Available online at www.sciencedirect.com
ScienceDirect
Journal of the Franklin Institute 000 (2017) 1–22
www.elsevier.com/locate/jfranklin

# Fast distributed algebraic connectivity estimation in large scale networks

Eduardo Montijano[a,*], Juan I. Montijano[b], Carlos Sagues[a]

[a] *Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, Spain*
[b] *Instituto Universitario de Matemáticas y Aplicaciones (IUMA), Universidad de Zaragoza, Spain*

Received 3 August 2015; received in revised form 15 January 2017; accepted 9 May 2017
Available online xxx

**Abstract**

This paper presents a distributed method to estimate the algebraic connectivity of fixed undirected communication graphs. The proposed algorithm uses bisection to estimate the second smallest eigenvalue of the Laplacian matrix associated to the graph. In order to decide the sub-interval in which the eigenvalue is contained, a distributed averaging algorithm based on Chebyshev polynomials is considered together with a max consensus algorithm. The information exchanged by neighbors in the graph each communication round is constant and independent of the size of the network, making it scalable to large networks. Besides, exploiting the convergence properties of Chebyshev polynomials we provide a direct estimation of the algebraic connectivity so that, instead of the midpoint of the bisection interval, the new approximation can be used. Finally, our algorithm also provides upper and lower bounds on the algebraic connectivity and an estimation of the Fiedler eigenvector associated to it. Simulations in large networks demonstrate the scalability and accuracy of the algorithm.

## 1. Introduction

Knowledge of the properties of the network topology can play a fundamental role when facing the problem of designing distributed algorithms to be run in ad hoc networks. The research community, aware of this issue, has devoted a lot of effort in the last decade to propose distributed solutions that allow a network to determine most of its properties. A parameter of

* Corresponding author.
*E-mail address:* emonti@unizar.es (E. Montijano).

http://dx.doi.org/10.1016/j.jfranklin.2017.05.021
0016-0032/© 2017 The Franklin Institute. Published by Elsevier Ltd. All rights reserved.

particular importance in communication networks is the Algebraic Connectivity (AC). This parameter is crucial for example in the convergence speed of linear iterations in distributed averaging [1] and provides information about whether the graph topology is connected or not [2,3]. However, since the algebraic connectivity depends on all the communication links present in the network its computation is not trivial in a distributed fashion.

Some works use the Fourier Transform to estimate the entire spectrum of the Laplacian matrix, instead of just the algebraic connectivity [4,5]. The second algorithm allows for an easy clusterization of the graph [6]. On the other hand, the computation of the Fourier Transform requires all the values of the measured signal along time, therefore needing to store large amounts of data in each node of the network.

A distributed algorithm to compute an orthogonal decomposition that contains all the eigenvalues of the graph is given in [7]. Distributed powers of a deflated matrix are computed using event-triggered consensus in [8], providing an estimation with upper and lower bounds of the algebraic connectivity [9]. In [10] they introduce a distributed algorithm to compute the coefficients of the minimal polynomial, which is equivalent to compute all the eigenvalues of the weight matrix. A similar approach is given in [11], where the coefficients of the minimal polynomial are computed by solving an optimization problem in a distributed manner. The number and the size of the messages the nodes need to exchange in these kind of approaches, [7–11], grows linearly with the size of the network, which implies that for large networks a lot of communications will be required.

The mathematical relationship between the eigenvalues of the Laplacian matrix, and their spectral moments, and local structural features, like the number of triangles in the network topology, is analyzed in [12,13], also giving distributed algorithms to compute and control these local features [14]. While these features can be used for more than spectral information, its distributed computation requires, besides distributed averaging linear iterations, information from two-hop neighbors.

Finally, several recent works exploit a power iteration algorithm [15] to distributively estimate the AC, along with the associated Fiedler vector. A mean correction algorithm together with the power iteration is introduced in [16] to avoid numerical problems of the latter. This algorithm is used to adapt the weights of the Laplacian in [17]. Similarly, the power iteration is used to estimate the AC and then optimize a distributed averaging procedure in [18]. A stochastic power iteration with nested consensus operations is used in [19] considering random graphs, and directed graph topologies are analyzed in [20]. The main advantage of using the power iteration is that the amount of information that nodes exchange at each communication round with their neighbors remains constant independently of the size of the network. On the other hand, the limitation of the power iteration is that it requires a large increasing number of communication rounds to estimate the AC with sufficient accuracy as the number of nodes grows, making it hardly applicable in large scale networks. This is caused by the convergence rate of the algorithm, which is determined by the quotient between the second and third largest eigenvalues of the weight matrix associated to the Laplacian. While this quotient is low for small networks, for large scale networks, with hundreds of nodes, it is close to the unity, implying that the number of communication rounds required to compute a good estimation of the algebraic connectivity will be quite large. Moreover, the nested iterations presented in methods like [19] increase even more the number of communication rounds required to estimate the AC with sufficient accuracy.

To overcome this limitation, our contribution is a novel fast distributed algorithm, able to compute the AC and the Fiedler vector in large scale networks. Our algorithm builds upon the results in [16,18], where power iterations are alternated with mean corrections. However, compared to these approaches, in our algorithm we exploit the Chebyshev iteration as an alternative to the power iteration to accelerate convergence. Among other applications, Chebyshev polynomials are typically used to improve the convergence speed of iterative methods [21]. In particular, we use the distributed algorithm presented in [22] that uses Chebyshev polynomials for fast consensus, to accelerate the convergence to the desired value of the AC. Besides, our algorithm includes upper and lower bounds of the AC. The information exchanged by neighbors in the graph each communication round is constant and independent of the size of the network, making it scalable to large networks. On the other hand, we restrict our analysis to fixed-undirected network topologies, and our algorithms can only compute the AC, instead of the entire spectrum.

In the paper we first present a bisection method that exploits the convergence properties of Chebyshev polynomials to estimate the AC, distinguishing when our estimation is above (or below) the right value. Since this first algorithm has a limited accuracy, we introduce a second algorithm that overcomes this limitation, allowing for arbitrary precision (up to round off errors) and also provides an estimation of the Fiedler vector. Simulations in large networks demonstrate the scalability and accuracy of the algorithm. A preliminary version of this article can be found in [23]. In this paper we have introduced substantial modifications to both algorithms, we have added an algorithm to compute the Fiedler vector and we have included measurements of performance with simulations of real networks of thousands of nodes, showing that the new algorithms work in practice much better.

The structure of the paper is the following: in Section 2 we provide a formal definition of the problem and include some background about the consensus algorithm using Chebyshev polynomials. In Section 3 we present the standard bisection algorithm to compute the AC using Chebyshev polynomials. Section 4 describes the new algorithm to compute the AC and the Fiedler vector with arbitrary accuracy. An empirical evaluation of the method with Monte Carlo simulations is given in Section 5. Finally in Section 6 the conclusions of the work are presented.

## 2. Problem formulation

In the paper we consider a network of $N$ nodes labeled by $i \in \mathcal{V} = \{1, \ldots, N\}$. Communications between the nodes are defined according to an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ represents the edge set. We say that nodes $i$ and $j$ directly exchange messages if and only if $(i, j) \in \mathcal{E}$. The set of neighbors of node $i$ is denoted by $\mathcal{N}_i = \{j \in \mathcal{V} \mid (i, j) \in \mathcal{E}\}$. We assume that the communication graph is fixed and connected, that is, there exists a path of one or more links between any two nodes in the network.

We let $\mathbf{L}$ be the Laplacian matrix associated to $\mathcal{G}$, with $l_{ij} = -1$ if $j \in \mathcal{N}_i$, $l_{ii} = |\mathcal{N}_i|$ and the rest of its components equal to zero. For an undirected connected graph, the eigenvalues $\lambda_i$ of $\mathbf{L}$, are all positive and, sorted in increasing order, satisfy $0 = \lambda_1 < \lambda_2 \leq \cdots \leq \lambda_N \leq N$. The algebraic connectivity of the network is characterized by $\lambda_2$. The main objective of the paper is to devise a distributed algorithm that allows all the nodes in $\mathcal{G}$ to obtain an accurate estimation of the value of $\lambda_2$ in a fast way. We are also interested in obtaining the eigenvector $\mathbf{v}_2$ associated to the second eigenvalue $\lambda_2$, usually called the Fiedler vector, in the sense that each node in $\mathcal{G}$ obtains one component of this vector.

For that purpose, we define $\mathbf{W} = \mathbf{I}_N - \epsilon\mathbf{L}$, with $\mathbf{I}_N$ the identity matrix of dimension $N$ and $\epsilon > 0$ a sufficiently small constant [1]. The eigenvalues $\mu_i$ of $\mathbf{W}$ are related to those of $\mathbf{L}$

by

$$\mu_i = 1 - \epsilon\lambda_i,$$

and clearly they are sorted in decreasing order $1 = \mu_1 > \mu_2 \geq \cdots \geq \mu_N$.

Note that both matrices **L** and **W** have the same eigenvectors. In particular, the eigenvector of **L** associated to $\lambda_1 = 0$ is the vector $\mathbf{1} = (1, \ldots, 1)^T$ and therefore, it is also the eigenvector of **W** associated to $\mu_1 = 1$. Also, the eigenvector of **L** associated to $\lambda_2$, can be obtained by computing the eigenvector of **W** associated to the eigenvalue $\mu_2$.

To obtain an approximation to the algebraic connectivity $\lambda_2$ we will develop algorithms that compute the second largest eigenvalue of the matrix **W**, then we must make $\mu_2$ be the second largest eigenvalue (in modulus).

**Assumption 2.1** (Small value of $\epsilon$)**.** The parameter $\epsilon$ is sufficiently small to ensure that $|\mu_N| < |\mu_2| < 1$.

Note that for any strongly connected graph that is not fully connected, i.e., such that $0 < \lambda_2 < \lambda_N \leq N$, this assumption can be fulfilled choosing $\epsilon = 1/N$.

Note also that if we compute an approximation $\tilde{\mu}_2$ to $\mu_2$ with an error $\delta = \mu_2 - \tilde{\mu}_2$, the approximated algebraic connectivity, given by $\tilde{\lambda}_2 = (1 - \tilde{\mu}_2)/\epsilon$, will have an error $\lambda_2 - \tilde{\lambda}_2 = \delta/\epsilon$. The error $\delta$ will be amplified by a factor $1/\epsilon$ which can be large for small $\epsilon$, for example in large-scale networks. This fact must be taken into account when computing $\tilde{\mu}_2$ in order to obtain the adequate accuracy for $\lambda_2$.

## 2.1. Average consensus using Chebyshev polynomials

The algorithm to compute the algebraic connectivity is based on the fast distributed average consensus proposed in [22]. This algorithm computes the distributed evaluation of the Chebyshev polynomial of first kind of degree $n$, $T_n(x)$ [24]. These polynomials are defined with the recurrence:

$$\begin{aligned} &T_0(x) = 1, \;\; T_1(x) = x \\ &T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x), \quad n \geq 2. \end{aligned} \tag{1}$$

For the sake of completeness of the paper we briefly describe the algorithm and its main properties here.

Assume the nodes in the network have some initial values $\mathbf{x}(0) = (x_1(0), \ldots, x_N(0))^T \in \mathbb{R}^N$. The distributed consensus rule using Chebyshev polynomials, expressed in vectorial form, is defined by

$$\begin{aligned} \mathbf{x}(1) &= \frac{1}{T_1(c-d)}(c\mathbf{W} - d\mathbf{I})\mathbf{x}(0), \\ \mathbf{x}(n+1) &= 2\frac{T_n(c-d)}{T_{n+1}(c-d)}(c\mathbf{W} - d\mathbf{I})\mathbf{x}(n) - \frac{T_{n-1}(c-d)}{T_{n+1}(c-d)}\mathbf{x}(n-1), \end{aligned} \tag{2}$$

with the parameters $c$ and $d$ equal to

$$c = \frac{2}{\zeta_M - \zeta_m}, \quad d = \frac{\zeta_M + \zeta_m}{\zeta_M - \zeta_m}, \tag{3}$$

so that the linear application $cx - d$ maps the interval $[\zeta_m, \zeta_M]$ onto $[-1, 1]$. The coefficients $\zeta_M$ and $\zeta_m$ are two real coefficients that need to be chosen satisfying $1 > \zeta_M > \zeta_m > -1$.

The recurrence (2) gives in fact

$$\mathbf{x}(n) = \frac{T_n(c\mathbf{W} - d\mathbf{I})}{T_n(c-d)}\mathbf{x}(0)$$

Any initial vector $\mathbf{x}(0)$, can be expressed as a weighted sum of the eigenvectors of $\mathbf{W}$,

$$\mathbf{x}(0) = \sum_{i=1,\ldots,N} \gamma_i \mathbf{v}_i,$$

with $\mathbf{v}_i$ the eigenvector associated to $\mu_i$ and $\gamma_i$ the corresponding coefficient. Recall that, as mentioned above, the eigenvector $\mathbf{v}_1$ is proportional to the vector $\mathbf{1}$ and $\mu_1 = 1$. Then, the iteration (2) will give a vector

$$\mathbf{x}(n) = \gamma_1\mathbf{v}_1 + \gamma_2\frac{T_n(c\mu_2 - d)}{T_n(c-d)}\mathbf{v}_2 + \gamma_3\frac{T_n(c\mu_3 - d)}{T_n(c-d)}\mathbf{v}_3 + \cdots + \gamma_N\frac{T_n(c\mu_N - d)}{T_n(c-d)}\mathbf{v}_N$$

If $c\mu_i - d < c - d$, $T_n(c\mu_i - d)/T_n(c-d)$ will tend to zero when $n \to \infty$ and $\mathbf{x}(n) \to \gamma_1\mathbf{v}_1$.

In Theorem 4.4 of [22] it was proved that the convergence rate of this iteration is minimum for the parameters $c, d$ such that $\zeta_M = \mu_2$ and $\zeta_m = \mu_N$. Therefore, to get the optimal convergence we must have a good approximation to the eigenvalues $\mu_2$ and $\mu_N$, and in consequence to the algebraic connectivity $\lambda_2 = (1-\mu_2)/\epsilon$.

In order to define the algorithm to compute the algebraic connectivity we will use the following properties of the Chebyshev polynomials [24]:

$$\begin{aligned}
&T_n(x) = \frac{1}{2\tau^n} + \frac{\tau^n}{2} \quad \text{for} \quad x > 1, \quad \text{with} \quad \tau = x + \sqrt{x^2-1}\\
&|T_n(x)| \leq 1 \quad \text{for } |x| < 1\\
&\lim_{n\to\infty} |T_n(x)| = \infty \quad \text{if and only if} \quad |x| > 1\\
&|T_n(x)| > |T_n(y)| \quad \text{if } |x| > |y| \geq 1
\end{aligned} \tag{4}$$

## 3. Distributed algebraic connectivity estimation using bisection

In this section we present the distributed algorithm based on bisection to estimate $\mu_2$. Recall that the power iteration [16] $\mathbf{x}(n) = \mathbf{W}\mathbf{x}(n)$ gives a vector

$$\mathbf{x}(n) = \gamma_1\mathbf{v}_1 + \mu_2^n\gamma_2\mathbf{v}_2 + \mu_3^n\gamma_3\mathbf{v}_3 + \cdots + \mu_N^n\gamma_N\mathbf{v}_N.$$

If $\mathbf{x}(0)$ is chosen in such a way that $\gamma_1 = 0$, the vector $\mathbf{x}(n)$ will give us the Fiedler vector with a convergence ratio $|\mu_3/\mu_2|$. If $\mu_3$ is very close to $\mu_2$ and $\gamma_3 \neq 0$, the convergence will be very slow.

To use the Chebyshev algorithm, since we do not know the values of $\mu_2$ and $\mu_N$ we will consider a symmetric interval around the origin for the iteration in Eq. (2), i.e., $-\zeta_m = \zeta_M = \zeta$ and therefore, $c = 1/\zeta$ and $d = 0$. The value of $\zeta$ will be chosen close (but not equal) to $\mu_2$, as explained below. For such a symmetric selection of the parameters it was proved in [22] that the Chebyshev consensus algorithm is always convergent. The iteration (2) will give in this case a vector

$$\mathbf{x}(n) = \gamma_1\mathbf{v}_1 + \gamma_2\frac{T_n(c\mu_2)}{T_n(c)}\mathbf{v}_2 + \gamma_3\frac{T_n(c\mu_3)}{T_n(c)}\mathbf{v}_3 + \cdots + \gamma_N\frac{T_n(c\mu_N)}{T_n(c)}\mathbf{v}_N. \tag{5}$$

If we select the parameter $c$ in such a way that $|T_n(c\mu_i)/T_n(c\mu_2)| < |\mu_3/\mu_2|$ for $i = 3, \dots, N$ the Chebyshev algorithm will give us the Fiedler vector, and the algebraic connectivity, with a convergence ratio smaller than the one of the power iteration. However, to select a proper value of $c$ we must have an approximation of the value of $\mu_2$, and we will do this by a bisection process.

The idea is based on the fact that if $c\mu_2 > 1$ and $c\mu_2 > c\mu_i$ for $i > 2$ then $T_n(c\mu_i)/T_n(c\mu_2)$ will tend to zero when $n \to \infty$ and for $n$ large $\mathbf{x}(n)$ will behave as $\gamma_1\mathbf{v}_1 + \gamma_2 T_n(c\mu_2)/T_n(c)\mathbf{v}_2$. In particular, $T_n(c)(\mathbf{x}(n) - \gamma_1\mathbf{v}_1)$ will behave as $\gamma_2 T_n(c\mu_2)\mathbf{v}_2$ and will tend to infinity when $n \to \infty$. On the contrary, if $|c\mu_i| < 1$ for $i \geq 2$, all $|T_n(c\mu_i)| \leq 1$ and $T_n(c\mu_2)\mathbf{v}_2$ will be bounded when $n \to \infty$. The boundedness of $T_n(c)(\mathbf{x}(n) - \gamma_1\mathbf{v}_1)$ will indicate us if $c\mu_2 < 1$.

This requires the following assumption:

**Assumption 3.1** (Initial conditions)**.** The initial conditions $\boldsymbol{x}(0)$ satisfy that $\gamma_2 \neq 0$.

For our algorithm, we select initial conditions, $\mathbf{x}(0)$, such that, for each $i \in \mathcal{V}$, $x_i(0)$ is randomly assigned either zero or one. Although there is no way of ensuring Assumption 3.1 without the spectral knowledge of $\mathbf{W}$, in practice almost any random initial conditions satisfy the assumption, as long as not all the components of $\mathbf{x}(0)$ are equal, i.e., the initial conditions are not already in consensus. In fact, in our simulations we did not encounter any problem with this assumption using this assignment. Without loss of generality and for the sake of clarity, since $\gamma_i\mathbf{v}_i$ is also an eigenvector of $\mathbf{W}$, throughout the rest of the paper we will consider the base of eigenvectors that make $\gamma_i = 1$ for all $i$.

## *3.1. Eigenvalues' position indicator*

Assume that we run the iteration (2) for a fixed number, $n$, of communication rounds choosing $c = 1/\zeta$, for some $\zeta \in (0, 1)$ with the previous initial conditions. Considering the infinity norm of a vector $\|x\|_\infty = \max |x_i|$, let us define

$$\kappa_n(c) = T_n(c)\frac{\|\mathbf{x}(n) - \mathbf{v}_1\|_\infty}{\|\mathbf{x}(0) - \mathbf{v}_1\|_\infty} \tag{6}$$

as an indicator of the position of the eigenvalues with respect to the parameter $c$. The vector $\mathbf{v}_1$ is the eigenvector associated to $\mu_1 = 1$, i.e., the average of the initial values whereas $\|\mathbf{x}(0) - \mathbf{v}_1\|_\infty$ and $\|\mathbf{x}(n) - \mathbf{v}_1\|_\infty$ are the initial and current errors in the averaging process.

The following proposition relates the value of $\kappa_n(c)$ with the position of the eigenvalues of $\mathbf{W}$:

**Proposition 3.1** (Eigenvalues' position indicator)**.** *Given $\zeta > 0$ and $c = 1/\zeta$ as the parameter for the iteration (2), if $\mu_i \in [-\zeta, \zeta]$, $\forall i = 2, \dots, N$, then $\kappa_n(c)$ is bounded by*

$$\kappa_n(c) \leq \frac{\sum_{i=2}^N \|\mathbf{v}_i\|_\infty}{\|\sum_{i=2}^N \mathbf{v}_i\|_\infty}, \quad \forall n. \tag{7}$$

*Otherwise, $\mu_2 \notin [-\zeta, \zeta]$ and $\lim_{n\to\infty} \kappa_n(c) = \infty$.*

**Proof.** The initial error is

$$\|\mathbf{x}(0) - \mathbf{v}_1\|_\infty = \left\| \sum_{i=2}^N \mathbf{v}_i \right\|_\infty . \tag{8}$$

The error after $n$ communication rounds is equal to

$$\|\mathbf{x}(n) - \mathbf{v}_1\|_\infty = \left\| \sum_{i=2}^{N} \frac{T_n(c\mu_i)}{T_n(c)} \mathbf{v}_i \right\|_\infty \tag{9}$$

Using these two expressions we get

$$\kappa_n(c) = T_n(c) \frac{\|\mathbf{x}(n) - \mathbf{v}_1\|_\infty}{\|\mathbf{x}(0) - \mathbf{v}_1\|_\infty} = \frac{\| \sum_{i=2}^N T_n(c\mu_i)\mathbf{v}_i\|_\infty}{\| \sum_{i=2}^N \mathbf{v}_i\|_\infty} \tag{10}$$

If $\mu_i \in [-\zeta, \zeta]$, it means that $|c\mu_i| \leq 1$, and using Eq. (4), $|T_n(c\mu_i)|$ is upper-bounded by 1. Then we get,

$$\kappa_n(c) = \frac{\| \sum_{i=2}^N T_n(c\mu_i)\mathbf{v}_i\|_\infty}{\| \sum_{i=2}^N \mathbf{v}_i\|_\infty} \leq \frac{\sum_{i=2}^N \|\mathbf{v}_i\|_\infty}{\| \sum_{i=2}^N \mathbf{v}_i\|_\infty}, \quad \forall n. \tag{11}$$

On the other hand, when $|c\mu_i| > 1$, for some $\mu_i$, $T_n(c\mu_i)$ goes to infinity as $n$ grows and

$$\lim_{n\to\infty} \kappa_n(c) = \lim_{n\to\infty} \frac{\| \sum_{i=2}^N T_n(c\lambda_i)\mathbf{v}_i\|_\infty}{\| \sum_{i=2}^N \mathbf{v}_i\|_\infty} = \infty. \tag{12}$$

□

Taking this into account, Eq. (7) can be used to discern when all the eigenvalues of $\mathbf{W}$ are contained in the interval $[-\zeta, \zeta]$ and when they are not.

The bound on the right hand side of Eq. (7) requires the knowledge of all the eigenvectors of $\mathbf{W}$, which are unknown. A more accessible bound is given in the following result:

**Proposition 3.2.** *When $c\mu_i < 1$ for all $i$, $\kappa_n(c)$ is bounded by $\kappa_n(c) \leq \sqrt{N}$ for all $n \geq 0$.*

**Proof.** The bound is obtained from the inequality

$$\|\mathbf{x}(n) - \mathbf{v}_1\|_\infty \leq \|\mathbf{x}(n) - \mathbf{v}_1\|_2$$

Taking into account that $\mathbf{x}(n) = T_n(c\mathbf{W})\mathbf{x}(0)/T_n(c)$ and that $\mathbf{W}\mathbf{v}_1 = \mathbf{v}_1$, then $T_n(c\mathbf{W})\mathbf{v}_1 = T_n(c)\mathbf{v}_1$ and

$$\begin{aligned} \|\mathbf{x}(n) - \mathbf{v}_1\|_\infty &\leq \|T_n(c\mathbf{W})/T_n(c)(\mathbf{x}(0) - \mathbf{v}_1)\|_2 \\ &\leq \max_{i\neq 1} T_n(c\mu_i)/T_n(c)\|\mathbf{x}(0) - \mathbf{v}_1\|_2 \\ &\leq \max_{i\neq 1} T_n(c\mu_i)/T_n(c)\|\mathbf{x}(0) - \mathbf{v}_1\|_\infty\sqrt{N}. \end{aligned}$$

Regrouping terms and considering that $T_n(c\mu_i) \leq 1$ yields $\kappa_n(c) \leq \sqrt{N}$. □

**Corollary 3.1.** *For large enough $n$, $|\mu_2| < \zeta = 1/c$ if and only if $\kappa_n(c) \leq \sqrt{N}$*

**Proof.** From Proposition 3.1 it is clear that if $\mu_2 > \zeta$, $\kappa_n(c) > \sqrt{N}$ for large enough $n$, since in this case $\lim_{n\to\infty} \kappa_n(c) = \infty$. On the other hand, if $\mu_2 < \zeta$, all the other eigenvalues satisfy $\mu_i \in [-\zeta, \zeta]$, $i = 2, \ldots, N$ and from Proposition 3.2, $\kappa_n(c) \leq \sqrt{N}$. □

The remaining issue is the distributed computation of $\kappa_n(c)$. The exact value of this indicator requires the knowledge of $\mathbf{v}_1$ which is not known by the nodes. Nevertheless, this vector is the average of the initial conditions, that is, ($\mathbf{v}_1 = \mathbf{1}^T/N)\mathbf{x}(0)\mathbf{1}$ and since $\mathbf{L1} = 0$, we can take as initial vector

$$\mathbf{x}(0) = \mathbf{Lu}, \quad \text{the elements of } \mathbf{u} \text{ randomly chosen in} \{0, 1\},$$

as done in [16]. In this way, for this vector the corresponding component $\mathbf{v}_1 = 0$ and consequently, the proposed approximation for the value of $\kappa_n(c)$ is

$$\kappa_n(c) \simeq T_n(c)\frac{\max_i |x_i(n)|}{\max_i |x_i(0)|}. \tag{13}$$

These values are obtained distributively using max-consensus [25] in a fixed number of iterations equal to the diameter of $\mathcal{G}$.

## 3.2. Bisection algorithm

We can now propose the adaptive algorithm to iteratively choose $\zeta$ and $c$ using $\kappa_n(c)$ and a bisection method. The algorithm starts with the definition of the extremes of the interval $[\zeta_{\min}, \zeta_{\max}]$ that contains $\mu_2$.

These extremes are defined in such a way that

$$\mu_i \in [-\zeta_{\max}, \zeta_{\max}], \ \forall i = 2, \ldots, N, \tag{14}$$

and at least $\mu_2 \notin [-\zeta_{\min}, \zeta_{\min}]$.

If there is no knowledge about the network, the initial values of these parameters can be $\zeta_{\min} = 0$ and $\zeta_{\max} = 1$. Following the bisection approach, the first parameter to run the consensus algorithm (2) is $\zeta = (\zeta_{\min} + \zeta_{\max})/2 = 0.5, \ c = 1/\zeta = 2$.

At each iteration of the algorithm, Eq. (2) is run for $n$ communication rounds using $c$. The number of rounds, $n$, needs to be chosen sufficiently large to ensure that $\kappa_n(c)$ is greater than $\sqrt{N}$ when there is some eigenvalue outside the range $[-\zeta, \zeta]$. In particular, it is required that $T_n(c\mu_2) > \sqrt{N}$, if $c\mu_2 > 1$, for the selected value of $n$. For networks of the order of hundreds of nodes we have determined empirically that a good value for $n$ is approximately 8 times the diameter of the communication graph.

After that, the max-consensus is performed to compute $\kappa_n(c)$. According to Corollary 3.1, if $\kappa_n(c) \le \sqrt{N}$, we know that all the eigenvalues are contained in the interval $[-\zeta, \zeta]$. On the other hand, if $\kappa_n(c) > \sqrt{N}$, it means that there is some eigenvalue outside the range. Once we have detected which of the two situations is happening, the bisection parameters are updated according to it,

$$\begin{aligned} \zeta_{\max} &= \zeta, \ \text{if } \kappa_n(c) \le \sqrt{N} \\ \zeta_{\min} &= \zeta, \ \text{if } \kappa_n(c) > \sqrt{N}. \end{aligned} \tag{15}$$

Then the consensus process is repeated with the parameter $c = 1/\zeta$ and $\zeta = (\zeta_{\min} + \zeta_{\max})/2$. At each consensus process, a new estimation of the parameter $\zeta$ is computed.

The algorithm updates the parameter $c$ of Eq. (2) to optimize the convergence speed of the Chebyshev consensus method, leading at the same time to the distributed estimation of the algebraic connectivity. The procedure is synthesized in Algorithm 1.

**Remark 3.1.** An additional advantage of the proposed algorithm is that it can make use of prior information available about the algebraic connectivity. The standard initialization of the

**Algorithm 1** Algebraic connectivity estimation with bisection.

```
Require: $N$, $\mathbf{W} = \mathbf{I}_N - \epsilon\mathbf{L}$ and $\mathbf{x}(0)$, s.t. $x_i(0) \in \{0, 1\}$
 1: Initialize $\zeta_{\min} = 0$, $\zeta_{\max} = 1$, $n = 8\text{Diam}(\mathcal{G})$
 2: Remove the component $\mathbf{v}_1$ of $\mathbf{x}(0)$, $\mathbf{x}(0) = \mathbf{L}\mathbf{x}(0)$
 3: while $\zeta_{\max} - \zeta_{\min} >$ tolerance do
 4:     $\zeta = (\zeta_{\min} + \zeta_{\max})/2$, $c = 1/\zeta$
 5:     Starting from $x_i(0)$, compute $x_i(n)$ using (2)
 6:     Use max consensus to estimate $\kappa_n(c)$ in (13)
 7:     if $\kappa_n(c) \le \sqrt{N}$ then $\zeta_{\max} = \zeta$
 8:     else $\zeta_{\min} = \zeta$
 9:     end if
10: end while
11: $\lambda_2 = (1 - (\zeta_{\min} + \zeta_{\max})/2)/\epsilon$
```

interval to cover all possibilities is $\zeta_{\min} = 0$ and $\zeta_{\max} = 1$. However, if we know that the algebraic connectivity $\lambda_2$ is smaller than some value, e.g., 0.1, we know that $\mu_2 > 1 - \epsilon 0.1$ and we can start the algorithm assigning to $\zeta_{\min}$ this value, requiring less bisection iterations, and therefore, less communication rounds. This property is of high interest in large scale networks, where we can assume that $\mu_2 = 1 - \epsilon\lambda_2$ is going to be close to 1.

**Remark 3.2.** The bisection algorithm is based on Corollary 3.1 which states that for a number of rounds, $n$, large enough the value of $\kappa_n(c)$ let us discriminate if $|\mu_2| < 1/c$ or else $|\mu_2| > 1/c$. However, this minimum value of communication rounds depends on $c$ in such a way that when $c \to 1/\mu_2$, $n(c) \to \infty$. Thus, to obtain $\mu_2$ with high accuracy, the value of $n$ must be large. Unfortunately, in practice for large values of $n$ the iteration becomes numerically unstable. This kind of computation is made stable in the case of the power iteration by means of some rescaling of the vectors. However, in our algorithm such a scaling cannot be performed in a cheap way because the magnitude of the vector $\mathbf{x}(n) - \mathbf{v}_1$ is relevant. In conclusion, the bisection algorithm can only be used with a limited value of $n$ and therefore, the accuracy of the approximated algebraic connectivity is limited. In the next section we propose an algorithm that, knowing an approximation of $\mu_2$, can provide the algebraic connectivity as well as the Fiedler vector both with high accuracy.

## *3.3. Speed up using* k*-section method*

The bisection method presented above reduces the estimation error, $|\zeta - \mu_2|$, by a constant factor of 0.5. In communication networks, the cost of sending several small messages is usually bigger than the cost of sending a unique message with more information. Taking this into account, our method can execute several copies of Eq. (2) in parallel with different values of $\zeta$. In this way, the bounds of $\mu_2$ are delimited with more accuracy and the optimal convergence rate is reached sooner. Specifically, given $\zeta_{\min}$ and $\zeta_{\max}$, the $k$-section method executes $k - 1$ consensus iterations in parallel with parameters

$$\zeta_j = j(\zeta_{\min} + \zeta_{\max})/k, \quad c_j = 1/\zeta_j, \quad j = 1, \ldots, k-1. \tag{16}$$

Once all the different estimations of $\kappa_n(c_j)$ have been computed, the interval to consider in the next consensus iteration is defined by

$$\begin{aligned}\zeta_{\min} &= \max_j \zeta_j \text{ s.t. } \kappa_n(c_j) \leq \sqrt{N},\\ \zeta_{\max} &= \min_j \zeta_j \text{ s.t. } \kappa_n(c_j) > \sqrt{N}.\end{aligned} \tag{17}$$

With this algorithm, the size of the messages exchanged by the nodes will increase in $k-1$ additional elements instead of the one sent by the bisection method. However, the error in the estimation of $\mu_2$ will be reduced by a factor of $1/k$ after each update in the estimation.

## 4. Accurate computation of the Fiedler vector and the algebraic connectivity

The above bisection algorithm can only provide an approximation to the algebraic connectivity with limited accuracy and does not estimate the Fiedler vector of the network. In this section we propose solutions to compute, in a distributed way, the algebraic connectivity, as well as its associated eigenvector, with high accuracy if required. The overall idea of this procedure is the fact that $\mathbf{x}(n)$ in Eq. (5) converges to a vector proportional to $\mathbf{v}_2$ if $\mathbf{v}_1 = 0$ and $c\mu_2 > 1$. By running several iterations of Eq. (2) we can improve the estimation of $\mu_2$ and this will improve the convergence ratio of the algorithm. This section discusses also a stop criterion to end the algorithm as soon as the approximations have the required accuracy.

### *4.1. Computing the Fiedler vector*

Let us suppose that we have computed an approximation $\tilde{\mu}_2$ to the second largest eigenvalue $\mu_2$ of the matrix $\mathbf{W}$. We have also a vector $\mathbf{x}(0)$ that has no component in $\mathbf{v}_1$, that is

$$\mathbf{x}(0) = \mathbf{v_2} + \cdots + \mathbf{v}_N.$$

where the eigenvectors $\mathbf{v}_i$ do not need to have norm unity. We take a value $c = 1/\zeta$ with $\zeta = \tilde{\mu}_2 - \delta < \mu_2$ and starting with $\mathbf{x}(0)$ perform the iteration (2) obtaining

$$\mathbf{x}(n) = \frac{1}{T_n(c)}(T_n(c\mu_2)\mathbf{v_2} + \cdots + T_n(c\mu_N)\mathbf{v}_N).$$

Since $c\mu_2 > 1$ and $c\mu_2 > c\mu_i$ for $i > 2$, the vector $\mathbf{x}(n)/\|\mathbf{x}(n)\|$ tends to a unitary vector $\mathbf{x}$ which is proportional to $\mathbf{v}_2$, that is, to the Fiedler vector. However, if $c\mu_2 \leq 1$ the convergence of $\mathbf{x}(n)$ cannot be guaranteed. The reason to use $\zeta = \tilde{\mu}_2 - \delta$ instead of $\zeta = \tilde{\mu}_2$ is precisely to guarantee that $c\mu_2 > 1$, and as we will show later, to improve the convergence of the iteration.

We are interested now in finding the optimal value of the parameter $c$. First note that if $c\mu_i \leq 1$ for all $i > 2$, then $|T_n(c\mu_i)| \leq 1$ and the convergence ratio, defined as the limit

$$r = \lim_{n\to\infty}\left(\left\|\frac{\mathbf{x}(n)}{\|\mathbf{x}(n)\|} - \frac{\mathbf{v}_2}{\|\mathbf{v}_2\|}\right\|\right)^{1/n},$$

will be $\lim_{n\to\infty} 1/T_n(c\mu_2)^{1/n}$, where both vectors are normalized to guarantee that they indeed represent the same eigenvector. On the other hand, if $c\mu_3 > 1$, the convergence ratio will be $\lim_{n\to\infty}(T_n(c\mu_3)/T_n(c\mu_2))^{1/n}$. Using this fact we obtain the optimal value of $c$ as the value that minimizes the convergence ratio.

**Theorem 4.1.** *For $c \geq 1/\mu_3$, the convergence ratio of $\mathbf{x}(n)/\|\mathbf{x}(n)\|$ to the Fiedler vector is a decreasing function of c. Therefore, the smallest convergence ratio is obtained for $c = 1/\mu_3$ and it is given by $r_{opt} = \mu_3/(\mu_2 + \sqrt{\mu_2^2 - \mu_3^2})$. Moreover, for any c such that $c > 1/\mu_3$, $r < \mu_3/\mu_2$, that is, the convergence ratio is smaller than the one for the power iteration.*

**Proof.** If $c\mu_3 \geq 1$, using Eq. (4) the convergence ratio is given by

$$r = \lim_{n\to\infty} \left(\frac{T_n(c\mu_3)}{T_n(c\mu_2)}\right)^{1/n} = \lim_{n\to\infty} \left(\frac{1+\tau_3^{2n}}{1+\tau_2^{2n}}\frac{\tau_2^n}{\tau_3^n}\right)^{1/n} = \frac{\tau_3}{\tau_2} = \frac{c\mu_3 + \sqrt{c^2\mu_3^2 - 1}}{c\mu_2 + \sqrt{c^2\mu_2^2 - 1}}$$

where we have denoted $\tau_i = c\mu_i + \sqrt{c^2\mu_i^2 - 1} > 1,\ \ i = 1, 2$. For any $\mu_2$, $\mu_3$ such that $c\mu_2 > c\mu_3 \geq 1$ the function $r = r(c, \mu_2, \mu_3)$ satisfies $\partial r/\partial c < 0$ and therefore is a decreasing function on $c$ that attains its minimum for $c = 1/\mu_3$.

Moreover, if $c\mu_3 > 1$

$$r = \frac{c\mu_3 + \sqrt{c^2\mu_3^2 - 1}}{c\mu_2 + \sqrt{c^2\mu_2 2 - 1}} \leq \frac{\mu_3}{\mu_2} \iff \mu_2\sqrt{c^2\mu_3^2 - 1} < \mu_3\sqrt{c^2\mu_2^2 - 1} \iff \mu_3^2 < \mu_2^2$$

$\square$

On the other side, if $c\mu_2 > 1$ but $c\mu_3 < 1$, then $T_n(c\mu_i) \leq 1$ for all $i = 2, \dots, N$ and the convergence ratio is bounded by

$$r \leq \lim_{n\to\infty} \left(\frac{1}{T_n(c\mu_2)}\right)^{1/n} = \tau_2 = \frac{1}{c\mu_2 + \sqrt{c^2\mu_2^2 - 1}}, \quad \text{if } 1/\mu_3 > c > 1/\mu_2.$$

According to this Theorem, the optimal convergence is obtained for $c = 1/\mu_3$. Unfortunately, $\mu_3$ is not known, but if we have an approximation $\tilde{\mu}_2$ to $\mu_2$ we can take $c = 1/(\tilde{\mu}_2 - \delta)$. This guarantees that $c > 1/\mu_2$ only if $\tilde{\mu}_2$ is accurate enough, and therefore the algorithm presents faster convergence than the power iteration for $\delta$ small enough. The displacement $\delta$ must be chosen by some heuristic criteria. From our experiments, we have observed that a good value is $\delta = 1/2N$.

### *4.2. Scaling and filtering the approximation to the Fiedler vector*

It is clear that $\mathbf{x}(n)$ tends to the null vector as $n \to \infty$. Therefore, it must be normalized to obtain an accurate approximation to the Fiedler vector. On the other side, even though the vector $\mathbf{x}(0)$ has no component in the vector $\mathbf{v}_1$, that is, it has average zero, and theoretically the iteration (2) preserves this property, the round-off errors make the vector $\mathbf{x}(n)$ have such a component and it must be removed at some iterations. To deal with both situations we will follow the ideas applied to the power iteration in [16].

First, to remove the $\mathbf{v}_1$ component we can multiply, after $n$ communication rounds, the approximation $\mathbf{x}(n)$ by the Laplacian matrix $\mathbf{L}$, so that, since $\mathbf{W}$ commutes with $\mathbf{L}$, the filtered approximation $\mathbf{Lx}(n)$ will be given by

$$\mathbf{Lx}(n) = \frac{T_n(c\mu_2)}{T_n(c)}\lambda_2\mathbf{v_2} + \dots + \frac{T_n(c\mu_N)}{T_n(c)}\lambda_N\mathbf{v}_N.$$

Then, to guarantee the convergence of the iteration, $n$ must be large enough to ensure that the coefficient of $\mathbf{v_2}$, $\frac{T_n(c\mu_2)}{T_n(c)}\lambda_2$ is larger than the other coefficients. Then, it must satisfy

$$\frac{|T_n(c\mu_i)|}{|T_n(c\mu_2)|}\frac{|\lambda_i|}{|\lambda_2|} < 1, \text{ for } c\mu_i > 1, i > 2$$

$$\frac{1}{T_n(c\mu_2)}\frac{\lambda_i}{\lambda_2} < 1, \text{ for } c\mu_i \leq 1, i > 2$$

In [16] it was proved that the power iteration with this correction converges if $\lambda_i\mu_i^n < \lambda_2\mu_2^n$, and this is satisfied if the filtering is done every $n$ communication rounds with $n > 1/(\epsilon\mu_2)$. For the case in which $c\mu_i > 1$, if $\lambda_i\mu_i^n < \lambda_2\mu_2^n$, then

$$\frac{T_n(c\mu_i)}{T_n(c\mu_2)}\frac{\lambda_i}{\lambda_2} < \frac{T_n(c\mu_i)}{T_n(c\mu_2)}\frac{\mu_2^n}{\mu_i^n} = \frac{T_n(c\mu_i)/(c\mu_i)^n}{T_n(c\mu_2)/(c\mu_2)^n} < 1,$$

where we have used that $T_n(x)/x^n$ is an increasing function for $x > 1$ and $c\mu_2 > c\mu_i > 1$. Consequently, the lower bound for $n$ of the power iteration will also be sufficient to have convergence in our case. Note that the above bound is equivalent to say that $T_n(c\mu_i)/T_n(c\mu_2) < \mu_i^n/\mu_2^n$, in agreement with the fact that this algorithm has a faster convergence than the power iteration.

Once we have filtered the approximation, to scale it we recall that, by means of a max consensus process, the norm $\|\mathbf{x}(n)\|$ is computed to obtain an approximation to $\kappa_n(c)$. If we make coincide both actions, we can scale the vector and take as new initial vector $\mathbf{x}(0) = \mathbf{L}\mathbf{x}(n)/\|\mathbf{x}(n)\|$. Then we can restart the Chebyshev iteration with this vector and with the parameter $c = 1/(\tilde{\mu}_2 - \delta)$

### 4.3. Direct estimation of $\mu_2$

The convergence of the algorithm depends on the parameter $c = 1/(\tilde{\mu}_2 - \delta)$ and its quality depends on the accuracy of the approximation $\tilde{\mu}_2$. The bisection algorithm can give us an approximation $\tilde{\mu}_2$, but we will see next that a more accurate direct approximation to $\mu_2$ can be obtained with minimum additional cost after each application of the Chebyshev algorithm (2) with $n$ rounds. This will provide a better value of $c$ to be used in the next $n$ Chebyshev iterations.

**Lemma 4.1.** *Let* $\mathbf{x}(0)$ *be such that* $\mathbf{v}_2 \neq \mathbf{0}$ *and let* $c = 1/\zeta$ *with* $\zeta < \mu_2$. *Then,*

$$\lim_{n\to\infty}\frac{\|\mathbf{x}(n) - \mathbf{v}_1\|_\infty}{\|\mathbf{x}(n-1) - \mathbf{v}_1\|_\infty} = \lim_{n\to\infty}\frac{T_n(c\mu_2)T_{n-1}(c)}{T_{n-1}(c\mu_2)T_n(c)} \tag{18}$$

**Proof.** According to the recurrence (5) of the Chebyshev algorithm,

$$\mathbf{x}(n) = \mathbf{v}_1 + \sum_{j=2}^{N}\frac{T_n(c\mu_j)}{T_n(c)}\mathbf{v}_j \tag{19}$$

and therefore

$$\|\mathbf{x}(n) - \mathbf{v}_1\|_\infty = \frac{|T_n(c\mu_2)|}{T_n(c)}\left\|\mathbf{v}_2 + \sum_{j=3}^{N}\frac{T_n(c\mu_j)}{T_n(c\mu_2)}\mathbf{v}_j\right\|_\infty.$$

Since $c\mu_2 > 1$, $\lim_{n\to\infty} |T_n(c\mu_j)|/|T_n(c\mu_2)| = 0$ for $j = 3, \dots, N$ and the result follows immediately. □

**Lemma 4.2.** *Let* $\mathbf{x}(0)$ *such that* $\mathbf{v}_2 \neq \mathbf{0}$ *and let* $c = 1/\zeta$ *with* $\zeta < \mu_2$. *Then, the factor*

$$\rho_n(c) = \frac{\|\mathbf{x}(n) - \mathbf{v}_1\|_\infty}{\|\mathbf{x}(n-1) - \mathbf{v}_1\|_\infty} \frac{T_n(c)}{T_{n-1}(c)}, \tag{20}$$

*satisfies*

$$\lim_{n\to\infty} \rho_n(c) = c\mu_2 + \sqrt{(c\mu_2)^2 - 1} > 1 \tag{21}$$

**Proof.** According to Lemma 4.1,

$$\lim_{n\to\infty} \rho_n(c) = \lim_{n\to\infty} \frac{T_n(c\mu_2)}{T_{n-1}(c\mu_2)} \tag{22}$$

and taking into account that, as given in Eq. (4), the direct expression of $T_n(c\mu_2)$ is characterized by

$$T_n(c\mu_2) = \frac{1 + \tau^{2n}}{2\tau^n}, \quad \tau = c\mu_2 + \sqrt{(c\mu_2)^2 - 1}, \tag{23}$$

the result holds just taking limits and considering that $c\mu_2 > 1$ and $\tau > 1$. □

With these two results, after $n$ rounds of Eq. (2), if we use the max-consensus stage to compute $\|\mathbf{x}(n-1)\|_\infty$ (recall that the initial vector $\mathbf{x}(0)$ was selected to have $\mathbf{v}_1 = 0$), in the same fashion as we did with $\|\mathbf{x}(n)\|_\infty$, we can compute the factor $\rho_n(c)$ which, assuming $n$ is sufficiently large can be approximated by

$$\rho_n(c) \simeq \frac{T_n(c\mu_2)}{T_{n-1}(c\mu_2)}. \tag{24}$$

Therefore, a direct estimation of $\mu_2$ can be obtained by solving the equation

$$\rho_n(c) T_{n-1}(c\mu_2) = T_n(c\mu_2). \tag{25}$$

In the following we explain how to clear $\mu_2$ in Eq. (25). Plugging Eq. (23) into Eq. (25) yields

$$\rho_n(c) \frac{1 + \tau^{2(n-1)}}{\tau^{n-1}} = \frac{1 + \tau^{2n}}{\tau^n}. \tag{26}$$

This is a non-linear equation that has no explicit solution, which means that an iterative solver is needed. To solve for $\tau$ in this implicit equation we use the Newton–Raphson method. Taking into account that $\lim_{n\to\infty} \rho_n(c) = 1/\tau$, the value of $1/\rho_n(c)$ can be assigned as initial guess $\tau_0$. After that, the nodes iterate

$$\tau_{j+1} = \tau_j - \frac{\rho_n(c)(1 + \tau_j^{2n-2})\tau_j - 1 - \tau_j^{2n}}{\rho_n(c)\left(1 + (2n-1)\tau_j^{2n-2}\right) - 2n\tau_j^{2n-1}}, \tag{27}$$

until $\tau_j$ converges. In practice, a fixed number of Newton iterations, let us say 2, are enough to get a very accurate $\tau$. Note that this procedure is run locally by each node of the network without exchanging any additional information. It is also worth mentioning, that since $\rho_n(c)$ is the same for all the nodes, then the final value of $\tau$ will also be the same for everybody.

Finally, clear $\mu_2$ from Eq. (23),

$$\mu_2 = \frac{1+\tau^2}{2c\tau}. \tag{28}$$

Let us note that the value of $\mu_2$ obtained is still an approximation of the real algebraic connectivity because $\rho_n(c)$ is approximated in Eq. (24). To obtain an accurate estimation we would need to run Eq. (2) several times.

Note also that Lemmas 4.1 and 4.2 and therefore this approximation of $\mu_2$ are only valid if $c\mu_2 > 1$. In case that after the Chebyshev iteration it is detected that $c\mu_2 > 1$ a new value of $c$ must be chosen by means of bisection. In this case, for numerical accuracy, we apply

$$\begin{aligned} &\zeta_{\max} = \alpha\zeta + (1-\alpha)\zeta_{\max}, \\ &\zeta = (\zeta_{\min} + (1/c))/2, \\ &c = 1/\zeta. \end{aligned} \tag{29}$$

with $\alpha \in (0, 1]$ (0.5 in our simulations).

### 4.4. Stop criterion

In the standard bisection we know that at each iteration of the algorithm the size of the interval is reduced by half its original value. However, if we use $\rho_n(c)$ to adapt the value of $\zeta$, this is no longer true because the reduction of the interval is generally larger.

Let us denote by $\Delta\zeta$ the difference between two successive values of $\zeta$. Clearly, as $\zeta$ approaches to the algebraic connectivity, $\Delta\zeta$ approaches to zero. Therefore, if we detect that $\Delta\zeta$ is sufficiently close to zero we deduce that our estimation is close to the real value of $\mu_2$ and that further iterations of the algorithm are not going to improve much the result. Thus, besides the standard bisection stop criterion, we consider a second case that triggers when $\Delta\zeta$ is smaller than the desired tolerance error.

### 4.5. Algorithm

To conclude, the modified method to compute the algebraic connectivity and the Fiedler vector is synthesized in Algorithm 2.

The algorithm is composed of several high level iterations, described in the *while* loop of lines 5–28, that represent new updates of the AC estimation by the network. Each of these high level iterations requires $n$ communication rounds for the Chebyshev iteration to update $\mathbf{x}(n)$, lines 6–8, plus an additional number of $\text{Diam}(\mathcal{G})$ communication rounds for the max consensus to estimate $\kappa_n(c)$ and $\rho_n(c)$, lines 9–11. Additionally, some high level iterations include one additional communication round to update the initial condition $\mathbf{x}(0)$ in line 22. In terms of amount of information, each communication round requires to send one float, Chebyshev and update of the initial conditions, or three floats, max-consensus to estimate $\max(x_i(0))$, $\max(x_i(n))$ and $\max(x_i(n-1))$, to the neighbors. The rest of the operations are performed locally by each node. We show in the simulations that, compared to the power iteration or similar solutions, the extra communications required by the max-consensus are compensated by the convergence speed of the Chebyshev iteration and the procedure to estimate directly $\lambda_2$ at each high level iteration.

**Algorithm 2** Algebraic connectivity estimation with modified bisection.

```
Require: $N, \mathbf{W} = \mathbf{I}_N - \epsilon\mathbf{L},\ n,\ \alpha$ and $\mathbf{x}(0)$, s.t. $x_i(0) \in \{0, 1\}$
1: – Initialization –
2:     $\zeta_{\min} = 0,\ \zeta_{\max} = 1,\ \zeta = 1/2,\ c = 2,\ \Delta\zeta = 1,\ \delta = 1/2N$
3: Remove the component $\mathbf{v}_1$ of $\mathbf{x}(0)$, $\mathbf{x}(0) = \mathbf{L}\mathbf{x}(0)$
4: – Algebraic connectivity estimation –
5: while ($\zeta_{\max} - \zeta_{\min} >$ tolerance) AND $\Delta\zeta >$tolerance do
6:    – Compute $\mathbf{x}(n)$ using (2) with $d = 0$ –
7:    $\mathbf{x}(1) = \frac{c}{T_1(c)}\mathbf{x}(0)$
8:    $\mathbf{x}(j) = \frac{2cT_{j-1}(c)}{T_j(c)}\mathbf{W}\mathbf{x}(j-1) - \frac{T_{j-2}(c)}{T_j(c)}\mathbf{x}(j-2)$, for $j = 2, \dots, n$.
9:    – Use max consensus to estimate $\kappa_n(c)$ in (13) and $\rho_n(c)$ in (20) –
10:   $\kappa_n(c) = T_n(c)\frac{\max_i |x_i(n)|}{\max_i |x_i(0)|}$
11:   $\rho_n(c) = \frac{\max_i |x_i(n)|}{\max_i |x_i(n-1)|}\frac{T_n(c)}{T_{n-1}(c)}$
12:   – Update estimation –
13:   if $\kappa_n(c) \le \sqrt{N}$ then
14:       $\zeta_{\max} = \alpha(1/c) + (1-\alpha)\zeta_{\max}$
15:       $\zeta = (\zeta_{\min} + (1/c))/2,\ c = 1/\zeta$
16:   else
17:       $\zeta_{\min} = 1/c$
18:       Initialize $\tau_0 = 1/\rho_n(c)$
19:       Compute $\tau_{j+1} = \tau_j - \frac{\rho_n(c)(1+\tau_j^{2n-2})\tau_j - 1 - \tau_j^{2n}}{\rho_n(c)\left(1 + (2n-1)\tau_j^{2n-2}\right) - 2n\tau_j^{2n-1}}$, for $j = 0, 1$.
20:       $\zeta = \frac{1+\tau_2^2}{2c\tau_2}$
21:       $c = 1/(\zeta - \delta)$
22:       $\mathbf{x}(0) = \mathbf{L}x(n)/\|\mathbf{x}(n)\|_\infty$
23:       Compute $\Delta\zeta$, the difference between the two last $\zeta$
24:   end if
25:   $\mu_2 = \zeta$
26:   $\lambda_2 = (1-\mu_2)/\epsilon$
27:   $\mathbf{v}_2 = \mathbf{x}(0)$
28: end while
```

## 5. Simulations

We have analyzed our algorithm in a simulated environment considering two different cases. One part of the analysis has been done generating different networks and evaluating the results of our algorithms with Monte Carlo methods. The second consists in the execution of our algorithms considering a real large scale network.

## *5.1. Evaluation with Monte Carlo experiments*

In these simulations we consider proximity graphs of different size and with different communication radii. We have chosen these two parameters because they are highly correlated with the algebraic connectivity, and by considering different values of both of them we can observe how the algorithm behaves when executed in a wide variety of networks. Additionally, with the different communication radii we can observe the effect of different values of the number, $n$, of communication rounds of iteration (2), which in our approach is dependent on the diameter of the graphs. Since in large networks computing the exact diameter is a costly operation, in our simulations we use instead the pseudo-diameter, which is a lower bound approximation that can be computed very efficiently.

To generate a particular network we randomly place each node within a square of 150 $\times$ 150 m. Then we include a link between two nodes if the corresponding points are at a distance smaller than the considered radius (1.5, 6 and 24 m), forcing that every node has at least one neighbor to ensure connectedness. Fig. 1 shows one example of a network with $N = 150$ and the communication radius equal to 1.5 m. For each different number of nodes and communication radius, we have repeated the process for 100 random networks. Ten different random initial values have been tested for each network, giving a total of 1000 trials for each number of nodes. In all the experiments the matrix $\mathbf{W}$ has been computed from the Laplacian matrix using $\epsilon = 1/\max(|\mathcal{N}_i| + 1)$. This value is given in [1] as a sufficient condition for convergence and, although for this value we cannot ensure Assumption 2.1, in all our experiments this assumption has been satisfied.

We have analyzed the evolution of the algebraic connectivity estimation at each communication round for Algorithm 2, accounting for all the stages of the algorithm, i.e., Chebyshev iteration, max-consensus and update of initial conditions. Since Algorithm 1 has a limitation on the accuracy of the estimation, see Remark 3.2, we have left it out of the analysis. Instead, we have compared our solution with the algorithm based on the power iteration presented in [16], with the code available at [26]. We have also considered a variation of this algorithm, where instead of performing the re-scaling every 250 communication rounds, we perform it every $3N$ rounds.

The results for the AC estimation are shown in Fig. 2. Straight lines represent the median of the estimation error, whereas dashed lines represent the percentiles 25 and 75 for the three algorithms. Note how Algorithm 2 does not update the estimation of $\lambda_2$ every communication round, but every 9 Diam($\mathcal{G}$) communication rounds, 8 times the diameter due to the computation of $x_i(n)$ in lines 6–8 of the algorithm plus an extra Diam($\mathcal{G}$) communication rounds to compute $\max_i|x_i(n)|$ using a max consensus procedure, in line 9. This is mostly visible at the beginning, where our estimation error remains constant for a while. Particularly, our initial guess for the AC is computed using the initial value given to $\mu_2$, equal to the initial value of $\zeta$, which is 0.5. Therefore, the initial value of the AC is given by $0.5/\epsilon$. Afterwards, since the diameter of the network is not the same all the time, the aggregate results are smoother, yet the peaks can still be observed.

Note that with the direct estimation procedure of $\mu_2$ in lines 18–21 of Algorithm 2, the error is reduced at each round by a factor greater than 0.5. In terms of number of communication rounds, as expected, for larger values of $N$ the algorithm needs more communication rounds (having a larger diameter) to obtain the same accuracy. Similarly, as the communication radius increases, the algorithm needs less rounds. On the other hand, in this case our algorithm presents less accuracy than in the rest of cases. We believe that this happens because for

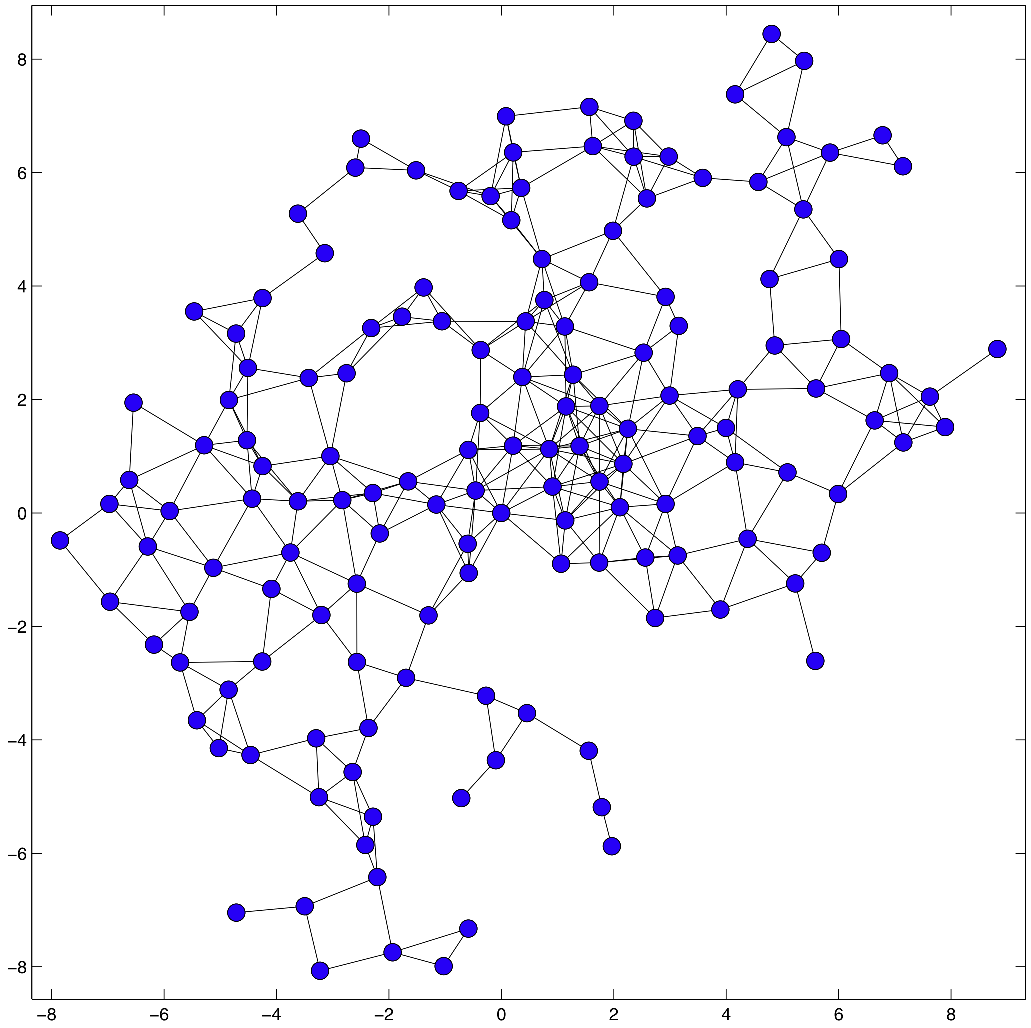

Fig. 1. Example of one network used in the simulations.

these type of networks (large $N$ and dense topologies) the general parameters chosen for all the simulations, $\delta = 1/2N$ and $n = 8$ Diam$(\mathcal{G})$, could be improved. Nevertheless, the algorithm still gets an accuracy around $10^{-10}$.

Additionally, except for the initial iterations, in all the cases our algorithm outperforms the power iteration by several orders of magnitude in considerably few communication rounds, reaching in most cases values with round-off precision. The re-scaling every $3N$ communication rounds of the power iteration performs better than the re-scaling at a fixed number of rounds when the communication radius or the size of the network are small. However, as these parameters grow, it seems preferable to re-scale every 250 communication rounds, as proposed originally in [16].

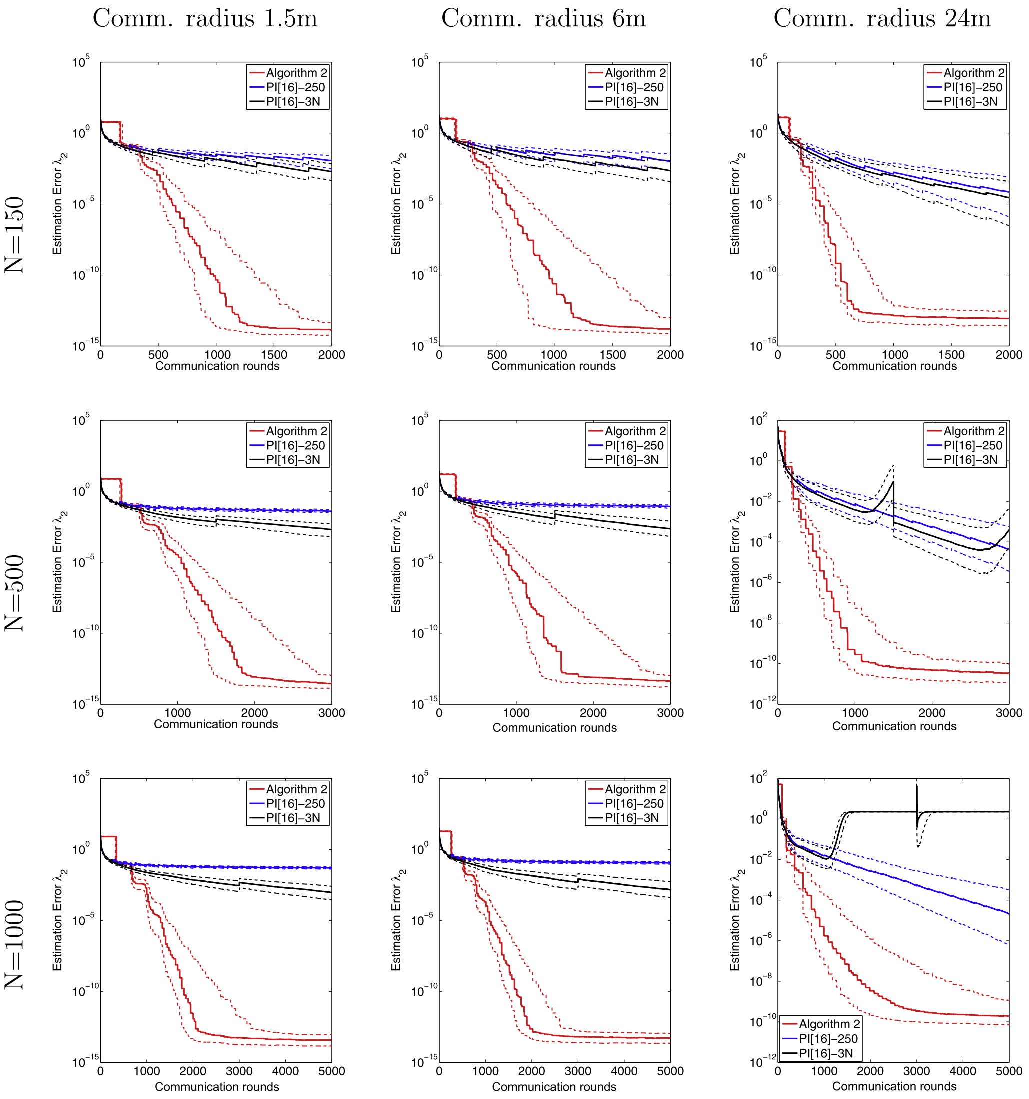

Fig. 2. Median estimation error of the algebraic connectivity for networks of different size and communication radius.

Fig. 3 shows the results of the estimation of the Fiedler vector for the same experiment. The estimation error in this case is computed as

$$e_{\mathbf{v}_2} = \min\left(\left\| \frac{\mathbf{x}(n)}{\|\mathbf{x}(n)\|} - \mathbf{v}_2 \right\|, \left\| \frac{\mathbf{x}(n)}{\|\mathbf{x}(n)\|} + \mathbf{v}_2 \right\|\right),$$

where the minimum is chosen to account for the possibility of changed signs. Since this estimation is contained in the vector $\mathbf{x}(n)$, these plots do not have discrete updates. The peaks that generate small increments on the estimation error coincide with the moments where the value of AC is updated , because we either modify $\mathbf{x}(0)$ in line 23 of Algorithm 2 or re-start the consensus procedure. Once again, our algorithm performs better than the power iteration by

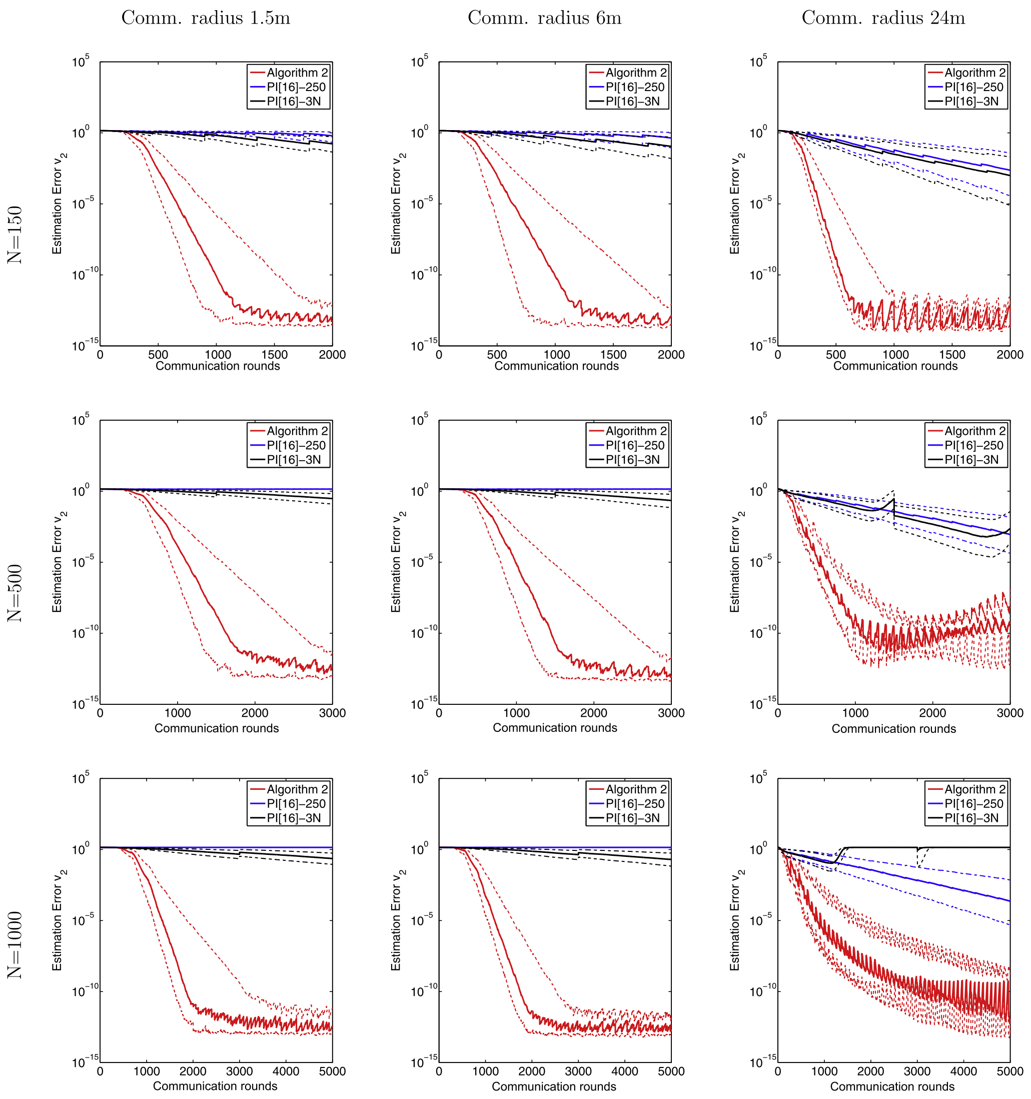

Fig. 3. Median estimation error of the Fiedler vector for networks of different size and communication radius.

several orders of magnitude. The modification of $\mathbf{x}(0)$ affects the estimation more negatively in large, densely connected networks, probably for the same reason as in the estimation of $\lambda_2$, i.e., there is still room for improvement choosing the parameters of the algorithm.

Finally, in order to analyze the scalability of our algorithm as $N$ grows in terms of communication rounds, we have counted the median number of communication rounds required to achieve an accuracy of 4 orders of magnitude. As in the previous simulation, the number of communication rounds includes the rounds required to compute the max-consensus procedure and the update of the initial conditions. The results are shown in Fig. 4, where in this case we have only analyzed our algorithm. Note that as $N$ grows our algorithm performs better and

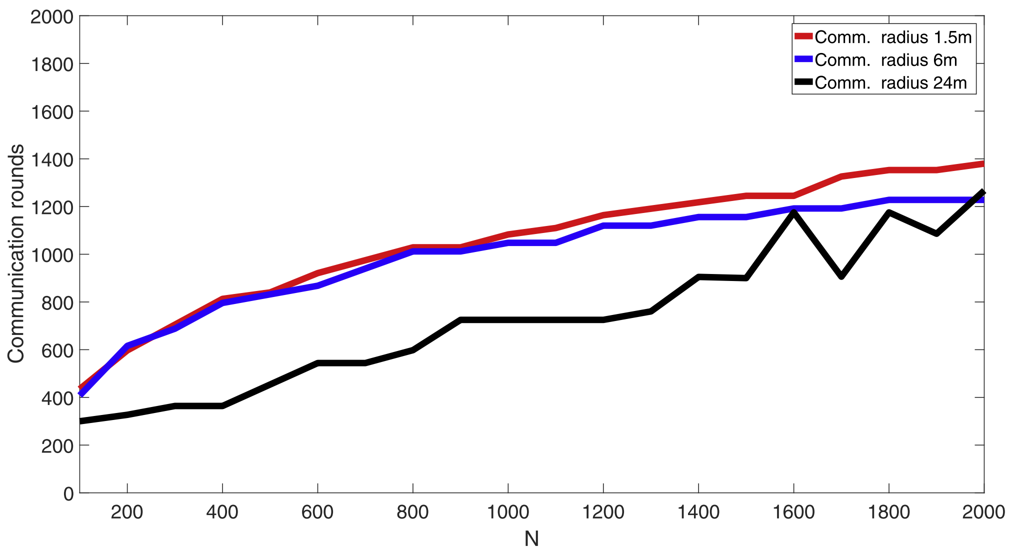

Fig. 4. Median number of communication rounds required to reach an accuracy of $10^{-4}$ for an increasing network size.

better, requiring less than $N$ communication rounds to reach the desired accuracy. Besides, as it would be expected, the denser the network, the less rounds are required due to a smaller diameter of the communication graph.

### 5.2. Simulation of a real network

In this simulation we run the tested algorithms using as network topology the structure of the high-voltage transmission network of US (the adjacency of this network is available, in MatLab format [27]), which contains 4941 nodes. The approximate diameter of this network is equal to 47 hops, and the eigenvalues of interest are $\mu_2 \simeq 0.99996$ and $\lambda_2 \simeq 0.00075$.

The sparse nature of this network, combined with the large amount of nodes, renders the empirical value of $n$ given for the previous networks too small to obtain an accurate estimation of the algebraic connectivity in Algorithm 2. Instead, for this network we have considered $n = 1880$ which corresponds to 40 times the pseudo-diameter of the graph. Additionally, since in the first bisection iterations this value can lead to Chebyshev coefficients beyond numerical representation (NaN), whenever $T_n(c)$ is larger than $10^{70}$ we have stopped the Chebyshev iteration for that bisection round.

The estimation errors of both the algebraic connectivity (AC) and the Fiedler vector are depicted in Fig. 5. The left plot shows the estimation error of the AC, where we can clearly observe the discrete updates the AC, which at the beginning occur in less rounds due to the early termination of the Chebyshev iteration. As $\tilde{\mu}_2 \to \mu_2$, the iteration can be run for the fixed 1880 communication rounds. In 12,000 communication rounds the AC estimation reaches a precision of approximately 8 digits and the Fiedler vector of 4 digits.

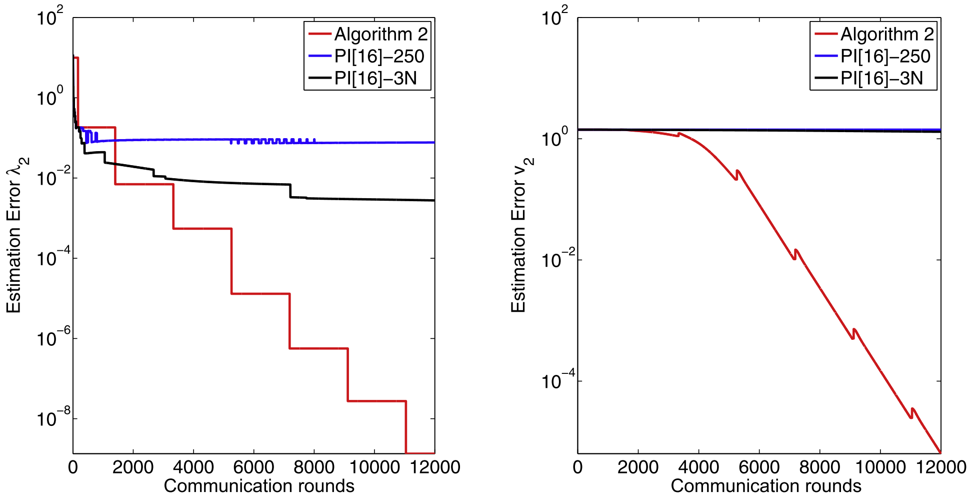

Fig. 5. Estimation error of the algebraic connectivity (left) and the Fiedler eigenvector (right) for the high-voltage transmission network of US, with 4941 nodes.

## 6. Conclusions

We have presented a distributed algorithm to efficiently estimate the algebraic connectivity of a communication network. The method is based on a bisection procedure, combining distributed averaging using Chebyshev polynomials and Max-consensus. The AC estimation is obtained with a small number of communication rounds. We have also proposed several modifications on the basic algorithm to improve its accuracy and accelerate the convergence, also including the estimation of the Fiedler vector. Finally, we have evaluated our method with simulations of multiple large-scale networks and with a large scale real network, showing the performance of our proposal.

## Acknowledgments

This work was funded by the Spanish projects Ministerio de Economía y Competitividad DPI2015-69376-R and CUD2016-17.

## References

[1] L. Xiao, S. Boyd, Fast linear iterations for distributed averaging, Syst. Control Lett. 53 (1) (2004) 65–78.
[2] F. Bullo, J. Cortés, S. Martínez, Distributed Control of Robotic Networks, Applied Mathematics Series, Princeton University Press, 2009. Electronically available at http://coordinationbook.info
[3] D. Nanongkai, Graph and geometric algorithms on distributed networks and databases, Georgia Institute of Technology, 2011 (Ph.D. thesis).
[4] M. Franceschelli, A. Gasparri, A. Giua, C. Seatzu, Decentralized estimation of Laplacian eigenvalues in multi--agent systems, Automatica 49 (4) (2013) 1031–1036.
[5] T. Sahai, A. Speranzon, A. Banaszuk, Wave equation based algorithm for distributed eigenvector computation, in: Proceedings of 49th IEEE International Conference on Decision and Control, 2010, pp. 7308–7315.
[6] T. Sahai, A. Speranzon, A. Banaszuk, Hearing the clusters of a graph: a distributed algorithm, Automatica 48 (1) (2012) 15–24.

[7] D. Kempe, F. McSherry, A decentralized algorithm for spectral analysis, in: Proceedings of the 36th Annual ACM Symposium on Theory of Computing, ACM, 2004, pp. 561–568.
[8] R. Aragues, G. Shi, D.V. Dimarogonas, C. Sagues, K.H. Johansson, Distributed algebraic connectivity estimation for adaptive event-triggered consensus, in: Proceedings of American Control Conference, 2012, pp. 32–37.
[9] R. Aragues, G. Shi, D.V. Dimarogonas, C. Sagues, K.H. Johansson, Distributed algebraic connectivity estimation for undirected graphs with upper and lower bounds, Automatica 50 (12) (2014) 3253–3259.
[10] S. Sundaram, C.N. Hadjicostis, Finite-time distributed consensus in graphs with time-invariant topologies, in: Proceedings of American Control Conference, New York, 2007, pp. 711–716.
[11] T.M.D. Tran, A.Y. Kibangou, Consensus-based distributed estimation of Laplacian eigenvalues of undirected graphs, in: Proceedings of European Control Conference, 2013, pp. 227–232.
[12] V.M. Preciado, A. Jadbabaie, Moment-based spectral analysis of large-scale networks using local structural information, IEEE/ACM Trans. Netw. 21 (2) (2013) 373–382.
[13] V.M. Preciado, A. Jadbabaie, G.C. Verghese, Structural analysis of Laplacian spectral properties of large-scale networks, IEEE Trans. Autom. Control 58 (9) (2013) 2338–2343.
[14] V.M. Preciado, M.M. Zavlanos, A. Jadbabaie, G.J. Pappas, Distributed control of the Laplacian spectral moments of a network, in: Proceedings of American Control Conference, 2010, pp. 4462–4467.
[15] P. Yang, R. Freeman, G. Gordon, K. Lynch, S. Srinivasa, R. Sukthankar, Decentralized estimation and control of graph connectivity for mobile sensor networks, Automatica 46 (2) (2010) 390–396.
[16] A. Bertrand, M. Moonen, Distributed computation of the Fiedler vector with application to topology inference in ad hoc networks, Signal Process. 93 (5) (2013a) 1106–1117.
[17] A. Bertrand, M. Moonen, Topology-aware distributed adaptation of Laplacian weights for in-network averaging, in: Proceedings of the 21st European Signal Processing Conference, 2013b, pp. 1–5.
[18] B. Oreshkin, M. Coates, M. Rabbat, Optimization and analysis of distributed averaging with short node memory, IEEE Trans. Signal Process. 58 (5) (2010) 2850–2865.
[19] P.D. Lorenzo, S. Barbarossa, Distributed estimation and control of algebraic connectivity over random graphs, IEEE Trans. Signal Process. 62 (21) (2014) 5615–5628.
[20] C. Li, Z. Qu, Distributed estimation of algebraic connectivity of directed networks, Syst. Control Lett. 62 (6) (2013) 517–524.
[21] L. Hagemaan, D. Young, Applied Iterative Methods, Academic Press, New York, 1981.
[22] E. Montijano, J. Montijano, C. Sagues, Chebyshev polynomials in distributed consensus applications, IEEE Trans. Signal Process. 61 (3) (2013) 693–706.
[23] E. Montijano, J.I. Montijano, C. Sagues, Adaptive consensus and algebraic connectivity estimation in sensor networks with Chebyshev polynomials, in: Proceedings of the 50th Conference on Decision and Control and European Control Conference, 2011, pp. 4296–4301.
[24] J.C. Mason, D.C. Handscomb, Chebyshev Polynomials, Chapman and Hall, 2002.
[25] N. Lynch, DistributedAlgorithms, Morgan Kaufmann Publishers, 1997.
[26] http://homes.esat.kuleuven.be/~abertran/software.html.
[27] http://code.google.com/p/optimal-synchronization-bounds.