ASYNCHRONOUS DISTRIBUTED EDGE-VARIANT GRAPH FILTERS

Mario Coutino, Geert Leus

Delft University of Technology
Delft, The Netherlands

ABSTRACT

As the size of the sensor network grows, synchronization starts to become the main bottleneck for distributed computing. As a result, efforts in several areas have been focused on the convergence analysis of asynchronous computational methods. In this work, we aim to cross-pollinate distributed graph filters with results in parallel computing to provide guarantees for asynchronous graph filtering. To alleviate the possible reduction of convergence speed due to asynchronous updates, we also show how a slight modification to the graph filter recursion, through operator splitting, can be performed to obtain faster convergence. Finally, through numerical experiments the performance of the discussed methods is illustrated.

Index Terms— asynchronous filtering, distributed signal processing, edge-variant graph filters, graph filters, graph signal processing.

1. INTRODUCTION

In recent years, graph filters (GFs) [1] have become the workhorse for processing data living on irregular domains, e.g., social networks [2], biological structures [3], etc. Extending fundamental notions of signal processing, graph signal processing [4] leverages such graph structures to perform filtering [5], estimation [6, 7] and detection [8, 9] tasks.

In order to improve scalability, the *locality* of such graph filters is exploited. That is, graph filters, in their finite impulse response (FIR) [10] and infinite impulse response (IIR) [11] variants, can be implemented distributively in order to reduce the computational cost incurred by centralized operations. Furthermore, recent advances in graph filters [12] have shown that by an appropriate selection of edge weights, constrained edge-variant (CEV) GFs are able to achieve a significant reduction in communication costs, i.e., local exchanges, while maintaining the distributed implementation.

Several works have been devoted to implementing and designing graph filters [1, 5, 11, 13], but all of them assume synchronization. This assumption presents a real challenge as most practical distributed systems are not synchronous. Hence, asynchronous methods are of the utmost importance for overcoming the lack of synchronization of (practical) distributed systems.

Extensive research has been carried out within the realm of asynchronous methods [14–18]. Results from distributed computation have shown, both analytically and experimentally, that asynchronous methods, for linear problems, *can be faster* than synchronous methods, and that for certain problems, they *converge* even when its synchronous counterpart does not [19].

In this paper, we provide conditions and analytical guarantees for the convergence of a family of asynchronous distributed CEV GFs. Differently from other works, we provide both strong and weak guarantees for the convergence in the asynchronous regime without invoking diagonal dominance or its relation with inexact multiplicative block relaxation [19]. We introduce our results through lemmas with sufficient conditions that facilitate the selection of filter coefficients during the design stage. In addition, our results hold for both exact and inexact, i.e., in a noisy setting, update regimes.

2. LINEAR OPERATORS AS GRAPH FILTERS

Let us consider the following linear system

$$\boldsymbol{A}\boldsymbol{y} = \boldsymbol{x} \in \mathbb{R}^N, \tag{1}$$

and its solution $\boldsymbol{y}_{\rm sol} = \boldsymbol{A}^{-1}\boldsymbol{x}$. Such systems arise in many kind of network processes and physical problems such as convection-diffusion systems [20], partial differential equations on graphs [21] or network inference problems [22].

In many instances, the solution is computed *iteratively* using the following recurrence relation [23]

$$\boldsymbol{y}_{l+1} = \boldsymbol{y}_l + (\boldsymbol{x} - \boldsymbol{A}\boldsymbol{y}_l), \tag{2}$$

for some initial vector $\boldsymbol{y}_0$ (typically the zero vector). For this recurrence, under certain conditions on the matrix $\boldsymbol{A}$ [23], we can guarantee that

$$\lim_{l\to\infty} \boldsymbol{y}_l = \boldsymbol{A}^{-1}\boldsymbol{x} = \boldsymbol{y}_{\rm sol}. \tag{3}$$

This iterative procedure is performed to alleviate the cost of performing the matrix (pseudo)inversion, to leverage the parallelizable matrix structure or to satisfy communication constraints.

In many network inference problems such as Tikhonov interpolation [22], $\boldsymbol{A}$ has a structure that can be expressed by *local matrices*. That is, the linear operator $\boldsymbol{A}$ can be represented through local operations. To illustrate this scenario, we consider the following filter operation $\boldsymbol{y}_{\rm sol} = \boldsymbol{A}^{-1}\boldsymbol{x} = \boldsymbol{H}\boldsymbol{x}$, where $\boldsymbol{H}$ is a so-called CEV GF, i.e.,

$$\boldsymbol{H} = (\boldsymbol{I} - \sum_{k=1}^{K} \boldsymbol{\Phi}_k \boldsymbol{S}^{k-1})^{-1}, \tag{4}$$

with $\boldsymbol{S} \in \mathbb{R}^{N\times N}$ the so-called *graph shift operator* [4], i.e., the matrix representation of the network (graph), and $\boldsymbol{\Phi}_k \in \mathbb{R}^{N\times N}$ an edge-weighting matrix which shares the support with $\boldsymbol{S}+\boldsymbol{I}$ (both $\boldsymbol{S}$ and $\boldsymbol{\Phi}_k$ can be implemented distributively).

In some problems though, $\boldsymbol{A}$ does not exhibit parallelizable features over the communication graph of the distributed system. To

The authors are with the faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, 2826 CD Delft, The Netherlands. This research is supported in part by the ASPIRE project (project 14926 within the STW OTP programme), financed by the Netherlands Organization for Scientific Research (NWO). Mario Coutino is partially supported by CONACYT.

deal with this, a way to fully distribute the operations (and meet the communication constraints) is to approximate the system matrix inverse by a cascade of two CEV GFs, one with the form of (4) and the other with its inverse form. More specifically, we then compute the (approximate) solution of the system true local operations as

$$\boldsymbol{y}_{\text{sol}} = \boldsymbol{A}^{-1}\boldsymbol{x} = \boldsymbol{H_B}\boldsymbol{H}_{\text{A}}\boldsymbol{x}, \tag{5}$$

where $\boldsymbol{H}_{\text{A}}$ has the form of the CEV GF (4) and $\boldsymbol{H}_{\text{B}}$ has the inverse form of (4). Notice that this structure can be viewed as the output of an autoregressive moving average (ARMA) CEV GF [1], matching the traditional ARMA GF definition if the matrices $\boldsymbol{H}_{\text{A}}$ and $\boldsymbol{H}_{\text{B}}$ commute.

In this work, we provide guarantees for ensuring the convergence of iterative methods of the form (2) when there is lack of synchronization between the processing units (nodes) for structured linear systems as (5). To do so, we first build lemmas around the asynchronous application of the GF $\boldsymbol{H}_{\text{A}}$, and then, we consider the subsequent application of $\boldsymbol{H}_{\text{B}}$.

3. NODE UPDATE MODEL

First, let us consider $\boldsymbol{y}^*$ as the output of the filter operation

$$\boldsymbol{y}^* = \boldsymbol{H}_{\text{A}}\boldsymbol{x}, \tag{6}$$

or equivalently, as the solution of the linear system

$$(\boldsymbol{I} - \sum_{k=1}^{K} \boldsymbol{\Phi}_k \boldsymbol{S}^{k-1})\boldsymbol{y}^* = \boldsymbol{x}. \tag{7}$$

Using the structure of $\boldsymbol{H_A}$ and its relation to the solution of a linear system [cf. (7)], we can substitute the equivalent system matrix in the recurrence relation (2) and obtain

$$\begin{aligned} \boldsymbol{y}_{l+1} &= \boldsymbol{x} + \sum_{k=1}^{K} \boldsymbol{\Phi}_k \boldsymbol{S}^{k-1}\boldsymbol{y}_l \\ &= \boldsymbol{x} + \boldsymbol{B}\boldsymbol{y}_l, \end{aligned} \tag{8}$$

From (8), we observe that convergence is guaranteed when the *iteration matrix*, i.e., $\boldsymbol{B}$, has a spectral radius, $\rho(\boldsymbol{B})$, strictly smaller than one. This result can be obtained by observing that the error $\boldsymbol{e}_l = \boldsymbol{y}^* - \boldsymbol{y}_l$ at the lth iteration is given by

$$\boldsymbol{e}_{l+1} = \boldsymbol{B}\boldsymbol{e}_l. \tag{9}$$

Hence, the error is guaranteed to vanish (asymptotically) when $\rho(\boldsymbol{B}) < 1$.

In the recurrence expression (8), we observe that a graph filter of order K is required to update the current solution. This implies that a set of K communication rounds are required *before* recursion (8) can be updated. Hence, this model might not be suitable when disruptions in the communication among nodes occur. A possible way to deal with such a problem is to endow the nodes with *memory* as in the case of linear recursive sequences [24] where based on the characteristic polynomial of the recurrence, it is possible to generate the sequence using the so-called companion matrix [25] and the previous values in the series. Following this idea, we can consider that each diffusion step is stored and is available for transmission. More formally, by defining the kth shift of the recurrence vector as

$$\boldsymbol{y}_l^{(k)} = \boldsymbol{S}\boldsymbol{y}_l^{(k-1)}, \tag{10}$$

and stacking the vectors $\{\boldsymbol{y}_l^{(k)}\}_{k=0}^{K-1}$ in the column vector $\bar{\boldsymbol{y}}_l \in \mathbb{R}^{NK}$, we can write an extended recurrence equation as follows

$$\begin{aligned} \bar{\boldsymbol{y}}_{l+1} &= \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \end{bmatrix} + \left[\begin{array}{cccc|c} \boldsymbol{\Phi}_1 & \boldsymbol{\Phi}_2 & \cdots & & \boldsymbol{\Phi}_K \\ \hline \boldsymbol{S} & & & & \boldsymbol{0} \\ & & \ddots & & \vdots \\ & & & \boldsymbol{S} & \boldsymbol{0} \end{array}\right] \bar{\boldsymbol{y}}_l \\ &= \bar{\boldsymbol{x}} \quad + \quad \bar{\boldsymbol{B}}\bar{\boldsymbol{y}}_l, \end{aligned} \tag{11}$$

for some initial vector $\bar{\boldsymbol{y}}_0$ (typically the zero vector). Here, the first N elements of $\bar{\boldsymbol{y}}_{l+1}$ and $\bar{\boldsymbol{y}}_l$ are the vectors $\boldsymbol{y}_{l+1}$ and $\boldsymbol{y}_l$ [cf. (8)], respectively.

Despite that (11) provides a recurrence equation for the node variables, (11) does not directly reveal the locality of the operations. To show this, let us stack $\boldsymbol{y}_l$ row-wise in the $K \times N$ matrix $\boldsymbol{Y}_l$, i.e.,

$$\boldsymbol{Y}_l := [\boldsymbol{y}_l^{(0)}, \ldots, \boldsymbol{y}_l^{(K-1)}]^T, \tag{12}$$

so that we can interpret the nth column of $\boldsymbol{Y}_l$ as the nth node variable related to the lth iteration. Then, (11) can be rewritten as

$$\boldsymbol{Y}_{l+1} = \begin{bmatrix} \boldsymbol{x}^T \\ \boldsymbol{0} \end{bmatrix} + \begin{bmatrix} \sum_{k=1}^{K} [\boldsymbol{Y}_l \boldsymbol{\Phi}_k^T]_{k,:} \\ \boldsymbol{Y}_l \boldsymbol{S}^T \end{bmatrix}, \tag{13}$$

which shows that the nth node variable can be updated solely by combining the node variables of the neighbors of node n. As a result, by a mere exchange of local variables among neighboring nodes, convergence of (11) can be achieved in a distributed fashion. To simplify the further discussion, we will only work with (11) from now on and keep in mind that we can always distribute it following (13).

From the recurrence relation in (11), we note that the condition for convergence is now $\rho(\bar{\boldsymbol{B}}) < 1$. Despite the fact that this condition on the spectrum of the iteration matrix guarantees the convergence of (11), we can derive a *weaker* sufficient condition for convergence, involving the matrices that comprise $\bar{\boldsymbol{B}}$. The following lemma provides such a guarantee.

Lemma 1. *Consider the recurrence equation*

$$\bar{\boldsymbol{y}}_{l+1} = \bar{\boldsymbol{x}} + \bar{\boldsymbol{B}}\bar{\boldsymbol{y}}_l,$$

where $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{B}}$ are defined as in (11)*. Then, for matrices $\boldsymbol{S}$ and $\{\boldsymbol{\Phi}_k\}_{k=1}^{K}$ such that*

$$\rho \leq 1, \qquad \gamma := a\frac{1-\rho^K}{1-\rho} < 1,$$

where $\rho := \|\boldsymbol{S}\|$ and $a := \max_{k\in\{1,\ldots,K\}} \|\boldsymbol{\Phi}_k\|$, as $l \to \infty$, the recurrence converges and the first N elements of $\bar{\boldsymbol{y}}_l$ converge to $\boldsymbol{y}^*$ in the induced operator $\|\ \|$-norm chosen for ρ and a.

Proof. See Appendix[1]. □

This guarantee provides a certificate of convergence, in any norm, as long as the conditions are met. Conditions on norms other than the spectral norm are usually computationally easier to guarantee (and check). This leads to simpler optimization problems, i.e., we could avoid to solve a semidefinite program (due to the spectral norm) during the graph filter design stage.

[1]Appendix available in the extended version: `http://cas.et.tudelft.nl/~mariocoutino/`

As we explained before, (11) requires the exchange of information from neighbors. In most instances, these exchanges are *inexact*. That is, errors in the communication between nodes corrupt the information transmitted in the network. Hence, the updates are not performed correctly. In the following lemma, we show weak sufficient conditions for the ϵ-approximate convergence of the iterates towards the optimal solution in the noisy setting.

Lemma 2. *Consider the noisy recurrence equation*

$$\bar{\boldsymbol{y}}_{l+1}^{\mathrm{in}} = \bar{\boldsymbol{x}} + \bar{\boldsymbol{B}}\bar{\boldsymbol{y}}_l^{\mathrm{in}} + \boldsymbol{v}_l,$$

where $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{B}}$ are defined as in (11) *and $\boldsymbol{v}_l$ is a perturbation from the true update due to communication or fixed-precision errors satisfying*

$$\|\boldsymbol{v}_l\| \leq \beta, \ \forall \ l \in \mathbb{N}.$$

Then, for $\boldsymbol{S}$ and $\{\boldsymbol{\Phi}_k\}_{k=1}^K$ meeting the conditions of Lemma 1, as $l \to \infty$, the first N elements of $\bar{\boldsymbol{y}}_l^{\mathrm{in}}$ eventually lie within a $\| \ \|$-ball centered at $\boldsymbol{y}^*$ of radius

$$\epsilon = \frac{\beta}{1-\gamma},$$

with γ defined in Lemma 1.

Proof. See Appendix. □

From this result, we observe that if there is noise during the transmission, i.e., inexact updates, we gravitate around the desired output. Although, in general, this result might look discouraging, we show that if the perturbation is *structured*, as in the case of noiseless asynchronous updates, the recurrence asymptotically converges to the desired output.

4. ASYNCHRONOUS UPDATES

So far we have only considered synchronous updates. To model asynchronous updates, we can recast the simple recurrence (11) into its asynchronous version

$$\bar{\boldsymbol{y}}_{l+1}^{\mathrm{a}} = [\boldsymbol{D}_l\bar{\boldsymbol{x}} + \boldsymbol{D}_l\bar{\boldsymbol{B}}\bar{\boldsymbol{y}}_l^{\mathrm{a}}] + [(\boldsymbol{I}-\boldsymbol{D}_l)\bar{\boldsymbol{y}}_l^{\mathrm{a}}], \tag{14}$$

where the first term corresponds to the updated entries, and the second term denotes the entries that remain unchanged. In (14), $\boldsymbol{I}$ is the $KN \times KN$ identity matrix and

$$\boldsymbol{D}_l := \mathrm{diag}(\boldsymbol{w}_l) \in \{0,1\}^{KN\times KN}.$$

Here, $\mathrm{diag}(\cdot)$ constructs a diagonal matrix using its argument, $\boldsymbol{w}_l = [(\boldsymbol{w}_l^{(0)})^T, \ldots, (\boldsymbol{w}_l^{(K-1)})^T]^T$, where $\boldsymbol{w}_l^{(k)}$ is an N-binary vector with $[\boldsymbol{w}_l^{(k)}]_i = 1$ if the ith entry of $\boldsymbol{y}_l^{(k)}$ is updated or zero otherwise. Notice that no further structure is enforced on $\boldsymbol{D}_l$. This setting encompasses cases where not all memory entries are updated within a node, i.e., asynchronous updates within nodes.

Using the asynchronous update model (14), in the following lemma, we provide sufficient guarantees for the convergence of the recurrence using properties of the structured perturbation corresponding to the asynchronous update model.

Lemma 3. *Let each node update its local variable using the asynchronous recurrence* (14). *Further, let $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{B}}$ be defined as in* (11) *and let the sequence matrices $\boldsymbol{D}_l$ be sufficiently exciting, i.e., $\sum_l^\infty [\boldsymbol{D}_l]_{i,i} \gg 0 \, \forall \, i$. Then, for $\boldsymbol{S}$ and $\{\boldsymbol{\Phi}_k\}_{k=1}^K$ meeting the conditions of Lemma 1, $(\bar{\boldsymbol{y}}_l^{\mathrm{a}})^{(0)}$ converges to $\boldsymbol{y}^*$ as $l \to \infty$.*

Proof. See Appendix. □

In Lemma 3, the condition on the sequence $\boldsymbol{D}_l$ enforces a sufficient exchange of information over the network. That is, all the entries are sufficiently seen, i.e., updated, throughout the recurrences.

After these results, we are ready to prove the main theorem with respect to the behavior of asynchronous constrained edge-variant graph filters.

Theorem 1. *Let a network perform the filtering operation*

$$\boldsymbol{y}_{\mathrm{sol}} = \boldsymbol{H}\boldsymbol{x} = \boldsymbol{H}_{\mathrm{B}}\boldsymbol{H}_{\mathrm{A}}\boldsymbol{x},$$

where $\boldsymbol{H}_{\mathrm{A}}$ is a filter of order K of the form (4) *and $\boldsymbol{H}_{\mathrm{B}}$ is a filter of order P of the inverse form of* (4). *Then, if the involved matrices $\boldsymbol{S}$ and $\{\boldsymbol{\Phi}_k\}_{k=1}^K$ in the graph filter $\boldsymbol{H}_{\mathrm{A}}$ and the sequence of matrices $\boldsymbol{D}_l$ satisfy the conditions of Lemma 3, the asynchronous implementation of $\boldsymbol{H}$ converges to $\boldsymbol{y}_{\mathrm{sol}}$.*

Proof. We can apply $\boldsymbol{H}_{\mathrm{A}}$ and use the results of Lemma 3 to show that the filtering operation converges to the desired output. Further, as each node is endowed with memory, as long as they can communicate with their neighbors, they can locally compute the output of $\boldsymbol{H}_{\mathrm{B}}$ when the filter order, P, of $\boldsymbol{H}_{\mathrm{B}}$ is lower than or equal to K. In the case that $P > K$, we simply increase the memory of $\boldsymbol{H}_{\boldsymbol{A}}$ and set the corresponding matrices $\boldsymbol{\Phi}_k$ to zero. □

Corollary 1. *The exchange of exact updates for inexact updates, as defined in Lemma 2, in the $\boldsymbol{H}_{\mathrm{A}}$ filter operation defined in Theorem 1 forces the sequence, as $l \to \infty$, to eventually lie within a $\| \ \|$-ball centered at $\boldsymbol{y}_{\mathrm{sol}}$ of radius*

$$\epsilon_{\boldsymbol{H}_{\mathrm{B}}} = \|\boldsymbol{H}_{\boldsymbol{B}}\|\frac{\beta}{1-\gamma}.$$

The result from Theorem 1 provides an interesting insight with respect to the cascade application of general asynchronous graph filters. That is, in the case of *classical* graph filters, where any pair of GFs commute, this result implies that any ARMA GF can be implemented asynchronously. However, for a more general family of graph filters, e.g., node-variant [13] or edge-variant [1], the ordering of the MA and AR of the GF is *critical*.

4.1. Classical Graph Filter Asynchronous Case

Here, we particularize our results to the case of the simplest GF: the classical GF. To do so, we replace each $\boldsymbol{\Phi}_k$ by $\alpha_k\boldsymbol{S}$ in (11) and obtain the following relation

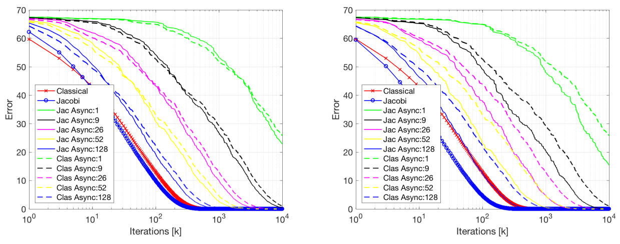
$$\bar{\boldsymbol{y}}_{l+1} = \bar{\boldsymbol{x}} + (\boldsymbol{C}_q^T \otimes \boldsymbol{S})\bar{\boldsymbol{y}}_l, \tag{15}$$

where

$$\boldsymbol{C}_q = \begin{bmatrix} \phi_1 & 1 & 0 & \cdots & 0 \\ \phi_2 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ \phi_{K-1} & 0 & 0 & \cdots & 1 \\ \phi_K & 0 & 0 & \cdots & 0 \end{bmatrix} \tag{16}$$

is the companion matrix (with mirrored rows and columns w.r.t the standard definition) of the polynomial q defined as

$$q(t) = t^K - \sum_{i=1}^K \phi_i t^{K-i}. \tag{17}$$



(a) (b)

Fig. 1: Convergence rate for different ARMA$_1$ implementations. (a) Jacobi updates with $\boldsymbol{M} = 4\mathrm{diag}(\boldsymbol{H}^{-1})$. (b) Jacobi updates with $\boldsymbol{M} = 2.5\mathrm{diag}(\boldsymbol{H}^{-1})$. Here, $\mathrm{diag}(\cdot)$ denotes the diagonal part of the matrix argument and the number in the legend corresponds to the number of coordinated nodes.

Using (15), we can state that the recursion converges if

$$\rho(\boldsymbol{C}_q)\rho(\boldsymbol{S}) < 1. \tag{18}$$

For a normalized shift, i.e., $\rho(\boldsymbol{S}) = 1$, the condition reduces to $\rho(\boldsymbol{C}_q) < 1$. Therefore, by noticing that the eigenvalues of $\boldsymbol{C}_q$ are the same as the roots of $q(t)$, we can perform *stable* filter design for the recursion (15) by enforcing constraints on the roots of $q(t)$. Notice that if (18) is satisfied, all the results from the lemmas follow.

In the following section, we present a series of numerical experiments demonstrating the behavior of the asynchronous edge-variant graph filters. In addition, operator splitting is discussed to complement the discussion on distributed GFs.

5. NUMERICAL SIMULATION

In this section, we illustrate the convergence results for general graph filters and briefly discuss the splitting of the system to provide an improvement in convergence speed. In literature [23], recurrence equations like (2) are sometimes referred to as *splitting methods*. The term comes from the fact that (2) can be rewritten as

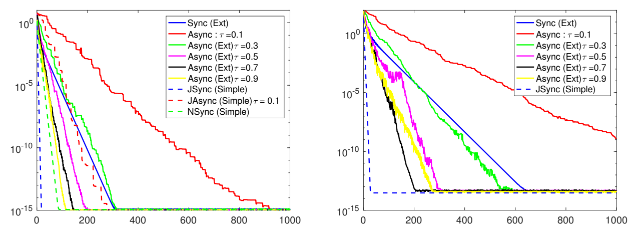
$$\boldsymbol{y}_{l+1} = (\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})\boldsymbol{y}_l + \boldsymbol{M}^{-1}\boldsymbol{x}, \tag{19}$$

where $\boldsymbol{A} = \boldsymbol{M} - \boldsymbol{N}$. Notice that in this case, the iteration matrix $\boldsymbol{B}$ is given by $\boldsymbol{M}^{-1}\boldsymbol{N}$. Hence, the (asymptotic) convergence rate is now controlled by $\rho(\boldsymbol{M}^{-1}\boldsymbol{N})$ instead of $\rho(\boldsymbol{I} - \boldsymbol{A})$.

Typical choices for $\boldsymbol{M}^{-1}$ are the diagonal part and the lower triangular part of $\boldsymbol{A}$. The former and the latter choices lead to the so-called *Jacobi* and *Gauss-Seidel* methods, respectively. While it is well-known that Gauss-Seidel exhibits better convergence properties, the Jacobi method is, in general, the only one that can be efficiently implemented in a distributed fashion, i.e., the inversion of a diagonal matrix is easy to distribute. In the following example, we show a comparison between the classical implementation and the Jacobi implementation for different numbers of random coordinate nodes, i.e., nodes that update at the same time. Here, we consider that the underlying graph is a community graph with $N = 256$ nodes (generated using the GSPToolbox [26]) and we assume a classical ARMA$_1$ [11]

$$\boldsymbol{A} = 1/\varphi(\boldsymbol{I} - \psi\boldsymbol{S}) \rightarrow \boldsymbol{H} = \varphi(\boldsymbol{I} - \psi\boldsymbol{S})^{-1}, \tag{20}$$

with $\varphi = 1$ and $\psi = 1/(\rho(\boldsymbol{S}) + \epsilon)$. In this case, $\rho(\boldsymbol{I} - \boldsymbol{H}) = 0.9903$. In Fig. 1, we show a comparison between two different splittings and the algorithm without splitting. We can see how the Jacobi implementation converges faster than the implementation without splitting, while maintaining its distributed nature. In addition, by a proper selection, i.e., optimizing $\rho(\boldsymbol{M}^{-1}\boldsymbol{N})$, we observe that the convergence speed can be increased. In this example, it is shown that the asynchronous implementation always converges independently of the number of nodes that are updated at each iteration as long as there is sufficient communication among the network.



(a) (b)

Fig. 2: (a) Results for the time-varying Poisson equation. (b) Results for the ARMA graph filter. A community graph with $N = 64$ nodes has been used for the simulations.

In our analysis, we have provided guarantees for the behavior of asynchronous GFs for general norms. However, we want to remark that despite the fact that global properties, such as bounded spectral norm, provide guarantees for asymptotic convergence, they do not guarantee monotonic convergence to the true solution, i.e., to the desired filtered signal. Hence, if $\rho(\boldsymbol{B}) < 1$, then the recurrence converges, but the error at each iteration is not strictly decreasing.

In the following, we show this behavior for two different instances. First, we consider a Poisson equation with weighted feedback, i.e.,

$$\partial_{xx}\boldsymbol{y} = \boldsymbol{f} \rightarrow \boldsymbol{L}\boldsymbol{y} = \boldsymbol{g} + \boldsymbol{D}\boldsymbol{y}, \tag{21}$$

where $\boldsymbol{g}$ is a constant field, $\boldsymbol{L}$ the discretized Laplacian, $\boldsymbol{D}$ a diagonal matrix with weights and $\boldsymbol{H} = \boldsymbol{L} - \boldsymbol{D}$ with $\rho(\boldsymbol{H}) = 1.6812$. Second, we consider an ARMA GF, i.e.,

$$\boldsymbol{H} = \boldsymbol{H}_{\mathrm{B}}\boldsymbol{H}_{\mathrm{A}} = [\Sigma_{l=0}^{1}\phi_l\boldsymbol{S}^l][\Sigma_{k=1}^{3}\boldsymbol{\Phi}_k\boldsymbol{S}^{k-1}]^{-1}, \tag{22}$$

with $\rho(\boldsymbol{H}_{\mathrm{B}}) = 2.048$. For these experiments, a community graph of $N = 64$ nodes has been considered.

In Fig. 2a and 2b, we can observe that in both cases, the asynchronous versions of the GFs converge to the true solution. In addition, we notice that for certain values of τ, i.e, fraction of coordinate nodes, the convergence is faster than the synchronous version, i.e., $\tau = 1$. This behavior has been seen before in the parallel computing literature, where it is argued that this is due to the shrinkage of the iteration matrix at every step. That is, under certain circumstances, using the interlacing theorem, one can show that the eigenvalues of the asynchronous system are smaller than the ones of the synchronous system. Finally, we want to remark that there is indeed a loss in convergence speed with respect to the *global* updating scheme [c.f. (8)]. This is seen in the fast convergence of the dotted lines in the figures. Those lines represent the Jacobi (J) and classical (N) updates in their synchronous and asynchronous versions.

6. CONCLUSION

In this work, we presented general guarantees for the convergence of an asynchronous implementation of a cascade of edge-variant graph filters. Differently from other approaches, we use weaker conditions, more amenable for filter design, to provide convergence guarantees in both exact and inexact settings. The latter encompasses cases where there is miscommunication between nodes, errors within information packages or noisy communication channels. Through numerical simulations, we demonstrated the predicted convergent behavior of asynchronous edge-variant graph filters.

7. REFERENCES

[1] Mario Coutino, Elvin Isufi, and Geert Leus, "Advances in distributed graph filtering," *arXiv preprint arXiv:1808.03004*, 2018.

[2] Eric D Kolaczyk, *Statistical analysis of network data: methods and models*, Springer Science & Business Media, 2009.

[3] Olaf Sporns, *Networks of the Brain*, MIT press, 2010.

[4] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," IEEE *Sig. Proc. Mag.*, vol. 30, no. 3, pp. 83–98, 2013.

[5] David I Shuman, Pierre Vandergheynst, Daniel Kressner, and Pascal Frossard, "Distributed signal processing via chebyshev polynomial approximation," *IEEE Transactions on Signal and Information Processing over Networks*, 2018.

[6] Sunil K Narang, Akshay Gadde, and Antonio Ortega, "Signal processing techniques for interpolation in graph structured data," in *Proc.* IEEE *Int. Conf. Acoust., Speech Signal Process. (ICASSP)*. IEEE, 2013, pp. 5445–5449.

[7] Santiago Segarra, Antonio G Marques, Geert Leus, and Alejandro Ribeiro, "Reconstruction of graph signals through percolation from seeding nodes," IEEE *Trans. Signal Process*, vol. 64, no. 16, pp. 4363–4378, 2016.

[8] Andreas Loukas, Marco Zuniga, Ioannis Protonotarios, and Jie Gao, "How to identify global trends from local decisions? event region detection on mobile networks," in *INFOCOM, 2014 Proceedings IEEE*. IEEE, 2014, pp. 1177–1185.

[9] Chenhui Hu, Jorge Sepulcre, Keith A Johnson, Georges E Fakhri, Yue M Lu, and Quanzheng Li, "Matched signal detection on graphs: Theory and application to brain imaging data classification," *NeuroImage*, vol. 125, pp. 587–600, 2016.

[10] Elvin Isufi, Geert Leus, and Paolo Banelli, "2-dimensional finite impulse response graph-temporal filters," in *Sig. and Inf. Proc. (GlobalSIP), 2016 IEEE Global Conference on*. IEEE, 2016, pp. 405–409.

[11] Elvin Isufi, Andreas Loukas, Andrea Simonetto, and Geert Leus, "Autoregressive moving average graph filtering," IEEE *Trans. Signal Process*, vol. 65, no. 2, pp. 274–288, 2017.

[12] Mario Coutino, Elvin Isufi, and Geert Leus, "Distributed edge-variant graph filters," in IEEE *7th Int. Workshop Comp. Adv. in Multi-Sensor Adap. Proc.(CAMSAP)*. IEEE, 2017.

[13] Santiago Segarra, Antonio Marques, and Alejandro Ribeiro, "Optimal graph-filter design and applications to distributed linear network operators," IEEE *Trans. Signal Process*, 2017.

[14] Dimitri P Bertsekas, "Distributed asynchronous computation of fixed points," *Mathematical Programming*, vol. 27, no. 1, pp. 107–120, 1983.

[15] Daniel Chazan and Willard Miranker, "Chaotic relaxation," *Linear algebra and its applications*, vol. 2, no. 2, pp. 199–222, 1969.

[16] Andreas Frommer and Daniel B Szyld, "On asynchronous iterations," *Journal of computational and applied mathematics*, vol. 123, no. 1-2, pp. 201–216, 2000.

[17] Jacques Mohcine Bahi, Sylvain Contassot-Vivier, and Raphael Couturier, *Parallel iterative algorithms: from sequential to grid computing*, Chapman and Hall/CRC, 2007.

[18] Oguzhan Teke and PP Vaidyanathan, "The asynchronous power iteration: A graph signal perspective," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4059–4063.

[19] Edmond Chow, "Convergence models and surprising results for the asynchronous jacobi method," in *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2018, pp. 940–949.

[20] CT Kelley, "Iterative methods for linear and nonlinear equations," *Frontiers in applied mathematics*, vol. 16, pp. 575–601, 1995.

[21] Aizik Isaakovich Vol'pert, "Differential equations on graphs," *Mathematics of the USSR-Sbornik*, vol. 17, no. 4, pp. 571, 1972.

[22] Elvin Isufi, Andreas Loukas, and Geert Leus, "Autoregressive moving average graph filters a stable distributed implementation," in *Proc.* IEEE *Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2017.

[23] Yousef Saad, *Iterative methods for sparse linear systems*, vol. 82, siam, 2003.

[24] Alfred Brousseau, *Linear Recursion and Fibonacci Sequences*, Fibonacci Assoc., 1971.

[25] Richard Bellman, *Introduction to matrix analysis*, vol. 19, Siam, 1997.

[26] Nathanaël Perraudin, Johan Paratte, David Shuman, Lionel Martin, Vassilis Kalofolias, Pierre Vandergheynst, and David K. Hammond, "GSPBOX: A toolbox for signal processing on graphs," *ArXiv e-prints*, Aug. 2014.