Distributed Finite-Time Computation of Digraph Parameters: Left-Eigenvector, Out-Degree and Spectrum

Themistoklis Charalambous, *Member, IEEE*, Michael G. Rabbat, *Member, IEEE,*
Mikael Johansson, *Member, IEEE*, and Christoforos N. Hadjicostis, *Senior Member, IEEE*

***Abstract*—Many of the algorithms that have been proposed in the field of distributed computation rely on assumptions that require nodes to be aware of some global parameters. In this work, we propose algorithms to compute some network parameters in a distributed fashion and in a finite number of steps. More specifically, given an arbitrary strongly connected network of interconnected nodes, by adapting a distributed finite-time approach, we develop distributed strategies that enable nodes to compute the following network parameters: the left-eigenvector, the out-degree and the spectrum of weighted adjacency matrices.**

***Index Terms*—network parameter computation, distributed finite-time algorithms, left-eigenvector, out-degree, spectrum.**

I. Introduction

Multi-agent coordination has received a lot of attention in the last two decades due to its numerous applications (see, e.g., [1]–[3] and references therein). A multi-agent system consists of a set of agents (nodes) that can share information via connection links (edges), forming a directed communication topology (a directed graph or digraph). Most of the literature assumes bidirectional links and models communication among agents as an undirected graph. However, communication (wireless communication in particular) is non-isotropic and the communication range of each agent is not the same; hence, communication links are directed in general (see, for example, [4]) and results relying on the undirected graph assumption are not always valid.

Numerous algorithms proposed in the literature claim to be distributed. However, most of them rely on the assumption that a node is aware of some network parameters, which may not be justified from the distributed nature of the algorithms. To alleviate this, there have been numerous attempts to estimate network parameters in a distributed fashion. In this work, we adapt a distributed *finite-time* approach, first developed by Yuan *et al.* [5], [6] for finite-time average consensus in undirected graphs, to compute network parameters of directed graphs in a finite number of steps; namely, the left-eigenvector, the out-degree and the spectrum of weighted adjacency matrices. Finite-time algorithms are, in general, more desirable; besides the fact that they converge in finite-time with the exact computation of parameters (thus allowing the results to be used as sub-routines in more involved algorithms), it has also been reported that closed-loop systems under finite-time control usually demonstrate better disturbance rejection properties [7].

Themistoklis Charalambous and Mikael Johansson are with the ACCESS Linnaeus Center, School of Electrical Engineering, KTH-Royal Institute of Technology, SE-100 44 Stockholm, Sweden. Emails: {themisc, mikaelj}@kth.se.

Michael G. Rabbat is with the Department of Electrical and Computer Engineering, McGill University, Montréal, Québec, Canada. Email: michael.rabbat@mcgill.ca. This work has been carried out while Prof. M. G. Rabbat was visiting KTH.

Christoforos N. Hadjicostis is with the Department of Electrical and Computer Engineering, University of Cyprus, Nicosia, Cyprus. E-mail: chadjic@ucy.ac.cy.

A. Motivation

Left-eigenvector computation. In distributed coordination it is often the case that some parameters of the network are required for computing something else (e.g., weight balancing [8] and average consensus [9] require knowledge of the left-eigenvector corresponding to the zero eigenvalue of the Laplacian of a graph). However, many works assume prior knowledge of the left-eigenvector, or, they use asymptotic algorithms to obtain a possibly inaccurate estimate of it after a pre-specified number of iterations. On the contrary, using the finite-time approach proposed in this paper, it is possible to compute the left eigenvector in a finite number of steps and in a distributed manner (i.e., each agent locally accumulates successive state values which are then used to compute the asymptotic final value of the network without any prior knowledge of the system dynamics), so that it can be subsequently used by nodes for other computations, such as for computing their out-degree in a distributed manner.

Out-degree computation. In systems where the communication graph is directed there are many limitations on what can be computed using only local information. In order to overcome these limitations, most previous work assumes that each agent knows the number of agents receiving their messages (i.e., their out-degree) or even the identities of their out-neighbors. For example, a gossip algorithm, called *push-sum* (see [10], [11]), was proposed to compute the average in directed graphs based on the assumption that each node is aware of its out-neighbors, and it assigns nonnegative weights on the links to its out-neighbors such that their sum is unity. Along the same lines, *ratio consensus* [12]–[14] was proposed as a consensus algorithm based on multi-casting (or broadcasting) where each node is required to know its out-degree and assign weights to outgoing links which sum to unity, in order to compute the average in directed graphs. Many algorithms have also been proposed for weight balancing (see, for example, [15], [16]) and for forming bistochastic (doubly stochastic) matrices (e.g., [17], [18]). All of these approaches rely on the assumption that the out-degree of each agent is known to the agent (and hence nodes can adjust the weights on the outgoing edges), or even require each agent to be able

to address individual messages to each of its out-neighbors. Hence, a mechanism for the computation of the out-degree or even the identification of the out-neighborhood is required.

Spectrum computation. The importance of the information conveyed in the eigenvalues of the adjacency and the Laplacian matrices has been the centerpiece of many works in the past (see, for example, [19] and references therein). It is both theoretically interesting and practically useful if the spectrum can be determined explicitly, based on local information in directed or undirected graphs. An application of the spectrum computation considered in this paper is the design of self-configuring protocols for finite-time consensus, i.e., protocols that achieve average consensus in the fastest possible time for all nodes (see, for example, [20]). The existence of such a protocol is very useful in static networks, since there are many instances where the nodes need to compute some parameters quickly and in finite-time without requiring memory and heavy computations (that drain the battery).

B. Related work

Left-eigenvector computation. There has been some work on computing the left-eigenvector, but mostly in undirected graphs (see, e.g., [21]). For digraphs, the most notable work is that of Qu *et al.* [22] in which a method is proposed for computing the left eigenvector associated with the zero eigenvalue of the Laplacian matrix of a digraph.

Out-degree computation. In the literature on distributed coordination the problem of out-degree computation has not been addressed. However, it has been addressed from a neighbor discovery perspective (see, e.g., [23], [24] and the references therein) and various algorithms have been proposed based on flooding, which requires an enormous communication overhead and storage capability. We revisit comparisons with flooding once we have a chance to establish notation and describe our approach in detail.

Spectrum computation. Distributed methods based on power iterations have been proposed (e.g., [25], [26]), but several iterations 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; even then, only approximate solutions can be guaranteed.

In [27], a novel decentralized algorithm is presented to estimate the eigenvalues of the Laplacian of the digraph. The agents construct distributed oscillators whose states oscillate at frequencies corresponding to eigenvalues of the graph Laplacian. Then, they apply the Fast Fourier Transform (FFT) to their state trajectory to extract the eigenvalues whose mode of operation is observable to the agents, i.e., there are modes which are not observable at some agents and hence some eigenvalues cannot be computed. A framework to study the observability of the Laplacian from a graph theoretical perspective was developed in [28]. The seminal idea in [27] was used in [29], in which the extracted information is used to study the controllability and observability of network topologies. Finally, [30] extends the results of [27] to remove components at null frequency and characterize analytically the amplitude and phase of the oscillations as a function of the eigenvectors of the Laplacian and the initial conditions.

C. Contributions

We adapt a distributed *finite-time* approach to compute various network parameters. More specifically, we compute:

(a) The left-eigenvector. Unlike other work in the literature, our proposed approach does not require a termination condition to be defined a priori, and subsequently there is no associated error with respect to the final value introduced when terminating an asymptotic algorithm in finite time. Our approach yields the *exact* entries of the left-eigenvector in a *finite* number of steps.

(b) The out-degree. Departing from flooding approaches that require big data chunks to be broadcast across the network and impose large memory requirements on nodes, a consensus approach for the computation of the out-degree or even the identification of the out-neighborhood is proposed.

(c) The spectrum of weighted adjacency matrices. We report the first method to compute the *exact* spectrum of weighted adjacency matrices for *digraphs* in a *finite number of steps*. The application of spectrum computation to self-configuration protocols for finite-time consensus is also discussed.

D. Outline of the paper

The remainder of the paper is organized as follows. In Section II, we review necessary notation and background. Sections III, IV and V present our algorithms for deriving the left-eigenvector, the out-degree and the spectrum for each node in a distributed fashion. Illustrative examples are presented throughout the paper to demonstrate the algorithms proposed. Common ground and some limitations of the proposed approaches are also discussed in Section VI. Finally, Section VII presents concluding remarks and future directions.

II. Notation and Preliminaries

A. Notation

The set of real (integer) numbers is denoted by $\mathbb{R}$ ($\mathbb{Z}$) and the set of positive numbers (integers) is denoted by $\mathbb{R}_+$ ($\mathbb{Z}_+$). $\mathbb{R}_+^n$ denotes the non-negative orthant of the n-dimensional real space $\mathbb{R}^n$. Vectors are denoted by small letters whereas matrices are denoted by capital letters. The transpose of a matrix A is denoted by A^{T}. For $A \in \mathbb{R}^{n\times n}$, A_{ij} denotes the entry in row i and column j. The i^{th} component of a vector $\mathbf{x}$ is denoted by x_i, and the notation $\mathbf{x} \geq \mathbf{y}$ implies that $x_i \geq y_i$ for all components i. By $\mathbb{1}$ we denote the all-ones vector and by I we denote the identity matrix (of appropriate dimensions). We also denote by $\mathbf{e}_j^{\mathrm{T}} = [0, \ldots, 0, 1_{j^{th}}, 0, \ldots, \ldots, 0] \in \mathbb{R}^{1\times n}$, where the single "1" entry is at the j^{th} position. $|A|$ is the element-wise absolute value of matrix A (i.e., $|A| \triangleq [|A_{ij}|]$), $A \leq B$ ($A < B$) is the (strict) element-wise inequality between matrices A and B. A matrix whose elements are nonnegative, called a nonnegative matrix, is denoted by $A \geq 0$, and a matrix whose elements are positive, called positive matrix, is denoted by $A > 0$. $\mathrm{diag}(x_i)$ denotes the matrix with elements x_1, x_2 , $\ldots$ on the leading diagonal and zeros elsewhere. The l_1 norm is defined by $\|\mathbf{x}\|_1 = \sum_i |x_i|$.

In multi-component systems with fixed communication links (edges), the exchange of information between components (nodes) can be conveniently captured by a directed graph (digraph) $\mathcal{G}(\mathcal{N},\mathcal{E})$ of order n $(n \geq 2)$, where $\mathcal{N} = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes and $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ is the set of edges. A directed edge from node v_i to node v_j is denoted by $\varepsilon_{ji} = (v_j, v_i) \in \mathcal{E}$ and represents a communication link that allows node v_j to receive information from node v_i.

A graph is said to be undirected if and only if $\varepsilon_{ji} \in \mathcal{E}$ implies $\varepsilon_{ij} \in \mathcal{E}$. In this paper, links are not required to be bidirectional, i.e., we deal with digraphs; for this reason, we use the terms "graph" and "digraph" interchangeably. A digraph is called *strongly* connected if there exists a path from each vertex v_i in the graph to each vertex v_j $(v_j \neq v_i)$. In other words, for any $v_j, v_i \in \mathcal{N}$, $v_j \neq v_i$, one can find a sequence of nodes $v_i = v_{l_1}, v_{l_2}, v_{l_3}, \ldots, v_{l_t} = v_j$ such that link $(v_{l_{m+1}}, v_{l_m}) \in \mathcal{E}$ for all $m = 1, 2, \ldots, t-1$.

All nodes that can transmit information to node v_j directly are said to be in-neighbors of node v_j and belong to the set $\mathcal{N}_j^- = \{v_i \in \mathcal{N} \mid \varepsilon_{ji} \in \mathcal{E}\}$. The cardinality of $\mathcal{N}_j^-$, is called the *in-degree* of v_j and is denoted by $\mathcal{D}_j^- = \left|\mathcal{N}_j^-\right|$. The nodes that receive information from node v_j belong to the set of out-neighbors of node v_j, denoted by $\mathcal{N}_j^+ = \{v_l \in \mathcal{N} \mid \varepsilon_{lj} \in \mathcal{E}\}$. The cardinality of $\mathcal{N}_j^+$, is called the *out-degree* of v_j and is denoted by $\mathcal{D}_j^+ = \left|\mathcal{N}_j^+\right|$. The adjacency matrix $\mathcal{A}$ is defined as the $n \times n$ matrix with entries

$$\mathcal{A}_{ji} = \begin{cases} 1, & \text{if } (v_j, v_i) \in \mathcal{E}, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The Laplacian matrix is defined as $\mathcal{L} = \mathcal{D} - \mathcal{A}$, where $\mathcal{D}$ is the diagonal in-degree matrix, i.e., $\mathcal{D} \triangleq \operatorname{diag}(\mathcal{D}_i^-)$.

In the algorithms we consider, we associate a positive weight p_{ji} with each edge $\varepsilon_{ji} \in \mathcal{E} \cup \{(v_j, v_j) \mid v_j \in \mathcal{N}\}$. The nonnegative matrix $P = [p_{ji}] \in \mathbb{R}_+^{n \times n}$ (with p_{ji} as the entry at its jth row, ith column position) is a weighted adjacency matrix (also referred to as weight matrix) that has zero entries at locations that do not correspond to directed edges (or self-edges) in the graph. In other words, apart from the main diagonal, the zero-nonzero structure of the adjacency matrix P matches exactly the given set of links in the graph.

We use $x_j[k] \in \mathbb{R}$ to denote the information state of node v_j at time t_k. In the synchronous setting, each node v_j updates and sends its information to its neighbors at discrete times $t_0, t_1, t_2, \ldots$.

Assumption 1: We assume that

1) the network is static, and,

2) each node has a unique ID.

Assumption 1.1 states that any change in the topology will require re-initiation of the schemes proposed in this work, which means that they would not work in a time-varying network topology. In addition, Assumption 1.2 is satisfied if each node has a unique identifier (e.g., MAC address).

B. Distributed linear iterations

Each node updates its information state $x_j[k]$ by combining the available information received by its neighbors $x_i[k]$ $(v_i \in \mathcal{N}_j^-)$ using the positive weights $p_{ji}[k]$, that capture the weight of the information inflow from agent v_i to agent v_j at time k. In this work, we assume that each node v_j can choose its self-weight and the weights on its in-coming links $\mathcal{N}_j^-$ only. Hence, in its general form, each node updates its information state according to the following relation:

$$x_j[k+1] = p_{jj}x_j[k] + \sum_{v_i \in \mathcal{N}_j^-} p_{ji}x_i[k], \tag{2}$$

for $k \geq 0$, where $x_j[0] \in \mathbb{R}$ is the initial state of node v_j. Let $\mathbf{x}[k] = (x_1[k] \;\; x_2[k] \;\; \ldots \;\; x_n[k])^{\mathrm{T}}$ and $P = [p_{ji}] \in \mathbb{R}_+^{n \times n}$. Then (2) can be written in matrix form as

$$\mathbf{x}[k+1] = P\mathbf{x}[k], \tag{3}$$

where $\mathbf{x}[0] = (x_1[0] \;\; x_2[0] \;\; \ldots \;\; x_n[0])^{\mathrm{T}} \triangleq x_0$. We say that the nodes asymptotically reach average consensus if

$$\lim_{k \to \infty} x_j[k] = \frac{\sum_{v_i \in \mathcal{N}} x_i[0]}{n}\ , \quad \forall v_j \in \mathcal{N}\ .$$

The necessary and sufficient conditions for (3) to reach average consensus are the following: (a) P has a simple eigenvalue at one, with left eigenvector $\mathbb{1}^{\mathrm{T}}$ and right eigenvector $\mathbb{1}$, and (b) all other eigenvalues of P have magnitude less than 1. If $P \geq 0$ (as in our case), the necessary and sufficient condition is that P is a primitive doubly stochastic matrix. In an undirected graph, assuming each node knows n (or an upper bound n') and the graph is connected, each node v_j can choose the weights on its outgoing links to be $\frac{1}{n'}$ and set its diagonal to be $1 - \frac{\mathcal{D}_j}{n'}$ (where $\mathcal{D}_j^+ = \mathcal{D}_j^- \triangleq \mathcal{D}_j$). The resulting P is primitive doubly stochastic, and nodes do not require global topological information other than n' to implement P. However, in a digraph, this choice does not necessarily lead to a doubly stochastic weight matrix [18], [31].

In this work, two distributed weight selection schemes are considered, but any weighted adjacency matrix can be used and the two examples below are used for demonstration purposes.

Distributed weight selection 1: The update of the states is such that the update matrix P is a nonnegative row-stochastic matrix, given by

$$P_{ji} \triangleq \begin{cases} 1 - c_j, & \text{if } i = j, \\ c_j/\mathcal{D}_j^-, & \text{if } v_i \in \mathcal{N}_j^-, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

where $c_j \in (0, 1)$.

Distributed weight selection 2: The update of the states is such that the update matrix P is the *Perron matrix*, i.e.,

$$P = I - \theta\mathcal{L}, \quad 0 < \theta < \frac{1}{\max_i\{\mathcal{D}_i^-\}}. \tag{5}$$

Note that one can easily compute the Laplacian eigenvalues from the eigenvalues of the Perron matrix, since

$$\lambda(P) = 1 - \theta\lambda(\mathcal{L}). \tag{6}$$

C. Finite-time consensus

In [5], [6] a distributed algorithm is proposed by which any arbitrarily chosen agent in an *undirected* graph can compute

its asymptotic final consensus value in a finite number of steps. The algorithm is based on letting each node execute (2) and store a sequence of consecutive iterates. The result hinges on the use of the concept of the minimal polynomial associated with the consensus dynamics (3) in conjunction with the final value theorem.

First, we provide definitions on minimal polynomials that comprise the salient feature of the analysis.

Definition 1: (Minimal polynomial of a matrix) The minimal polynomial associated with a matrix P denoted by

$$q(t) = t^{D+1} + \sum_{i=0}^{D} \alpha_i^{(j)} t^i \tag{7}$$

is the monic polynomial of minimum degree $D+1$ that satisfies $q(P) = 0$.

Note that the minimal polynomial will have a degree at most $|\mathcal{N}|$ (i.e., $D+1 \leq |\mathcal{N}|$), which means that $q(t)$, if not the same, divides the *characteristic polynomial* $\chi(t)$ of a matrix.

Definition 2: (Minimal polynomial of a matrix pair) The minimal polynomial associated with the matrix pair $[P, e_j^{\mathrm{T}}]$ denoted by

$$q_j(t) = t^{M_j+1} + \sum_{i=0}^{M_j} \alpha_i^{(j)} t^i \tag{8}$$

is the monic polynomial of minimum degree $M_j + 1$ that satisfies $e_j^{\mathrm{T}} q_j(P) = 0$.

The minimal polynomial of a matrix pair $q_j(t)$ is unique and is not necessarily the same as that of a matrix $q(t)$ (i.e., $M_j \leq D$). In fact, $q_j(t)$ divides $q(t)$ (the proof of these statements can be found in [32]).

Considering the iteration in (3) with weight matrix P, it is shown that

$$\sum_{i=0}^{M_j+1} \alpha_i^{(j)} x_j[k+i] = 0, \quad \forall k \in \mathbb{Z}_+ \,, \tag{9}$$

where $\alpha_{M_j+1}^{(j)} = 1$.

We denote the z-transform of $x_j[k]$ as $X_j(z) \stackrel{\triangle}{=} \mathcal{Z}(x_j[k])$. From (9) and the time-shift property of the $z-$transform,

$$X_j(z) = \frac{\sum_{i=1}^{M_j+1} \alpha_i^{(j)} \sum_{\ell=0}^{i-1} x_j[\ell] z^{i-\ell}}{q_j(z)}. \tag{10}$$

If the network is strongly connected, the minimal polynomial $q_j(z)$ of $[P, e_j^{\mathrm{T}}]$, does not have any unstable poles apart from one at 1; we can then define the following polynomial:

$$p_j(z) \triangleq \frac{q_j(z)}{z-1} \triangleq \sum_{i=0}^{M_j} \beta_i^{(j)} z^i \,. \tag{11}$$

The application of the final value theorem [5], [6] yields:

$$\phi_x(j) = \lim_{k\to\infty} x_j[k] = \lim_{z\to 1}(z-1)X_j(z) = \frac{x_{M_j}^{\mathrm{T}} \boldsymbol{\beta}_j}{\mathbb{1}^{\mathrm{T}} \boldsymbol{\beta}_j} \,, \tag{12}$$

where $x_{M_j}^{\mathrm{T}} = (x_j[0], x_j[1], \ldots, x_j[M_j])$ and $\boldsymbol{\beta}_j$ is the vector of coefficients of the polynomial $p_j(z)$. From the analysis above and in line with [5, Theorem 1], we have the following proposition.

Proposition 1: Consider a strongly connected digraph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. Let $x_j[k]$ (for all $v_j \in \mathcal{V}$ and $k = 0, 1, 2, \ldots$) be the result of the iteration (2), where $P = [p_{ji}] \in \mathbb{R}_+^{n\times n}$ is any set of weights that adhere to the graph structure and form a primitive column stochastic weight matrix. The solution to the iteration can be distributively obtained in finite-time, i.e., $x_j^* \triangleq \lim_{k\to\infty} x_j[k] = \phi_w(j)$, where $\phi_x(j)$ is given by (12).

The next question is how one can obtain the coefficient vector $\boldsymbol{\beta}_j$ in the computation of final values, i.e., eq. (12). Consider the vectors of $2k+1$ successive discrete-time values at node v_j, given by

$$x_{2k}^{\mathrm{T}} = (x_j[0], x_j[1], \ldots, x_j[2k]),$$

for the iteration $x_j[k]$ at node v_j . Let us define their associated Hankel matrix:

$$\Gamma\{x_{2k}^{\mathrm{T}}\} \triangleq \begin{bmatrix} x_j[0] & x_j[1] & \ldots & x_j[k] \\ x_j[1] & x_j[2] & \ldots & x_j[k+1] \\ \vdots & \vdots & \ddots & \vdots \\ x_j[k] & x_j[k+1] & \ldots & x_j[2k] \end{bmatrix}.$$

We also consider the vector of differences between successive values of $x_j[k]$, i.e., $\overline{x}_{2k}^{\mathrm{T}} = (x_j[1] - x_j[0], \ldots, x_j[2k+1] - x_j[2k])$. It has been shown in [6] that $\boldsymbol{\beta}_j$ can be computed as the kernel of the first defective Hankel matrix $\Gamma\{\overline{x}_{2k}^{\mathrm{T}}\}$ for arbitrary initial conditions x_0, except for a set of initial conditions that has Lebesgue measure zero.

III. Finite-time left eigenvector computation

In this section, we present a distributed finite-time algorithm that computes the left eigenvector in a finite number of steps. The algorithm is based on the following lemma, which is a consequence of the Perron-Frobenius theorem (see, for example, [33]).

Lemma 1: Let P be a primitive[1] matrix, and let $\mathbf{w}$ and $\mathbf{v}$ denote the unique normalized (i.e., $\|\mathbf{v}\|_1 = \|\mathbf{w}\|_1 = 1$) left and right eigenvectors of P corresponding to the Perron-Frobenius eigenvalue[2] ρ; i.e., the spectral radius of P is equal to ρ, with $P\mathbf{v} = \rho\mathbf{v}$ and $\mathbf{w}^{\mathrm{T}}P = \rho\mathbf{w}^{\mathrm{T}}$. Then,

$$\lim_{k\to\infty} \left(\frac{P}{\rho}\right)^k = \frac{\mathbf{v}\mathbf{w}^T}{\mathbf{w}^T\mathbf{v}}. \tag{13}$$

For a row stochastic matrix P, $\rho = 1$ and $\mathbf{v} = \mathbb{1}/n$, and provided that $\mathbf{w}$ is normalized, then

$$\lim_{k\to\infty} P^k \mathbf{e}_j = \frac{\mathbb{1}\mathbf{w}^T}{\mathbf{w}^T\mathbb{1}}\mathbf{e}_j = \mathbb{1}\mathbf{w}^T\mathbf{e}_j = w_j\mathbb{1}.$$

Hence, by having an iteration of the form of (3) with initial value $\mathbf{x}[0] = \mathbf{e}_j$, the coordinate $x_j[k]$ converges to the j-th element of the left eigenvector $\mathbf{w}$, w_j, as $k \to \infty$. When the

[1] A nonnegative matrix $P = [P_{ji}] \in \mathbb{R}_+^{n\times n}$ is said to be *primitive* if there exists k such that $P^k > 0$. A sufficient condition for matrix P to be primitive is for the matrix to be a nonnegative, irreducible (matrix P is irreducible if, and only if, its associated graph $\mathcal{G}$ is strongly connected) with at least a positive element on the main diagonal [33].

[2] The Perron-Frobenius eigenvalue is a simple positive eigenvalue and any other eigenvalue λ (possibly, complex) is strictly smaller than ρ in magnitude, i.e., $|\lambda| < \rho$ [33].

non-zero entries of P match the structure of the network $\mathcal{G}$ (i.e., $(v_j, v_i) \in \mathcal{E} \Leftrightarrow P_{ji} > 0$), then the linear iteration (3) can be implemented over the network via message passing. Let $x_{ij}[k]$ denote the state of node v_i for an iteration regarding the computation of w_j at time k. This suggests that if each node v_j initiates an iteration in which it only has initial value $x_{jj}[0] = 1$ and all other nodes v_i ($v_i \in \mathcal{N}, v_i \neq v_j$) have initial value $x_{ij}[0] = 0$, then all the nodes will eventually compute w_j. More specifically, for $k = 0, 1, 2, \ldots$, each node $v_j \in \mathcal{N}$ chooses a proportionality gain $c_j \in (0, 1)$ and updates the values x_{jl} for the iteration initiated by node v_l, $\forall v_l \in \mathcal{N}$, as

$$x_{jl}[k+1] = x_{jl}[k] + c_j \left(\frac{\sum_{v_i \in \mathcal{N}_j^-} x_{il}[k]}{\mathcal{D}_j^-} - x_{jl}[k] \right), \quad (14)$$

where

$$x_{jl}[0] = \begin{cases} 1, & \text{if } l = j, \\ 0, & \text{otherwise.} \end{cases} \quad (15)$$

Let $\mathbf{x}_l = (x_{1l} \;\; x_{2l} \;\; \ldots \;\; x_{nl})^{\mathrm{T}}$. The evolution of equation (14) can be written in matrix form as

$$\mathbf{x}_l[k+1] = P\mathbf{x}_l[k], \quad \mathbf{x}_l[0] = \mathbf{e}_l, \quad (16)$$

where P is a nonnegative row-stochastic matrix given by (4).

Remark 1: Note that all nodes are required to compute all the values of the eigenvector. Also, since each node has a unique identifier in the network (e.g., a MAC address), node v_j is not required to know the number of nodes in the network a priori, but initiates a new iteration whenever it receives a packet associated with a new identifier (i.e., we can associate node v_j's identifier with the iteration initiated by node v_j).

Proposition 2: Assume that the initial value of $x_{jl}[0]$ is chosen by node v_j, according to (15), and let $x_{jl}[k]$ be iteration (14). Then, for all j, $\lim_{k\to\infty} x_{jl}[k] = w_l$, where w_l is the l-th element of the left eigenvector $\mathbf{w}$, i.e., $\mathbf{w} = (w_1 \;\; w_2 \;\; \ldots \;\; w_n)^{\mathrm{T}}$ and satisfies $\mathbf{w}P = \mathbf{w}$.

Proof: It can be easily shown that P in (4) is primitive and row stochastic. Then, the proof follows from the Perron-Frobenius theorem, i.e., the iterations converge to $\lim_{k\to\infty} P^k \mathbf{e}_l = w_l \mathbb{1}$. ∎

A. Finite-time estimation of the left-eigenvector

We propose an algorithm with which every node v_j using the asymptotic algorithm (14) can compute the elements of the left eigenvector $\mathbf{w}$ in a finite number of steps. Using the arguments for finite-time consensus, it is easy to prove the following proposition (similar to Theorem 1 in [5]).

Proposition 3: Consider a strongly connected digraph $\mathcal{G}(\mathcal{N}, \mathcal{E})$. Let $x_{jl}[k]$ (for all $v_j \in \mathcal{N}$ and $k = 0, 1, 2, \ldots$) be the result of the iteration (14), where $P = [P_{ji}] \in \mathbb{R}_+^{n\times n}$ is any set of weights that adhere to the graph structure and form a primitive column stochastic weight matrix. The unique fixed-point of (14) with initialization (15) can be obtained in finite-time and in a distributed manner, i.e., $x_{jl}^* \triangleq \lim_{k\to\infty} x_{jl}[k] = \phi_{\mathbf{x}_l}(j)$, where $\phi_{\mathbf{x}_l}(j)$ is given by (12).

Consider the vectors of $2k+1$ successive discrete-time values at node v_j, given by $\mathbf{x}_{l,2k}^{\mathrm{T}} = (x_{jl}[0], x_{jl}[1], \ldots, x_{jl}[2k])$, for the iteration $x_{jl}[k]$ (as given in (14)). Let us define its associated Hankel matrix as

$$\Gamma\{\mathbf{x}_{l,2k}^{\mathrm{T}}\} \triangleq \begin{bmatrix} x_{jl}[0] & x_{jl}[1] & \ldots & x_{jl}[k] \\ x_{jl}[1] & x_{jl}[2] & \ldots & x_{jl}[k+1] \\ \vdots & \vdots & \ddots & \vdots \\ x_{jl}[k] & x_{jl}[k+1] & \ldots & x_{jl}[2k] \end{bmatrix},$$

and the vector of differences between successive values of $x_{jl}[k]$ as $\overline{\mathbf{x}}_{lM_j}^{\mathrm{T}} = (x_{jl}[1] - x_{jl}[0], \ldots, x_{jl}[2k+1] - x_{jl}[2k])$. We now introduce an algorithm, herein called *Algorithm 1*, in which the nodes compute the left eigenvector in a finite number of steps.

Algorithm 1 Distributed finite-time left-eigenvector computation

Input: A strongly connected digraph $\mathcal{G}(\mathcal{N}, \mathcal{E})$.

Data: Successive observations for $x_{jl}[k]$, $\forall v_j \in \mathcal{N}$, $k = 0, 1, \ldots$ via iteration (14), with initial conditions $\mathbf{x}_l[0] = \mathbf{e}_l^{\mathrm{T}}$.

Step 1: For $k = 0, 1, 2, \ldots$, each node $v_j \in \mathcal{N}$ runs iteration (14) and stores the vectors of differences $\overline{\mathbf{x}}_{lM_j}^{\mathrm{T}}$ between successive values of $x_{jl}[k]$.

Step 2: Increase the dimension k of the square Hankel matrices $\Gamma\{\overline{\mathbf{x}}_{lM_j}^{\mathrm{T}}\}$ until they lose rank and store their first defective matrix.

Step 3: The kernel $\boldsymbol{\beta}_j = \left(\beta_0, \ldots, \beta_{M_j-1}, 1\right)^{\mathrm{T}}$ of the first defective matrix gives the coefficients of $\phi_{\mathbf{x}_l}$.

Step 4: The final value which is the jth element of the left eigenvector w_j is computed by

$$x_{jl}^* = \frac{\mathbf{x}_{\mathbf{l}M_j}^{\mathrm{T}} \boldsymbol{\beta}_j}{\mathbb{1}^{\mathrm{T}} \boldsymbol{\beta}_j}.$$

Output: Assign $x_{jl}^* = w_l$, $l \in \mathcal{N}$.

B. Memory complexity and early termination

Each node does not need to know a priori how many values it needs to store. However, as soon as the square Hankel matrices lose rank, the nodes can stop storing information. Throughout this work we assume that nodes have enough memory to store at least as many values as necessary to obtain the resulting defective matrix. Nevertheless, it has been shown in [5] that the number of steps required to predict their final value is less than $2|\mathcal{N}|$, so no more than $2|\mathcal{N}|$ are stored.

The asymptotic approach proposed by Qu *et al.* [22] for computing the left-eigenvector has been adapted by Priolo *et al.* in [8], where a predefined maximum value of the number of iterations is introduced. This termination value is chosen large enough such that the error from the actual final value is small enough. This termination criterion, however, depends on the network size and structure (parameters that affect the predefined maximum value), needs to be known to all nodes prior to the execution of the algorithm, and the value obtained is an approximation to the final value.

In Algorithm 1, nodes cannot stop the iterations, even if they have computed the exact left-eigenvector, unless they know that $2|\mathcal{N}|$ steps have already been executed (and hence all other nodes have computed their final values). Knowledge of

the number of nodes is indirectly obtained by the number of iterations initiated; i.e., since each node initiates an iteration that leads to the computation of the corresponding value in the left-eigenvector, eventually each node will be executing and finding the final value to exactly $|\mathcal{N}|$ iterations. Hence, the termination value is inherited through the iterations and the termination value, by which all nodes have already computed the *exact* left-eigenvector, can be determined.

The importance of computing the exact left-eigenvector will be demonstrated in the next section, where it is used by each node for the computation of its out-degree.

IV. Finite-time out-degree computation

The problem we consider in this section is to design a distributed coordination algorithm that allows every node in a *directed graph* to compute the out-degree in a finite number of steps. The main contribution of this section is a method by which nodes can compute, in a distributed manner, their out-degree and in many cases can identify their out-neighborhood. Once the left eigenvector is computed, each agent solves a subset-sum problem to find its out-degree and, in some cases, to identify its out-neighborhood. If the subset-sum problem has multiple solutions, some of which yield different out-degrees, then it is not possible to directly determine the correct answer. For these situations, we propose another distributed approach to compute the out-degree that requires additional communication.

A. Out-degree computation via integer linear programming

Since $\mathbf{w}$ is the left eigenvector of matrix P, we have that $\mathbf{w}^{\mathrm{T}}P = \mathbf{w}^{\mathrm{T}}$. For a single node v_j, this is equivalent to

$$w_j c_j = \sum_{v_i \in \mathcal{N}_j^+} w_i \frac{c_i}{\mathcal{D}_i^-} \,. \tag{17}$$

Note that the summation is over the set of out-neighbors of node v_j. Once node v_j has computed w_j and w_i for all v_i in $\mathcal{N} - \{v_j\}$ for which it received a message (i.e., once v_j has the left eigenvector $\mathbf{w}$), it is required to solve

$$w_j c_j = \sum_{v_i \in \mathcal{N}} w_i \frac{c_i}{\mathcal{D}_i^-} \mathbb{I}_i^j, \tag{18}$$

where $\mathbb{I}_i^j$ is a binary variable that indicates whether v_i belongs to the out-neighborhood of node v_j; i.e.,

$$\mathbb{I}_i^j = \begin{cases} 1, & \text{if } v_i \in \mathcal{N}_j^+, \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

This is a *subset sum problem* (a special case of the *knapsack problem* [34]), often formulated as follows: given n items with sizes $p_1, p_2, \ldots, p_n$ with $p_i > 0$, and given a capacity $W > 0$, maximize the sum of size of items chosen, provided that it remains below or equal to the capacity W; i.e.,

$$\text{maximize} \quad \sum_{i=1}^{n} p_i z_i, \tag{20a}$$

$$\text{subject to} \quad \sum_{i=1}^{n} p_i z_i \leq W, \tag{20b}$$

$$z_i \in \{0, 1\}, \quad i = 1, \ldots, n. \tag{20c}$$

As a result, each node v_j executes a subset sum algorithm via *integer linear programming* (ILP), with $p_i \triangleq w_i \frac{c_i}{\mathcal{D}_i^-}$ and $W \triangleq w_j c_j$ (all of which are parameters known, or can be made known, to node v_j). Note that the maximization guarantees that the inequality constraint will be activated, since we know that at least (17) holds and the maximum value of the utility function will be equal to the capacity W.

Each node v_j initiating an iteration should also communicate the ratio $c_j/\mathcal{D}_i^-$ (or even the individual values of c_j and $\mathcal{D}_i^-$), for making the solution of the optimization problem possible. It is also possible to use the initial condition $\frac{c_l}{\mathcal{D}_l^-}\mathbf{e}_l^{\mathrm{T}}$, that provides $\frac{c_l}{\mathcal{D}_l^-} w_l$ directly. However, the values of the left-eigenvector $\mathbf{w}$ might be required by Algorithm 2 that will be introduced in Section IV-B.

Remark 2: As previously mentioned, other approaches in the literature assume that iterations for finding the left-eigenvector are executed for a pre-determined finite number of steps (e.g., [8], [9]). Such iterations do not necessarily converge to the exact values after a finite number of steps. Depending on the resulting error, using an approximate left-eigenvector may cause the ILP to return an incorrect out-degree, since the summation will not be necessarily exact. Hence, such approaches may become inappropriate within the context of the subset sum optimization.

Example 1: Consider the directed network in Fig. 1 where each node v_j chooses $c_j = 0.5$ (the precise value of c_j does not change the final result, but it affects the convergence rate).

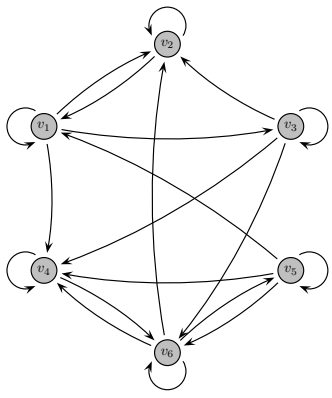

Fig. 1. A digraph consisting of six nodes.

When each node updates its information state $x_{jl}[k]$ using equation (16), the information state for the whole network is captured by $\mathbf{x}_l[k+1] = P\mathbf{x}_l[k]$, where P is given by

$$P = \begin{pmatrix} 1/2 & 1/4 & 0 & 0 & 1/4 & 0 \\ 1/6 & 1/2 & 1/6 & 0 & 0 & 1/6 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 1/8 & 0 & 1/8 & 1/2 & 1/8 & 1/8 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 1/6 & 1/6 & 1/6 & 1/2 \end{pmatrix}.$$

The left eigenvector computed by each node is given by

$$\mathbf{w} = (0.1978 \quad 0.0989 \quad 0.1429 \quad 0.0879 \quad 0.2088 \quad 0.2637)^{\mathrm{T}}.$$

Each node runs an ILP (the modeling and solution of the ILP formulation (20) was performed using the Gurobi optimization solver [35]) to find its out-neighbors. The out-degree computed and out-neighbors identified by each node agree with the network. Note that agents might not be able to use such a solver for the subset sum problem. However, there are other *exact* approaches with reduced complexity, such as the one proposed in [36] that uses a hybrid algorithm combining a tree search and a dynamic programming method specialized to the problem.

Example 2: Now consider the network shown in Fig. 2, where each node v_j again uses $c_j = 0.5$.

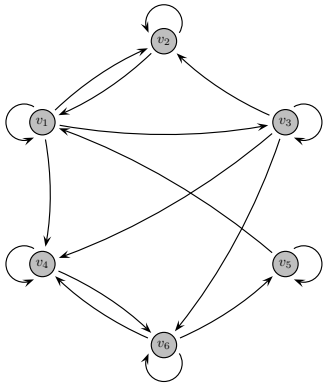

Fig. 2. A digraph consisting of six nodes.

The left eigenvector computed by each node is given by

$$\mathbf{w} = (0.2817 \quad 0.1408 \quad 0.1831 \quad 0.0845 \quad 0.1408 \quad 0.1690)^{\mathrm{T}}.$$

Note that $w_1/\mathcal{D}_1^- = w_5/\mathcal{D}_5^- = 0.1408$ and hence for some nodes the optimization has more than one solution. Note that although $w_2/\mathcal{D}_2^- = 0.1408/2 = 0.0704$, it does not cause more than one solution in this example. While more than one solution exists, all of them give the same out-degree. By running the ILP for this problem, one solution gives that node v_6 has an outgoing link to node v_1 instead of v_5, but the out-degree is still correct.

The above example illustrates that, on some occasions, the size of the sum of one or more items p_i may be equal to the size of the sum of two or more other items, e.g., $p_i = \sum_{\ell \in \mathcal{S}} p_\ell$, where $\mathcal{S}$ is the subset of items whose size sums to p_i. In this case, there exists more than one solution to (20) and v_j cannot deduce which is its correct out-degree.

Example 3: Now we consider a different example that shows the limitation of the proposed approach. Consider the network given by Fig. 3 where each node v_j chooses again $c_j = 0.5$. Matrix P can be easily constructed, and the left

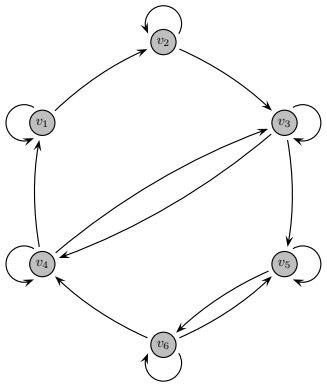

Fig. 3. A digraph consisting of six nodes.

eigenvector computed by each node is given by

$$\mathbf{w} = (0.1 \quad 0.1 \quad 0.2 \quad 0.2 \quad 0.2 \quad 0.2)^{\mathrm{T}}.$$

We have that $w_1/\mathcal{D}_1^- = w_2/\mathcal{D}_2^- = \ldots = w_4/\mathcal{D}_4^- = w_6/\mathcal{D}_6^-$ and $w_5/\mathcal{D}_5^- = 2w_1/\mathcal{D}_1^-$. As a result, while v_6 is the only out-neighbor of v_5 the optimization returns more than one answer and most of these answers consider that v_5 has 2 out-neighbors (2 of the nodes v_1, v_2, v_3 and v_4).

As seen in Example 3, the possibility of the subset sum optimization to uniquely identify the out-degree of a node v_j is not negligible. Therefore, we next develop an alternative way for nodes to determine their out-degree if the optimization fails, or, if the nodes do not have the computational power to do so and the additional coordination between the agents in the form of local communication might be a better option (e.g., when the number of nodes in the network is large).

B. Out-degree computation via a consensus approach

In what follows, we propose an algorithm with which a node is guaranteed to compute its exact out-degree. This approach, which requires additional communication, can be used when the optimization fails.

The algorithm proceeds as follows. Once node v_j runs the ILP and obtains more than one solution, some of which have different out-degrees, it initiates a new iteration, as in (14), with initial values as described next. Nodes v_i that directly receive information from node v_j (i.e., the out-neighbors of node v_j) set their initial value to $1/w_i$, while all the other nodes set their initial value for this iteration to zero. With this initialization, for iteration (14) initiated by node v_j, the value to which each node will asymptotically converge is given by

$$\sum_{i=1}^{n} w_i x_{ij}[0] = \sum_{v_i \in \mathcal{N}_j^+} w_i \left(\frac{1}{w_i}\right) = \mathcal{D}_j^+.$$

This value is computed in a finite number of steps via a finite-time approach that finds the final value, as in Algorithm 1, with the modified initial condition described above. This second method for computing the out-degree of node v_j is summarized in Algorithm 2.

Remark 3: Note that the optimization approach to compute the out-degree is preferred because it does not require any

Algorithm 2 Distributed finite-time out-degree computation

Input: A strongly connected digraph $\mathcal{G}(\mathcal{N}, \mathcal{E})$.

Data: Successive observations for $x_{jl}[k]$, $\forall v_j \in \mathcal{N}$, $k = 0, 1, \ldots$ via iteration (14), with initial conditions

$$x_{jl}[0] = \begin{cases} \dfrac{1}{w_j}, & \text{if } v_j \in \mathcal{N}_l^+, \\ 0, & \text{otherwise.} \end{cases}$$

Step 1: For $k = 0, 1, 2, \ldots$, each node $v_j \in \mathcal{N}$ runs iteration (14) and stores the vectors of differences $\overline{\mathbf{x}}_{l\,M_j}^{\mathrm{T}}$ between successive values of $x_{jl}[k]$.

Step 2: Increase the dimension k of the square Hankel matrices $\Gamma\{\overline{\mathbf{x}}_{l\,M_j}^{\mathrm{T}}\}$ until they lose rank and store their first defective matrix.

Step 3: The kernel $\boldsymbol{\beta}_j = \left(\beta_0, \ldots, \beta_{M_j-1}, 1\right)^T$ of the first defective matrix gives the coefficients of $\phi_{\mathbf{x}_l}$.

Step 4: The final value is computed by

$$x_{jl}^* = \frac{\mathbf{x}_{\mathbf{l}\,M_j}^{\mathrm{T}} \boldsymbol{\beta}_j}{\mathbb{1}^{\mathrm{T}} \boldsymbol{\beta}_j} = \mathcal{D}_l^+.$$

further coordination between the nodes. As a result, Algorithm 2 should be used only when the optimization approach fails to provide a definite answer. Also, it should be mentioned that Algorithm 2 cannot be used unless the values for the left eigenvector are already known to each node.

Example 4: This example revisits the network from Example 3 (see Fig. 3), where the ILP fails for node v_5. Node v_5 initializes a subsequent iteration, and all nodes execute Algorithm 2 for finding the out-degree of node v_5. The results of this iteration are shown in Fig. 4. No other nodes initiate subsequent iterations of Algorithm 2 since their out-degrees are uniquely determined after the ILP finishes.

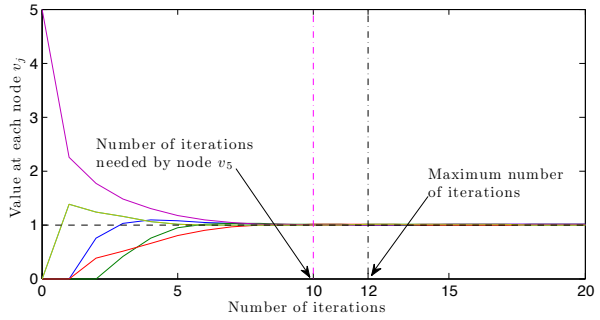

Fig. 4. For the network in Fig. 3, node v_5 initializes an iteration that executes Algorithm 2 and each node can compute the out-degree of node v_5 in a finite number of steps. The maximum number of steps required is 12 (by nodes v_3 and v_5, whereas node v_5 required 10 steps only.

C. Flooding approach

The problem of out-degree computation has been also considered from a neighbor discovery perspective (see [23], [24], [37] and references therein) and various algorithms have been proposed. With such an approach, each node v_j would need to broadcast (a) $\mathcal{D}_j^-$ packets describing the links of the form (v_j, v_i) (where $v_i \in \mathcal{N}_j^-$), and (b) all packets received from other nodes that have not been forwarded to its out-neighbors. This requires the storing of $O(n\mathcal{D}_{\max}^-)$ entries at each node, where $\mathcal{D}_{\max}^-$ is the maximum in-degree among all nodes), and it takes $O(n\mathcal{D}_{\max}^- d_{\max})$ steps to complete, where $d_{\max}$ is the diameter of the network. The algorithm proposed here requires running n parallel iterations, each of which require a maximum memory of $2d_{\max}$.

D. Performance evaluation

To further evaluate the performance of optimization (20), we simulate Erdős-Rényi random networks (a link between two nodes exists with probability q) of sizes $10, 20, \ldots, 90, 100$. We explore different connectivity probabilities $q \in \{0.1, 0.25, 0.5, 0.75, 0.9\}$. For each network size and connectivity probability q, we sample 100 random graphs and run Algorithm 1. The proportion of nodes that do not uniquely determine their out-degree after running the ILP is shown in Fig. 5.

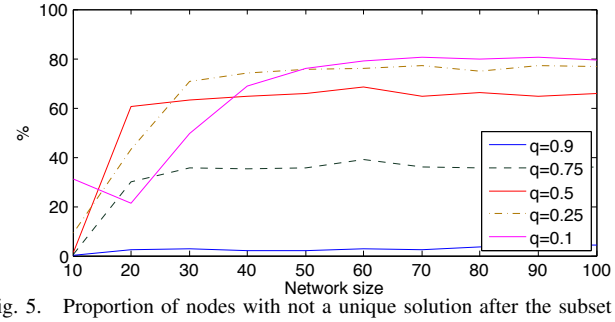

Fig. 5. Proportion of nodes with not a unique solution after the subset sum optimization problem. For each network size and "connectivity" probability q, we consider 100 random graphs.

The connectivity of the network affects the proportion of nodes for which the ILP does not uniquely determine the out-degree; i.e., the higher the connectivity the lower is the proportion of nodes having more than one solution for the out-degree problem via the subset optimization (see Fig. 5); hence, nodes are more likely to find their out-degree using the ILP when the connectivity of the network is higher.

V. Finite-time spectrum computation

In this work, we compute the spectrum of weighted adjacency matrices (such as the Perron matrix) in a distributed setting where the network topology is unknown and directed. More specifically, by having the agents gather information about the history of their own states, without imposing an additional interaction rule between the agents, each agent can compute in a finite number of steps the eigenvalues of the modes that are observable to that agent. Note that the undirected graph constitutes a special case.

From equation (11), the correspondence between the coefficients $\boldsymbol{\beta}_j$ and the coefficients of the minimal polynomial of the matrix pair $[P, \mathbf{e}_j^{\mathrm{T}}]$ are as follows:

$$t^{M_j+1} + \alpha_{M_j}^{(j)} t^{M_j} + \ldots + \alpha_1^{(j)} t + \alpha_0^{(j)} = (t-1)(t^{M_j} + \beta_{M_j-1}^{(j)} t^{M_j-1} + \ldots + \beta_1^{(j)} t + \beta_0^{(j)}). \quad (21)$$

Hence, having the coefficients $\boldsymbol{\beta}_j$, the coefficients of the minimal polynomial of the matrix pair $[P, \mathbf{e}_j^{\mathrm{T}}]$ can be found via (21). Kernel $\boldsymbol{\beta}_j$ provides all the eigenvalues corresponding to modes of system (2) that are observable from state x_j, excluding the eigenvalue at 1.

Next, we provide an algorithm for computing the exact eigenvalues of the observable modes in a distributed fashion and finite number of steps for strongly connected digraphs. The algorithm is similar to Algorithms 1 and 2. In this case, for any initial condition (which could be one of the initial conditions from Algorithms 1 and 2), each node computes the roots of the monic polynomial given in (21), i.e., the observable eigenvalues of the weighted adjacency matrix under consideration.

Algorithm 3 Distributed finite-time spectrum computation

Input: A strongly connected digraph $\mathcal{G}(\mathcal{N}, \mathcal{E})$.

Data: Successive observations for $x_j[k]$, $\forall v_j \in \mathcal{N}$, $k = 0, 1, 2, \ldots$, using iterations (2), with initial conditions

$$x[0] = x_0.$$

Step 1: For $k = 0, 1, 2, \ldots$, each node $v_j \in \mathcal{N}$ runs the consensus algorithm (2) and stores the vectors of differences $\overline{x}_{M_j}^{\mathrm{T}}$ between successive values of $x_j[k]$.

Step 2: Increase the dimension k of the square Hankel matrices $\Gamma\{\overline{x}_{M_j}^{\mathrm{T}}\}$ for each of the iterations, until they lose rank; store their first defective matrix.

Step 3: The kernel $\boldsymbol{\beta}_j = \left(\beta_0, \ldots, \beta_{M_j-1}, 1\right)^{\mathrm{T}}$ of the first defective matrix is thus extracted.

Step 4: The roots of the monic polynomial given in (21) are the eigenvalues of the weighted adjacency matrix considered.

A. Observability of states

Each node v_j can independently compute all the eigenvalues corresponding to modes of system (2) that are observable from that node. A framework to study the observability of the Laplacian from a graph theoretical perspective was developed in [28] and for general weighted adjacency matrices was developed by [32].

Here, we stress the connection of the observability Gramian with the minimal polynomial of a matrix pair and hence, the number of distinct eigenvalues observable from node v_j. By Definition 2, we have that $e_j^{\mathrm{T}} q_j(P) = 0$; this can be written as $\begin{pmatrix} a_0 & a_1 & \ldots & a_{M_j} & 1 \end{pmatrix}^T \mathcal{O}_{j,M_j+1} = 0$, where $\mathcal{O}_{j,M_j+1}$ is the observability Gramian for a matrix pair and it is given by

$$\mathcal{O}_{j,M_j+1} = (e_j^{\mathrm{T}} \;\; e_j^{\mathrm{T}} P \;\; e_j^{\mathrm{T}} P^2 \;\; \ldots \;\; e_j^{\mathrm{T}} P^{M_j+1})^{\mathrm{T}}.$$

The coefficients of $q_j(t)$ can be found by forming the matrix $\mathcal{O}_{j,M_j+1}$ and increasing M_j until $\mathcal{O}_{j,M_j+1}$ loses rank; the coefficients of $q_j(t)$ can then be obtained from the left nullspace of $\mathcal{O}_{j,M_j+1}$. However, this requires knowledge of global parameters (such as the weighted adjacency matrix P) on each node v_j. The order of the characteristic equation for node v_j and subsequently, the rank of its observability Gramian is given by the observable modes of operation from that node.

Note that information about the observability of the system is not required for the execution of Algorithm 3, but, of course, each node can only compute the eigenvalues that are observable to it.

B. Illustrative examples

Illustrative examples show how node v_j uses Algorithm 3 to compute the eigenvalues corresponding to modes of system (2) that are observable from v_j.

Example 5: Consider the network given by Fig. 6. When

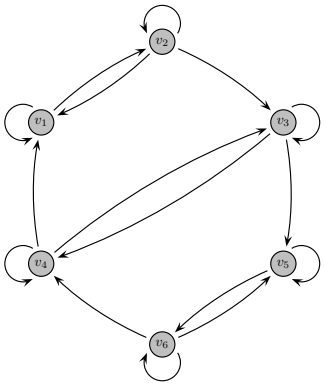

Fig. 6. A digraph consisting of six nodes.

each node updates its information state $x_{jl}[k]$ using equation (2), the information state for the whole network is given by $\mathbf{x}_l[k+1] = P\mathbf{x}_l[k]$, where P is based on (4) with $c_j = 0.5$ for all $v_j \in \mathcal{N}$. The spectrum of the weighted adjacency matrix P is $\sigma(P) = (1 \;\; 0.75 \;\; 0 \;\; 0.25 \;\; 0.5 \;\; 0.5)^{\mathrm{T}}$. Node v_6 computes $\boldsymbol{\beta}_6$ to be $\boldsymbol{\beta}_6 = (0 \;\; -0.0938 \;\; 0.6875 \;\; -1.5 \;\; 1)^{\mathrm{T}}$, with roots $\boldsymbol{\lambda}_6 = (0.75 \;\; 0.5 \;\; 0.25 \;\; 0)^{\mathrm{T}}$. From $\boldsymbol{\beta}_6$ it is deduced that $M_6 = 4$. Using the observability Gramian we confirm that the rank of the observability Gramian increases up to 5 and then it remains unchanged. This confirms that there is one eigenvalue which is unobservable.

Example 6: Now we consider the same network as Fig. 6 with an additional link from node v_6 to v_1.

For distributed weight selection 1, each node v_j chooses $c_j = 0.5$. The spectrum of the *weighted adjacency matrix* P is $\sigma(P) = (1 \;\; 0.5 \;\; 0.6317 \pm 0.1370i \;\; 0.1573 \;\; 0.0793)^{\mathrm{T}}$, where complex roots exist, since the graph is directed. Node v_6 computes roots $\boldsymbol{\lambda}_6(P) = (0.6317 \pm 0.1370i \;\; 0.1573 \;\; 0.0793)^{\mathrm{T}}$. The mode corresponding to eigenvalue 0.5 is not observable by node v_6.

For distributed weight selection 2 and $\theta = (\max_j(\mathcal{D}_j^-) + 1)^{-1} = 0.25$, the spectrum of the *Perron matrix* P is

$$\sigma(P) = (1 \;\; 0.75 \;\; 0.6545 \;\; 0.5 \;\; 0.25 \;\; 0.0955)^{\mathrm{T}},$$

while node v_6 observes (and hence computes) all the modes of system (2) apart from eigenvalue 1, i.e.,

$$\boldsymbol{\lambda}_6(P) = (0.75 \;\; 0.6545 \;\; 0.5 \;\; 0.25 \;\; 0.0955)^{\mathrm{T}}.$$

The eigenvalues of the Laplacian can be computed via (6), i.e., $\boldsymbol{\lambda}_6(\mathcal{L}) = \big(1 - \boldsymbol{\lambda}_6(P)\big)/0.25$.

Example 7: In this example, we investigate how Algorithm 3 behaves in large-scale networks. Towards this end, we perform Algorithm 3 on (a) Erdős-Rényi (E-R) random graphs of size 1000 and on (b) Random Geometric Graphs (RGG) consisting of 1000 nodes uniformly distributed in a unit square (note that the RGGs are undirected graphs). We note that the nodes are able to observe only a limited part of the spectrum. Indicatively, in Figure 7 we include two histograms: the top figure shows the number of nodes that computed a given number of modes of the system for 100 E-R random graphs of size 1000 and connectivity probability $q = 0.01$; the bottom figure does the same for 100 RGG of size 1000 and connectivity radius of $q = 0.1175$. Interestingly, most of the nodes are able to discover about the same number of modes and this occurs for all networks of the same connectivity. By comparing the modes of the graph and the ones computed by each node, one finds that most of the modes computed have small deviations from the actual modes illustrating potential difficulties due to numerical conditioning. This may occur due to the fact that inverse problems on large-scale systems are ill-conditioned and mainly the dominant modes can be observed (see, e.g., [38] and references therein).

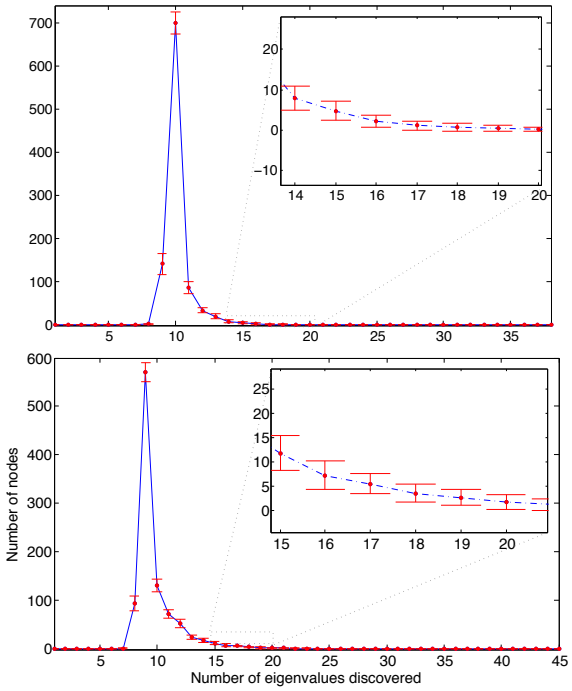

Fig. 7. Top figure: histogram of modes of operation discovered by how many nodes for 100 E-R random graphs of size 1000 and connectivity probability $q = 0.01$. Bottom figure: histogram of modes of operation discovered by how many nodes for 100 RGG of size 1000 and connectivity radius of $q = 0.1175$. Nodes are able to observe only a limited part of the spectrum.

C. Self-configuration protocols for finite-time consensus

There has been a lot of interest in designing protocols that achieve average consensus in the fastest possible time for all nodes, possibly as fast as the diameter of the underlying graph (see, for example, [20], [39]–[41]) and without the need of memory (whereas the approach used herein requires memory). The existence of such a protocol is very useful in static networks, since there will be a lot of instances where they will need to compute some parameters fast and in finite-time without requiring memory and heavy computations (that drain the battery).

However, all the aforementioned approaches assume that each node has knowledge of global parameters; for example, [20], [39] assume that each node has knowledge of the number of steps required, while [40], [41] require the whole weighted adjacency matrix at each node.

Using our approach, once a node has computed its observable modes, it can distribute its findings to the rest of the network. As a result, in at most $2|\mathcal{N}| + d_{\max}$ steps (where $d_{\max}$ is the diameter), all nodes know all k distinct eigenvalues of the weighted adjacency matrix considered. The minimum number of steps required to compute the final value by the network is equal to $k - 1$ [40]. So, the parameter assumed known in [20], [39] is found in a distributed manner here in a finite number of steps. Furthermore, k and the distinct eigenvalues can be used to compute the parameters needed in [40], [41] for the protocol design. Note that once these parameters are computed, the distributed consensus algorithm in [40], [41] can terminate in $k - 1$ steps.

VI. Discussion

All three algorithms proposed in this paper follow a similar structure. More specifically, Algorithms 1 and 2 merely differ in the initial condition (and subsequently in the final value found), while Algorithm 3 uses the kernel $\boldsymbol{\beta}_j$ differently to compute the eigenvalues of the weighted adjacency matrix. Note that the eigenvalues of the weighted adjacency matrix can be computed as a subroutine while the node computes the final value of its iterations. There is no need to run an extra iteration especially for computing the eigenvalues.

As previously mentioned, there exists a set of initial conditions $x[0] = x_0$ for which the final value cannot be found by all nodes, at least not if nodes stop storing data the first time the associated Hankel matrix loses rank. We are not able to characterize the set of Lebesgue measure zero completely; however, we provide some intuition here. Most of the cases in which the Hankel matrix loses rank and fails to compute the final value are due to the fact that the Hankel matrix of some node loses rank for the first time before information from far away has arrived, i.e., they maintain the same state value for two consecutive steps and the Hankel matrix loses rank.

In general, this problem can be alleviated if the nodes have knowledge of the network diameter or, at least, an upper bound on the network diameter (e.g., the number of nodes in the network, or even an upper bound of the number of nodes). If such knowledge is available, nodes neither stop executing iterations nor stop storing data (assuming the buffer capacity allows) until enough steps have been executed, and hence information is transferred by each node to every other node (due to the strong connectivity assumption). In our particular problem, Algorithms 1 and 2 are designed such that a node v_i participates in an iteration initiated by another node v_j as soon as information from node v_j has traversed to node v_i. At that time instant, node v_i has its first time step in this

iteration with initial state $x_i[0] = 0$. The arrived information causes the state of v_i to change to a non-zero value (given that $x_j[0]$ is non-zero) that keeps changing until it has converged, meaning that the problem of losing rank in the Hankel matrix due to the initial condition remaining the same does not apply. Algorithm 3 does not have a specific initial condition and works for any initial condition outside the Lebesgue measure zero set. If it is run as a subroutine of Algorithms 1 or 2, then the computation is guaranteed by the successful operation of these algorithms.

VII. Conclusions and Future Directions

Distributed finite-time algorithms are proposed to compute some important parameters of directed networks; namely the spectrum of weighted adjacency matrices, the left-eigenvector and the out-degree. Illustrative examples showcase our proposed algorithms.

It would be interesting to study operation of the proposed algorithms in the presence of delays on the communication links, noisy measurements of the states, dropped packets/collisions.

References

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

[2] A. Nedic, A. Olshevsky, A. Ozdaglar, and J. Tsitsiklis, "On distributed averaging algorithms and quantization effects," *IEEE Transactions on Automatic Control*, vol. 54, no. 11, pp. 2506–2517, Nov. 2009.

[3] A. Olshevsky and J. Tsitsiklis, "Convergence speed in distributed consensus and averaging," *SIAM Review*, vol. 53, no. 4, pp. 747–772, 2011.

[4] B. Krishnamachari, *Networking* W*ireless* S*ensors*. New York, USA: Cambridge University Press, 2005.

[5] Y. Yuan, G.-B. Stan, L. Shi, and J. Gonçalves, "Decentralised final value theorem for discrete-time LTI systems with application to minimal-time distributed consensus," in *Proceedings of the* 48^{th} *IEEE Conference on Decision and Control (CDC)*, Dec. 2009, pp. 2664–2669.

[6] Y. Yuan, G.-B. Stan, M. Barahona, L. Shi, and J. Gonçalves, "Decentralised minimal-time consensus," *Automatica*, vol. 49, no. 5, pp. 1227–1235, 2013.

[7] S. Bhat and D. Bernstein, "Finite-time stability of continuous autonomous systems," *SIAM Journal on Control and Optimization*, vol. 38, no. 3, pp. 751–766, 2000.

[8] A. Priolo, A. Gasparri, E. Montijano, and C. Sagues, "A decentralized algorithm for balancing a strongly connected weighted digraph," in *Proceedings of the American Control Conference*, June 2013, pp. 6547–6552.

[9] ——, "A distributed algorithm for average consensus on strongly connected weighted digraphs," *Automatica*, vol. 50, no. 3, pp. 946–951, 2014.

[10] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proceedings of the* 44^{th} *Annual IEEE Symposium on Foundations of Computer Science*, ser. FOCS '03, 2003, pp. 482–491.

[11] F. Bénézit, V. Blondel, P. Thiran, J. Tsitsiklis, and M. Vetterli, "Weighted gossip: Distributed averaging using non-doubly stochastic matrices," in *Proceedings of the IEEE International Symposium on Information Theory*, June 2010, pp. 1753–1757.

[12] A. D. Domínguez-García and C. N. Hadjicostis, "Coordination and control of distributed energy resources for provision of ancillary services," in *Proceedings of the First IEEE International Conference on Smart Grid Communications*, Oct. 2010, pp. 537–542.

[13] F. Iutzeler, P. Ciblat, and W. Hachem, "Analysis of sum-weight-like algorithms for averaging in wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 61, no. 11, pp. 2802–2814, June 2013.

[14] C. N. Hadjicostis and T. Charalambous, "Average consensus in the presence of delays in directed graph topologies," *IEEE Transactions on Automatic Control*, vol. 59, no. 3, pp. 763–768, March 2014.

[15] B. Gharesifard and J. Cortés, "Distributed strategies for generating weight-balanced and doubly stochastic digraphs," *European Journal of Control*, vol. 18, no. 6, pp. 539–557, 2012.

[16] A. Rikos, T. Charalambous, and C. Hadjicostis, "Distributed weight balancing over digraphs," *IEEE Transactions on Control of Network Systems*, vol. 1, no. 2, pp. 190–201, June 2014.

[17] T. Charalambous and C. N. Hadjicostis, "Distributed formation of balanced and bistochastic weighted digraphs in multi-agent systems," *Proceedings of the European Control Conference (ECC)*, pp. 1752–1757, July 2013.

[18] A. D. Domínguez-García and C. N. Hadjicostis, "Distributed matrix scaling and application to average consensus in directed graphs," *IEEE Transactions on Automatic Control*, vol. 58, no. 3, pp. 667–681, March 2013.

[19] R. Merris, "Laplacian matrices of graphs: a survey," *Linear Algebra and its Applications*, vol. 197–198, no. 0, pp. 143–176, 1994.

[20] A. Kibangou, "Graph Laplacian based matrix design for finite-time distributed average consensus," in *American Control Conference (ACC)*, June 2012, pp. 1901–1906.

[21] M. De Gennaro and A. Jadbabaie, "Decentralized control of connectivity for multi-agent systems," in *IEEE Conference on Decision and Control (CDC)*, Dec 2006, pp. 3628–3633.

[22] Z. Qu, C. Li, and F. Lewis, "Cooperative control with distributed gain adaptation and connectivity estimation for directed networks," *International Journal of Robust and Nonlinear Control*, vol. 24, no. 3, pp. 450–476, 2014.

[23] N. Karowski, A. C. Viana, and A. Wolisz, "Optimized asynchronous multi-channel neighbor discovery," in *Proceedings of the IEEE INFOCOM*, 2011, pp. 536–540.

[24] S. Vasudevan, M. Adler, D. Goeckel, and D. Towsley, "Efficient algorithms for neighbor discovery in wireless networks," *IEEE/ACM Transactions on Networking*, vol. 21, no. 1, pp. 69–83, Feb 2013.

[25] 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.

[26] R. Aragues, G. Shi, D. Dimarogonas, C. Sagues, and K. Johansson, "Distributed algebraic connectivity estimation for adaptive event-triggered consensus," in *American Control Conference (ACC)*, June 2012, pp. 32–37.

[27] M. Franceschelli, A. Gasparri, A. Giua, and C. Seatzu, "Decentralized Laplacian eigenvalues estimation for networked multi-agent systems," in *Proceedings of the 48th IEEE Conference on Decision and Control (CDC)*, 2009, pp. 2717–2722.

[28] M. Ji and M. Egerstedt, "Observability and estimation in distributed sensor networks," in *46th IEEE Conference on Decision and Control (CDC)*, Dec. 2007, pp. 4221–4226.

[29] M. Franceschelli, S. Martini, M. Egerstedt, A. Bicchi, and A. Giua, "Observability and controllability verification in multi-agent systems through decentralized Laplacian spectrum estimation," in *Proceedings of the 49th IEEE Conference on Decision and Control (CDC)*, 2010, pp. 5775–5780.

[30] M. Franceschelli, A. Gasparri, A. Giua, and C. Seatzu, "Decentralized estimation of Laplacian eigenvalues in multi-agent systems," *Automatica*, vol. 49, no. 4, pp. 1031–1036, 2013.

[31] A. Dominguez-Garcia and C. Hadjicostis, "Distributed strategies for average consensus in directed graphs," in *Proceedings of the* 50*th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, Dec 2011, pp. 2124–2129.

[32] S. Sundaram and C. N. Hadjicostis, "Finite-time distributed consensus in graphs with time-invariant topologies," in *Proceedings of the American Control Conference (ACC)*, July 2007, pp. 711–716.

[33] R. A. Horn and C. R. Johnson, *Matrix* A*nalysis*. Cambridge, CB2 2RU, UK: Cambridge University Press, 1985.

[34] G. B. Mathews, "On the partition of numbers," *Proceedings of the London Mathematical Society*, vol. 28, pp. 486–490, June 1897.

[35] Gurobi Optimization, Inc., "Gurobi Optimizer Reference Manual," 2014. [Online]. Available: http://www.gurobi.com

[36] S. Martello and P. Toth, "A mixture of dynamic programming and branch-and-bound for the subset-sum problem," *Management Science*, vol. 30, no. 6, pp. 765–771, 1984.

[37] S. Borbash, A. Ephremides, and M. J. McGlynn, "An asynchronous neighbor discovery algorithm for wireless sensor networks," *Ad Hoc Networks*, vol. 5, no. 7, pp. 998–1016, Sept. 2007.

[38] Z. Strakoš and J. Liesen, "On numerical stability in large scale linear algebraic computations," *Journal of Applied Mathematics and Mechanics (ZAMM)*, vol. 85, no. 5, pp. 307–325, 2005.

[39] T.-M. D. Tran and A. Y. Kibangou, "Distributed design of finite-time average consensus protocols," in *IFAC Workshop on Distributed Estimation and Control in Networked Systems*, Sep. 2013, pp. 227–233.
[40] A. Sandryhaila, S. Kar, and J. M. F. Moura, "Finite-time distributed consensus through graph filters," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2014, pp. 1085–1089.
[41] S. Safavi and U. A. Khan, "Revisiting finite-time distributed algorithms via successive nulling of eigenvalues," *IEEE Signal Processing Letters*, vol. 22, no. 1, pp. 54–57, Jan. 2015.

Themistoklis Charalambous received his BA and M.Eng in Electrical and Information Sciences from Cambridge University in 2005. He pursued his Ph.D. in the Control Laboratory, of the Engineering Department, Cambridge University in 2009. He worked as a Research Associate at Imperial College London and as a Visiting Lecturer at the Department of Electrical and Computer Engineering, University of Cyprus. He is currently working at the Automatic Control Lab of the School of Electrical Engineering at the Royal Institute of Technology (KTH) as a Research Associate. His research involves distributed coordination and control, distributed decision making, and control to various resource allocation problems in complex and networked systems.

Michael G. Rabbat received the B.Sc. degree from the University of Illinois at Urbana-Champaign in 2001, the M.Sc. degree from Rice University, Houston, TX, in 2003, and the Ph.D. from the University of Wisconsin-Madison in 2006, all in electrical engineering. He joined McGill University, Montreal, Canada, in 2007 and currently holds the rank of associate professor. He was a Visiting Researcher at Applied Signal Technology, Inc., during summer 2003. He conducts research at the intersection of statistical signal processing, networking, and machine learning. His current research focuses on distributed signal processing, network inference, and signal processing of data supported on graphs, with applications to sensor networks and network monitoring.

Dr. Rabbat co-authored the paper which received the Best Paper Award (Signal Processing and Information Theory Track) at the 2010 International Conference on Distributed Computing in Sensor Systems (DCOSS). He received Honourable Mention for Outstanding Student Paper Award at the 2006 Conference on Neural Information Processing Systems (NIPS) and a Best Student Paper Award at the 2004 ACM/IEEE International Symposium on Information Processing in Sensor Networks (IPSN). He is currently an Associate Editor for IEEE Signal Processing Letters.

Mikael Johansson received the MSc and PhD degrees in Electrical Engineering from the University of Lund, Sweden in 1994 and 1999, respectively. He held postdoctoral positions at Stanford University and UC Berkeley before joining KTH in 2002, where he now serves as full professor. His research interests are in distributed optimization, wireless networking, and control. He has published one book and over one hundred papers, several of which are ISI-highly cited. He has served on the editorial board of Automatica and on the program committee for conferences such as IEEE CDC, IEEE INFOCOM, ACM SenSys, and played a leading role in several national and international research projects in control and communications.

Christoforos N. Hadjicostis (M'99, SM'05) received the S.B. degrees in electrical engineering, computer science and engineering, and in mathematics, the M.Eng. degree in electrical engineering and computer science in 1995, and the Ph.D. degree in electrical engineering and computer science in 1999, all from the Massachusetts Institute of Technology, Cambridge. In 1999, he joined the Faculty at the University of Illinois at Urbana–Champaign, where he served as Assistant and then Associate Professor with the Department of Electrical and Computer Engineering, the Coordinated Science Laboratory, and the Information Trust Institute. Since 2007, he has been with the Department of Electrical and Computer Engineering. University of Cyprus. His research focuses on fault diagnosis and tolerance in distributed dynamic systems, error control coding, monitoring, diagnosis and control of large-scale discrete-event systems, and applications to network security, anomaly detection, energy distribution systems, medical diagnosis, biosequencing, and genetic regulatory models.