Distributed Algebraic Connectivity Estimation for Adaptive Event-triggered Consensus

R. Aragues G. Shi D. V. Dimarogonas C. Sagues K. H. Johansson

***Abstract*— In several multi agent control problems, the convergence properties and speed of the system depend on the algebraic connectivity of the graph. We discuss a particular event-triggered consensus scenario, and show that the availability of an estimate of the algebraic connectivity could be used for adapting the behavior of the average consensus algorithm. We present a novel distributed algorithm for estimating the algebraic connectivity, that relies on the distributed computation of the powers of matrices. We provide proofs of convergence, convergence rate, and upper and lower bounds at each iteration of the estimated algebraic connectivity.**

I. INTRODUCTION

Consensus problems are connected to diverse applications in multi-agent systems, including sensor fusion, flocking, formation control or rendezvous among others [1]. Event-triggered control strategies [2], [3] are appropriate for scenarios where the state variables evolve in continuous time but where the agents may exchange data only at specific time instances. The algebraic connectivity is an important network property for all the previous systems to reach convergence and it characterizes the convergence rate. We propose a distributed method for estimating this algebraic connectivity.

Connectivity control methods establish agent motions that preserve or maximize some network connectivity property. The $k-$connectivity matrix of the graph is computed in a centralized fashion in [4]. Several distributed methods compute spanning subgraphs [5], specific Laplacian eigenvectors [6], moments (mean, variance, skewness and kurtosis) of the Laplacian eigenvalue spectrum [7], or maximize the algebraic connectivity trough motion control without actually computing it [8]. Although these control methods improve the network connectivity, they do not characterize any particular Laplacian eigenvalue, as required in our case.

A Laplacian eigenvalues estimation method is given in [9]. Nodes execute a local interaction rule that makes their states oscillate at frequencies corresponding to these eigenvalues, and use the Fast Fourier Transform (FFT) on their states to identify these eigenvalues. The main limitation of this work is that the proper adjustment of the FFT, so that the eigenvalues can be correctly identified, is nontrivial. In addition, some nodes may observe only a subset of the eigenvalues and thus they need to execute additional coordination rules for propagating their data. Several solutions to the computation of the Laplacian spectra rely on the power iteration method or variations [10]–[12]. Power iteration [13] selects an initial vector and then repeatedly multiplies it by a matrix and normalizes it. This vector converges to the eigenvector associated to the leading eigenvalue (the one with the largest absolute value) of the matrix. The original matrix can be previously deflated so that a particular eigenvalue becomes the leading one. The main limitation of distributed power iteration approaches consists on the normalization and orthonormalization of the vectors at each step. For [11] it involves a gossip-based information aggregation algorithm, and for [10] a distributed averaging method. Therefore, several iterations of the previous algorithms must be executed by the nodes between consecutive steps of the power method in order to ensure that they have achieved the required accuracy in the vector normalization. Besides, the previous methods only ensure convergence but they do not give any upper or lower bound relating the true algebraic connectivity and the estimates at each iteration.

We propose a distributed method for computing the algebraic connectivity where, at each step k, the agents compute the $k-$th power of a deflated Laplacian. When the nodes want to obtain an estimate of the algebraic connectivity, they run a $\max-$consensus [14]. The agents do not need to wait for the $\max-$consensus to finish before starting the next step $k+1$. Instead, they can continue executing the matrix power algorithm in parallel. We provide proofs of convergence and convergence speed, and give upper and lower bounds for the true algebraic connectivity at each iteration. We combine the previous ideas with [3] and present an adaptive triggered consensus method where the most recent estimate of the algebraic connectivity is used at each step.

This paper is organized as follows: Section II states the problem; Section III presents the distributed algebraic connectivity estimation method; and Section IV evaluates our method in a simulated scenario.

II. PRELIMINARIES

We use the notation in Table I. Consider $n \in \mathbb{N}$ agents which can exchange information with nearby nodes. This information is represented by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, n\}$ are the agents, and $\mathcal{E}$ are the edges. There is an edge $(i, j) \in \mathcal{E}$ between nodes i and j if they can exchange data. We say a $n \times n$ matrix C is

R. Aragues and C. Sagues are with the Instituto de Investigación en Ingeniería de Aragón, Universidad de Zaragoza, Spain raragues@unizar.es, csagues@unizar.es

G. Shi, D. V. Dimarogonas and K. H. Johansson are with Access Linnaeus Centre, Royal Institute of Technology (KTH), Sweden. guodongs@kth.se, dimos@kth.se, kallej@ee.kth.se

This work was supported by project Ministerio de Ciencia e Innovacion DPI2009-08126 and grant MEC BES-2007-14772.

The third author is also affiliated with the Centre of Autonomous Systems (CAS) at KTH and is supported by the Swedish Research Council through contract VR-2009-3948.

TABLE I
NOTATION.

Indices			
n	Number of agents.	k	Iteration number, $k \in \mathbb{N}$.
i, j	Agent indices.	t	Time, $t \in \mathbb{R}_{t\geq 0}$.
Matrix operations, eigenvalues and eigenvectors			
A_{ij}, $[A]_{ij}$	(i,j) entry of matrix A.		
$\text{diag}(b_1, \ldots, b_r)$	matrix A with $A_{ii} = b_i$ and $A_{ij} = 0$.		
$\lambda_i(A)$, $\mathbf{v}_i(A)$	i^{th} eigenvalue and eigenvector of A.		
λ_A	$\text{diag}(\lambda_1(A), \ldots, \lambda_r(A))$.		
V_A	$[\mathbf{v}_1(A), \ldots, \mathbf{v}_r(A)]$.		
$\|A\|_\infty$	Induced $\infty-$norm of A, $\max_i \sum_{j=1}^n \|A_{ij}\|$.		
$\|A\|_2$	Spectral norm of A, $\max_i \sqrt{\lambda_i(A^T A)}$.		
$\rho(A)$	Spectral radius of A, $\max_i \|\lambda_i(A)\|$.		
Special matrices			
$\mathbf{I}_r$	$r \times r$ identity matrix.		
$\mathbf{0}_r$, $\mathbf{1}_r$	Column vectors with the r entries equal to 0 and 1.		
$\mathcal{A}$	Adjacency matrix of the graph.		
$\mathcal{L}$	Laplacian matrix of the graph, $\mathcal{L} = \text{diag}(\mathcal{A}\mathbf{1}) - \mathcal{A}$.		
$\lambda_\star(\mathcal{L})$	Algebraic connectivity, the second-smallest $\lambda_i(\mathcal{L})$.		

compatible with $\mathcal{G}$ if $C_{ij} = 0$ iff $(i,j) \notin \mathcal{E}$ for $j \neq i$; we let the elements in the diagonal C_{ii} be either equal or different than 0. The adjacency matrix $\mathcal{A} \in \{0,1\}^{n\times n}$ of $\mathcal{G}$ is

$$\mathcal{A}_{ij} = 1 \text{ if } (i,j) \in \mathcal{E}, \mathcal{A}_{ij} = 0 \text{ otherwise, for } i,j \in \mathcal{V}. \quad (1)$$

We assume $\mathcal{G}$ is connected. We use $\mathcal{N}_i$ for the set of neighbors of a node i with whom i can exchange data, $\mathcal{N}_i = \{j \mid (i,j) \in \mathcal{E}\}$, and we let d_i be the *degree* of node i defined as the cardinality of $\mathcal{N}_i$, and $d_{\max} = \max_{i\in\mathcal{V}} d_i$. The Laplacian matrix $\mathcal{L} \in \mathbb{R}^{n\times n}$ of $\mathcal{G}$ is $\mathcal{L} = \text{diag}(\mathcal{A}\mathbf{1}) - \mathcal{A}$, and we sort its eigenvalues as follows $\lambda_1(\mathcal{L}) \leq \lambda_2(\mathcal{L}) \leq \cdots \leq \lambda_n(\mathcal{L})$. Both $\mathcal{A}$ and $\mathcal{L}$ are compatible with $\mathcal{G}$. The Laplacian $\mathcal{L}$ has the following well known properties [1]:

(i) It has an eigenvector $\mathbf{v}_1(\mathcal{L}) = \mathbf{1}/\sqrt{n}$ with associated eigenvalue $\lambda_1(\mathcal{L}) = 0$, $\mathcal{L}\mathbf{1}/\sqrt{n} = \mathbf{0}$;

(ii) When the graph $\mathcal{G}$ is connected, then all the other eigenvalues are strictly greater than 0,

$$0 = \lambda_1(\mathcal{L}) < \lambda_2(\mathcal{L}) \leq \cdots \leq \lambda_n(\mathcal{L}); \text{ and}$$

(iii) Its eigenvalues are upper-bounded by $\lambda_n(\mathcal{L}) \leq 2d_{\max}$.

The *algebraic connectivity* of $\mathcal{G}$ denoted by $\lambda_\star(\mathcal{L})$ is the second-smallest eigenvalue $\lambda_2(\mathcal{L})$ of the Laplacian $\mathcal{L}$.

A. Consensus Protocol and Event-based Control

Consider each agent $i \in \mathcal{V}$ has single-integrator dynamics

$$\dot{x}_i(t) = u_i(t), \text{ with } x_i(t), u_i(t) \in \mathbb{R}, \quad (2)$$

where u_i denotes the control input at agent i given by

$$u_i(t) = -\sum_{j\in\mathcal{N}_i} (x_i(t) - x_j(t)). \quad (3)$$

With stack vectors $\mathbf{x} = [x_1, \ldots, x_n]^T$, $\mathbf{u} = [u_1, \ldots, u_n]$,

$$\dot{\mathbf{x}}(t) = -\mathcal{L}\mathbf{x}(t) = \mathbf{u}(t). \quad (4)$$

A well known result [1] is that if $\mathcal{G}$ is undirected and connected, then the previous algorithm globally asymptotically solves the average consensus problem, i.e.,

$$\lim_{t\to\infty} x_i(t) = (x_1(0) + \cdots + x_n(0))/n. \quad (5)$$

However in general agents cannot exchange their states continuously, and the continuous-time law needs to be implemented on a digital platform. A triggered-based control method is proposed in [3] where agents monitor their own states $x_i(t)$ continuously but propagate their most recent states at some time instances. The trigger condition is

$$|e_i(t)| \geq c_1 e^{-\alpha t}, \quad c_1 > 0, \quad 0 < \alpha < \lambda_\star(\mathcal{L}), \quad (6)$$

where $e_i(t)$ is the difference between the actual current state $x_i(t)$ at agent $i \in \mathcal{V}$ at time t, and $\hat{x}_i(t)$ the last transmitted one, $e_i(t) = \hat{x}_i(t) - x_i(t)$. An event for agent i is triggered as soon as the condition in eq. (6) is satisfied, resulting in agent i sending its most recent state $\hat{x}_i$. Each agent i updates its control-law when the event is triggered, or when it receives an updated state $\hat{x}_j$ from one of its neighbors. Thus, the control-law is piecewise constant. As stated by [3, Th. 4], for connected graphs if nodes execute the previous procedure, then for all initial conditions $\mathbf{x}(0)$ the overall system converges to average consensus asymptotically. Furthermore the closed-loop system does not exhibit Zeno-behavior.

In general, nodes do not know $\lambda_\star(\mathcal{L})$. However, if they had a lower-bound $\hat{\lambda}(k) \leq \lambda_\star(\mathcal{L})$ of $\lambda_\star(\mathcal{L})$, for $k \in \mathbb{N}$, they could use the following trigger condition,

$$|e_i(t)| \geq c_1 e^{-\alpha(t)t}, \quad \alpha(t) = \gamma\hat{\lambda}(k), \text{ for } t \in [k, k+1),$$

with $0 < \gamma < 1$, where instead of using a fixed α, nodes adapt this value depending on the most recent and accurate estimate $\hat{\lambda}(k)$ of the algebraic connectivity $\lambda_\star(\mathcal{L})$. Based on similar analysis as in [3], it can be shown that the resulting algorithm is convergent, and does not exhibit Zeno-behavior.

B. Problem description

Our goal is to design distributed algorithms to allow the agents to compute $\lambda_\star(\mathcal{L})$, and/or a lower bound of $\lambda_\star(\mathcal{L})$.

III. ALGEBRAIC CONNECTIVITY ESTIMATION

We present a novel distributed algorithm for computing the algebraic connectivity $\lambda_\star(\mathcal{L})$ of the graph. This algorithm relies on the observation that the induced infinite norm of a matrix $\|C\|_\infty$ can be easily computed in a distributed fashion with a $\max-$consensus method, provided that each node knows a row of this matrix; and that $\|C^k\|_\infty^{\frac{1}{k}}$ successively approaches the spectral radius $\rho(C)$ of matrix C. First, we present some results on matrices *compatible* with $\mathcal{G}$.

A. Distributed computation of power of matrices

We show that the powers a matrix C *compatible* with the graph, can be computed in a distributed fashion. Our discussion refers to fixed undirected graphs, although the method can be easily extended to time-varying graphs. Our algorithm was originally proposed in [15] for adjacency matrices defined by blocks. Here we propose an improved version that does not require the knowledge of n.

Algorithm III.1 (Basic Distributed Power Computation) *Let each node $i \in \mathcal{V}$ maintain an estimate $\hat{C}_{ij}(k)$ of the*

(i,j) entries of the $k-$th power of C, $[C^k]_{ij}$, for all $j \in \mathcal{V}$. At $k=0$, node i initializes its variables $\hat{C}_{ij}(k)$ with

$$\hat{C}_{ii}(0) = 1, \text{ and } \hat{C}_{ij}(0) = 0 \text{ for } j \in \mathcal{V} \setminus \{i\}. \quad (7)$$

At each $k \geq 1$, node i updates these variables as follows,

$$\hat{C}_{ij}(k+1) = \sum_{j' \in \mathcal{N}_i \cup \{i\}} C_{ij'} \hat{C}_{j'j}(k), \text{ for } j \in \mathcal{V}. \quad (8)$$

Proposition III.1 *When C is compatible with the graph, the outcomes of algorithm* (8) *at each step $k \geq 0$ are exactly the entries of the $k-$th power of C, C^k.*

Proof: Each node i maintains exactly the $i-$th row of C^k. For $k=0$, it is straightforward to see that eq. (7) gives the identity matrix $\mathbf{I}$ which is exactly C^0. For $k \geq 1$, we consider the explicit expression for $C^{k+1} = CC^k$, and take into account that, since C is compatible with the graph, then $C_{ij'} = 0$ for $j' \notin \mathcal{N}_i \cup \{i\}$. Each (i,j) entry of C^{k+1} is

$$[C^{k+1}]_{ij} = \sum_{j'=1}^{n} C_{ij'}[C^k]_{j'j} = \sum_{j' \in \mathcal{N}_i \cup \{i\}} C_{ij'}[C^k]_{j'j}, \quad (9)$$

which is exactly what algorithm (8) does. ■

Note that each node i updates its variables using only its own and its neighbors' data; it stores n scalars, and exchanges n scalars at each iteration k. Our algorithm exactly computes C^k at each step k (not an estimate of it). Observe that it remains valid if the communication graph is time-varying, in which case each agent i computes

$$[\hat{C}(k)]_{ij} = [C(k)C(k-1)\dots C(0)]_{ij}, \text{ for } j \in \mathcal{V}.$$

The main limitation of the previous procedure is that it assumes that in the initial phase (eq. (7)) each agent knows the total amount of agents in the network n, and that at each step k (eq. (8)) it knows the identities j, j' associated to each piece of data $\hat{C}_{j'j}(k)$. We show here that the algorithm can be slightly modified so that the previous requirement is not necessary. We only impose the assumption that each agent i has assigned a unique identifier $ID(i)$, e.g., its IP address.

Algorithm III.2 (Distributed Power Computation) *Each node $i \in \mathcal{V}$ maintains a set of node identifiers $l_i(k)$, and an estimate $\tilde{C}_{ij}(k)$ of the (i,j) entries of the $k-$th power of C, $[C^k]_{ij}$, associated to the nodes j such that $ID(j) \in l_i(k)$.*

1: At $k=0$, each node $i \in \mathcal{V}$ initializes a single variable $\tilde{C}_{ii}(k)$ and a single identifier,

$$\tilde{C}_{ii}(0) = 1, \qquad l_i(0) = \{ID(i)\}, \quad (10)$$

and sends this data to its neighbors $\mathcal{N}_i$.

2: At each step $k \geq 1$, node i first looks for new nodes in the information $l_j(k)$ received from its neighbors, and updates its identifiers $l_i(k)$ accordingly,

$$l_i(k+1) = \bigcup_{j \in \mathcal{N}_i \cup \{i\}} l_j(k). \quad (11)$$

3: Then, node i creates a new variable $\tilde{C}_{ij}(k)$ initialized with 0, $\tilde{C}_{ij}(k) = 0$, for each recently discovered node j,

$$ID(j) \in l_i(k+1) \text{ and } ID(j) \notin l_i(k).$$

4: Finally, node i updates all its variables $\tilde{C}_{ij}(k)$, for $ID(j) \in l_i(k+1)$, by

$$\tilde{C}_{ij}(k+1) = \sum_{j' \in \mathcal{N}_i \cup \{i\}, ID(j) \in l_{j'}(k)} C_{ij'} \tilde{C}_{j'j}(k), \quad (12)$$

and sends these variables $\tilde{C}_{ij}(k)$, for $ID(j) \in l_i(k)$, to its neighbors as well as its discovered identifiers $l_i(k)$.

Proposition III.2 *Let us define for each node $i \in \mathcal{V}$ and each step $k \geq 0$ variables $\tilde{C}_{ij}(k) = 0$ for all $ID(j) \notin l_i(k)$. Then, when C is compatible with the graph, the outcomes of algorithm* (12) *are exactly the outcomes of* (8)*. As a result, they are exactly the entries of the $k-$th power of C, C^k.*

Proof: We first consider Algorithm III.1 together with the node identifier management rule,

$$l_i(0) = \{ID(i)\}, \ \hat{C}_{ii}(0) = 1, \ \hat{C}_{ij}(0) = 0, \text{ for } j \neq i, \quad (13)$$

$$l_i(k+1) = \bigcup_{j \in \mathcal{N}_i \cup \{i\}} \{l_j(k)\}, \text{ and,}$$

$$\hat{C}_{ij}(k+1) = \sum_{j' \in \mathcal{N}_i \cup \{i\}} C_{ij'} \hat{C}_{j'j}(k), \text{ for } j \in \mathcal{V}. \quad (14)$$

We want to show that if $j \notin l_i(k)$ then the element $\hat{C}_{ij}$ is zero. This is proved by induction. It is true for $k=0$, see eq. (13). Let us assume that for k it is true that, for all $i \in \mathcal{V}$, if $j \notin l_i(k)$ then $\hat{C}_{ij} = 0$. Consider a j which at $k+1$ satisfies $j \notin l_i(k+1)$. By eq. (14) it means that $j \notin \bigcup_{j' \in \mathcal{N}_i \cup \{i\}} \{l_{j'}(k)\}$ and therefore for all $j' \in \mathcal{N}_i \cup \{i\}$ the element $\hat{C}_{j'j} = 0$. Then, the update rule (14) gives

$$\hat{C}_{ij}(k+1) = \sum_{j' \in \mathcal{N}_i \cup \{i\}} C_{ij'} 0 = 0. \quad (15)$$

Now we prove by induction that the outcomes $\tilde{C}_{ij}(k)$ of algorithm (III.2) are exactly equal to $\hat{C}_{ij}(k)$ for all $k \geq 0$, $i \in \mathcal{V}$ and all $j \in l_i(k)$. Note that for all $k \geq 0$ all $i \in \mathcal{V}$ and all $j \notin l_i(k)$ the elements $\tilde{C}_{ij}(k)$ do not exist whereas $\hat{C}_{ij}(k) = 0$ as shown above. We pay attention to eq. (14) for $ID(j) \in l_i(k+1)$. For $j \notin l_{j'}(k)$ we have $\hat{C}_{j'j}(k) = 0$. If $\hat{C}_{j'j}(k) = \tilde{C}_{j'j}(k)$ at k, then at $k+1$ we have

$$\hat{C}_{ij}(k+1) = \sum_{\substack{j' \in \mathcal{N}_i \cup \{i\}, \\ ID(j) \in l_{j'}(k)}} C_{ij'} \tilde{C}_{j'j}(k) + \sum_{\substack{j' \in \mathcal{N}_i \cup \{i\}, \\ ID(j) \notin l_{j'}(k)}} C_{ij'} 0,$$

which is the update rule for $\tilde{C}_{ij}(k+1)$ in eq. (12). ■

Algorithm III.2 provides each agent i with all the entries of the $i-th$ row of the power matrix C^k, and only requires each node i to have a unique identifier $ID(i)$. The results presented so far hold for both fixed and time-varying graphs. Now we show that in addition, for fixed graphs, the previous method can be used for obtaining the number of nodes n.

Proposition III.3 *For each node $i \in \mathcal{V}$, let k_i be the first instant for which $l_i(k) = l_i(k-1)$,*

$$k_i = \min\{k \mid l_i(k) = l_i(k-1)\}. \tag{16}$$

Then, $n = |l_i(k_i)|$.

Proof: Note that $l_i(k-1)$ contains the identifiers of the $(k-1)-$hop neighbors of i. By the definition of a path, if there are no new nodes at distance k, then there cannot be new nodes at distances greater than k. Therefore $l_i(k-1)$ already contains the identifiers of all the nodes that are connected with i. Since the graph is connected, these nodes are all the nodes in the network and $n = |l_i(k_i)|$. ■

For fixed communication graphs, the previous result can be further used in Algorithm III.2 for improving the network usage. Since the first time $l_i(k) = l_i(k-1)$, agent i can stop executing steps 2 : to 3 : and exchanging variables $l_i(k)$.

B. Distributed computation of the spectral radius

Now we present a distributed algorithm that allows the computation of the spectral radius of a symmetric matrix C compatible with the graph. It relies on the observation that, for any induced norm $\|.\|$, [16, Chap. 5.6]

$$\rho(C) \leq \|C\|, \text{ and } \rho(C) = \lim_{k\to\infty} \|C^k\|^{\frac{1}{k}}. \tag{17}$$

We propose to use the induced $\infty-$norm $\|.\|_\infty$,

$$\|C^k\|_\infty = \max_{i\in\mathcal{V}}\{|[C^k]_{i1}| + \cdots + |[C^k]_{in}|\}, \tag{18}$$

since it can be easily computed by the agents using a distributed $\max-$consensus algorithm provided that each agent i knows the $i-th$ row of C^k.

Algorithm III.3 (Distributed Spectral Radius) *Consider the agents executing Algorithm III.2 for a symmetric matrix C compatible with the graph. Let $c_i(k)$ be the sum of the absolute values of variables $\tilde{C}_{ij}(k)$ at agent i, step k,*

$$c_i(k) = \sum_{ID(j)\in l_i(k)} |\tilde{C}_{ij}(k)|. \tag{19}$$

Nodes run a $\max-$consensus [14] on their variables $c_i(k)$,

$$\beta_i(k) = c_i(k), \quad \beta_i(k+\tau+1) = \max_{j\in\mathcal{N}_i\cup\{i\}} \beta_j(k+\tau), \tag{20}$$

which finishes after $T = \text{diam}(\mathcal{G})$ communication rounds with variables $\beta_i(k+T)$ at all the nodes $i \in \mathcal{V}$ containing the maximum of the inputs $c_i(k)$ over all the network,

$$\beta_1(k+T) = \cdots = \beta_n(k+T) = \max_{i\in\mathcal{V}} c_i(k). \tag{21}$$

The spectral radius $\beta_i^*(k)$ estimated at node i, step $k \geq 1$ is

$$\beta_i^*(k) = (\beta_i(k+T))^{\frac{1}{k}} = (\max_{j\in\mathcal{V}} c_j(k))^{\frac{1}{k}}. \tag{22}$$

Observe that this computation of $c_i(k)$ in eq. (19) is local to each node i, since it maintains the $i-th$ row of C^k. Note that the estimated spectral radius $\beta_i^*(k)$ associated to step k is available at the nodes T iterations later (at iteration $k+T$). However, the $\max-$consensus iterations (20) are executed independently (in parallel) to the Algorithm III.2. This means that agents do not have to wait T iterations for the $\max-$consensus to converge before executing a new iteration of Algorithm III.2. Now we present a result that establishes the convergence of the previous algorithm to the spectral radius of the matrix C.

Theorem III.4 *Consider each node i executing Algorithm III.3 with a symmetric matrix C compatible with the graph. Then, as $k \to \infty$ all the variables $\beta_i^*(k)$ converge to the spectral radius $\rho(C)$ of matrix C,*

$$\lim_{k\to\infty} \beta_i^*(k) = \rho(C), \text{ for all } i \in \mathcal{V}, \text{ and} \tag{23}$$

$$(\sqrt{n})^{\frac{-1}{k}} \beta_i^*(k) \leq \rho(C) \leq \beta_i^*(k), \text{ for all } k \geq 1. \tag{24}$$

Proof: First note that Algorithm III.3 computes the $k-$th root of the induced infinite norm of C^k. Since we showed that $\tilde{C}_{ij}(k) = 0$ for $ID(j) \notin l_i(k)$, then $c_i(k)$ in eq. (19) is exactly the absolute sum of the $i-th$ row of C^k. The $\max-$consensus (20) provides each agent with the induced infinite norm of C^k. Thus, $\beta_i^*(k)$ in eq. (22) equals $\beta_i^*(k) = (\|C^k\|_\infty)^{\frac{1}{k}}$, which combined with eq. (17) gives

$$\rho(C) = \lim_{k\to\infty} \|C^k\|_\infty^{\frac{1}{k}} = \lim_{k\to\infty} \beta_i^*(k), \tag{25}$$

as stated in eq. (23). Now we focus on eq. (24); from (17),

$$\rho(C) = (\rho(C^k))^{\frac{1}{k}} \leq \|C^k\|_\infty^{\frac{1}{k}} = \beta_i^*(k), \tag{26}$$

which gives the right part in eq. (24). Since matrix C is symmetric, then its spectral norm $\|C\|_2 = \max_i \sqrt{\lambda_i(C^2)}$ equals its spectral radius $\rho(C) = \max_i |\lambda_i(C)|$,

$$\rho(C) = \|C\|_2 = \|C^k\|_2^{\frac{1}{k}}. \tag{27}$$

The spectral $\|C^k\|_2$ and induced infinite $\|C^k\|_\infty$ norms of a matrix C^k are related by $(\sqrt{n})^{-1}\|C^k\|_\infty \leq \|C^k\|_2$, [16, Chap. 5.6], which combined with eq. (27) gives

$$(\sqrt{n})^{-\frac{1}{k}} \beta_i^*(k) = (\sqrt{n})^{-\frac{1}{k}} \|C^k\|_\infty^{\frac{1}{k}} \leq \|C^k\|_2^{\frac{1}{k}} = \rho(C), \tag{28}$$

as stated in eq. (24) and the proof is completed. ■

Now we are ready to show how the algebraic connectivity $\lambda_\star(\mathcal{L})$ is computed with the previous algorithm.

C. Distributed Computation of the Algebraic Connectivity

We transform the Laplacian $\mathcal{L}$ of the undirected and connected $\mathcal{G}$ into a matrix C with $\rho(C)$ depending on $\lambda_\star(\mathcal{L})$.

Proposition III.5 *Consider the following deflated version of the Perron matrix [1], [10] of the Laplacian,*

$$C = \mathbf{I} - \beta\mathcal{L} - \mathbf{1}\mathbf{1}^T/n. \tag{29}$$

The relationship between the eigenvalues of C and $\mathcal{L}$ is

$$\lambda_1(C) = 0, \quad \lambda_i(C) = 1 - \beta\lambda_i(\mathcal{L}), \text{ for } i \in \{2, \ldots, n\}. \tag{30}$$

$\rho(C)$ is associated to the algebraic connectivity $\lambda_\star(\mathcal{L})$ by

$$\lambda_\star(\mathcal{L}) = (1 - \rho(C))/\beta, \quad \text{if } 0 < \beta < 1/\lambda_n(\mathcal{L}). \tag{31}$$

Proof: The previous result can be proven as follows. Similar results appear in [1], [10]. Let $\mathcal{K}$ be

$$\mathcal{K} = \mathcal{L} + r\mathbf{1}\mathbf{1}^T/n. \tag{32}$$

First, we analyze the relationship between the eigenvalues of $\mathcal{L}$ and $\mathcal{K}$. Consider the following orthogonal matrix $V_{\mathcal{L}}$ composed of eigenvectors of $\mathcal{L}$,

$$V_{\mathcal{L}} = \left[\mathbf{1}/\sqrt{n}, \mathbf{v}_2(\mathcal{L}), \ldots, \mathbf{v}_n(\mathcal{L})\right] = \left[\mathbf{1}/\sqrt{n}, \tilde{V}_{\mathcal{L}}\right], \tag{33}$$

so that $V_{\mathcal{L}}^T \mathcal{L} V_{\mathcal{L}} = \lambda_{\mathcal{L}} = \mathrm{diag}(0, \lambda_2(\mathcal{L}), \ldots, \lambda_n(\mathcal{L}))$, which exists since $\mathcal{L}$ is symmetric. We apply this operation to $\mathcal{K}$,

$$V_{\mathcal{L}}^T \mathcal{K} V_{\mathcal{L}} = \lambda_{\mathcal{L}} + r \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} = \mathrm{diag}(r, \lambda_2(\mathcal{L}), \ldots, \lambda_n(\mathcal{L})),$$

since $\mathbf{1}^T\mathbf{1} = n$ and $\mathbf{1}^T\tilde{V}_{\mathcal{L}} = \mathbf{0}$, what yields

$$\lambda_1(\mathcal{K}) = r, \quad \lambda_i(\mathcal{K}) = \lambda_i(\mathcal{L}) \text{ for } i \in \{2, \ldots, n\}. \tag{34}$$

The eigenvalues of $\mathcal{L}$ and C in eq. (29) satisfy

$$\lambda_i(C) = 1 - \beta\lambda_i\left(\mathcal{L} + (1/\beta)\mathbf{1}\mathbf{1}^T/n\right), \text{ for } i \in \mathcal{V}, \tag{35}$$

which combined with eq. (34), with $r = 1/\beta$, gives eq. (30). Expressing $\beta = \varepsilon/\lambda_n(\mathcal{L})$ for some $\varepsilon \in (0, 1)$, then

$$\lambda_1(C) = 0, \quad \lambda_i(C) = 1 - \varepsilon\lambda_i(\mathcal{L})/\lambda_n(\mathcal{L}), \text{ for } i \in \{2, \ldots, n\}.$$

Recall that $\lambda_n(\mathcal{L}) \geq \lambda_i(\mathcal{L}) > 0$ for all $i \in \{2, \ldots, n\}$. Then,

$$1 > \lambda_2(C) \geq \cdots \geq \lambda_n(C) > \lambda_1(C) = 0, \tag{36}$$

and $\rho(C) = \lambda_2(C)$, what concludes the proof. ■

The previous deflated Perron matrix $C = \mathbf{I} - \beta\mathcal{L} - \mathbf{1}\mathbf{1}^T/n$ is not compatible with the graph and thus Algorithm III.3 cannot be immediately applied in a distributed fashion. Note however that, since $\mathbf{1}/\sqrt{n}$ is the eigenvector $\mathbf{v}_1(C)$ of C associated to the eigenvalue $\lambda_1(C) = 0$, then, for all $k \geq 1$,

$$C^k = (\mathbf{I} - \beta\mathcal{L} - \mathbf{1}\mathbf{1}^T/n)^k = (\mathbf{I} - \beta\mathcal{L})^k - \mathbf{1}\mathbf{1}^T/n, \tag{37}$$

where matrix $\mathbf{I} - \beta\mathcal{L}$ is compatible with the graph. We propose to use a variation (Algorithm III.4) of Algorithm III.3.

Before presenting Algorithm III.4, we discuss some issues regarding the number of nodes n. Note that the number of nodes n is used in the computation of β. In case the agents do not know n from the beginning, they can compute $\beta = \varepsilon/(2d_{\max})$, which satisfies $\beta < 1/\lambda_n(\mathcal{L})$ as in Proposition III.5 by executing a $\max-$consensus algorithm on the nodes degrees in an initial phase. Once β has been computed, agents can start Algorithm III.4 for computing the powers of matrix $\hat{C} = \mathbf{I} - \beta\mathcal{L}$. However, they can only execute eqs. (39)-(40) for getting the output $\hat{\lambda}_i(k)$ when they know n. At each step k agents use Proposition III.3 to find out if they have already found n and thus if they can proceed with eqs. (39)-(40).

Algorithm III.4 (Distributed Algebraic Connectivity) *Let $\varepsilon \in (0, 1)$, $\beta = \varepsilon/(2n)$. Agents execute Algorithm III.2 to compute the powers of $\hat{C} = \mathbf{I} - \beta\mathcal{L}$. Then, at each step k, each agent i has variables $\hat{C}_{ij}(k)$, for $ID(j) \in l_i(k)$, containing $[\hat{C}^k]_{ij}$. Let C being as in eq.* (29)*, then*

$$[\hat{C}^k]_{ij} = [C^k]_{ij} + 1/n. \tag{38}$$

At each step k, each node i computes

$$\hat{c}_i(k) = \sum_{ID(j) \in l_i(k)} |\hat{C}_{ij}(k) - 1/n| + (n - |l_i(k)|)/n, \tag{39}$$

and runs a $\max-$consensus to get $\max_{j\in\mathcal{V}} \hat{c}_j(k)$. The estimated algebraic connectivity at agent i, step k is

$$\hat{\lambda}_i(k) = (1 - \hat{\beta}_i^*(k))/\beta, \quad \hat{\beta}_i^*(k) = (\max_{j\in\mathcal{V}} \hat{c}_j(k))^{\frac{1}{k}}. \tag{40}$$

Theorem III.6 *Let each node i execute Algorithm III.4 with $\mathcal{G}$ connected. As $k \to \infty$, all the variables $\hat{\lambda}_i(k)$ converge to the algebraic connectivity $\lambda_*(\mathcal{L})$,*

$$\lim_{k\to\infty} \hat{\lambda}_i(k) = \lambda_*(\mathcal{L}), \quad \text{for } i \in \mathcal{V}, \tag{41}$$

and for all k we have lower- and upper-bounds for $\lambda_\star(\mathcal{L})$:

$$\hat{\lambda}_i(k) \leq \lambda_\star(\mathcal{L}) \leq (\sqrt{n})^{\frac{-1}{k}} \hat{\lambda}_i(k) + (1 - (\sqrt{n})^{\frac{-1}{k}})/\beta. \tag{42}$$

Proof: First note that $\beta = \varepsilon/(2n)$ satisfies $0 < \beta < 1/\lambda_n(\mathcal{L})$ since $\varepsilon \in (0, 1)$ and $\lambda_n(\mathcal{L}) \leq 2d_{\max} < 2n$, where $d_{\max}$ is the maximum degree in the graph. Therefore, as stated in Proposition III.5, the algebraic connectivity is $\lambda_*(\mathcal{L}) = (1-\rho(C))/\beta$, where C is the deflated Perron matrix $C = \mathbf{I} - \beta\mathcal{L} - \mathbf{1}\mathbf{1}^T/n = \hat{C} - \mathbf{1}\mathbf{1}^T/n$. From Proposition III.2, for all $i \in \mathcal{V}$, the variables $\hat{C}_{ij}(k)$ are equal to $[\hat{C}^k]_{ij}$ for $ID(j) \in l_i(k)$, whereas $[\hat{C}^k]_{ij} = 0$ for $ID(j) \notin l_i(k)$. Linking this with eqs. (37), (38) yields

$$[C^k]_{ij} = [\hat{C}^k]_{ij} - 1/n, \text{ for } ID(j) \in l_i(k),$$
$$[C^k]_{ij} = -1/n, \text{ for } ID(j) \notin l_i(k), \text{ for all } i \in \mathcal{V}, k \geq 1.$$

Thus $\hat{c}_i(k)$ in eq. (39) is the absolute $i-$th row sum of C^k, and $\hat{\beta}_i^*(k)$ in eq. (40) equals $\|C^k\|_\infty^{\frac{1}{k}}$. From eqs. (25)-(28),

$$(\sqrt{n})^{\frac{-1}{k}} \|C^k\|_\infty^{\frac{1}{k}} \leq \rho(C) \leq \|C^k\|_\infty^{\frac{1}{k}}, \tag{43}$$

since C is symmetric. Combining this with eqs. (40) and (31) we get eqs. (41) and (42) and the proof is completed. ■

IV. SIMULATIONS

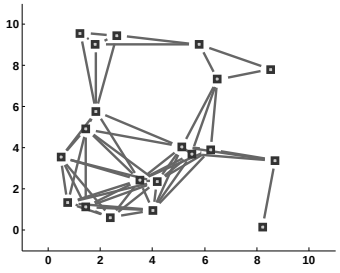

Fig. 1. 20 agents (squares) are placed randomly in a region of 10 × 10 meters and exchange data (links, lines) if they are closer than 4 meters.

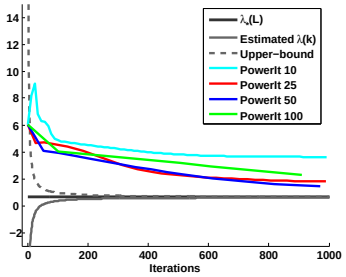

Fig. 2. Our estimated algebraic connectivity $\hat{\lambda}_i(k)$ (light gray solid) and our expression $(\sqrt{n})^{\frac{-1}{k}}\hat{\lambda}_i(k) + (1-(\sqrt{n})^{\frac{-1}{k}})/\beta$ (light gray dashed) respectively lower- and upper- bound the true algebraic connectivity $\lambda_*(\mathcal{L})$ (dark gray solid) for each step k, and converge to it. The Power iteration estimates (colored solid lines), with $T_{\text{cons}} = 10, 20, 50, 100$, exhibit a slower convergence speed, and do not converge exactly to $\lambda_*(\mathcal{L})$.

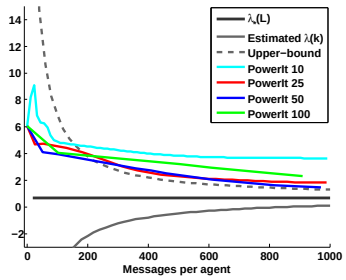

Fig. 3. For the same communication usage, the estimates produced by our algorithm (light gray dashed and solid lines) are more precise than the ones obtained with the power iteration algorithms (colored solid lines).

We have performed a set of simulations with $n = 20$ nodes as in Fig. 1. The algebraic connectivity $\hat{\lambda}_i(k)$ estimated by the agents $i \in \mathcal{V}$ at each step k using the proposed method is depicted in Fig. 2. $\hat{\lambda}_i(k)$ (light gray solid) is the same for all of them at each step k. It is a lower-bound for the true algebraic connectivity $\lambda_*(\mathcal{L})$ (dark gray solid), and asymptotically converges to $\lambda_*(\mathcal{L})$. The expression $(\sqrt{n})^{\frac{-1}{k}}\hat{\lambda}_i(k) + (1-(\sqrt{n})^{\frac{-1}{k}})/\beta$ (light gray dashed) upper-bounds $\lambda_*(\mathcal{L})$ for each step k and converges to it. We compare the performances of our method and of the distributed power iteration (Figs. 2, 3),

$$\mathbf{y}(k) = \mathbf{w}(k)/\text{normalization cons.}(\mathbf{w}(k)),$$
$$\mathbf{w}(k+1) = (\mathbf{I} - \beta\mathcal{L})\mathbf{y}(k) - \text{deflation cons.}(\mathbf{y}(k)).$$

After each power iteration step k, agents deflate $\mathbf{I} - \beta\mathcal{L}$ and normalize $\mathbf{w}(k)$ by running $T_{\text{cons}} = 10, 25, 50$, and 100 iterations of a classical discrete-time averaging rule, $\mathbf{z}(t+1) = \mathcal{W}\mathbf{z}(t)$, being $\mathcal{W}$ the Metropolis weights [17]. We display the Rayleigh quotient $\mathbf{w}(k+1)^T\mathbf{y}(k)/\mathbf{y}^T(k)\mathbf{y}(k)$, that considers simultaneously the estimates at all the agents, for the power iteration methods (Fig. 2). In all cases, our algorithm converges much faster, since we do not need to wait for the consensus process to finish between steps k. In addition, power iteration methods only converge to a neighborhood of the true $\lambda_*(\mathcal{L})$, closer to $\lambda_*(\mathcal{L})$ as T_{cons} increases. The messages sent by our agents have size n, whereas the messages of power iteration methods have constant size. However, for the same communication usage, the estimates produced by our method are more accurate than for the power iteration methods (Fig. 3).

V. CONCLUSIONS

We have presented a distributed method to compute the algebraic connectivity for networked agent systems with limited communication. At each iteration, the algorithm produces both an upper and a lower bound estimates of the algebraic connectivity, converging both to the true algebraic connectivity. As future work, we will combine our method with higher level algorithms for adaptive consensus in a parallel fashion, by taking advantage of our upper- and lower-bound estimates of the algebraic connectivity.

REFERENCES

[1] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[2] P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks," *IEEE Transactions on Automatic Control*, vol. 52, no. 9, pp. 1680 –1685, 2007.

[3] G. S. Seyboth, D. V. Dimarogonas, and K. H. Johansson, "Control of multi-agent systems via event-based communication," in *IFAC World Congress*, Milano, Italy, Aug. 2011.

[4] M. M. Zavlanos and G. J. Pappas, "Controlling connectivity of dynamic graphs," in *IEEE Conf. on Decision and Control*, Seville, Spain, Dec. 2005, pp. 6388 –6393.

[5] ——, "Distributed connectivity control of mobile networks," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1416 –1428, Dec. 2008.

[6] Z. Qu, C. Li, and F. Lewis, "Cooperative control based on distributed estimation of network connectivity," in *American Control Conference*, San Francisco, CA, USA, Jul. 2011, pp. 3441 – 2446.

[7] V. M. Preciado, M. M. Zavlanos, A. Jadbabaie, and G. J. Pappas, "Distributed control of the laplacian spectral moments of a network," in *American Control Conference*, Baltimore, Maryland, USA, Jul. 2010, pp. 4462 –4467.

[8] A. Simonetto, T. Keviczky, and R. Babuska, "On distributed maximization of algebraic connectivity in robotic networks," in *American Control Conference*, San Francisco, CA, USA, Jul. 2011, pp. 2180 – 2185.

[9] M. Franceschelli, A. Gasparri, A. Giua, and C. Seatzu, "Decentralized laplacian eigenvalues estimation for networked multi-agent systems," in *IEEE Conf. on Decision and Control*, Shanghai, P. R. China, Dec. 2009, pp. 2717 –2722.

[10] P. Yang, R. Freeman, G. Gordon, K. Lynch, S. Srinivasa, and R. Sukthankar, "Decentralized estimation and control of graph connectivity for mobile sensor networks," *Automatica*, vol. 46, no. 2, pp. 390 – 396, 2010.

[11] D. Kempe and F. McSherry, "A decentralized algorithm for spectral analysis," *Journal of Computer and System Sciences*, vol. 74, no. 1, pp. 70 – 83, 2008.

[12] M. C. D. Gennaro and A. Jadbabaie, "Decentralized control of connectivity for multi-agent systems," in *IEEE Conf. on Decision and Control*, San Diego, CA, Dec. 2006, pp. 3628 –3633.

[13] A. Householder, *The Theory of Matrices in Numerical Analysis*. New York: Dover Publications, 1964.

[14] A. Tahbaz-Salehi and A. Jadbabaie, "A one-parameter family of distributed consensus algorithms with boundary: From shortest paths to mean hitting times," in *IEEE Conf. on Decision and Control*, San Diego, CA, USA, Dec. 2006, pp. 4664–4669.

[15] R. Aragues, E. Montijano, and C. Sagues, "Consistent data association in multi-robot systems with limited communications," in *Robotics: Science and Systems*, Zaragoza, Spain, Jun. 2010.

[16] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge, UK: Cambridge University Press, 1985.

[17] L. Xiao, S. Boyd, and S. Lall, "A space-time diffusion scheme for peer-to-peer least-square estimation," in *Symposium on Information Processing of Sensor Networks (IPSN)*, Nashville, TN, Apr. 2006, pp. 168 – 176.