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 the convergence speed of the system depend on the algebraic connectivity of the graph. We present a novel distributed algorithm where the agents estimate this algebraic connectivity, obtaining a more accurate estimate at each iteration. This algorithm relies on the distributed computation of the powers of the adjacency matrix. We provide proofs of convergence and convergence rate of the algebraic connectivity estimation algorithm. In addition, we give upper and lower bounds at each iteration of the estimated algebraic connectivity. We apply this method to an event-triggered consensus scenario, where the most recent estimate of the algebraic connectivity is used for adapting the behavior of the average consensus algorithm. We show that both processes can be executed in parallel, i.e., the nodes do not need to wait for obtaining a good estimate of the algebraic connectivity before starting the averaging algorithm.

I. INTRODUCTION

Distributed consensus problems in networked multi-agent systems have received a lot of attention during the last years. They 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]–[4] are appropriate for scenarios where the state variables evolve in continuous time but where the agents may exchange data only at specific time instances. They have the advantage that they save communication costs and that they keep the control law piecewise constant. The algebraic connectivity is an important network property for all the previous systems to reach convergence and it characterizes the convergence rate. In this paper, we address the problem of computing this algebraic connectivity in a distributed fashion. We provide a successive solution where, at every iteration, each agent obtains a more accurate estimate of the algebraic connectivity. As an application, we use this algebraic connectivity estimation algorithm to derive an adaptive version of the event-triggered consensus protocol in [4], where at each iteration, the nodes adjust their behavior according to their most recent estimate of the algebraic connectivity.

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 Innovación 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.

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 [5]. There are several distributed methods that compute spanning subgraphs [6], the left Laplacian eigenvector with eigenvalue 1 for directed unbalanced graphs [7], or the first four moments (mean, variance, skewness and kurtosis) of the Laplacian eigenvalue spectrum [8]. In [9], the motion control strategy maximizes the algebraic connectivity without actually computing it. Although the previous control methods improve the network connectivity, they do not characterize any particular eigenvalue of the Laplacian as required in our case.

A method that computes and tracks the eigenvalues of the network topology is given in [10]. 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 [11]–[13]. Power iteration [14] 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 greatest absolute value) of the matrix. The original matrix can be previously deflated so that a particular eigenvalue becomes the leading one. A continuous-time version of the power iteration is proposed in [11] for computing the Fiedler eigenvector, which is the one associated to the algebraic connectivity of the graph. In [12] the orthogonal iteration method is used for simultaneously computing the k leading eigenvectors of a matrix. This algorithm can be seen as a generalization of the power iteration where, at each step, the k vectors are multiplied by the matrix and orthonormalized. The previous method is used in [13] for computing the Fiedler eigenvector. The interest of power iteration methods is that each agent only needs to maintain its own component within the estimate of the eigenvector. Then, the product of this vector by the Laplacian can be executed locally by the nodes. However, the main limitation of these approaches consists on the normalization and orthonormalization of the vectors at each step. For [12], it involves a gossip-based information aggregation algorithm [15] and for [11] a dis-

tributed averaging method [16]. 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.

In this paper we propose a method for computing the algebraic connectivity where, at each step k , the agents compute the k -th power of a deflated Laplacian in a distributed fashion. The computation of the powers of the adjacency matrix was first discussed in [17]. Whenever the nodes want to obtain an estimate of the algebraic connectivity, they run a max-consensus method [18]. 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. An additional benefit of our method is that it automatically provides the total number of nodes n . We provide proofs of convergence of the algebraic connectivity computation algorithm, we characterize its convergence speed, and give upper and lower bounds for the true algebraic connectivity at each iteration. Then, we combine the previous ideas with [4] and present an adaptive triggered consensus algorithm where the most recent estimate of the algebraic connectivity is used at each step. We show that both processes can be executed simultaneously due to the upper and lower bounds of the true algebraic connectivity.

This paper is organized as follows. Section II states the problem. Section III presents the distributed algorithm for computing the algebraic connectivity. Section IV applies the previous algorithm to the triggered average consensus scenario. Finally, Section V shows the performance of the method in a simulated scenario.

II. PRELIMINARIES

We use the notation defined in Table I.

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 .		
$\text{diag}(b_1, \dots, b_r)$	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	$\text{diag}(\lambda_1(A), \dots, \lambda_r(A))$.		
V_A	$[\mathbf{v}_1(A), \dots, \mathbf{v}_r(A)]$.		
$\ A\ _\infty$	Induced ∞ -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, $\mathcal{L} = \text{diag}(\mathbf{A}\mathbf{1}) - \mathcal{A}$.		
$\lambda_*(\mathcal{L})$	Algebraic connectivity, the second-smallest $\lambda_i(\mathcal{L})$.		

Consider a set of $n \in \mathbb{N}$ agents with indices $i \in$

$\{1, \dots, n\}$. The agents can exchange information with nearby nodes. This information can be represented by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \dots, n\}$ 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 *compatible* with $\mathcal{G}$ if $C_{ij} = 0$ iff $(i, j) \notin \mathcal{E}$ for $j \neq i$; note that 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 defined by

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

We assume that the undirected communication graph $\mathcal{G}$ 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 the positive-semidefinite matrix

$$\mathcal{L} = \text{diag}(\mathcal{A}\mathbf{1}) - \mathcal{A}. \quad (2)$$

Note that both $\mathcal{A}$ and $\mathcal{L}$ are compatible with the graph. In this paper we sort the eigenvalues of $\mathcal{L}$ as follows,

$$\lambda_1(\mathcal{L}) \leq \lambda_2(\mathcal{L}) \leq \dots \leq \lambda_n(\mathcal{L}). \quad (3)$$

The Laplacian matrix $\mathcal{L}$ has the following well known properties, see e.g., [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,

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

- (iii) Its eigenvalues are upper-bounded by $\lambda_n(\mathcal{L}) \leq 2d_{\max}$.
- The *algebraic connectivity* of $\mathcal{G}$ denoted by $\lambda_*(\mathcal{L})$ 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

$$\dot{x}_i(t) = u_i(t), \text{ with } x_i(t), u_i(t) \in \mathbb{R}, \quad (4)$$

where u_i denotes the control input at agent i given by

$$u_i(t) = - \sum_{j \in \mathcal{N}_i} (x_i(t) - x_j(t)). \quad (5)$$

With stack vectors $x = [x_1, \dots, x_n]^T$, $u = [u_1, \dots, u_n]$,

$$\dot{x}(t) = -\mathcal{L}x(t), \quad x(0) = x_0. \quad (6)$$

A well known result [19] is that if $\mathcal{G}$ is undirected and connected, then the previous algorithm globally asymptotically solves the average consensus problem, i.e.,

$$\lim_{t \rightarrow \infty} x_i(t) = \sum_{i=1}^n x_i(0)/n. \quad (7)$$

However in general it is necessary to implement the continuous-time law on a digital platform. This has motivated the research on event-based and time-scheduled control. It is

of special interest the triggered-based control method in [4], where each agent monitors its own state $x_i(t)$ continuously. However, the agents do not have continuous communication but instead propagate their most recent states at some time instances. Also the control law is piecewise constant. The following rule given in [4, Th. 4] allows each agent i to locally determine when to trigger and transmit its new state. Let $e_i(t)$ be the measurement error at node i containing the difference between its actual current state $x_i(t)$ at time t and the last transmitted one $\hat{x}_i(t)$,

$$e_i(t) = \hat{x}_i(t) - x_i(t), \quad \text{for } i \in \mathcal{V}, \quad (8)$$

with stack vector $e(t) = [e_1(t), \dots, e_n(t)]^T$. Let c_1, α be constants satisfying

$$c_1 > 0 \quad \text{and} \quad 0 < \alpha < \lambda_*(\mathcal{L}). \quad (9)$$

The authors define the time-dependent trigger function

$$f_i(t, e_i(t)) = |e_i(t)| - c_1 e^{-\alpha t}, \quad (10)$$

where an event for agent i is triggered as soon as $f_i(t, e_i(t)) > 0$ resulting in agent i sending its most recent state $\hat{x}_i$ to its neighbors. Whenever an event is triggered, i.e., node i transmitting its new state $x_i(t)$ or receiving an updated state $x_j(t)$ from one of its neighbors, it (i) updates its control-law immediately, and (ii) reevaluates the trigger function and computes the next triggering time. As stated by [4, Th. 4], for connected graphs if nodes execute the previous procedure, then for all initial conditions x_0 the overall system converges to average consensus asymptotically. Furthermore the closed-loop system does not exhibit Zeno-behavior.

Note however that condition (9) requires all the nodes to know $\lambda_*(\mathcal{L})$ or a lower bound of it.

B. Problem description

Our goal is to design distributed algorithms to allow the agents to compute $\lambda_*(\mathcal{L})$ (or a lower bound of) in a distributed fashion. Then, we modify the trigger function so that instead of using a fixed α , nodes adapt this value depending on the most recent and accurate estimate of the algebraic connectivity $\lambda_*(\mathcal{L})$.

III. DISTRIBUTED COMPUTATION OF THE ALGEBRAIC CONNECTIVITY

First of all, we present a novel algorithm for computing the algebraic connectivity $\lambda_*(\mathcal{L})$ of the graph in a distributed fashion. This algorithm relies on the observation that (i) 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 (ii) $\|C^k\|_\infty^{\frac{1}{k}}$ successively approaches the spectral radius $\rho(C)$ of matrix C . This allows us to propose a method for estimating the algebraic connectivity that is suited for distributed implementation.

We start this section presenting some results about the computation of the powers of matrices which are *compatible* with the graph.

A. Distributed computation of power of matrices

We show that if matrix C is *compatible* with the graph structure, the computation of its powers can be done in a distributed fashion. Our discussion refers to fixed undirected communication graphs, although the method can be easily extended to time-varying graphs. The presented algorithm was originally proposed in [17] 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

$$\hat{C}_{ii}(0) = 1, \quad \text{and} \quad \hat{C}_{ij}(0) = 0 \quad \text{for } j \in \mathcal{V} \setminus \{i\}. \quad (11)$$

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

$$\hat{C}_{ij}(k+1) = \sum_{j' \in \mathcal{N}_i \cup \{i\}} C_{ij'} \hat{C}_{j'j}(k), \quad \text{for } j \in \mathcal{V}. \quad (12)$$

Note that this algorithm is distributed and each node i updates its variable using exclusively information from its neighbors $j \in \mathcal{N}_i$ and from itself. It requires each node i to store n scalars and to exchange n scalars at each iteration k . We discuss now the outcomes of this algorithm.

Proposition III.1 *When C is compatible with the graph, the outcomes of algorithm (12) 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. (11) gives the identity matrix $\mathbf{I}$ which is exactly C^0 . For $k \geq 1$, we consider the explicit expression for $C^{k+1} = CC^k$, where each (i, j) entry is given by

$$[C^{k+1}]_{ij} = \sum_{j'=1}^n C_{ij'} [C^k]_{j'j}. \quad (13)$$

Since C is compatible with the graph, then $C_{ij'} = 0$ for $j' \notin \mathcal{N}_i \cup \{i\}$ and the previous expression gives

$$[C^{k+1}]_{ij} = \sum_{j' \in \mathcal{N}_i \cup \{i\}} C_{ij'} [C^k]_{j'j}, \quad (14)$$

which is exactly what algorithm (12) does. ■

Observe that the previous algorithm does not compute an estimate of the centralized one. Instead, it exactly computes the same solution as a centralized system, but distributing the operations over the nodes. Observe that it remains valid if the communication graph is time-varying, in which case each agent i computes

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

The main limitation of the previous procedure is that it assumes that in the initial phase (eq. (11)) each agent knows the total amount of other agents in the network n , and

that at each step k (eq. (12)) 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) *Let each node $i \in \mathcal{V}$ maintain a set $l_i(k)$ with the identifiers $ID(j)$ of the nodes j it has discovered, and an estimate $\tilde{C}_{ij}(k)$ of the (i, j) entries of the k -th power of C , $[C^k]_{ij}$, associated to these nodes j , $ID(j) \in l_i(k)$.*

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

$$\tilde{C}_{ii}(0) = 1, \quad l_i(0) = \{ID(i)\}, \quad (15)$$

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

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,

$$l_i(k+1) = \bigcup_{j' \in \mathcal{N}_i \cup \{i\}} l_{j'}(k). \quad (16)$$

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 ,

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

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

$$\tilde{C}_{ij}(k+1) = \sum_{j' \in \mathcal{N}_i \cup \{i\}, ID(j) \in l_{j'}(k)} C_{ij'} \tilde{C}_{j'j}(k), \quad (17)$$

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 (17) are exactly the outcomes of (12). 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, with initialization

$$l_i(0) = \{ID(i)\}, \quad \hat{C}_{ii}(0) = 1, \quad \hat{C}_{ij}(0) = 0, \text{ for } j \neq i, \quad (18)$$

and update rule

$$l_i(k+1) = \bigcup_{j' \in \mathcal{N}_i \cup \{i\}} \{l_{j'}(k)\}, \text{ and,} \quad (19)$$

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

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. (18). 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. (19) 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 (19) gives

$$\hat{C}_{ij}(k+1) = \sum_{j' \in \mathcal{N}_i \cup \{i\}} C_{ij'} \hat{C}_{j'j} = 0. \quad (20)$$

Now we prove 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. This can be shown by induction by taking into account the following issue. We pay attention to eq. (17) for $ID(j) \in l_i(k+1)$,

$$\tilde{C}_{ij}(k+1) = \sum_{j' \in \mathcal{N}_i \cup \{i\}, ID(j) \in l_{j'}(k)} C_{ij'} \tilde{C}_{j'j}(k). \quad (21)$$

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

$$\begin{aligned} \tilde{C}_{ij}(k+1) &= \sum_{j' \in \mathcal{N}_i \cup \{i\}, ID(j) \in l_{j'}(k)} C_{ij'} \hat{C}_{j'j}(k) \\ &+ \sum_{j' \in \mathcal{N}_i \cup \{i\}, ID(j) \notin l_{j'}(k)} C_{ij'} 0, \end{aligned} \quad (22)$$

which is exactly the update rule for $\hat{C}_{ij}(k+1)$. $\blacksquare$

Therefore Algorithm III.2 provides each agent i with all the elements $[C^k]_{ij}$ for $j \in \mathcal{V}$ of the i -th row of the power matrix C^k . It is fully distributed 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 $l_i(k) = l_i(k-1)$,*

$$k_i = \min\{k \mid l_i(k) = l_i(k-1)\}. \quad (23)$$

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 $n = |l_i(k_i)|$. $\blacksquare$

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 $\|\cdot\|$, [20, Chap. 5.6]

$$\rho(C) \leq \|C\|, \text{ and } \rho(C) = \lim_{k \rightarrow \infty} \|C^k\|^{\frac{1}{k}}. \quad (24)$$

We propose to use the induced ∞ -norm $\|\cdot\|_\infty$, which is the maximum absolute row sum of the matrix,

$$\|C^k\|_\infty = \max_{i \in \mathcal{V}} \sum_{j=1}^n |[C^k]_{ij}|, \quad (25)$$

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. At each iteration k we define for each node the sum $c_i(k)$ of the absolute values of its variables $\tilde{C}_{ij}(k)$,

$$c_i(k) = \sum_{ID(j) \in l_i(k)} |\tilde{C}_{ij}(k)|. \quad (26)$$

Nodes execute the following *max*-consensus [18] on their variables $c_i(k)$,

$$\beta_i(k) = c_i(k), \quad \beta_i(k + \tau + 1) = \max_{j \in \mathcal{N}_i \cup \{i\}} \beta_j(k + \tau), \quad (27)$$

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

$$\beta_1(k + T) = \dots = \beta_n(k + T) = \max_{i \in \mathcal{V}} c_i(k). \quad (28)$$

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

$$\beta_i^*(k) = (\beta_i(k + T))^{\frac{1}{k}} = (\max_{j \in \mathcal{V}} c_j(k))^{\frac{1}{k}}. \quad (29)$$

Observe that this computation of $c_i(k)$ in eq. (26) 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 (27) 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 $k \rightarrow \infty$ all the variables $\beta_i^*(k)$ converge to the spectral radius $\rho(C)$ of matrix C ,

$$\lim_{k \rightarrow \infty} \beta_i^*(k) = \rho(C), \text{ for all } i \in \mathcal{V}, \quad (30)$$

and for all $k \geq 1$,

$$(\sqrt[n]{n})^{-\frac{1}{k}} \beta_i^*(k) \leq \rho(C) \leq \beta_i^*(k). \quad (31)$$

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. (26) is exactly the absolute sum of the i -th row of C^k . The *max*-consensus (27) provides each agent with the induced infinite norm of C^k . Therefore, $\beta_i^*(k)$ in eq. (29) equals $\beta_i^*(k) = (\|C^k\|_\infty)^{\frac{1}{k}}$, which combined with eq. (24) gives

$$\rho(C) = \lim_{k \rightarrow \infty} \|C^k\|_\infty^{\frac{1}{k}} = \lim_{k \rightarrow \infty} \beta_i^*(k), \quad (32)$$

as stated in eq. (30).

Now we focus on the inequalities in eq. (31). From (24),

$$\rho(C) = (\rho(C^k))^{\frac{1}{k}} \leq \|C^k\|_\infty^{\frac{1}{k}} = \beta_i^*(k), \quad (33)$$

which gives the right part in eq. (31). Since matrix C is symmetric, then its spectral norm $\|C\|_2 = \max_i \sqrt{\lambda_i(C^2)}$ equals its spectral radius $\rho(C) = \max_i |\lambda_i(C)|$,

$$\rho(C) = \|C\|_2 = \|C^k\|_2^{\frac{1}{k}}. \quad (34)$$

The spectral $\|C^k\|_2$ and induced infinite $\|C^k\|_\infty$ norms of a matrix C^k are related by $(\sqrt{n})^{-1} \|C^k\|_\infty \leq \|C^k\|_2$, [20, Chap. 5.6], which combined with eq. (34) gives

$$(\sqrt{n})^{-\frac{1}{k}} \beta_i^*(k) = (\sqrt{n})^{-\frac{1}{k}} \|C^k\|_\infty^{\frac{1}{k}} \leq \|C^k\|_2^{\frac{1}{k}} = \rho(C), \quad (35)$$

as stated in eq. (31) and the proof is completed. ■

Observe that in the previous process the nodes are not required to know n . Now we are ready to show how the algebraic connectivity of the graph $\lambda_*(\mathcal{L})$ can be computed by using the previous algorithm.

C. Distributed Computation of the Algebraic Connectivity

The method roughly consists of transforming the Laplacian $\mathcal{L}$ matrix associated to the undirected and connected graph $\mathcal{G}$ into a new matrix C where the algebraic connectivity $\lambda_*(\mathcal{L})$ becomes the leading eigenvalue, i.e., the spectral radius $\rho(C)$ is an expression that depends on $\lambda_*(\mathcal{L})$. Then, the procedure presented in the previous sections is used for computing $\rho(C)$ in a distributed fashion. The estimated algebraic connectivity is finally obtained by reversing the transformation.

Proposition III.5 Consider the following deflated version of the Perron matrix [1], [11], [21] of the Laplacian,

$$C = \mathbf{I} - \beta \mathcal{L} - \mathbf{1}\mathbf{1}^T/n. \quad (36)$$

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

$$\lambda_1(C) = 0, \quad \lambda_i(C) = 1 - \beta \lambda_i(\mathcal{L}), \text{ for } i \in \{2, \dots, n\}. \quad (37)$$

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

$$\lambda_*(\mathcal{L}) = (1 - \rho(C))/\beta, \quad \text{if } 0 < \beta < 1/\lambda_n(\mathcal{L}). \quad (38)$$

The previous result can be easily proven as follows. Note that similar results can be found at [1], [11], [21].

Proof: First of all, we show that $\mathcal{L} + r\mathbf{1}\mathbf{1}^T/n$ has the same eigenvalues as $\mathcal{L}$ for $i \in \{2, \dots, n\}$, but the first one that is equal to r ,

$$\begin{aligned} \lambda_1(\mathcal{L} + r\mathbf{1}\mathbf{1}^T/n) &= r, \text{ and,} \\ \lambda_i(\mathcal{L} + r\mathbf{1}\mathbf{1}^T/n) &= \lambda_i(\mathcal{L}) \text{ for } i \in \{2, \dots, n\}. \end{aligned} \quad (39)$$

Consider the following orthogonal matrix $V_{\mathcal{L}}$ composed of eigenvectors of $\mathcal{L}$,

$$V_{\mathcal{L}} = [\mathbf{1}/\sqrt{n}, \mathbf{v}_2(\mathcal{L}), \dots, \mathbf{v}_n(\mathcal{L})] = [\mathbf{1}/\sqrt{n}, \tilde{V}_{\mathcal{L}}], \quad (40)$$

which exists since $\mathcal{L}$ is symmetric. Then,

$$V_{\mathcal{L}}^T \mathcal{L} V_{\mathcal{L}} = \lambda_{\mathcal{L}} = \text{diag}(0, \lambda_2(\mathcal{L}), \dots, \lambda_n(\mathcal{L})), \quad (41)$$

We apply the same operation to $\mathcal{L} + r\mathbf{1}\mathbf{1}^T/n$ and get

$$\begin{aligned} V_{\mathcal{L}}^T (\mathcal{L} + r\mathbf{1}\mathbf{1}^T/n) V_{\mathcal{L}} &= \lambda_{\mathcal{L}} + r \begin{bmatrix} \mathbf{1}^T \mathbf{1}/n & \mathbf{1}^T \tilde{V}_{\mathcal{L}}/\sqrt{n} \\ \tilde{V}_{\mathcal{L}}^T \mathbf{1}/\sqrt{n} & \tilde{V}_{\mathcal{L}}^T \mathbf{1}\mathbf{1}^T \tilde{V}_{\mathcal{L}} \end{bmatrix} \\ &= \lambda_{\mathcal{L}} + r \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} = \text{diag}(r, \lambda_2(\mathcal{L}), \dots, \lambda_n(\mathcal{L})), \end{aligned} \quad (42)$$

since $\mathbf{1}^T \mathbf{1} = n$ and $\mathbf{1}^T \tilde{V}_{\mathcal{L}} = \mathbf{0}$, what yields eq. (39). Now note that the eigenvalues of the Laplacian $\mathcal{L}$ and of the deflated Perron matrix C in eq. (36) satisfy

$$\lambda_i(C) = 1 - \beta \lambda_i(\mathcal{L} + (1/\beta)\mathbf{1}\mathbf{1}^T/n), \text{ for } i \in \mathcal{V}, \quad (43)$$

which combined with eq. (39), with $r = 1/\beta$, gives the relationship in eq. (37).

Now let us define β as follows

$$\beta = \varepsilon/\lambda_n(\mathcal{L}), \text{ for some } \varepsilon \in (0, 1), \quad (44)$$

and express the eigenvalues of C accordingly,

$$\begin{aligned} \lambda_1(C) &= 0, \text{ and for } i \in \{2, \dots, n\}, \\ \lambda_i(C) &= 1 - \varepsilon \lambda_i(\mathcal{L})/\lambda_n(\mathcal{L}). \end{aligned} \quad (45)$$

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

$$1 > \lambda_2(C) \geq \dots \geq \lambda_n(C) > \lambda_1(C) = 0, \quad (46)$$

and the eigenvalue of C with largest absolute value is $\lambda_2(C)$, 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 $k \geq 1$,

$$C^k = (\mathbf{I} - \beta\mathcal{L} - \mathbf{1}\mathbf{1}^T/n)^k = (\mathbf{I} - \beta\mathcal{L})^k - \mathbf{1}\mathbf{1}^T/n, \quad (47)$$

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. At each step k of

Algorithm III.4, agents can always execute Algorithm III.2 for computing the powers of matrix $\hat{C} = \mathbf{I} - \beta\mathcal{L}$. However, they can only execute eqs. (49)-(51) 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. (49)-(51). Alternatively, n can be computed in an initial phase as, for instance, in [22].

Algorithm III.4 (Distributed Algebraic Connectivity)

Let $\varepsilon \in (0, 1)$ be known by all the agents and $\beta = \varepsilon/(2n)$. Consider the agents execute Algorithm III.2 for computing the powers of matrix $\hat{C} = \mathbf{I} - \beta\mathcal{L}$. At each step $k \geq 1$, each node $i \in \mathcal{V}$ has variables $\hat{C}_{ij}(k)$ with the (i, j) entries of the k -th power of matrix $\hat{C}$, $[\hat{C}^k]_{ij}$, for all the nodes it has discovered so far, $ID(j) \in l_i(k)$. Note that

$$[\hat{C}^k]_{ij} = [C^k]_{ij} + 1/n, \quad (48)$$

with C being the deflated Perron matrix in eq. (36).

At each step k , each node i computes

$$\hat{c}_i(k) = \sum_{ID(j) \in l_i(k)} |\hat{C}_{ij}(k) - 1/n| + (n - |l_i(k)|)/n, \quad (49)$$

it executes a max-consensus algorithm to obtain $\max_{j \in \mathcal{V}} \hat{c}_j(k)$ and lets $\hat{\beta}_i^*(k)$ be

$$\hat{\beta}_i^*(k) = (\max_{j \in \mathcal{V}} \hat{c}_j(k))^{1/k}. \quad (50)$$

The algebraic connectivity estimated by each agent $i \in \mathcal{V}$ at step $k \geq 1$ is given by

$$\hat{\lambda}_i(k) = (1 - \hat{\beta}_i^*(k)) / \beta. \quad (51)$$

Theorem III.6 Consider each node i executes Algorithm III.4 for a connected graph. Then, as $k \rightarrow \infty$ all the variables $\hat{\lambda}_i(k)$ asymptotically converge to the algebraic connectivity $\lambda_*(\mathcal{L})$,

$$\lim_{k \rightarrow \infty} \hat{\lambda}_i(k) = \lambda_*(\mathcal{L}), \text{ for } i \in \mathcal{V}, \quad (52)$$

and for each step $k \geq 1$ we have the following lower- and upper-bounds for $\lambda_*(\mathcal{L})$:

$$\hat{\lambda}_i(k) \leq \lambda_*(\mathcal{L}) \leq (\sqrt[n]{n})^{1/k} \hat{\lambda}_i(k) + (1 - (\sqrt[n]{n})^{1/k})/\beta. \quad (53)$$

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. (47), (48) yields

$$\begin{aligned} [C^k]_{ij} &= [\hat{C}^k]_{ij} - 1/n, \text{ for } ID(j) \in l_i(k), \\ [C^k]_{ij} &= -1/n, \text{ for } ID(j) \notin l_i(k), \end{aligned} \quad (54)$$

for all $i \in \mathcal{V}$, $k \geq 1$. Therefore $\hat{c}_i(k)$ in eq. (49) is the absolute row sum of the i -th row of C^k , and $\hat{\beta}_i^*(k)$ in eq. (50) is $\hat{\beta}_i^*(k) = \|C^k\|_\infty^{\frac{1}{k}}$. As stated from eqs. (32)-(35),

$$(\sqrt{n})^{\frac{-1}{k}} \|C^k\|_\infty^{\frac{1}{k}} \leq \rho(C) \leq \|C^k\|_\infty^{\frac{1}{k}}, \quad (55)$$

since C is symmetric. Combining this with eqs. (51) and (38) we get eqs. (52) and (53) and the proof is completed. ■

IV. DISTRIBUTED ADAPTIVE TRIGGERED AVERAGE CONSENSUS

In this section we apply the Distributed Algebraic Connectivity algorithm previously presented to an event-triggered consensus scenario.

The adaptive triggered average consensus algorithm is as the one presented in Section II,

$$\begin{aligned} \dot{\mathbf{x}}(t) &= -\mathcal{L}\hat{\mathbf{x}}(t) = \mathbf{u}(t), \\ e_i(t) &= \hat{x}_i(t) - x_i(t), \end{aligned} \quad (56)$$

except for the trigger function, which now is

$$|e_i(t)| \leq c_1 e^{-\alpha(t)t}, \quad (57)$$

where now α is not fixed but is adapted depending on the previously estimated algebraic connectivity as follows:

$$\begin{aligned} \hat{\alpha}(k) &= \hat{\lambda}(k), \text{ for } k \in \mathbb{N}, \\ \alpha(t) &= \gamma \hat{\alpha}(k), \text{ for } t \in [k, k+1), \end{aligned} \quad (58)$$

where $0 < \gamma < 1$ is a constant and $\hat{\lambda}(k)$ is the outcome of the algorithm in the previous section, satisfying

$$\hat{\alpha}(k) = \hat{\lambda}(k) \leq \lambda_*(\mathcal{L}). \quad (59)$$

Theorem IV.1 *Algorithm (56)-(58) asymptotically converges to the average of the initial states and does not exhibit Zeno-behavior.*

Proof: The result holds based on similar analysis as in [4]. ■

V. SIMULATIONS

In order to show the performance of the algorithm, we have performed a set of simulations with $n = 20$ nodes randomly placed 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) is an upper-bound for $\lambda_*(\mathcal{L})$ for each step k and it asymptotically converges to $\lambda_*(\mathcal{L})$.

We have studied the performance of our algorithm compared to the distributed power iteration algorithm for the matrix $\mathbf{I} - \beta\mathcal{L} - \mathbf{1}\mathbf{1}^T/n$ (Figs. 3, 4),

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

Fig. 1. 20 agents (black squares) are placed randomly in a region of 10×10 meters. There is an edge $e = (i, j) \in \mathcal{E}$ (gray lines) between pairs of agents that are closer than 4 meters.

Fig. 2. The algebraic connectivity $\hat{\lambda}_i(k)$ estimated by the agents $i \in \mathcal{V}$ (light gray solid) using the proposed method 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) is an upper-bound for $\lambda_*(\mathcal{L})$ for each step k and it asymptotically converges to $\lambda_*(\mathcal{L})$.

After each power iteration step, agents execute $T_{\text{cons}} = 10, 25, 50$, and 100 consensus iterations for deflating $\mathbf{I} - \beta\mathcal{L}$ and normalizing $\mathbf{w}(k)$. The consensus iterations are computed with a classical discrete-time averaging rule, $\mathbf{z}(t+1) = \mathcal{W}\mathbf{z}(t)$, and using the Metropolis weight matrix $\mathcal{W}$ [23].

In Fig. 3 we show the estimates produced by our algorithm, $\hat{\lambda}_i(k)$ (light gray solid), which are the lower-bound, and the upper-bound $(\sqrt{n})^{\frac{-1}{k}} \hat{\lambda}_i(k) + (1 - (\sqrt{n})^{\frac{-1}{k}})/\beta$ (light gray dashed). Recall that in our algorithm, these estimates are equal for all the agents $i \in \mathcal{V}$. For the power iteration estimates, we are displaying 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. The estimates of our algorithm converge much faster than the ones produced by the power iteration algorithms, due to the fact that we do not need to wait for a consensus process to finish before starting a new step. In addition, our estimates asymptotically converge to the true algebraic connectivity, whereas the power iteration methods only converge to a neighborhood which is further from the true $\lambda_*(\mathcal{L})$ as the number of consensus iterations T_{cons} for normalizing and deflating decreases.

A benefit of power iteration methods compared to our proposal is that, at each iteration, the agents send constant size messages, whereas in our case the messages have size n . We have made an analysis of the evolution of the estimates of the algorithms versus the total size of messages sent per

Fig. 3. The estimates obtained with the proposed method (light gray dashed and solid lines) converge very fast to $\lambda_*(\mathcal{L})$ (dark gray solid). The estimates produced with the Power iteration algorithm (colored solid lines), with $T_{\text{cons}} = 10, 20, 50$ and 100 consensus iterations for normalizing and deflating, need much more iterations to converge. Besides, they do not converge exactly to $\lambda_*(\mathcal{L})$, but to a value which is closer to $\lambda_*(\mathcal{L})$ as the number of consensus iterations T_{cons} increases.

agent (Fig. 4). As it can be observed, also in this case the estimates produced by our method are more accurate than for the power iteration methods.

Fig. 4. With the proposed algorithm, each agent i sends messages of size n per iteration, whereas for power iteration methods, messages have constant size. However, also in this case our method behaves better than the power iteration approaches. 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).

VI. 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. We have proved theoretically and experimentally that both estimates asymptotically converge to the true algebraic connectivity. We have shown that our method outperforms the classical distributed power iteration, providing more accurate estimates, and improving the network communication usage. The ability to give upper and lower bounds of the algebraic connectivity has been demonstrated to have a great importance for combining this method with higher level algorithms for adaptive consensus in a parallel fashion, i.e., where both processes are executed simultaneously. We have combined this algorithm with an event-triggered adaptive consensus scenario where, at each iteration, the most recent estimate of the algebraic connectivity is used by the consensus method.

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] D. V. Dimarogonas and K. H. Johansson, "Event-triggered control for multi-agent systems," in *IEEE Conf. on Decision and Control*, Shanghai, China, Dec. 2009, pp. 7131–7136.
- [4] 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.
- [5] 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.
- [6] —, "Distributed connectivity control of mobile networks," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1416–1428, Dec. 2008.
- [7] 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.
- [8] 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.
- [9] 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.
- [10] 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.
- [11] P. Yang, R. Freeman, G. Gordon, K. Lynch, S. Srinivasa, and R. Suktankar, "Decentralized estimation and control of graph connectivity for mobile sensor networks," *Automatica*, vol. 46, no. 2, pp. 390–396, 2010.
- [12] 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.
- [13] 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.
- [14] A. Householder, *The Theory of Matrices in Numerical Analysis*. New York: Dover Publications, 1964.
- [15] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *IEEE Symposium on Foundations of Computer Science*, Cambridge, MA, USA, Oct. 2003, pp. 482–491.
- [16] R. A. Freeman, P. Yang, and K. M. Lynch, "Stability and convergence properties of dynamic average consensus estimators," in *IEEE Conf. on Decision and Control*, San Diego, CA, 2006, pp. 398–403.
- [17] R. Aragues, E. Montijano, and C. Sagües, "Consistent data association in multi-robot systems with limited communications," in *Robotics: Science and Systems*, Zaragoza, Spain, Jun. 2010.
- [18] 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.
- [19] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.
- [20] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge, UK: Cambridge University Press, 1985.
- [21] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, pp. 65–78, 2004.
- [22] E. Montijano, S. Martínez, and C. Sagües, "De-RANSAC: Robust distributed consensus in sensor networks," *IEEE Transactions on Systems, Man, and Cybernetics: part B*, submitted, May 2010.
- [23] 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.