similarity transform. Since the condition becomes more relaxed as the state transition matrix $\mathbf{A}$ has a smaller spectral norm, it is important to select the similarity transform $\mathbf{T}$ in (39) in such a way that $\mathbf{A}$ has the minimum spectral norm.

Due to their robustness, minimum-norm realizations of digital filters have been studied extensively in signal processing [41], [53], [54]. A minimum-norm implementation corresponds to an appropriate selection of the similarity transform $\mathbf{T}$ in (39) due to the following inequality:

$$\|\mathbf{A}\|_2 \geq \rho(\mathbf{A}) = \rho(\widehat{\mathbf{A}}). \tag{58}$$

The lower bound $\rho(\widehat{\mathbf{A}})$ depends only on the coefficients of the polynomial $q(x)$ due to the definition of $\widehat{\mathbf{A}}$ in (40).

The lower bound in (58) may *not* be achieved with equality in general, and we will consider one such example in the next section. Nevertheless, it is known that the companion matrix $\widehat{\mathbf{A}}$ is diagonalizable if and only if the digital filter in (38) has L distinct poles [55]. That is to say, when there are L distinct nonzero z_n's such that $q(z_n^{-1}) = 0$, we can write the following eigenvalue decomposition:

$$\widehat{\mathbf{A}} = \mathbf{V}_{\widehat{\mathbf{A}}}\;\mathbf{\Lambda}_{\widehat{\mathbf{A}}}\;\mathbf{V}_{\widehat{\mathbf{A}}}^{-1}, \tag{59}$$

where $\mathbf{\Lambda}_{\widehat{\mathbf{A}}}$ is a diagonal matrix with z_n^{-1}'s on the diagonal, and $\mathbf{V}_{\widehat{\mathbf{A}}}$ is a Vandermonde matrix corresponding to z_n^{-1}'s. If the similarity transform $\mathbf{T}$ is selected according to (59), then the bound in (58) is indeed achieved. More precisely,

$$\mathbf{T} = \mathbf{V}_{\widehat{\mathbf{A}}} \;\Rightarrow\; \mathbf{A} = \mathbf{\Lambda}_{\widehat{\mathbf{A}}} \;\Rightarrow\; \|\mathbf{A}\|_2 = \rho(\widehat{\mathbf{A}}). \tag{60}$$

Thus, the most relaxed version of the sufficiency condition of Theorem 3 is obtained when the updates of Algorithm 1 are implemented using the similarity transform given in (60).

When the filter (38) has repeated poles, the companion matrix $\widehat{\mathbf{A}}$ is not diagonalizable, hence an implementation achieving the bound (58) does not exist [53]. Nevertheless, the study [53] discussed that for any $\epsilon > 0$, there exists a realization with a state transition matrix $\mathbf{A}$ such that

$$\|\mathbf{A}\|_2 \leq \rho(\widehat{\mathbf{A}}) + \epsilon. \tag{61}$$

Therefore, it is always possible to obtain "almost minimum" realizations with the spectral norm arbitrarily close to the lower bound in (58). As a particular example, the case of FIR graph filters will be considered in the next section.

B. The Case of Polynomial Filters

Polynomial (FIR) graph filters can be considered as a special case of the rational graph filter (33), in which the denominator is selected as $q(x) = 1$ so that $q(\mathbf{G}) = \mathbf{I}$, and the filtered signal in (34) reduces to $\widetilde{\mathbf{u}} = p(\mathbf{G})\,\mathbf{u}$. In this case, the companion matrix

Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on July 02,2021 at 16:04:36 UTC from IEEE Xplore. Restrictions apply.

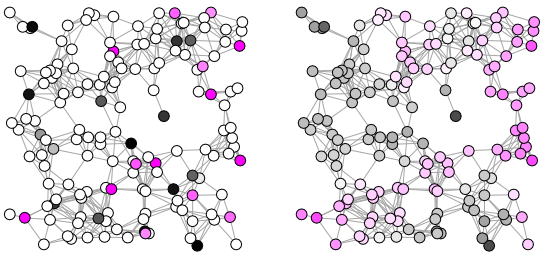


Fig. 2. Visualization of the signals on the graph. Colors black and pink represent positive and negative values, respectively. Intensity of a color represents the magnitude. (a) The graph signal $\mathbf{u}$ that has nonzero values on 30 nodes. (b) The filtered signal $\widetilde{\mathbf{u}}$ on the graph with the filter in (71).

$\widehat{\mathbf{A}}$ (direct form implementation) has the following form:

$$\widehat{\mathbf{A}} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{L \times L}, \tag{62}$$

which has all eigenvalues equal to zero, so that $\rho(\widehat{\mathbf{A}}) = 0$. As a result, no realization of a polynomial filter can achieve the lower bound (58) since $\|\mathbf{A}\|_2 = 0$ implies $\mathbf{A} = \mathbf{0}$. However, the spectral norm of a realization can be made arbitrarily small. In particular, consider the following similarity transform:

$$\mathbf{T} = \operatorname{diag}\left(\begin{bmatrix} 1 & \epsilon & \epsilon^2 & \cdots & \epsilon^{L-1} \end{bmatrix} \right), \tag{63}$$

where ϵ is an arbitrary nonzero complex number. Then, the corresponding realization $\mathbf{A}$ can be found as follows:

$$\mathbf{A} = \mathbf{T}^{-1}\, \widehat{\mathbf{A}}\, \mathbf{T} = \epsilon\, \widehat{\mathbf{A}} \quad \Rightarrow \quad \|\mathbf{A}\|_2 = |\epsilon|. \tag{64}$$

Thus, it is possible to select a value for ϵ (with a sufficiently small magnitude) in order to satisfy the condition (55). (See [45, Fact 2.5.4].) Such a selection is not unique in general, and one can easily find a value for ϵ satisfying the following:

$$|\epsilon| < \left(\|\mathbf{G}\|_2 \sqrt{\|\mathbf{P}\|_2 \left\|\mathbf{P}^{-1}\right\|_2} \right)^{-1}, \tag{65}$$

which ensures that the condition (55) is met.

As a result, for *any* graph operator $\mathbf{G}$ and average node selection matrix $\mathbf{P}$, it is always possible to implement *any* polynomial filter in a random node-asynchronous manner that is guaranteed to converge in the mean-squared sense. However, we note that $\mathbf{T}$ given in (63) may not be the optimal similarity transform in general.

We also note that when a polynomial filter is implemented in a synchronous manner, Theorem 2 shows that the algorithm reaches the error floor after L iterations since $\widehat{\mathbf{A}}$ in (62) is a nilpotent matrix and $\widehat{\mathbf{A}}^n = \mathbf{A}^n = \mathbf{0}$ for $n \geq L$. This convergence behavior will be verified numerically later in Section V-D. The error bound still depends on $\mathbf{T}$ because of $\|\mathbf{A}\|_2$.

V. Numerical Simulations

We now simulate the proposed algorithm on the graph visualized in Fig. 2. This is a random geometric graph on $N = 150$ nodes, in which nodes are distributed over the region $[0\ 1] \times [0\ 1]$ uniformly at random. Two nodes are connected to each other if the distance between them is less than 0.15, and the graph is undirected. The graph operator, the matrix $\mathbf{G} \in \mathbb{R}^{N \times N}$, is selected as the Laplacian matrix whose eigenvalues can be sorted as follows:

$$0 = \lambda_1 < \lambda_2 \leq \cdots \leq \lambda_N = \rho(\mathbf{G}) = \|\mathbf{G}\|_2 = 16.8891, \tag{66}$$

where the spectral norm of $\mathbf{G}$ is computed numerically, and the equality between the spectral radius and the spectral norm follows from the fact that $\mathbf{G}$ is a real symmetric matrix.

For the numerical simulations we consider the following smoothing problem: assume that we are given the graph signal $\mathbf{u} \in \mathbb{R}^N$ that has only 30 nonzero entries, which is visualized in Fig. 2(a). It is clear that the signal $\mathbf{u}$ is not smooth on the graph. In order to obtain a smoothed version of $\mathbf{u}$, which will be denoted by $\widetilde{\mathbf{u}} \in \mathbb{R}^N$, we will apply a low-pass graph filter to the signal $\mathbf{u}$. In this regard, we will consider examples of rational (IIR) graph filters in Sections V-A, V-B and V-C, and consider a polynomial (FIR) filter in Section V-D.

Throughout the simulations we will consider a particular stochastic model for the selection of the nodes. That is, in each iteration of Algorithm 1 we will select a subset of size μ uniformly randomly among all subsets of size μ. For this particular model, the average node selection matrix becomes:

$$\mathbf{P} = \frac{\mu}{N}\, \mathbf{I}. \tag{67}$$

We note that the case of $\mu = N$ corresponds to the synchronous implementation of the algorithm. With $\mathbf{P}$ as in (67), we note that the sufficiency condition (55), which ensures the convergence of Algorithm 1, reduces to the following form:

$$\|\mathbf{A}\|_2\, \|\mathbf{G}\|_2 < 1, \tag{68}$$

which does not depend on μ. Furthermore, the bound on the noise floor given by (56) reduces to the following form:

$$\lim_{k \to \infty} \mathbb{E}\left[\|\mathbf{y}_{(k)} - \widetilde{\mathbf{u}}\|_2^2 \right] \leq \operatorname{tr}(\mathbf{\Gamma}) \left(\frac{\|\mathbf{b}\|_2^2\, \|\mathbf{c}\|_2^2\, \|\mathbf{G}\|_2^2}{1 - \|\mathbf{A}\|_2^2\, \|\mathbf{G}\|_2^2} + |d|^2 \right). \tag{69}$$

For the sake of simplicity we will assume that the covariance matrix of the measurement noise is as follows:

$$\mathbf{\Gamma} = \sigma^2\, \mathbf{I}, \tag{70}$$

where σ^2 will denote the variance of input noise.

A. An Example of a Rational Graph Filter

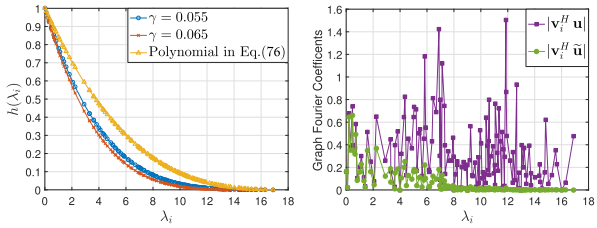
In this section we will consider a rational filter (30) constructed with the following polynomials of order $L = 3$:

$$p(x) = (1 - \gamma\, x)^3, \quad q(x) = 1 + \sum_{n=1}^{3} \gamma^n\, x^n, \quad \gamma = 0.055, \tag{71}$$

where the value of γ is selected in such a way that it normalizes the spectrum of $\mathbf{G}$, that is, $|\gamma|\, \|\mathbf{G}\|_2 < 1$ is satisfied.

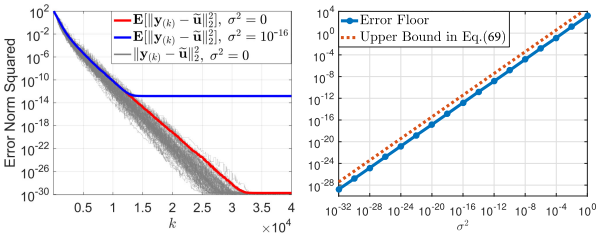
The frequency response of the filter in (71) on the graph is visualized in Fig. 3(a), which shows that the filter has low-pass characteristics on the graph. When compared with the input signal $\mathbf{u}$, the filtered signal $\widetilde{\mathbf{u}}$ has a lower amount of projection on the eigenvectors with larger eigenvalues as shown in Fig. 3(b). Since $\widetilde{\mathbf{u}}$ mainly contains low frequency components (eigenvectors with small eigenvalues [4]), $\widetilde{\mathbf{u}}$ is smoother on the graph as visualized in Fig. 2(b).

Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on July 02,2021 at 16:04:36 UTC from IEEE Xplore. Restrictions apply.



(a) (b)

Fig. 3. (a) Response of the rational filter $h(\lambda)$ constructed with (71). (b) Magnitude of the graph Fourier transforms of $\mathbf{u}$ and $\widetilde{\mathbf{u}}$ where $(\lambda_i, \mathbf{v}_i)$ denotes an eigenpair of $\mathbf{G}$.



(a) (b)

Fig. 4. (a) Squared error norm in 100 different independent realizations together with the mean squared error of Algorithm 1 with the implementation in (73). (b) Error floor of the algorithm with respect to the amount of input noise together with the bound in (69).

We now consider the implementation of the filter (71) using Algorithm 1. In this regard, we first construct the direct form implementation of the corresponding digital filter as in (40):

$$\widehat{\mathbf{A}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -\gamma^3 & -\gamma^2 & -\gamma \end{bmatrix}, \ \widehat{\mathbf{b}} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \ \widehat{\mathbf{c}}^{\mathrm{T}} = \begin{bmatrix} -2\gamma^3 \\ 2\gamma^2 \\ -4\gamma \end{bmatrix}, \ \widehat{d} = 1. \tag{72}$$

It is readily verified that the matrix $\widehat{\mathbf{A}}$ in (72) has $L = 3$ distinct eigenvalues that are given as $\{-\gamma,\ j\gamma,\ -j\gamma\}$. Thus, the similarity transform $\mathbf{T}$ can be selected as the eigenvectors of $\widehat{\mathbf{A}}$ as in (60), which corresponds to the Vandermonde matrix constructed with $\{-\gamma,\ j\gamma,\ -j\gamma\}$. As a result, the corresponding realization of the filter according to (39) is given as follows:

$$\mathbf{A} = \gamma \begin{bmatrix} -1 & 0 & 0 \\ 0 & j & 0 \\ 0 & 0 & -j \end{bmatrix}, \ \mathbf{b} = \frac{-1}{4\gamma^2} \begin{bmatrix} -2 \\ 1+j \\ 1-j \end{bmatrix}, \ \mathbf{c}^{\mathrm{T}} = \gamma^3 \begin{bmatrix} -8 \\ 2+2j \\ 2-2j \end{bmatrix}. \tag{73}$$

Since $\|\mathbf{A}\|_2 = \rho(\mathbf{A}) = |\gamma|$, we note that (68) is satisfied for the value of γ in (71), thus Algorithm 1 converges in the mean-squared sense when no input noise is present, and when there is noise, it reaches an error floor upper bounded as in (69).

In the first set of simulations of Algorithm 1 we consider the case of $\mu = 1$, i.e., only one randomly selected node is updated per iteration. In order to verify the convergence numerically, we simulated independent runs of Algorithm 1 with the filter realization in (73) and computed the mean-squared error by averaging over 10^4 independent runs. In order to present the effect of the measurement noise, we consider the case of $\sigma^2 = 10^{-16}$ as well as the noise-free case. Fig. 4(a) presents the corresponding mean-squared errors together with the error in the noise-free case for 100 different realizations. Due to the random selection of the nodes, the residual itself is a random quantity, which does not decrease monotonically as seen in Fig. 4(a). Nevertheless, the *expectation* of the error norm decreases monotonically until it reaches the error floor. We note that the error floor in the noise-free case corresponds to the numerical precision of the numerical environment (MATLAB).

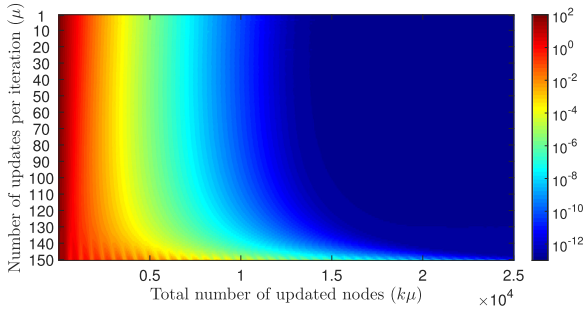


Fig. 5. The mean squared error of Algorithm 1 when more than one node is updated simultaneously with noise variance $\sigma^2 = 10^{-16}$. The first row in the figure corresponds to Fig. 4(a).

In order to present the effect of the noise variance on the error floor, we run Algorithm 1 for different values of σ^2 for $k_{\max} = 4 \cdot 10^4$ iterations (which ensures that the algorithm reaches an error floor as seen in Fig. 4(a)) while selecting only $\mu = 1$ node per iteration. The error floor corresponding to different values of σ^2 together with the upper bound in (69) are presented in Fig. 4(b). In addition to the upper bound (69) scaling linearly with the noise variance, Fig. 4(b) shows that the error floor itself scales almost linearly with the noise variance as well.

We note that the filter realization in (73) ensures the convergence of the algorithm irrespective of the value of μ. However, the *convergence rate* of the algorithm does depend on the value of μ in general. This point will be demonstrated in the following set of simulations, in which we use the filter realization in (73) and set the noise variance as $\sigma^2 = 10^{-16}$. In order to obtain a fair comparison between different values of μ, we fix the total number of updates to be 25 000, so the algorithm gets $\lceil 25\,000/\mu \rceil$ iterations. We run the algorithm independently 10^5 times for each value of $\mu = \{1, \ldots, N\}$ and present the corresponding mean-squared errors with respect to the number of updated nodes in Fig. 5.

We first point out that Fig. 5 verifies the convergence of the algorithm for all possible values of μ. More interestingly, the figure shows also that *the algorithm gets faster as it gets more asynchronous (small μ)*. Equivalently, for a given fixed amount of computational budget (total number of nodes to be updated), having nodes updated randomly and asynchronously results in a smaller error than having synchronous updates. However, it is important to emphasize that the behavior shown in Fig. 5 is *not* typical for the algorithm; rather, it depends on the underlying filter. Indeed we will find a similar behavior in Section V-C, but an opposite behavior later in Section V-D. We also note that for the case of zero-input, the study [37] theoretically discussed the conditions under which randomized asynchronicity results in a faster convergence.

B. Updates With Failing Broadcasts

Algorithm 1 assumes that when a node broadcasts the most recent value of its state vector to its outgoing neighbors (Line 11 of Algorithm 1), all the recipient nodes reliably receive the message. However, in a more realistic scenario the broadcasted

Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on July 02,2021 at 16:04:36 UTC from IEEE Xplore. Restrictions apply.

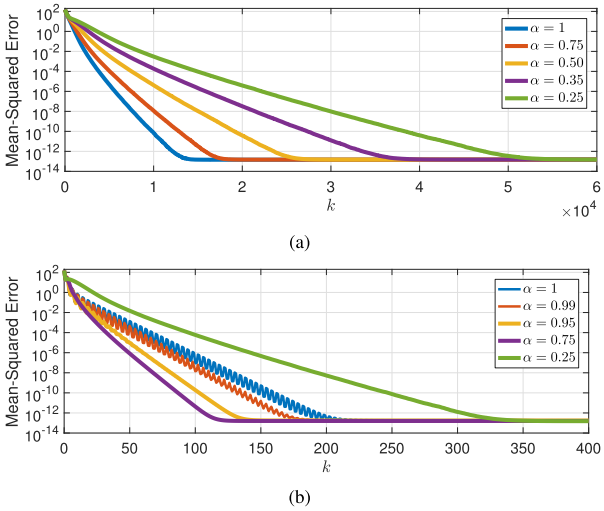


Fig. 6. The mean-squared error of the algorithm with (a) $\mu = 1$, (b) $\mu = N$, when the broadcasted messages are delivered successfully with probability α.

message may not be received by some of the recipients due to unreliable communication between the nodes. In the case of such communication failures, the theoretical analysis presented in Section IV (Theorems 2 and 3) becomes inconclusive regarding the convergence of the algorithm. Nevertheless, in this section we will numerically verify that the proposed algorithm is robust to such communication failures.

Similar to the previous section, in this set of simulations we use the filter realization in (73) (with γ selected as in (71)) and set the noise variance as $\sigma^2 = 10^{-16}$. However, we modify the implementation in such a way that when a node broadcasts its state vector, a recipient node is assumed to receive the message with probability α independent of the other nodes. Thus, $\alpha = 1$ corresponds to the case where convergence is guaranteed by Theorem 3.

We consider two cases, namely $\mu = 1$ (one node is activated randomly in each iteration) and $\mu = N$ (all nodes are activated synchronously). In both cases the broadcasted messages are delivered with probability α. The mean squared errors for these two cases are given in Figs. 6(a) and (b), respectively.

Both Figs. 6(a) and (b) verify the convergence of the algorithm even in the case of unreliable communication. In the case of $\mu = 1$, Fig. 6(a) suggests that the convergence rate of the algorithm decreases as the communications become more unreliable (the value of α gets smaller). However, for the case of $\mu = N$, Fig. 6(b) presents an unexpected behavior. The case of reliable communications ($\alpha = 1$) does *not* result in the fastest rate of convergence. When the communications fail with some probability, the algorithm may converge *faster*. While the behavior is surprising, it is consistent with Fig. 5 in the sense that fully synchronous iterations are slower than asynchronous counterparts for the specific filter in (73). Even when the nodes get updated synchronously, failed broadcasts break the overall synchrony over the network, hence the algorithm converges faster. However, when the communications fail with high probability (e.g., the case of $\alpha = 0.25$ in Fig. 6(b)), the convergence is indeed slower. We also note that the behaviors demonstrated in Figs. 6(a) and (b) remain the same even for the noise-free ($\sigma^2 = 0$) case.

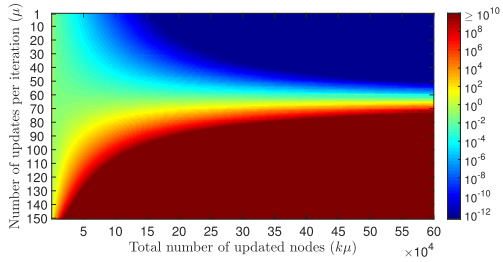


Fig. 7. The mean-squared error of Algorithm 1 for a case of an unstable augmented state transition matrix $\overline{\mathbf{A}}$. Input noise variance is $\sigma^2 = 10^{-16}$.

C. A Case of Convergence Only With Asynchronous Iterations

The results in Section V-A (namely Fig. 5) showed that the proposed algorithm may converge faster as the iterations get more asynchronous (i.e., the value of μ gets smaller). In this section we will demonstrate an even more interesting behavior, where the algorithm converges *only if the iterations are sufficiently asynchronous (μ is smaller than a threshold).*

In this subsection, we will use the same filter realization as in (73), but use the following value for the parameter γ:

$$\gamma = 0.065, \tag{74}$$

which results in a slight change in the response of the filter as presented in Fig. 3(a). More importantly, for the value of γ in (74), the sufficiency condition in (68) is not satisfied, thus Theorem 3 is *inconclusive* regarding the convergence of the algorithm with asynchronous iterations. In fact, Theorem 2 tells that the algorithm *diverges* in the synchronous case since the matrix $\overline{\mathbf{A}}$ is unstable for the value of γ in (74):

$$\rho(\overline{\mathbf{A}}) = \rho(\mathbf{A} \otimes \mathbf{G}) = \rho(\mathbf{A})\,\rho(\mathbf{G}) = |\gamma|\,\rho(\mathbf{G}) \approx 1.0978. \tag{75}$$

In order to examine the convergence behavior of the algorithm, we repeat the simulations done in Section V-A with the value of γ set as in (74). That is, the noise variance is set to be $\sigma^2 = 10^{-16}$, and the algorithm is simulated independently 10^4 times for each value of $\mu = \{1, \ldots, N\}$. The corresponding mean-squared errors are presented in Fig. 7.

For the specific filter considered in this simulations, Fig. 7 shows that the convergence of the algorithm displays an obvious phase-transition in terms of the amount of asynchronicity. That is to say, the algorithm convergences only if the number of simultaneously updated nodes satisfies $\mu \leq 66$, and the algorithm diverges otherwise. Therefore, a specific amount of asynchronicity is in fact required for the convergence in this example.

Although the theoretical analysis of the algorithm presented in Section IV does not explain the phenomena observed here, for the zero-input case the study [36], [37] proved that the convergence can be achieved for some unstable systems as long as the iterations are sufficiently asynchronous. Simulation results presented in Fig. 7 shows that a similar behavior exists even when the input is nonzero.

D. An Example of a Polynomial Graph Filter

Now consider the implementation of a polynomial (FIR) graph filter with the proposed algorithm. In particular, we consider the following filter of order $L = 3$:

$$p(x) = (1 - \gamma\, x)^3, \qquad q(x) = 1, \qquad \gamma = 0.055, \tag{76}$$

Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on July 02,2021 at 16:04:36 UTC from IEEE Xplore. Restrictions apply.

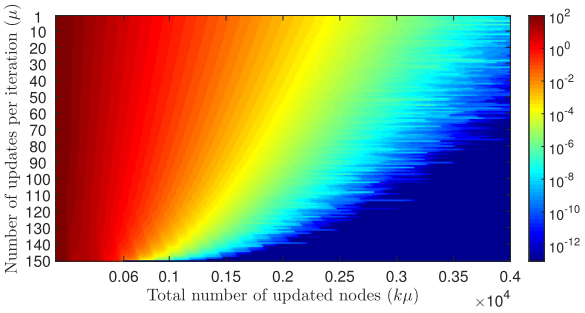


Fig. 8. The mean-squared error of Algorithm 1 for the case of the polynomial (FIR) filter described in (76) with the input noise variance $\sigma^2 = 10^{-16}$.

which has low-pass characteristics on the graph as visualized in Fig. 3(a). In the implementation of the filter we use the following similarity transformation $\mathbf{T} = \mathrm{diag}([1 \;\; \gamma \;\; \gamma^2])$ so that the realization of the filter has the following form:

$$\mathbf{A} = \gamma \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \; \mathbf{b} = \frac{1}{\gamma^2} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \; \mathbf{c}^{\mathrm{T}} = \gamma^3 \begin{bmatrix} -1 \\ 3 \\ -3 \end{bmatrix}, \; d = 1, \quad (77)$$

which satisfies $\|\mathbf{A}\|_2 \, \|\mathbf{G}\|_2 = |\gamma| \, \|\mathbf{G}\|_2 < 1$ for the value of γ in (76), thus Theorem 3 ensures the convergence of the algorithm irrespective of the value of μ. For the particular case of synchronous iterations, $\mu = N = 150$, Theorem 2 shows that the algorithm converges after $L = 3$ iterations, i.e., the algorithm reaches the error floor in (53) when $k\mu \geq 600$.

In order to examine the convergence behavior of the algorithm with a polynomial filter, we repeat the simulations done in Sections V-A and V-C with the filter realization in (77). That is, the noise variance is set to be $\sigma^2 = 10^{-16}$, and the algorithm is simulated for each value of $\mu = \{1, \ldots, N\}$. The corresponding mean-squared errors are presented in Fig. 8.

We note that the convergence behavior of the algorithm with the polynomial filter in (77) differs from that of the filter considered in Section V-A. In particular, the algorithm reaches the error floor after $k\mu \geq 600$ in the synchronous case (which is proven by Theorem 2), and *the algorithm gets slower as it gets more asynchronous.* When the results presented in Figs. 5 and 8 are considered together, a definite conclusion cannot be drawn regarding the effect of the asynchronicity on the convergence rate. Depending on the underlying graph filter, the asynchronicity may result in a faster or a slower convergence of the proposed algorithm.

VI. Conclusion

In this paper, we proposed a node-asynchronous implementation of rational graph filters, in which nodes on the graph follow a collect-compute-broadcast scheme in a randomized manner: in the passive stage a node only collects data, and when it gets activated randomly it runs a local state recursion for the filter, and then broadcasts its most recent value. In order to analyze the proposed method, we first studied a more general case of randomized asynchronous state recursions and presented a sufficiency condition that ensures the convergence in the mean-squared sense. Based on these results, we proved the convergence of the proposed algorithm in the mean-squared sense when the graph operator, the average update rate of the nodes and the filter of interest satisfy a certain condition. We simulated the proposed algorithm under different conditions and verified its convergence numerically.

Simulation results indicated that the presented sufficient condition is not necessary for the convergence of the algorithm. Moreover, the algorithm was observed to be robust to the communication failures between the nodes. It was observed also that the asynchronicity may increase the rate of convergence. Furthermore, simulations revealed that the proposed algorithm can converge even with an unstable filter if the nodes behave sufficiently asynchronously. Deeper theoretical analysis of some of these experimental observations is left for the future. For future studies, it would be interesting to consider the randomized asynchronous scenario in which nodes get updated depending on the values they have.

Appendix A
Proof of Theorem 1

The update model (14) can be written as follows:

$$\mathbf{x}_k = \sum_{i \notin \mathcal{T}_k} \mathbf{e}_i \, \mathbf{e}_i^{\mathrm{H}} \, \mathbf{x}_{k-1} + \sum_{i \in \mathcal{T}_k} \mathbf{e}_i \, \mathbf{e}_i^{\mathrm{H}} \, (\mathbf{S} \, \mathbf{x}_{k-1} + \mathbf{u} + \mathbf{w}_{k-1}) \,, \quad (78)$$

$$= \mathbf{x}_{k-1} + \mathbf{P}_{\mathcal{T}_k} \, (\, (\mathbf{S} - \mathbf{I}) \, \mathbf{x}_{k-1} + \mathbf{u} + \mathbf{w}_{k-1} \,) \,, \quad (79)$$

which can be re-written in terms of the residual vector $\mathbf{r}_k$ defined in (7) as follows:

$$\mathbf{r}_k = (\, \mathbf{I} + \mathbf{P}_{\mathcal{T}_k} (\mathbf{S} - \mathbf{I}) \,) \;\; \mathbf{r}_{k-1} + \mathbf{P}_{\mathcal{T}_k} \, \mathbf{w}_{k-1}. \quad (80)$$

Using the assumption that the residual vector $\mathbf{r}_{k-1}$, the index-selection matrix $\mathbf{P}_{\mathcal{T}_k}$ and the noise term $\mathbf{w}_{k-1}$ are uncorrelated with each other, and the assumption that $\mathbf{w}_{k-1}$ has a zero mean, the expected residual norm $\mathbf{r}_k$ conditioned on the previous residual $\mathbf{r}_{k-1}$ can be written as follows:

$$\mathbb{E} \left[\|\mathbf{r}_k\|_2^2 \mid \mathbf{r}_{k-1} \right] = \mathbb{E} \left[\left\| \left(\mathbf{I} + \mathbf{P}_{\mathcal{T}_k} (\mathbf{S} - \mathbf{I}) \right) \, \mathbf{r}_{k-1} \right\|_2^2 \right] + \mathbb{E} \left[\|\mathbf{P}_{\mathcal{T}_k} \, \mathbf{w}_{k-1}\|_2^2 \right]. \quad (81)$$

The first term on the right-hand-side of (81) can be written as follows:

$$\mathbb{E} \left[\mathbf{r}_{k-1}^{\mathrm{H}} \, \left(\mathbf{I} + (\mathbf{S}^{\mathrm{H}} - \mathbf{I}) \, \mathbf{P}_{\mathcal{T}_k} \right) \, \left(\mathbf{I} + \mathbf{P}_{\mathcal{T}_k} (\mathbf{S} - \mathbf{I}) \right) \, \mathbf{r}_{k-1} \right] \quad (82)$$

$$= \mathbf{r}_{k-1}^{\mathrm{H}} \, \left(\mathbf{I} + \mathbf{S}^{\mathrm{H}} \, \mathbf{P} \, \mathbf{S} - \mathbf{P} \right) \, \mathbf{r}_{k-1}, \quad (83)$$

which can be upper and lower bounded as $\Psi \, \|\mathbf{r}_{k-1}\|_2^2$ and $\psi \, \|\mathbf{r}_{k-1}\|_2^2$, respectively, where Ψ and ψ are defined as in (21).

The second term on the right-hand-side of (81) can be written as follows:

$$\mathbb{E} \left[\mathbf{w}_{k-1}^{\mathrm{H}} \mathbf{P}_{\mathcal{T}_k} \mathbf{w}_{k-1} \right] = \mathrm{tr} \left(\mathbb{E} \left[\mathbf{P}_{\mathcal{T}_k} \mathbf{w}_{k-1} \mathbf{w}_{k-1}^{\mathrm{H}} \right] \right) = \mathrm{tr}(\mathbf{P} \, \mathbf{\Gamma}), \quad (84)$$

where we use the fact that $\mathbf{P}_{\mathcal{T}_k}^{\mathrm{H}} \, \mathbf{P}_{\mathcal{T}_k} = \mathbf{P}_{\mathcal{T}_k}$, and the assumption that the noise and the index-selection are uncorrelated.

Thus, the conditional expected residual norm can be upper and lower bounded as follows:

$$\psi \, \|\mathbf{r}_{k-1}\|_2^2 \leq \mathbb{E} \left[\|\mathbf{r}_k\|_2^2 \, | \, \mathbf{r}_{k-1} \right] - \mathrm{tr}(\mathbf{P} \, \mathbf{\Gamma}) \leq \Psi \, \|\mathbf{r}_{k-1}\|_2^2. \quad (85)$$

Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on July 02,2021 at 16:04:36 UTC from IEEE Xplore. Restrictions apply.

By taking expectation of (85) with respect to the previous residual $\mathbf{r}_{k-1}$, we obtain the following:

$$\psi\, \mathbb{E}\left[\|\mathbf{r}_{k-1}\|_2^2\right] \leq \mathbb{E}\left[\|\mathbf{r}_k\|_2^2\right] - \mathrm{tr}(\mathbf{P}\,\mathbf{\Gamma}) \leq \Psi\, \mathbb{E}\left[\|\mathbf{r}_{k-1}\|_2^2\right]. \tag{86}$$

The iterative use of the inequalities in (86) yields the results given in (19) and (20).

Appendix B
Proof of Lemma 1

We first note that by left and right multiplying with $\mathbf{P}^{-1/2}$, the condition (22) can be equivalently written as:

$$\mathbf{S}^{\mathrm{H}}\,\mathbf{P}\,\mathbf{S} \prec \mathbf{P} \quad \Longleftrightarrow \quad \left\| \mathbf{P}^{1/2}\,\mathbf{S}\,\mathbf{P}^{-1/2} \right\|_2 < 1, \tag{87}$$

which proves the implication in (25). (Consider $\mathbf{P} = p\,\mathbf{I}$.)

We now prove the first implication of (24): Lemma 2.7.25 of [45] shows that $\rho(|\mathbf{S}|) < 1$ if and only if $|\mathbf{S}| \in \mathcal{D}_{\mathrm{d}}$. Since $|\mathbf{S}|$ is Schur diagonally stable, there exits a positive diagonal $\mathbf{P}$ such that $\|\mathbf{P}^{1/2}|\mathbf{S}|\mathbf{P}^{-1/2}\|_2 < 1$ due to (87). Then,

$$\left\|\mathbf{P}^{1/2}|\mathbf{S}|\mathbf{P}^{-1/2}\right\|_2 = \left\| \,|\mathbf{P}^{1/2}\mathbf{S}\mathbf{P}^{-1/2}|\, \right\|_2 \geq \left\|\mathbf{P}^{1/2}\mathbf{S}\mathbf{P}^{-1/2}\right\|_2, \tag{88}$$

where the equality follows from the fact that $\mathbf{P}$ is a positive diagonal matrix, and the inequality follows from the fact that $\|\,|\mathbf{X}|\,\|_2 \geq \|\mathbf{X}\|_2$ holds true for any matrix $\mathbf{X}$ [56]. Then, we have that $\|\mathbf{P}^{1/2}\mathbf{S}\mathbf{P}^{-1/2}\|_2 < 1$, which implies $\mathbf{S} \in \mathcal{D}_{\mathrm{d}}$ due to the equivalence in (87).

We now prove the second implication of (24): assume that $\mathbf{S} \in \mathcal{D}_{\mathrm{d}}$ and further assume that there exists an eigenpair $(\lambda, \mathbf{v})$ of $\mathbf{S}$ such that $|\lambda| \geq 1$. Then,

$$\mathbf{v}^{\mathrm{H}}\,\left(\mathbf{S}^{\mathrm{H}}\,\mathbf{P}\,\mathbf{S} - \mathbf{P}\right)\,\mathbf{v} = (|\lambda|^2 - 1)\,\mathbf{v}^{\mathrm{H}}\,\mathbf{P}\,\mathbf{v} \geq 0, \tag{89}$$

which contradicts with the assumption that $\mathbf{S}^{\mathrm{H}}\,\mathbf{P}\,\mathbf{S} - \mathbf{P}$ is negative definite. Thus, $\rho(\mathbf{S}) < 1$ must hold true.

Appendix C
Proof of Theorem 2

In what follows $\mathbf{I}_m$ denotes the $m \times m$ identity matrix. Since the state variables evolves according to (50) in the synchronous case, we can write the following due to (8):

$$\mathbb{E}\left[\overline{\mathbf{r}}_k\,\overline{\mathbf{r}}_k^{\mathrm{H}}\right] = \overline{\mathbf{A}}^k\,\overline{\mathbf{r}}_0\,\overline{\mathbf{r}}_0^{\mathrm{H}}\,(\overline{\mathbf{A}}^k)^{\mathrm{H}} + \sum_{n=0}^{k-1} \overline{\mathbf{A}}^n\,\overline{\mathbf{\Gamma}}\,(\overline{\mathbf{A}}^n)^{\mathrm{H}}, \tag{90}$$

where we define $\overline{\mathbf{r}}_k = \overline{\mathbf{x}}_k - \overline{\mathbf{x}}^\star$ similar to (7). Here $\overline{\mathbf{x}}^\star$ denotes the fixed point of the vectorized model in (50) (which exists since $\overline{\mathbf{A}}$ is assumed to not have eigenvalue 1), and it can be written as follows:

$$\begin{aligned}
\overline{\mathbf{x}}^\star &= \left(\mathbf{I}_{LN} - \overline{\mathbf{A}}\right)^{-1}\,\overline{\mathbf{u}} = (\mathbf{I}_{LN} - (\mathbf{A}\otimes\mathbf{G}))^{-1}\,(\mathbf{b}\otimes\mathbf{u}) \\
&= (\mathbf{T}^{-1}\otimes\mathbf{I}_N)\,(\mathbf{I}_{LN} - \widehat{\mathbf{A}}\otimes\mathbf{G})^{-1}\,(\widehat{\mathbf{b}}\otimes\mathbf{u}) \qquad (91)\\
&= (\mathbf{T}^{-1}\otimes\mathbf{I}_N)\,\mathrm{vec}\left(\begin{bmatrix}\mathbf{G}^{L-1}\,\mathbf{z} & \cdots & \mathbf{G}\,\mathbf{z} & \mathbf{z}\end{bmatrix}\right), \qquad (92)
\end{aligned}$$

where the vector $\mathbf{z} \in \mathbb{C}^N$ is defined as follows:

$$\mathbf{z} = q(\mathbf{G})^{-1}\,\mathbf{u}. \tag{93}$$

We note that the equivalence between (91) and (92) follows from the following identity:

$$\begin{bmatrix} \mathbf{I} & -\mathbf{G} & 0 & \cdots & 0 \\ 0 & \mathbf{I} & -\mathbf{G} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -\mathbf{G} \\ q_L\mathbf{G} & q_{L-1}\mathbf{G} & \cdots & q_2\mathbf{G} & \mathbf{I}+q_1\mathbf{G} \end{bmatrix} \begin{bmatrix} \mathbf{G}^{L-1}\,\mathbf{z} \\ \mathbf{G}^{L-2}\,\mathbf{z} \\ \vdots \\ \mathbf{G}\,\mathbf{z} \\ \mathbf{z} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ q(\mathbf{G})\mathbf{z} \end{bmatrix} \tag{94}$$

which can be written as follows:

$$(\mathbf{I}_{LN} - \widehat{\mathbf{A}}\otimes\mathbf{G})\mathrm{vec}\left(\begin{bmatrix}\mathbf{G}^{L-1}\,\mathbf{z} & \cdots & \mathbf{G}\,\mathbf{z} & \mathbf{z}\end{bmatrix}\right) = \widehat{\mathbf{b}}\otimes\mathbf{u}, \tag{95}$$

where we use the fact that $q(\mathbf{G})\,\mathbf{z} = \mathbf{u}$.

Line 9 of the algorithm together with the result of (44) shows that the output vector $\mathbf{y}_{(k)}$ defined in (42) is related to the vectorized state variables as follows:

$$\mathbf{y}_{(k)} = \mathbf{G}\,\mathbf{X}_{(k-1)}^{\mathrm{T}}\mathbf{c}^{\mathrm{T}} + d\,\mathbf{u}_{(k)} = (\mathbf{c}\otimes\mathbf{G})\,\overline{\mathbf{x}}_{k-1} + d\,\mathbf{u} + d\,\mathbf{w}_{(k)}. \tag{96}$$

Furthermore, the fixed point of the vectorized model given in (92) satisfies the following equality:

$$\begin{aligned}
(\mathbf{c}\otimes\mathbf{G})\,\overline{\mathbf{x}}^\star + d\,\mathbf{u} &= (\widehat{\mathbf{c}}\otimes\mathbf{G})\begin{bmatrix}\mathbf{G}^{L-1}\,\mathbf{z}\\ \vdots \\ \mathbf{G}\,\mathbf{z}\\ \mathbf{z}\end{bmatrix} + d\,\mathbf{u} \\
&= \sum_{n=1}^{L}(p_n - p_0\,q_n)\,\mathbf{G}^n\,\mathbf{z} + p_0\,q(\mathbf{G})\,\mathbf{z} = \sum_{n=0}^{L} p_n\,\mathbf{G}^n\,\mathbf{z} \\
&= p(\mathbf{G})\,\mathbf{z} = p(\mathbf{G})\,q(\mathbf{G})^{-1}\,\mathbf{u} = \widetilde{\mathbf{u}}. \qquad (97)
\end{aligned}$$

Combining (96) and (97), we can write the following:

$$\mathbf{y}_{(k)} - \widetilde{\mathbf{u}} = (\mathbf{c}\otimes\mathbf{G})(\overline{\mathbf{x}}_{k-1} - \overline{\mathbf{x}}^\star) + d\,\mathbf{w}_{(k)}. \tag{98}$$

Then,

$$\begin{aligned}
\mathbb{E}\left[\|\mathbf{y}_{(k)} - \widetilde{\mathbf{u}}\|_2^2\right] &= \mathrm{tr}\left((\mathbf{c}\otimes\mathbf{G})\,\mathbb{E}\left[\overline{\mathbf{r}}_{k-1}\,\overline{\mathbf{r}}_{k-1}^{\mathrm{H}}\right]\,(\mathbf{c}\otimes\mathbf{G})^{\mathrm{H}}\right) \\
&\quad + |d|^2\,\mathrm{tr}(\mathbf{\Gamma}), \\
&= \left\|(\mathbf{c}\otimes\mathbf{G})\,\overline{\mathbf{A}}^{k-1}\,\overline{\mathbf{r}}_0\right\|_2^2 + |d|^2\,\mathrm{tr}(\mathbf{\Gamma}) \\
&\quad + \sum_{n=0}^{k-2}\mathrm{tr}\left((\mathbf{c}\otimes\mathbf{G})\overline{\mathbf{A}}^n\,\overline{\mathbf{\Gamma}}\,(\overline{\mathbf{A}}^n)^{\mathrm{H}}\,(\mathbf{c}\otimes\mathbf{G})^{\mathrm{H}}\right), \qquad (99)
\end{aligned}$$

where we use the result in (90).

Due to (48), we note that $\overline{\mathbf{A}}^n = (\mathbf{A}\otimes\mathbf{G})^n = \mathbf{A}^n\otimes\mathbf{G}^n$. Furthermore, the augmented noise covariance matrix $\overline{\mathbf{\Gamma}}$ can be written explicitly from (48) as follows:

$$\overline{\mathbf{\Gamma}} = \mathbb{E}\left[\overline{\mathbf{w}}_k\,\overline{\mathbf{w}}_k^{\mathrm{H}}\right] = \mathbb{E}\left[(\mathbf{b}\otimes\mathbf{w}_k)\,(\mathbf{b}\otimes\mathbf{w}_k)^{\mathrm{H}}\right] = \mathbf{b}\mathbf{b}^{\mathrm{H}}\otimes\mathbf{\Gamma}. \tag{100}$$

Thus,

$$(\mathbf{c}\otimes\mathbf{G})\overline{\mathbf{A}}^n\,\overline{\mathbf{\Gamma}}\,(\overline{\mathbf{A}}^n)^{\mathrm{H}}(\mathbf{c}\otimes\mathbf{G})^{\mathrm{H}} = |\mathbf{c}\,\mathbf{A}^n\,\mathbf{b}|^2\,\mathbf{G}^{n+1}\,\mathbf{\Gamma}\,(\mathbf{G}^{n+1})^{\mathrm{H}}. \tag{101}$$

We also note that the coefficients of the impulse response of the underlying digital filter is given as follows [51,

Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on July 02,2021 at 16:04:36 UTC from IEEE Xplore. Restrictions apply.

Eq. (13.4.13)]:

$$h_n = \begin{cases} d, & n = 0, \\ \mathbf{c}\ \mathbf{A}^{n-1}\ \mathbf{b}, & n > 0. \end{cases} \tag{102}$$

Using (102) in (101), we can re-write (99) as follows:

$$\mathbb{E}\left[\|\mathbf{y}_{(k)} - \widetilde{\mathbf{u}}\|_2^2\right] = \left\|(\mathbf{c} \otimes \mathbf{G})\,\overline{\mathbf{A}}^{k-1}\,\overline{\mathbf{r}}_0\right\|_2^2 + |h_0|^2\,\mathrm{tr}(\mathbf{\Gamma}) + \sum_{n=0}^{k-2} |h_{n+1}|^2 \mathrm{tr}\left(\mathbf{G}^{n+1}\,\mathbf{\Gamma}\,(\mathbf{G}^{n+1})^{\mathrm{H}}\right), \tag{103}$$

which is equivalent to the result in (51).

Appendix D
Proof of Theorem 3

Let $\overline{\mathbf{P}} \in \mathbb{R}^{LN \times LN}$ denote the average index-selection matrix for the vectorized model in (47). Then, the structure of the update sets in (49) imply the following:

$$\overline{\mathbf{P}} = \mathbf{I}_L \otimes \mathbf{P}. \tag{104}$$

where $\mathbf{I}_L$ denotes the identity matrix of size L.

We now show that the following holds true:

$$\overline{\mathbf{A}}^{\mathrm{H}}\,\overline{\mathbf{P}}\,\overline{\mathbf{A}} \prec \overline{\mathbf{P}}, \tag{105}$$

which also ensures that $\overline{\mathbf{A}}$ is a stable matrix, hence the fixed point $\overline{\mathbf{x}}^{\star}$ computed in (92) exists. In this regard, using (48), (104), and the mixed-product property in (1), we can write (105) explicitly as follows:

$$(\mathbf{A}^{\mathrm{H}}\,\mathbf{A}) \otimes (\mathbf{G}^{\mathrm{H}}\,\mathbf{P}\,\mathbf{G}) \prec (\mathbf{I}_L \otimes \mathbf{P}). \tag{106}$$

Let $\mathbf{V}$ denote the eigenvectors of $\mathbf{A}^{\mathrm{H}}\mathbf{A}$. By left-multiplying with $\mathbf{V}^{\mathrm{H}} \otimes \mathbf{I}_N$ and right-multiplying with $\mathbf{V} \otimes \mathbf{I}_N$, the condition (106) can be written equivalently as follows:

$$\mathbf{\Sigma}_{\mathbf{A}}^2 \otimes (\mathbf{G}^{\mathrm{H}}\,\mathbf{P}\,\mathbf{G}) \prec (\mathbf{I}_L \otimes \mathbf{P}), \tag{107}$$

where $\mathbf{\Sigma}_{\mathbf{A}}$ is a diagonal matrix consisting of the singular values of the matrix $\mathbf{A}$. Since both sides of (107) are block diagonal matrices, the condition (107) holds true if and only if all of the individual blocks satisfy the inequality, that is,

$$\sigma_i^2(\mathbf{A})\ \mathbf{G}^{\mathrm{H}}\,\mathbf{P}\,\mathbf{G} \prec \mathbf{P} \tag{108}$$

for all singular values, $\sigma_i(\mathbf{A})$, of the matrix $\mathbf{A}$, which can be expressed explicitly as follows:

$$\sigma_{\min}^2(\mathbf{A})\ \mathbf{G}^{\mathrm{H}}\,\mathbf{P}\,\mathbf{G} \preceq \cdots \preceq \sigma_{\max}^2(\mathbf{A})\ \mathbf{G}^{\mathrm{H}}\,\mathbf{P}\,\mathbf{G} \prec \mathbf{P}. \tag{109}$$

Since (55) is both necessary and sufficient to satisfy (109), we conclude that (55) is equivalent to the condition in (105).

Since the assumption (55) ensures that (105) is satisfied, we can apply Corollary 1 to the model in (47) and conclude that

$$\lim_{k\to\infty} \mathbb{E}\left[\,\|\overline{\mathbf{x}}_k - \overline{\mathbf{x}}^{\star}\|_2^2\,\right] \leq \frac{\mathrm{tr}(\overline{\mathbf{P}}\,\overline{\mathbf{\Gamma}})}{\lambda_{\min}\left(\overline{\mathbf{P}} - \overline{\mathbf{A}}^{\mathrm{H}}\,\overline{\mathbf{P}}\,\overline{\mathbf{A}}\right)}. \tag{110}$$

Using the equality (98) from Appendix C:

$$\|\mathbf{y}_{(k)} - \widetilde{\mathbf{u}}\|_2^2 \leq \|\mathbf{c}\|_2^2\ \|\mathbf{G}\|_2^2\ \|\overline{\mathbf{x}}_{k-1} - \overline{\mathbf{x}}^{\star}\|_2^2 + \|d\,\mathbf{w}_{(k)}\|_2^2, \tag{111}$$

which results in the following:

$$\lim_{k\to\infty} \mathbb{E}\left[\|\mathbf{y}_{(k)} - \widetilde{\mathbf{u}}\|_2^2\right] \leq \frac{\|\mathbf{c}\|_2^2\ \|\mathbf{G}\|_2^2\ \mathrm{tr}(\overline{\mathbf{P}}\,\overline{\mathbf{\Gamma}})}{\lambda_{\min}\left(\overline{\mathbf{P}} - \overline{\mathbf{A}}^{\mathrm{H}}\,\overline{\mathbf{P}}\,\overline{\mathbf{A}}\right)} + |d|^2\,\mathrm{tr}(\mathbf{\Gamma}). \tag{112}$$

We have the following due to (100) of Appendix C and (104):

$$\mathrm{tr}(\overline{\mathbf{P}}\,\overline{\mathbf{\Gamma}}) = \mathrm{tr}\left(\mathbf{b}\mathbf{b}^{\mathrm{H}} \otimes \mathbf{P}\mathbf{\Gamma}\right) = \|\mathbf{b}\|_2^2\,\mathrm{tr}(\mathbf{P}\,\mathbf{\Gamma}). \tag{113}$$

We also note that (106) and (109) implies the following:

$$\lambda_{\min}\left(\overline{\mathbf{P}} - \overline{\mathbf{A}}^{\mathrm{H}}\,\overline{\mathbf{P}}\,\overline{\mathbf{A}}\right) = \lambda_{\min}\left(\mathbf{P} - \|\mathbf{A}\|_2^2\,\mathbf{G}^{\mathrm{H}}\,\mathbf{P}\,\mathbf{G}\right). \tag{114}$$

The use of (113) and (114) in (112) gives the desired result.

References

[1] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, no. 7, pp. 1644–1656, Apr. 2013.

[2] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs: Frequency analysis," *IEEE Trans. Signal Process.*, vol. 62, no. 12, pp. 3042–3054, Jun. 2014.

[3] A. Sandryhaila and J. M. F. Moura, "Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure," *IEEE Signal Process. Mag.*, vol. 31, no. 5, pp. 80–90, Sep. 2014.

[4] 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, May 2013.

[5] O. Teke and P. P. Vaidyanathan, "Extending classical multirate signal processing theory to graphs—Part I: Fundamentals," *IEEE Trans. Signal Process.*, vol. 65, no. 2, pp. 409–422, Jan. 2017.

[6] A. Sandryhaila and J. M. F. Moura, "Extending classical multirate signal processing theory to graphs—Part II: M-Channel filter banks," *IEEE Trans. Signal Process.*, vol. 65, no. 2, pp. 423–437, Jan. 2017.

[7] S. Narang and A. Ortega, "Perfect reconstruction two-channel wavelet filter banks for graph structured data," *IEEE Trans. Signal Process.*, vol. 60, no. 6, pp. 2786–2799, Jun. 2012.

[8] S. Narang and A. Ortega, "Compact support biorthogonal wavelet filterbanks for arbitrary undirected graphs," *IEEE Trans. Signal Process.*, vol. 61, no. 19, pp. 4673–4685, Oct. 2013.

[9] A. Anis, A. Gadde, and A. Ortega, "Towards a sampling theorem for signals on arbitrary graphs," in *Proc. Int. Conf. Acoust. Speech, Signal Process.*, May 2014, pp. 3864–3868.

[10] A. Gadde and A. Ortega, "A probabilistic interpretation of sampling theory of graph signals," in *Proc. Int. Conf. Acoust. Speech, Signal Process.*, Apr. 2015, pp. 3257–3261.

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

[12] A. Agaskar and Y. M. Lu, "A spectral graph uncertainty principle," *IEEE Trans. Inf. Theory*, vol. 59, no. 7, pp. 4338–4356, Jul. 2013.

[13] M. Tsitsvero, S. Barbarossa, and P. D. Lorenzo, "Signals on graphs: Uncertainty principle and sampling," *IEEE Trans. Signal Process.*, vol. 64, no. 18, pp. 4845–4860, Sep. 2016.

[14] O. Teke and P. P. Vaidyanathan, "Uncertainty principles and sparse eigenvectors of graphs," *IEEE Trans. Signal Process.*, vol. 65, no. 20, pp. 5406–5420, Oct. 2017.

[15] D. I. Shuman, P. Vandergheynst, D. Kressner, and P. Frossard, "Distributed signal processing via chebyshev polynomial approximation," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 4, no. 4, pp. 736–751, Dec. 2018.

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

[17] A. Sandryhaila, S. Kar, and J. M. F. Moura, "Finite-time distributed consensus through graph filters," in *Proc. Int. Conf. Acoust. Speech, Signal Process.*, May 2014, pp. 1080–1084.

[18] S. Segarra, A. G. Marques, and A. Ribeiro, "Distributed implementation of linear network operators using graph filters," in *Proc. Allerton Conf. Commun., Control, Comput.*, Sep. 2015, pp. 1406–1413.

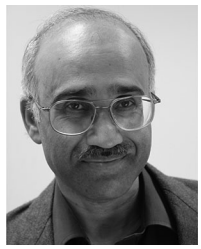
[19] X. Shi, H. Feng, M. Zhai, T. Yang, and B. Hu, "Infinite impulse response graph filters in wireless sensor networks," *IEEE Signal Process. Lett.*, vol. 22, no. 8, pp. 1113–1117, Aug. 2015.

Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on July 02,2021 at 16:04:36 UTC from IEEE Xplore. Restrictions apply.

[20] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, "Autoregressive moving average graph filtering," *IEEE Trans. Signal Process.*, vol. 65, no. 2, pp. 274–288, Jan. 2017.
[21] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, "Filtering random graph processes over random time-varying graphs," *IEEE Trans. Signal Process.*, vol. 65, no. 16, pp. 4406–4421, Aug. 2017.
[22] A. Loukas, A. Simonetto, and G. Leus, "Distributed autoregressive moving average graph filters," *IEEE Signal Process. Lett.*, vol. 22, no. 11, pp. 1931–1935, Nov. 2015.
[23] A. Loukas, "Distributed graph filters," Ph.D. dissertation, Delft Univ. Technol., Delft, The Netherlands, Mar. 2015.
[24] (2019) Giraph. [Online]. Available: https://giraph.apache.org
[25] (2019) Spark. [Online]. Available: https://spark.apache.org
[26] (2019) Dgraph. [Online]. Available: https://dgraph.io
[27] B. Awerbuch, "Complexity of network synchronization," *J. ACM*, vol. 32, no. 4, pp. 804–823, Oct. 1985.
[28] D. Chazan and W. Miranker, "Chaotic relaxation," *Linear Algebra Appl.*, vol. 2, pp. 199–222, Apr. 1969.
[29] G. M. Baudet, "Asynchronous iterative methods for multiprocessors," *J. ACM*, vol. 25, no. 2, pp. 226–244, Apr. 1978.
[30] D. P. Bertsekas, "Distributed asynchronous computation of fixed points," *Math. Program.*, vol. 27, no. 1, pp. 107–120, Sep. 1983.
[31] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Belmont, MA, USA: Athena Scientific, 1997.
[32] H. Avron, A. Druinsky, and A. Gupta, "Revisiting asynchronous linear solvers: Provable convergence rate through randomization," *J. ACM*, vol. 62, no. 6, pp. 51:1–51:27, Dec. 2015.
[33] Z. Peng, Y. Xu, M. Yan, and W. Yin, "Arock: An algorithmic framework for asynchronous parallel coordinate updates," *SIAM J. Scientific Comput.*, vol. 38, no. 5, pp. A2851–A2879, 2016.
[34] K. Avrachenkov, V. S. Borkar, and K. Saboo, "Distributed and asynchronous methods for semi-supervised learning," in *Proc. Int. Workshop Algorithms Models Web-Graph*, 2016, pp. 34–46.
[35] K. Avrachenkov, V. S. Borkar, and K. Saboo, "Parallel and distributed approaches for graph based semi-supervised learning," Inria, Tech. Rep. 8767, Centre Inria Sophia Antipolis – Méditerranée, Aug. 2015.
[36] O. Teke and P. P. Vaidyanathan, "Random node-asynchronous updates on graphs," *IEEE Trans. Signal Process.*, vol. 67, no. 11, pp. 2794–2809, Jun. 2019.
[37] O. Teke and P. P. Vaidyanathan, "The random component-wise power method," in *Proc. SPIE, Wavelets Sparsity XVIII*, Sep. 2019, vol. 11138, pp. 473–488.
[38] O. Teke and P. P. Vaidyanathan, "Node-asynchronous implementation of rational filters on graphs," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2019, pp. 7530–7534.
[39] O. Teke and P. P. Vaidyanathan, "Node-asynchronous spectral clustering on directed graphs," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2020, pp. 5325–5329.
[40] T. Kailath, A. H. Sayed, and B. Hassibi, *Linear Estimation*. Englewood Cliffs, NJ, USA: Prentice-Hall, 2000.
[41] W. Mills, C. Mullis, and R. Roberts, "Digital filter realizations without overflow oscillations," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-26, no. 4, pp. 334–338, Aug. 1978.
[42] P. Vaidyanathan and V. Liu, "An improved sufficient condition for absence of limit cycles in digital filters," *IEEE Trans. Circuits Syst.*, vol. CAS-34, no. 3, pp. 319–322, Mar. 1987.
[43] E. Kaszkurewicz, A. Bhaya, and D. Siljak, "On the convergence of parallel asynchronous block-iterative computations," *Linear Algebra Appl.*, vol. 131, pp. 139–160, 1990.
[44] A. Bhaya and E. Kaszkurewicz, "On discrete-time diagonal and D-stability," *Linear Algebra Appl.*, vol. 187, pp. 87–104, 1993.
[45] E. Kaszkurewicz and A. Bhaya, *Matrix Diagonal Stability in Systems and Computation*. Cambridge, MA, USA: Birkhäuser, 2000.
[46] O. Teke and P. P. Vaidyanathan, "The asynchronous power iteration: A graph signal perspective," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Apr. 2018, pp. 4059–4063.
[47] O. Teke and P. P. Vaidyanathan, "Asynchronous nonlinear updates on graphs," in *Proc. Asilomar Conf. Signals, Syst., Comput.*, Oct. 2018, pp. 998–1002.
[48] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proc. Int. Conf. Mach. Learn.*, 2003, pp. 912–919.
[49] K. Avrachenkov, A. Mishenin, P. Gonalves, and M. Sokol, "Generalized optimization framework for graph-based semi-supervised learning," in *Proc. SIAM Int. Conf. Data Mining*, 2012, pp. 966–974.
[50] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 1994.
[51] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1993.
[52] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Neural Inf. Process. Syst.*, 2017, pp. 1024–1034.
[53] C. Barnes and A. Fam, "Minimum norm recursive digital filters that are free of overflow limit cycles," *IEEE Trans. Circuits Syst.*, vol. CAS-24, no. 10, pp. 569–574, Oct. 1977.
[54] P. P. Vaidyanathan, "Low-noise and low-sensitivity digital filters," in *Handbook of Digital Signal Processing*, D. F. Elliott, Ed. New York, NY, USA: Academic, 1987, pp. 359–479.
[55] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 1990.
[56] R. Mathias, "The spectral norm of a nonnegative matrix," *Linear Algebra Appl.*, vol. 139, pp. 269–284, 1990.

Oguzhan Teke (Student Member, IEEE) received the B.S. and the M.S. degree in electrical and electronics engineering both from Bilkent University, Turkey in 2012, and 2014, respectively. He is expected to receive the Ph.D. degree in electrical engineering in 2020 at the California Institute of Technology, USA.

His research interests include signal processing, graph signal processing, and randomized algorithms.



Palghat P. Vaidyanathan (Life Fellow, IEEE) was born in Calcutta, India on Oct. 16, 1954. He received the B.Sc. (Hons.) degree in physics and the B.Tech. and M.Tech. degrees in radiophysics and electronics, all from the University of Calcutta, India, in 1974, 1977 and 1979, respectively, and the Ph.D. degree in electrical and computer engineering from the University of California at Santa Barbara in 1982. He was a Postdoctoral Fellow at the University of California, from Sep. 1982 to Mar. 1983. In Mar. 1983, he joined the Electrical Engineering Department of the California Institute of Technology as an Assistant Professor, and is currently the Kio and Eiko Tomiyasu Professor of electrical engineering there. His main research interests are in digital signal processing, multirate systems, wavelet transforms, signal processing for digital communications, genomic signal processing, and sparse array signal processing.

Prof. Vaidyanathan has authored over 550 papers in journals and conferences, and is the author/coauthor of the four books: Multirate systems and filter banks, Prentice Hall, 1993, Linear Prediction Theory, Morgan and Claypool, 2008, and (with Phoong and Lin) Signal Processing and Optimization for Transceiver Systems, Cambridge University Press, 2010, and Filter Bank Transceivers for OFDM and DMT Systems, Cambridge University Press, 2010. He was a recipient of the award for excellence in teaching at the California Institute of Technology multiple times. In 2016, he received the Northrup Grumman Prize for excellence in Teaching at Caltech. He also received the NSF's Presidential Young Investigator award in 1986. About 15 of his papers have received prizes from IEEE and other organizations. Dr. Vaidyanathan was elected Fellow of the IEEE in 1991, and is now a Life Fellow. He received the 1995 F. E. Terman Award of the American Society for Engineering Education, sponsored by Hewlett Packard for his contributions to engineering education. He has given several plenary talks at several conferences and has been chosen a Distinguished Lecturer for the IEEE Signal Processing Society for the year 1996–97. In 1999, he was chosen to receive the IEEE CAS Society's Golden Jubilee Medal. He is a recipient of the IEEE Gustav Kirchhoff Award (an IEEE Technical Field Award) in 2016. He is a recepient of the IEEE Signal Processing Society's Technical Achievement Award (2002), Education Award (2012), and the Society Award (2016). In 2019, he was elected to the U.S. National Academy of Engineering.

Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on July 02,2021 at 16:04:36 UTC from IEEE Xplore. Restrictions apply.