2011 50th IEEE Conference on Decision and Control and
European Control Conference (CDC-ECC)
Orlando, FL, USA, December 12-15, 2011

Adaptive Consensus and Algebraic Connectivity Estimation in Sensor Networks with Chebyshev Polynomials

Eduardo Montijano, Juan I. Montijano and Carlos Sagues

Abstract— **In the recent years a lot of effort has been devoted to the problem of finding distributed algorithms that achieve a fast consensus. The distributed evaluation of polynomials improves the convergence speed to the consensus keeping the good properties of standard methods. The drawback about using polynomials is that they usually require some knowledge about the network in order to have good convergence properties. In this paper we consider the consensus method using Chebyshev polynomials and present an algorithm to compute, in a distributed way, the parameters that make the method get the optimal convergence rate. One of the parameters coincides with the second largest eigenvalue of the weight matrix, i.e., the algebraic connectivity, and we prove the convergence of the algorithm to it. We also present three variants of the algorithm to converge to this parameter in a faster way and to consider changes in the communication topology. We evaluate our algorithm in a simulated environment showing its performance in a wide set of networks.**

Index Terms - Adaptive distributed consensus, Chebyshev polynomials, Algebraic Connectivity Estimation.

I. Introduction

Nowadays a lot of effort is being devoted to research distributed algorithms because they are proving to be useful tools in multi-agent systems and sensor networks. The consensus problem consists in designing a distributed iteration that makes all the nodes in a communication network reach the same value considering only local interactions. Many solutions based on the weighted adjacency matrix have been proposed, see e.g., [1] and the references therein. The appropriate selection of the weights is discussed in [2], but still convergence to the consensus can be slow.

The distributed evaluation of polynomials has turned out to be an easy way to speed up the consensus, also keeping the good properties found in standard methods. The minimal polynomial of the weight matrix is studied in [3], [4], reaching the exact solution in a finite number of iterations. The approach in [5] uses a polynomial of fixed degree with coefficients computed assuming the network is known. Second order recurrences with predictions are used in [6], [7] and Chebyshev polynomials are studied in [8]. The drawback of algorithms using polynomials is that they require some knowledge about the network in order to provide the best performance. This knowledge is usually related to the eigenvalues of the weight matrix of the network. For example, computing the minimal polynomial is equivalent to know all the eigenvalues of the weight matrix. In the case of Chebyshev polynomials, which will be the focus of our study, convergence speed depends on two input parameters. The optimal convergence rate is achieved when these parameters are equal to the second largest (algebraic connectivity) and the smallest eigenvalue of the weight matrix.

Due to the importance the eigenvalues of the weight matrix play in distributed algorithms, several works have considered its distributed computation. In [3] they introduce a distributed algorithm to compute the coefficients of the minimal polynomial. The number and the size of the messages the nodes need to exchange grows linearly with the size of the network, which implies that for large networks a lot of communications will be required. The Fast Fourier Transform (FFT) is used in [9] to estimate all the eigenvalues. In this case the algorithm is also able to pick changes in the communication topology. In order to compute the FFT, all the values of the measured signal along time are required, so that to get a good estimation, the nodes will need to store large amount of data. Finally, the power method is used in [10] to estimate the Fiedler eigenvector and in [6] to estimate the algebraic connectivity. The convergence speed of the power method is determined by the quotient between the two largest eigenvalues of the matrix. When this quotient is close to the unity, the number of iterations required to compute a good estimation of the algebraic connectivity is excessive.

In this paper we propose an extension to the Chebyshev polynomials consensus algorithm [8]. We design an algorithm, to be run at the same time as the consensus one, that adaptively selects the parameters that make the method in [8] reach the fastest convergence rate. Therefore, our algorithm can also be used as an estimator of the algebraic connectivity. The advantage of our approach with respect to prior work is that the estimation does not require big messages or large amounts of memory, giving a good approximation in a relatively small number of iterations. We also present three variants of the algorithm, two of them are used to reach the algebraic connectivity in a faster way and the last one detects changes in the communication topology.

The structure of the paper is the following: In section II we provide some background about the consensus algorithm using Chebyshev polynomials. In section III we present our adaptive consensus algorithm applying bisection techniques.

This work was supported by the project DPI2009-08126 and grant AP2007-03282 Ministerio de Educacion y Ciencia.

E. Montijano and C. Sagues are with Departamento de Informática e Ingeniería de Sistemas - Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, Spain. emonti@unizar.es, csagues@unizar.es

J.I. Montijano is with Departamento de Matemática Aplicada - Instituto Universitario de Matemáticas y Aplicaciones (IUMA), Universidad de Zaragoza, Spain. monti@unizar.es

Section IV provides the variants of the method. Finally in section VI the conclusions of the work are presented.

II. Consensus Using Chebyshev Polynomials

In the paper we consider a sensor network of N agents labeled by $i \in \mathcal{V} = \{1, \ldots, N\}$. Communications between the agents 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 agents i and j directly exchange messages if and only if $(i,j) \in \mathcal{E}$. The set of neighbors of agent 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 agents in the network.

The standard discrete time distributed consensus algorithm based on the weighted adjacency matrix associated to the communication graph [1] is

$$\mathbf{x}(n+1) = \mathbf{A}\mathbf{x}(n), \tag{1}$$

where $\mathbf{x}(n) = (x_1(n), \ldots, x_N(n))^T$ are the values of the state of the different nodes in vectorial form at time step n and $\mathbf{A} = [a_{ij}] \in \mathbb{R}^{N\times N}$, is the weight matrix.

***Assumption** 2.1 (Doubly Stochastic Weights):* $\mathbf{A}$ is symmetric, row stochastic and compatible with the underlying graph, $\mathcal{G}$, i.e., it is such that $a_{ii} > 0,\ a_{ij} = 0$ if $(i,j) \notin \mathcal{E}$, $a_{ij} > 0$ only if $(i,j) \in \mathcal{E}$, $\mathbf{A1} = \mathbf{1}$ and $\mathbf{1}^T\mathbf{A} = \mathbf{1}^T$.
With this assumption, the eigenvalues of $\mathbf{A}$, sorted in decreasing order, satisfy $1 = \lambda_1 > \lambda_2 \geq \ldots \geq \lambda_N > -1$ and the execution of (1) will make all the nodes in the network to asymptotically reach the average of the initial conditions $\mathbf{x}(0)$. However, the convergence speed of this rule is, in general, slow, which implies that many communications are required in order to obtain a good estimation of the average. Without loss of generality, we assume that the algebraic connectivity of the network is characterized by λ_2, i.e., $|\lambda_2| \geq |\lambda_N|$. In this paper we do not deal with the problem of how to choose the weights in $\mathbf{A}$ and we refer the reader, e.g., to [2], for a detailed discussion on this topic.

In [8] we propose a modification of the algorithm to speed up convergence using Chebyshev polynomials. For completeness of the paper we summarize it here. Chebyshev polynomials of first kind [11] are defined with the recurrence:

$$\begin{array}{l} T_0(x) = 1,\ T_1(x) = x \\ T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x), \quad n = 2,3,\ldots. \end{array} \tag{2}$$

The distributed consensus rule using Chebyshev polynomials is defined by

$$\begin{aligned} \mathbf{x}(1) &= \frac{1}{T_1(c-d)}(c\mathbf{A} - d\mathbf{I})\mathbf{x}(0), \\ \mathbf{x}(n+1) &= 2\frac{T_n(c-d)}{T_{n+1}(c-d)}(c\mathbf{A} - d\mathbf{I})\mathbf{x}(n) - \\ &\quad - \frac{T_{n-1}(c-d)}{T_{n+1}(c-d)}\mathbf{x}(n-1). \end{aligned} \tag{3}$$

The parameters c and d are

$$c = \frac{2}{\lambda_M - \lambda_m}, \quad d = \frac{\lambda_M + \lambda_m}{\lambda_M - \lambda_m}, \tag{4}$$

and $1 > \lambda_M > \lambda_m > -1$ are two real coefficients that bring the interval $[\lambda_m, \lambda_M]$ to $[-1, 1]$.

The convergence speed of the algorithm depends on the selection of the parameters λ_m and λ_M. The optimal convergence rate is found when $\lambda_m = \lambda_N$ and $\lambda_M = \lambda_2$.

III. Adaptive Consensus and Algebraic Connectivity Estimation

In this section we present the adaptive consensus algorithm using the Chebyshev Polynomials that leads to the estimation of the algebraic connectivity. The proposed algorithm is based on the bisection method. We describe the process considering a symmetric choice of the parameters, $-\lambda_m = \lambda_M = \lambda$, and therefore, $c = 1/\lambda$ and $d = 0$. The algorithm uses control data to adapt the parameter. For simplicity we explain the algorithm assuming that $\mathbf{x}(0)$ are exactly these control data. The execution of the algorithm with regular initial conditions can be executed in parallel. Each node randomly initializes $x_i(0) \in \{0, 1\}$. We require the following assumption:

***Assumption** 3.1 (Non-null Fiedler eigenvector):* The initial conditions $\mathbf{x}(0)$ expressed as a sum of eigenvectors of $\mathbf{A}$, $\mathbf{x}(0) = \sum_{i=1,\ldots,N} \mathbf{v}_i$, satisfy $\mathbf{v}_2 \neq \mathbf{0}$.

Let us suppose we run the algorithm (3) for a fixed number, n, of iterations choosing $c = 1/\lambda$, with $\lambda \in (0,1)$. Let us define

$$\kappa_n(c) = T_n(c)\frac{\|\mathbf{x}(n) - \mathbf{v}_1\|_\infty}{\|\mathbf{x}(0) - \mathbf{v}_1\|_\infty} \tag{5}$$

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 λ_1, i.e., the average of the initial values and $\|\mathbf{x}(0) - \mathbf{v}_1\|_\infty$ and $\|\mathbf{x}(n) - \mathbf{v}_1\|_\infty$ are the initial and current error in the averaging process. The distributed computation of these data is explained later in the text.

***Proposition** 3.1 (Eigenvalues position indicator):* Given λ and $c = 1/\lambda$ as the parameter for the iteration (3), if $\lambda_i \in [-\lambda, \lambda],\ \forall i = 2, \ldots, N$, then $\kappa_n(c)$ is bounded by

$$\kappa_n(c) \leq \frac{\sum_{i=2}^N \|\mathbf{v}_i\|_\infty}{\|\sum_{i=2}^N \mathbf{v}_i\|_\infty},\ \forall n. \tag{6}$$

Otherwise, at least $\lambda_2 \notin [-\lambda, \lambda]$ and $\lim_{n\to\infty} \kappa_n(c) = \infty$.
Proof. The initial error is

$$\|\mathbf{x}(0) - \mathbf{v}_1\|_\infty = \|\sum_{i=2}^N \mathbf{v}_i\|_\infty. \tag{7}$$

The error after n iterations [8] is equal to

$$\|\mathbf{x}(n) - \mathbf{v}_1\|_\infty = \|\sum_{i=2}^N \frac{T_n(c\lambda_i)}{T_n(c)}\mathbf{v}_i\|_\infty \tag{8}$$

Using these two expressions we get

$$\kappa_n(c) = T_n(c)\frac{\|\mathbf{x}(n) - \mathbf{v}_1\|_\infty}{\|\mathbf{x}(0) - \mathbf{v}_1\|_\infty} = \frac{\|\sum_{i=2}^N T_n(c\lambda_i)\mathbf{v}_i\|_\infty}{\|\sum_{i=2}^N \mathbf{v}_i\|_\infty} \tag{9}$$

If $\lambda_i \in [-\lambda, \lambda]$, it means that $c\lambda_i \leq 1$. Using the explicit representation of the Chebyshev polynomial, $T_n(c\lambda_i) =$

$\cos(n \arccos(c\lambda_i))$, which is upper-bounded (in norm) by 1, we get,

$$\kappa_n(c) = \frac{\|\sum_{i=2}^{N} T_n(c\lambda_i)\mathbf{v}_i\|_\infty}{\|\sum_{i=2}^{N} \mathbf{v}_i\|_\infty} \leq \frac{\sum_{i=2}^{N} \|\mathbf{v}_i\|_\infty}{\|\sum_{i=2}^{N} \mathbf{v}_i\|_\infty}, \ \forall n. \tag{10}$$

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

$$\lim_{n\to\infty} \kappa_n(c) = \lim_{n\to\infty} \frac{\|\sum_{i=2}^{N} T_n(c\lambda_i)\mathbf{v}_i\|_\infty}{\|\sum_{i=2}^{N} \mathbf{v}_i\|_\infty} = \infty \tag{11}$$

■

Taking this into account, for a sufficiently large n we are able to discern when all the eigenvalues of $\mathbf{A}$ are contained in the interval $[-\lambda, \lambda]$ and when they are not.

We can now propose the adaptive algorithm to iteratively choose λ and c using $\kappa_n(c)$ and a bisection method. The algorithm starts with an interval defined by the two extremes, $\lambda_{\min}$ and $\lambda_{\max}$, such that

$$\begin{aligned} &\lambda_i \in [-\lambda_{\max}, \lambda_{\max}], \ \forall i = 2, \ldots, N, \\ &\text{at least } \lambda_2 \notin [-\lambda_{\min}, \lambda_{\min}]. \end{aligned} \tag{12}$$

If there is no knowledge about the network, the initial values of these parameters can be $\lambda_{\min} = 0$ and $\lambda_{\max} = 1$. Following the bisection approach, the first parameter to run the consensus algorithm (3) is $\lambda = (\lambda_{\min} + \lambda_{\max})/2 = 0.5, \ c = 1/\lambda$.

At each consensus round, eq. (3) is run for n iterations using c. This number, n, must be chosen in such a way that $\kappa_n(c)$ has time to diverge when there is some eigenvalue outside the range $[-\lambda, \lambda]$. The bound in eq. (6) is unknown so an estimation, κ is used in the algorithm. If $\kappa_n(c) \leq \kappa$, we know that all the eigenvalues are contained in the interval $[-\lambda, \lambda]$. On the other hand, if $\kappa_n(c) > \kappa$, 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,

$$\begin{aligned} \lambda_{\max} &= \lambda, \text{ if } \kappa_n(c) \leq \kappa \\ \lambda_{\min} &= \lambda, \text{ if } \kappa_n(c) > \kappa. \end{aligned} \tag{13}$$

After that the consensus process is repeated, computing at each round a new estimation of the parameter $\lambda = (\lambda_{\min} + \lambda_{\max})/2$.

***Proposition** 3.2 (Convergence of the method):* Assuming n is large enough, the algorithm (13) is convergent to λ_2, that is $(\lambda_{\max} + \lambda_{\min})/2 \to \lambda_2$.

Proof. By Proposition 3.1, we know that for a sufficiently large n, $\kappa_n(c)$ correctly discriminates if $\lambda_i \in [\lambda, \lambda]$, $\forall i = 2, \ldots, N$. The algorithm in (13) is based on bisection and is convergent because of the use of $\kappa_n(c)$. The value of convergence is the border between the two limit situations, and it is $\lambda = (\lambda_{\max} + \lambda_{\min})/2 \to \max_{i\neq 1} \lambda_i = \lambda_2$. ■

Therefore, the adaptive consensus algorithm updates the parameter c of eq. (3) to optimize the convergence speed of the method. Moreover, as new values of λ are computed, a better estimation of the algebraic connectivity is available.

***Remark** 3.1:* Once a good approximation of λ_2 is available the same process can be applied to estimate λ_N. We need to consider again the standard parametrization of (3) with two parameters. The last estimation, λ, of λ_2 such that $\lambda > \lambda_2$ is assigned to λ_M and the new parameter to adaptively tune is λ_m. The iterative use of (13) to update λ_m will eventually assign the value of λ_N to this parameter.

A. Computation of $\mathbf{v}_1$ *and the errors*

We have not discussed the problem of computing the initial, $\|\mathbf{x}(0) - \mathbf{v}_1\|_\infty$, and final, $\|\mathbf{x}(n) - \mathbf{v}_1\|_\infty$, errors or the exact value of $\mathbf{v}_1$ yet. It is also convenient to discuss the selection of κ and n to have convergence guarantee.

The use of control data, composed by integers, allows us to compute the exact average using an easy rounding technique [12]. Let us recall that each node has initial value of $x_i(0) \in \{0, 1\}$. Therefore the average will be j/N, for some $j = 0, 1, \ldots, N$, and $\mathbf{v}_1 = j/N\mathbf{1}$. Let us assume the number of agents, N, is known by all the nodes. The approximation

$$y_i(n) = \frac{1}{N}\left[N x_i(n)\right], \tag{14}$$

with $[\cdot]$ a rounding operator to the closest integer, will provide us the exact value of $\mathbf{v}_1$, for all n such that $\|\mathbf{x}(n) - \mathbf{v}_1\|_\infty \leq 1/N$.

Since during the first iterations the exact value of $\mathbf{v}_1$ is not available, the errors cannot be computed either. Nevertheless, we can get a good approximation of the infinity norm by

$$\begin{aligned} \|\mathbf{x}(n) - \mathbf{v}_1\|_\infty &\simeq \max_i \{|x_i(n) - y_i(n)| + |y_i(n) - y_i(n-1)|\}, \\ \|\mathbf{x}(0) - \mathbf{v}_1\|_\infty &\simeq \max_i \{|x_i(0) - y_i(n)| + |y_i(n) - y_i(n-1)|\}, \end{aligned}$$

which can be easily computed using a max consensus algorithm [13] in a fixed number of iterations equal to the diameter of $\mathcal{G}$. Since $|x_i(n) - y_i(n)|$ is upper bounded by $1/N$ for all n and i, during the first iterations it will not provide a real estimation of the errors. For that reason the term $|y_i(n) - y_i(n-1)|$ is introduced in the estimation. Also note that, after a finite number of iterations, $\mathbf{y}(n) = \mathbf{y}(n-1) = \mathbf{v}_1$, and the above estimation of the error will be exact whereas using $|x_i(n) - x_i(n-1)|$ it would never be.

The only parameter that cannot be exactly computed is κ, but we can give some more accurate bounds of the value of $\kappa_n(c)$ that can be used by the algorithm as values of κ.

***Lemma** 3.1:* When $c\lambda_i < 1$, $\kappa_n(c)$ is bounded by $\kappa_n(c) \leq \sqrt{N}$

Proof. The bound is obtained from the following inequality:

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

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

Therefore, assigning $\kappa = \sqrt{N}$ will assure that the algorithm will not overestimate the value of λ_2. On the other hand, by choosing a conservative value of κ we require to choose bigger values of n to make $\kappa_n(c)$ diverge. In this way the algorithm also has time to have a good estimation of $y_i(n)$, and therefore of the errors.

The whole process is synthesized in Algorithm 1.

Algorithm 1 Adaptive consensus algorithm

```
Require: $\kappa$, $n$ and $\mathbf{x}(0)$, s.t. $x_i(0) \in \{0, 1\}$
 1: Initialize $\lambda_{\min} = 0$, $\lambda_{\max} = 1$
 2: while $\lambda_{\max} - \lambda_{\min} >$ tolerance do
 3:     $\lambda = (\lambda_{\min} + \lambda_{\max})/2,\ c = 1/\lambda$
 4:     for it$= 1, \ldots, n$ do
 5:         Compute $\mathbf{x}$(it) using (3)
 6:     end for
 7:     Use max consensus to estimate the errors
 8:     Compute $\kappa_n(c)$
 9:     if $\kappa_n(c) \leq \kappa$ then $\lambda_{\min} = \lambda$
10:     else $\lambda_{\max} = \lambda$
11:     end if
12: end while
```

IV. Variants of the Algorithm

In this section we provide three different variants of the basic algorithm that can be used to improve its behavior. The first two variants can be used to improve the convergence rate to the algebraic connectivity and the third one can be used to detect changes in the communication topology, making the algorithm to converge at each step to the current best parameter.

A. Direct estimation of λ_2

The first variant we present makes use of $\kappa_n(c)$ to provide a direct estimation of λ_2 when $c\lambda_2 > 1$. Let us recall that in this situation, $\kappa_n(c) \to \infty$ with speed determined by $T_n(c\lambda_2)$. The direct expression of $T_n(c\lambda_2)$ is characterized [11] by

$$T_n(c\lambda_2) = \frac{1 + \tau^{2n}}{2\tau^n},\ \tau = c\lambda_2 - \sqrt{(c\lambda_2)^2 - 1}. \qquad (15)$$

Therefore, the value of λ_2 can be estimated as follows:

1) Compute $\kappa_n(c)$ and assume $\kappa_n(c) \simeq T_n(c\lambda_2)$
2) Using (15) compute the value of τ from the second degree equation $2\kappa_n(c)\tau^n = 1 + \tau^{2n}$,
$$\tau = \left(\kappa_n(c) - \sqrt{\kappa_n^2(c) - 1}\right)^{1/n} \qquad (16)$$
3) Finally, clear λ_2 from (15) using (16)
$$\lambda_2 \simeq \lambda \frac{\tau^2 + 1}{2\tau}. \qquad (17)$$

Note that the value obtained is still an approximation of the real value of λ_2. Therefore, the bisection iteration still needs to be executed. Whenever a direct estimation of λ_2 is available, the interval is updated by

$$\begin{aligned} \lambda_{\max} &= \lambda_2 + \min(\lambda_{\max} - \lambda_2, \lambda_2 - \lambda_{\min}), \\ \lambda_{\min} &= \lambda_2 - \min(\lambda_{\max} - \lambda_2, \lambda_2 - \lambda_{\min}), \\ \lambda &= \lambda_2, \end{aligned} \qquad (18)$$

where λ_2 here is the estimation obtained in (17). In this way the interval to look for the algebraic connectivity is significantly reduced and so is the number of iterations and consensus rounds.

In order to have a good direct estimation of λ_2, it is desirable to have $T_n(c\lambda_i)\mathbf{v}_i/T_n(c) \simeq 0,\ i = 3, \ldots, N$, or at least $|T_n(c\lambda_2)\mathbf{v}_2| \gg |T_n(c\lambda_i)\mathbf{v}_i|$. To make this happen, at each new consensus round we update the initial conditions by $\mathbf{x}(0) = \mathbf{x}(n)$. With this update the average, $\mathbf{v}_1$, is preserved, but the initial conditions are closer to it, which is the same as to say that $\mathbf{v}_i$ is closer to zero. In addition, the estimations of the errors are also improved by the update because $y_i(n)$ will be closer to the average. Therefore, we are also obtaining a more exact value of $\kappa_n(c)$, improving even more the estimation.

However, we only do the update when $c\lambda_2 > 1$. The reason is that the convergence to zero is faster for the eigenvalues contained within $[-\lambda, \lambda]$ than for those outside the interval. Since λ_2 is not contained in the interval, $\mathbf{v}_2$ is not reduced as much as the other eigenvectors and $\mathbf{v}_2 \gg \mathbf{v}_i$. If we update the initial conditions when $c\lambda_2 < 1$, it is possible that the component associated to $\mathbf{v}_2$ is reduced by a larger factor than for other eigenvectors.

B. Speed up using k-section method

The bisection method has the property of reducing the estimation error, $|\lambda - \lambda_2|$, by a constant factor of 0.5. In sensor 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 the consensus algorithm in parallel with different parameters. In this way, the bounds of λ_2 are delimited with more accuracy and the optimal convergence rate is reached sooner. Specifically, given $\lambda_{\min}$ and $\lambda_{\max}$, the k-section method executes $k-1$ consensus in parallel with parameters

$$\lambda_j = j(\lambda_{\min}+\lambda_{\max})/k,\ c_j = 1/\lambda_j,\ j = 1, \ldots, k-1. \qquad (19)$$

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

$$\begin{aligned} \lambda_{\min} &= \max_j \lambda_j \text{ s.t. } \kappa_n(c_j) \leq \kappa, \\ \lambda_{\max} &= \min_j \lambda_j \text{ s.t. } \kappa_n(c_j) > \kappa. \end{aligned} \qquad (20)$$

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 λ_2 will be reduced by a factor of $1/k$ after each update in the estimation.

C. Detecting changes in the communication topology

Since the evaluation of the Chebyshev polynomial requires the topology to be fixed, we impose that during the n iterations used to estimate $\kappa_n(c)$ the topology remains fixed. However, let us assume that within different consensus rounds the communication topology can change. If we consider a network of mobile sensors, this situation could appear, for example, making the sensors remain static during the computation of $\mathbf{x}(n)$ and letting them move during the computation of $\kappa_n(c)$.

Whenever the eigenvalue λ_2 is contained in the interval $[\lambda_{\min}, \lambda_{\max}]$, the standard method will converge to the right value, even if it changes between estimations of $\kappa_n(c)$. However, as we approach to the algebraic connectivity, the interval $[\lambda_{\min}, \lambda_{\max}]$ will be small, and it will be very likely that a change of the topology displaces λ_2 outside of it. In such case, the standard method will not converge to λ_2.

Using a similar approach to the k-section method we can detect when the eigenvalue we are looking for has left the considered interval. Let us define $c_{\min} = 1/\lambda_{\min}$ and $c_{\max} = 1/\lambda_{\max}$. If we run the consensus iteration in parallel for the three parameters, $c_{\min}, c$ and $c_{\max}$, and $\lambda_2 \in [\lambda_{\min}, \lambda_{\max}]$, it must hold that $\kappa_n(c_{\min}) > \kappa$ and $\kappa_n(c_{\max}) \leq \kappa$. If one of these conditions does not hold we will know that the topology has changed and the algebraic connectivity has left the interval.

In case $\kappa_n(c_{\max}) < \kappa$ that means that λ_2 is above the interval and we can use (17) to obtain a direct estimation of it. If $\kappa_n(c_{\min}) \geq \kappa$ then the value of the algebraic connectivity is below the interval we are analyzing. In this case we do not have any means to estimate an approximate value. The policy we follow is to assign $\lambda_{\max} = \lambda_{\min}$ and $\lambda_{\min} = 0$. Although it is a conservative policy, it ensures that the eigenvalue we are looking for is again within the interval.

V. Simulations

We have analyzed our algorithm in a simulated environment using Monte Carlo methods. We have generated a set of networks of different sizes. For each different number of nodes, we have tried a hundred random networks. In each network the nodes have been randomly positioned in a square of 200×200 meters. Two nodes communicate if they are at a distance lower than 20 meters. The networks have also been forced to be connected. 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 $\mathbf{A}$ has been computed using the "local degree weights" [2].

The value of κ has been set to $\sqrt{N}$ and we have used the value of c in order to decide the number of iterations n executed at each round. We have chosen the minimum n such that $T_n(c) > 100$. In this way, the algorithm always executes enough iterations to converge to the right value.

TABLE I: Results for the standard bisection method

N	λ_2	Diam	Rounds	Iter	n	λ
10	0.910	4.68	6	118.08	90.00	0.917
50	0.982	10.52	6	170.56	107.45	0.978
100	0.985	13.62	6	198.67	116.95	0.982
250	0.992	18.64	6	240.84	129.00	0.993
500	0.992	43.00	6	376.40	150.20	0.994

The results obtained using the bisection method are in Table I. The second column (λ_2) shows the mean algebraic connectivity of all the networks analyzed and the third column (Diam) shows the mean diameter. The column "Rounds" represents the number of estimations of λ before satisfying that $\lambda_{\max} - \lambda_{\min} \leq 10^{-2}$ Therefore, the method is expected to have a tolerance of 10^{-2} in the estimation of λ_2. Since the error is reduced by the same factor at each round, using this method the number of rounds is constant for any size of the network. The total number of iterations (including the max consensus to estimate the errors) is written in the column "Iter" whereas next column (n) shows only the iterations required for the consensus part. Finally, the last column shows the mean of the estimations, λ.

Looking at the results, we can extract some interesting conclusions. First of all, the large values of the mean λ_2 indicate that consensus algorithms evaluated in these networks will require a large number of iterations to achieve consensus. Although our method uses several consensus rounds to estimate the algebraic connectivity, each one of these rounds requires a considerably smaller number of iterations than a standard consensus method. Since the algorithm is intended to be executed at the same time as the standard consensus with real data, we conclude that within one consensus execution with real data we will have the algebraic connectivity.

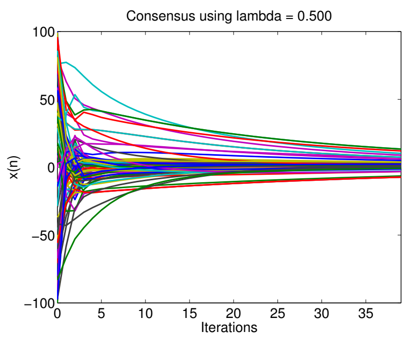
It is also remarkable how well the method escalates with the size of the network. For large networks in less than N iterations the algorithm reaches a good estimation of λ_2. Another thing to remark is that the number of iterations used in the max consensus represents a large fraction of the total, specially as N grows. This happens because the choice of n does not depend on the size of the network but on the connectivity. Therefore, for large networks, the bottleneck of the algorithm, in terms of communications, is the diameter of $\mathcal{G}$.

Figure 1 depicts three executions of the adaptive consensus in a network of 100 nodes using real initial conditions. The three pictures consider the same initial conditions, the same number of iterations and different input parameters, estimated using bisection. It can be seen that as new estimations of λ are computed, the consensus is reached faster.

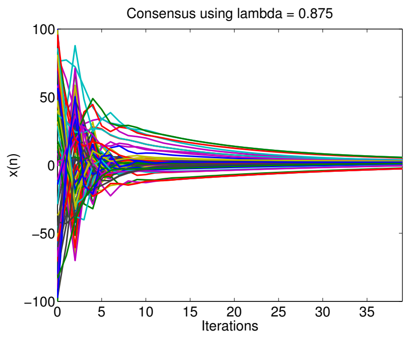
TABLE II: Results with direct estimation of λ_2

N	λ_2	Diam	Rounds	Iter	n	λ
10	0.910	4.68	4.40	86.62	66.20	0.915
50	0.982	10.52	4.02	118.68	76.60	0.979
100	0.985	13.62	4.45	155.02	94.36	0.982
250	0.992	18.64	5.23	215.66	118.31	0.994
500	0.991	43.00	5.10	322.00	129.72	0.994

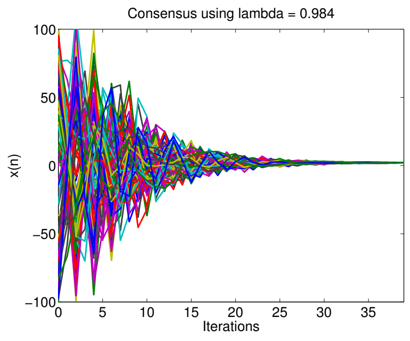
In Table II we show the results for the same experiment but using the direct estimation of λ_2 presented in section IV-A. In this case the number of rounds depends on the computed values of λ_2. Note that this number is smaller than the number of rounds required by the standard bisection to obtain the same tolerance error. As a consequence the number of iterations is also reduced. Finally, it is worth noticing that the estimation of λ_2 is also very precise.



(a) Round 1



(b) Round 3



(c) Round 6

Fig. 1: Adaptive consensus using real data. Evolution of $\mathbf{x}(n)$ using the same initial conditions and number of iterations but different parameters estimated using pure bisection. The algebraic connectivity of the network is 0.987. As new estimations of λ are computed, the consensus is reached faster.

A. *Estimation in switching topologies*

In this subsection we show an experiment where we allow the communication graph to change and how our method tracks the algebraic connectivity using the k-section method.

We have considered a random network composed by 50 nodes. Since during the computation of $\mathbf{x}(n)$ the network must be static, we have only modified the network during the estimation of the errors. The evolution of the estimations is depicted in Fig. 2. We can see that the algorithm detects the changes in the topology and adjust the intervals in consequence. If the topology remains fixed for enough iterations, the method estimates the value of the algebraic connectivity. When the algebraic connectivity is reduced, instead of assigning $\lambda_{\min} = 0$ we have used $\lambda_{\min} = \lambda_{\max} - 0.1$. Although sometimes, e.g., just before iteration 200, more than one consensus round is required to adapt the interval, with this assignment, the convergence is in general faster to the real value of λ_2.

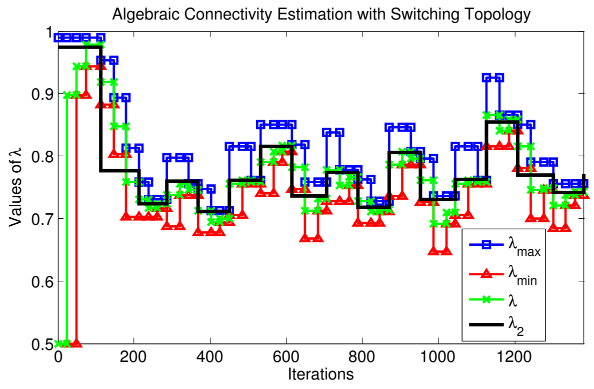


Fig. 2: Estimation of the algebraic connectivity under a switching communication network. The algorithm detects the changes and adapts the interval to estimate at each round the algebraic connectivity.

VI. **Conclusions**

We have presented a new distributed adaptive consensus algorithm using the bisection method. The method is designed to autonomously tune the input parameter of the consensus algorithm using Chebyshev polynomials to reach the optimal convergence rate. The obtained parameter coincides with the algebraic connectivity, and therefore our algorithm can also be used by other applications that make use of this parameter. The proposed method uses simple control data, for which an estimation of the exact average and the norm of the error can be computed with consensus methods in a finite number of iterations. We have also presented three variants of the basic method to speed up the convergence to the optimal input parameter, as well as to detect changes in the communication topology, still achieving convergence to the algebraic connectivity. We have evaluated our method with an extensive set of simulations, showing the goodness of our proposal.

References

[1] F. Bullo, J. Cortés, and S. Martínez. *Distributed Control of Robotic Networks*. Applied Mathematics Series. Princeton University Press, 2009. Electronically available at http://coordinationbook.info.

[2] L. Xiao and S. Boyd. Fast linear iterations for distributed averaging. *Systems and Control Letters*, 53:65–78, 2004.

[3] S. Sundaram and C. N. Hadjicostis. Finite-time distributed consensus in graphs with time-invariant topologies. In *American Control Conference*, pages 711–716, New York, 2007.

[4] Y. Yuan, G. Stan, L. Shi, and J. Gonçalves. Decentralised final value theorem for discrete-time lti systems with application to minimal-time distributed consensus. In *IEEE Int. Conference on Decision and Control*, pages 2664–2669, 2009.

[5] E. Kokiopoulou and P. Frossard. Polynomial filtering for fast convergence in distributed consensus. *IEEE Transactions on Signal Processing*, 57(1):342354, 2009.

[6] B. Oreshkin, M. Coates, and M. Rabbat. Optimization and analysis of distributed averaging with short node memory. *IEEE Transactions on Signal Processing*, 58(5):2850–2865, May 2010.

[7] S. Muthukrishnan, B. Ghosh, and M. H. Schultz. First- and second-order diffusive methods for rapid, coarse, distributed load balancing. *Theory of Computing Systems*, 31(4):331–354, 1998.

[8] E. Montijano, J. I. Montijano, and C. Sagues. Fast distributed consensus with chebyshev polynomials. In *American Control Conference*, 2011. to appear.

[9] M. Franceschelli, A. Gasparri, A. Giua, and C Seatzu. Decentralized laplacian eigenvalues estimation for networked multi-agent systems. In *IEEE Int. Conference on Decision and Control*, pages 2717–2722, 2009.

[10] P. Yang, R.A. Freeman, G.J. Gordon, K.M. Lynch, S.S. Srinivasa, and R. Sukthankar. Decentralized estimation and control of graph connectivity for mobile sensor networks. *Automatica*, 46(2):390–396, Feb 2010.

[11] J.C. Mason and D. C. Handscomb. *Chebyshev Polynomials*. Chapman and Hall, 2002.

[12] E. Montijano, S. Martínez, and C. Sagués. *De-RANSAC*: Robust consensus for robot formations. In *Network Science and Systems in Multi-Robot Autonomy, Workshop at the IEEE International Conference on Robotics and Automation 2010*, May 2010.

[13] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann publishers, 1997.