

# Distributed Finite-Time Average Consensus in Digraphs in the Presence of Time Delays

Themistoklis Charalambous, *Member, IEEE*, Ye Yuan, *Member, IEEE*, Tao Yang, *Member, IEEE*, Wei Pan, *Student Member, IEEE*, Christoforos N. Hadjicostis, *Senior Member, IEEE*, and Mikael Johansson, *Member, IEEE*

**Abstract**—Most algorithms for distributed averaging only guarantee asymptotic convergence. This paper introduces a distributed protocol that allows nodes to find the exact average of the initial values in a *finite* and *minimum* number of steps on interconnection topologies described by strongly connected directed graphs (digraphs). More specifically, under the assumption that each component has knowledge of the number of its outgoing links (i.e., the number of components to which it sends information), we show that the average value can be computed based on local observations over a finite time interval. The average can be obtained in a finite number of steps even when the information exchange is subject to delays. The proposed algorithm is the first in the literature that allows for distributed computation of the exact average in digraphs in finite time, with and without delays.

**Index Terms**—Average consensus, delays, digraphs, distributed algorithms, finite time convergence.

## I. INTRODUCTION

A DISTRIBUTED multicomponent system consists of a set of components (nodes) that can share information via connection links (edges), forming a directed interaction topology (naturally represented as a directed graph or digraph). The structure of the digraph has a critical impact on which tasks can be performed in a distributed manner, and might limit the achievable performance of distributed algorithms. The objective of a consensus problem is to have agents belong to a group that agrees upon a certain (*a priori* unknown) quantity of interest. When the agents have reached an agreement, we say that the distributed system has reached consensus.

While distributed coordination has been the subject of extensive research for a long time (see, for example, [1]–[3]), a furor

has been created during the last decade, due to the wide variety of applications and the potential to burgeon scientific advances in many different areas, ranging from computer science (e.g., parallel and distributed computation [3]), engineering (e.g., distributed optimization in sensor networks [4] and formation control of robotic networks [5]), to ecology (e.g., flocking phenomena [6], [7]) and social networks (e.g., dynamics of opinion forming [8]).

One of the most well-known consensus problems is the so-called *average consensus* problem where agents aim to reach the average of their initial values (see, for example, [9]). The initial value associated with each agent might be, for instance, a sensor measurement of some signal [10], the Bayesian belief of a decision to be taken [11], or the capacity of a distributed energy resource for the provisioning of ancillary services [12]. It has been shown in [13] that under a fixed interconnection topology, average consensus can be achieved by performing a linear iteration in a distributed fashion as long as the interconnection topology is strongly connected and balanced, while gossip algorithms [14] and convex optimization [15]–[17] require updating matrices to be doubly stochastic. In undirected interconnection topologies, weight balanced and doubly stochastic update matrices can be easily obtained, but this is significantly more challenging in directed interconnection topologies, which arise frequently in reality due to the heterogeneity of communication systems and the nonuniformity of their communication ranges. There exists a limited number of results (see, for example, [18]–[21]) that study how each agent in an unbalanced (directed) graph can ensure convergence to the exact average of the initial values. However, all existing consensus algorithms that can successfully reach the average in a directed graph only produce asymptotic convergence (i.e., exact average consensus is not reached in a finite number of steps). In addition, few of these existing algorithms (e.g., [21]) have addressed delays.

Finite-time consensus algorithms are, in general, more desirable; besides the fact that they converge in finite time, it is reported that closed-loop systems under finite-time control usually demonstrate better disturbance rejection properties [22]; this can be important in applications where the averaging operation is a first step toward a control or regularization task. Finite-time consensus with continuous-time dynamics is investigated in [23]–[25]. Finite-time consensus with discrete-time dynamics has been recently targeted in different ways. A strand of research (see, for example, [26]–[29]) is based on the factorization of the averaging matrix to orchestrate

Manuscript received May 1, 2014; revised January 1, 2015; accepted February 10, 2015. Date of publication April 28, 2015; date of current version December 14, 2015. Recommended by Associate Editor Y. Mostofi. (Corresponding author: Ye Yuan.)

T. Charalambous and M. Johansson are with ACCESS Linnaeus Center, School of Electrical Engineering, KTH-Royal Institute of Technology, Stockholm SE-100 44, Sweden (e-mail: themisc@kth.se; mikaelj@kth.se).

Y. Yuan is with the Control Group, Department of Engineering, University of Cambridge, Cambridge CB2 1PZ, U.K. (e-mail: yy311@cam.ac.uk).

T. Yang was with ACCESS Linnaeus Center, School of Electrical Engineering, KTH-Royal Institute of Technology, Stockholm SE-100 44, Sweden. He is now with the Electricity Infrastructure Group, Pacific Northwest National Laboratory, Richland, WA 99352 USA (e-mail: Tao.Yang@pnnl.gov).

W. Pan is with the Department of Bioengineering, Imperial College London, London SW7 2AZ, U.K. (e-mail: w.pan11@imperial.ac.uk).

C. N. Hadjicostis is with the Department of Electrical and Computer Engineering, University of Cyprus, Nicosia 1678, Cyprus (e-mail: chadjic@ucy.ac.cy).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TCNS.2015.2426732

update matrices at design time, so that average consensus is achieved in finite time. Another strand of research is related to the minimal polynomial of a matrix: first, [30] proposed a distributed method to compute the asymptotic final consensus value in finite time, provided that nodes have enough computing power to check rank conditions on matrices, memory to store previous states, and knowledge of an upper bound on the total number of nodes; next, [31] and [32] proposed a distributed algorithm with which any arbitrarily chosen agent can compute the asymptotic final consensus value in a minimum number of steps and without requiring any knowledge about the total number of nodes in the network. All of the aforementioned works have considered an average consensus in *undirected* graphs and assume the timely exchange of information between neighboring components.

In this paper, we first address the problem of designing a discrete-time distributed algorithm for exact average consensus in a minimum number of steps over a possibly directed communication topology. Then, we extend our proposed distributed algorithm to handle time-invariant and time-varying bounded delays. In addition, we provide an upper bound for the convergence time in the case of time-varying delays. Our work is based on the minimal polynomial of a matrix, as it is the case for [30]–[32]. However, none of the aforementioned works is able to distributively compute—using only local information at each iteration—the average consensus value in a finite number of steps in a digraph. Furthermore, none of them considered disturbances, such as delayed information.

The remainder of this paper is organized as follows. In Section II, we provide the necessary notation and background on linear algebra and graph theory. Sections III and IV present our main results in the delay-free and delayed cases, respectively, with illustrative examples. Finally, Section V presents concluding remarks and future directions.

## II. NOTATION AND PRELIMINARIES

The set of real (integer) numbers is denoted by  $\mathbb{R}$  ( $\mathbb{Z}$ ) and the set of non-negative real (integer) numbers 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 (assumed column vectors, unless otherwise stated) are denoted by small letters whereas matrices are denoted by capital letters.  $A^T$  denotes the transpose of matrix  $A$ . The  $i$ th component of a vector  $x \in \mathbb{R}^{n \times 1}$  is denoted by  $x_i$ , and the notation  $x \geq y$  implies that  $x_i \geq y_i$  for all components  $i$ . For  $A \in \mathbb{R}^{n \times n}$ ,  $a_{ij}$  denotes the entry at row  $i$  and column  $j$ . By  $\mathbb{1}$ , we denote the all-ones vector and by  $I$ , we denote the identity matrix (of appropriate dimensions). The  $n \times n$  zero matrix is denoted by  $0_{n \times n}$  and the zero vector is denoted by  $\mathbf{0}$ . We also denote by  $e_j^T = [0, \dots, 0, 1_{j\text{th}}, 0, \dots, 0] \in \mathbb{R}^{1 \times n}$  the row vector, whose single “1” entry is at the  $j$ th position. Also,  $|A|$  is the element-wise absolute value of matrix  $A$  (i.e.,  $|A| \triangleq [|A_{ij}|]$ ). A matrix whose elements are non-negative, called the non-negative matrix, is denoted by  $A \geq 0$  and a matrix whose elements are positive, called positive matrix, is denoted by  $A > 0$ .

**Lemma 1** [33]: For any non-negative matrix  $A \in \mathbb{R}_+^{n \times n}$ , the spectral radius (or the Perron-Frobenius eigenvalue),

denoted by  $\rho(A)$ , is a simple eigenvalue of  $A$  and there exists a non-negative vector  $x \in \mathbb{R}_+^{n \times 1}$ , such that  $Ax = \rho(A)x$ .

In multicomponent systems with fixed communication links, the exchange of information between components can be conveniently captured by a digraph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$  of order  $n$  ( $n \geq 2$ ), where  $\mathcal{V} = \{v_1, v_2, \dots, v_n\}$  is the set of nodes and  $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$  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  $\varepsilon_{jj} \in \mathcal{E}$  implies  $\varepsilon_{ij} \in \mathcal{E}$ . In this paper, links are not required to be bidirectional, that is, we deal with digraphs; for this reason, we use the terms “graph” and “digraph” interchangeably. Note that by convention and for notational purposes, we assume that the given graph does not include any self-loops (i.e.,  $\varepsilon_{jj} \notin \mathcal{E}$  for all  $v_j \in \mathcal{V}$ ) although each node  $v_j$  obviously has a link (access) to its own information. 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  $v_j, v_i \in \mathcal{V}$ ,  $v_j \neq v_i$ , one can find a sequence of nodes  $v_i = v_{l_1}, v_{l_2}, v_{l_3}, \dots, v_{l_t} = v_j$  such that link  $(v_{l_{k+1}}, v_{l_k}) \in \mathcal{E}$  for all  $k = 1, 2, \dots, t-1$ .

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  $\mathcal{N}_j^- = \{v_i \in \mathcal{V} | \varepsilon_{ji} \in \mathcal{E}\}$ . The cardinality of  $\mathcal{N}_j^-$ , is called the *in-degree* of  $v_j$  and is denoted by  $\mathcal{D}_j^- = |\mathcal{N}_j^-|$ . The nodes that receive information from node  $v_j$  belong to the set of out-neighbors of node  $v_j$ , denoted by  $\mathcal{N}_j^+ = \{v_l \in \mathcal{V} | \varepsilon_{lj} \in \mathcal{E}\}$ . The cardinality of  $\mathcal{N}_j^+$ , is called the *out-degree* of  $v_j$  and is denoted by  $\mathcal{D}_j^+ = |\mathcal{N}_j^+|$ .

In the type of algorithms we consider, we associate a positive weight  $p_{ji}$  for each edge  $\varepsilon_{ji} \in \mathcal{E} \cup \{(v_j, v_j) | v_j \in \mathcal{V}\}$ . The non-negative matrix  $P = [p_{ji}] \in \mathbb{R}_+^{n \times n}$  (with  $p_{ji}$  as the entry at its  $j$ th row,  $i$ th column position) is a weighted adjacency matrix (also referred to as the 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  $w_j[k] \in \mathbb{R}$  to denote the information state of node  $j$  at time step  $k$ .

Each node updates its information state  $w_j[k]$  by combining the available information received by its neighbors  $w_i[k]$  ( $v_i \in \mathcal{N}_j^-$ ) 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 paper, we assume that each node  $v_j$  can choose its self-weight and the weights on its outgoing links  $\mathcal{N}_j^+$  only, and that these weights are constant. Hence, each node updates its information state according to

$$w_j[k+1] = p_{jj}w_j[k] + \sum_{v_i \in \mathcal{N}_j^-} p_{ji}w_i[k] \quad (1)$$

for  $k \geq 0$ , where  $w_j[0] \in \mathbb{R}$  is the initial state of node  $v_j$ . If we let  $w[k] = (w_1[k] \ w_2[k] \ \dots \ w_n[k])^T$  and  $P = [p_{ji}] \in \mathbb{R}_+^{n \times n}$ , then (1) can be written in matrix form as

$$w[k+1] = Pw[k] \quad (2)$$

where  $w[0] = (w_1[0] \ w_2[0] \ \dots \ w_n[0])^T \triangleq w_0$ . We say that the nodes asymptotically reach average consensus if

$$\lim_{k \rightarrow \infty} w_j[k] = \frac{\sum_{v_i \in \mathcal{V}} w_i[0]}{n}, \quad \forall v_j \in \mathcal{V}.$$

The necessary and sufficient conditions for (2) to reach average consensus are as follows: 1)  $P$  has a simple eigenvalue at one with left eigenvector  $\mathbb{1}^T$  and right eigenvector  $\mathbb{1}$ , and 2) all other eigenvalues of  $P$  have a 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.<sup>1</sup> However, in a digraph, it is not possible to set up a doubly stochastic weight matrix without exchanging information among the nodes in the network. (See the discussion in [34].)

### III. DELAY-FREE CASE

In this section, we show that if each node in a directed graph can observe and store the evolution of its own values over a finite number of steps, then the average consensus value can be computed in finite time as follows: Each node runs two linear iterations (with identical coefficients but different initial conditions) and calculates the final value of the iterations after a finite number of successive iterations. The ratio of these final values is equal to the network-wide average of the initial values held by nodes. Hence, this distributed protocol allows each node to compute, based on its own local observations and after a minimum number of steps, the exact network-wide average.

In [18], an algorithm is suggested that solves the average consensus problem in a directed graph where each node  $v_j$  distributively sets the weights on its self link and outgoing links to be  $p_{lj} = (1/(1 + D_j^+)) \ \forall (l, j) \in \mathcal{E}$ , so that the resulting weight matrix  $P$  is column stochastic, but not necessarily row stochastic. Asymptotic average consensus is reached by using this weight matrix to run two iterations with appropriately chosen initial conditions. The algorithm is stated below for a specific choice of weights on each link that assumes that each node knows its out-degree; note, however, that the algorithm works for any set of weights that adhere to the graph structure and form a primitive column stochastic weight matrix.

**Proposition 1 [18]:** Consider a strongly connected digraph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$ . Let  $y_j[k]$  and  $x_j[k]$  (for all  $v_j \in \mathcal{V}$  and  $k = 0, 1, 2, \dots$ ) be the result of the iterations

$$y_j[k+1] = p_{jj}y_j[k] + \sum_{v_i \in \mathcal{N}_j^-} p_{ji}y_i[k] \quad (3a)$$

$$x_j[k+1] = p_{jj}x_j[k] + \sum_{v_i \in \mathcal{N}_j^-} p_{ji}x_i[k] \quad (3b)$$

where  $p_{lj} = (1/(1 + D_j^+))$  for  $v_l \in \mathcal{N}_j^+ \cup \{v_j\}$  (zeros otherwise), and the initial conditions are  $y[0] = y_0$  and  $x[0] = 1$ .

<sup>1</sup>A doubly stochastic matrix  $A$  is a square matrix of non-negative real numbers, whose rows and columns sum to 1. Matrix  $A$  is also primitive if for some  $k \in \mathbb{N}$ , matrix  $A^k$  has no entries equal to 0. A sufficient condition for matrix  $A$  to be primitive is for the matrix to be a non-negative, irreducible (matrix  $A$  is irreducible if, and only if, its associated graph  $\mathcal{G}$  is strongly connected) with at least one positive element on the main diagonal [33].

Then, the solution to the average consensus problem can be asymptotically obtained as

$$\lim_{k \rightarrow \infty} \mu_j[k] = \frac{\sum_{v_i \in \mathcal{V}} y_i[0]}{|\mathcal{V}|}, \quad \forall v_j \in \mathcal{V}$$

where  $\mu_j[k] = y_j[k]/x_j[k]$ .

**Remark 1:** Proposition 1 proposes a decentralized algorithm with which the exact average is *asymptotically* reached, even if the directed graph is not balanced.

In what follows, we propose an algorithm that is based on 1) the algorithm in Proposition 1 and 2) a distributed algorithm proposed in [31] and [32] where any arbitrarily chosen agent in an *undirected* graph can compute its asymptotic final consensus value in a finite number of steps. The proposed algorithm is based on letting each node execute (3) and store a sequence of consecutive iterations with which every node can compute  $\mu_j \triangleq \lim_{k \rightarrow \infty} \mu_j[k]$  in a *minimum* number of steps.

The result hinges on the use of the concept of the minimal polynomial associated with the linear dynamics of (3a) and (3b), in conjunction with the final value theorem. For this reason, we first provide definitions on minimal polynomials that are essential for the development of the analysis.

**Definition 1 (Minimal Polynomial of a Matrix):** The minimal polynomial associated with a matrix  $P$ , denoted by

$$q(t) = t^{D+1} + \sum_{i=0}^D \alpha_i^{(j)} t^i \quad (4)$$

is the monic polynomial of minimum degree  $D+1$  that satisfies  $q(P) = 0_{n \times n}$ .

Note that the minimal polynomial will have a degree of, at most,  $n$  (i.e.,  $D+1 \leq n$ ), 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  $[P, e_j^T]$ , denoted by  $q_j(t) = t^{M_j+1} + \sum_{i=0}^{M_j} \alpha_i^{(j)} t^i$ ,  $\alpha_i^{(j)} \in \mathbb{R}$ , is the monic polynomial of minimum degree  $M_j+1$  that satisfies  $e_j^T q_j(P) = 0$ .

Considering the iteration in (2) with weight matrix  $P$ , it is easy to show that

$$\sum_{i=0}^{M_j+1} \alpha_i^{(j)} w_j[k+i] = 0, \quad \forall k \in \mathbb{Z}_+ \quad (5)$$

where  $\alpha_{M_j+1}^{(j)} = 1$ . Let us now denote the  $z$ -transform of  $w_j[k]$  as  $W_j(z) \triangleq \mathcal{Z}(w_j[k])$ . From (5) and the time-shift property of the  $z$ -transform, it is easy to show (see [31], [32]) that

$$W_j(z) = \frac{\sum_{i=1}^{M_j+1} \alpha_i^{(j)} \sum_{\ell=0}^{i-1} w_j[\ell] z^{i-\ell}}{q_j(z)}. \quad (6)$$

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

$$p_j(z) \triangleq \frac{q_j(z)}{z-1} \triangleq \sum_{i=0}^{M_j} \beta_i^{(j)} z^i. \quad (7)$$



The application of the final value theorem [31], [32] yields

$$\phi_y(j) = \lim_{k \rightarrow \infty} y_j[k] = \lim_{z \rightarrow 1} (z-1)Y_j(z) = \frac{y_{M_j}^T \beta_j}{\mathbb{1}^T \beta_j} \quad (8a)$$

$$\phi_x(j) = \lim_{k \rightarrow \infty} x_j[k] = \lim_{z \rightarrow 1} (z-1)X_j(z) = \frac{x_{M_j}^T \beta_j}{\mathbb{1}^T \beta_j} \quad (8b)$$

where

$$y_{M_j}^T = (y_j[0], y_j[1], \dots, y_j[M_j])$$

$$x_{M_j}^T = (x_j[0], x_j[1], \dots, x_j[M_j])$$

and  $\beta_j$  is the vector of coefficients of the polynomial  $p_j(z)$ , defined in (7), that is,  $\beta_j^T = (\beta_0^{(j)}, \dots, \beta_{M_j}^{(j)})$ . The next question is how one can obtain the coefficient vector  $\beta_j$  in the computation of final values, for example, (8a) and (8b). Consider the vectors of  $2k+1$  successive discrete-time values at node  $v_j$ , given by

$$y_{2k}^T = (y_j[0], y_j[1], \dots, y_j[2k])$$

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

for the two iterations  $y_j[k]$  and  $x_j[k]$  at node  $v_j$  [as given in iterations (3a) and (3b)], respectively. Let us define their associated Hankel matrices as

$$\Gamma\{y_{2k}^T\} \triangleq \begin{bmatrix} y_j[0] & y_j[1] & \dots & y_j[k] \\ y_j[1] & y_j[2] & \dots & y_j[k+1] \\ \vdots & \vdots & \ddots & \vdots \\ y_j[k] & y_j[k+1] & \dots & y_j[2k] \end{bmatrix}$$

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

We also consider the vector of differences between successive values of  $y_j[k]$  and  $x_j[k]$

$$\bar{y}_{2k}^T = (y_j[1] - y_j[0], \dots, y_j[2k+1] - y_j[2k])$$

$$\bar{x}_{2k}^T = (x_j[1] - x_j[0], \dots, x_j[2k+1] - x_j[2k]).$$

It has been shown in [32] that  $\beta_j$  can be computed as the kernel of the first defective Hankel matrices  $\Gamma\{\bar{y}_{2k}^T\}$  and  $\Gamma\{\bar{x}_{2k}^T\}$  for arbitrary initial conditions  $y_0$  and  $x_0$  except a set of initial conditions with *Lebesgue measure zero*.<sup>2</sup>

Next, we provide our first main result, where it is stated that the exact average  $\mu$  can be distributively obtained in a minimum number of steps in strongly connected digraphs.

**Theorem 1:** Consider a strongly connected graph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$ . Let  $y_j[k]$  and  $x_j[k]$  (for all  $v_j \in \mathcal{V}$  and  $k = 0, 1, 2, \dots$ ) be the result of the iterations (3a) and (3b), where  $P = [p_{ji}] \in \mathbb{R}_+^{n \times n}$

<sup>2</sup>The Lebesgue measure is a way of assigning a measure to subsets of  $n$ -dimensional Euclidean space. A subset of  $\mathbb{R}^n$  is said to be Lebesgue measure zero if, for every  $\epsilon > 0$ , it can be covered with countably many products of  $n$  intervals whose total volume is, at most,  $\epsilon$ . All countable sets and all subsets of  $\mathbb{R}^n$ , whose dimension is smaller than  $n$ , have Lebesgue measure zero in  $\mathbb{R}^n$ . We elaborate more on this issue later on, in Appendix.

is a set of weights that adhere to the graph structure and form a primitive column stochastic weight matrix. Then, the solution to the average consensus can be distributively obtained in minimum number of steps at each node  $v_j$ , by computing

$$\mu_j \triangleq \lim_{k \rightarrow \infty} \frac{y_j[k]}{x_j[k]} = \frac{\phi_y(j)}{\phi_x(j)} = \frac{y_{M_j}^T \beta_j}{x_{M_j}^T \beta_j} \quad (9)$$

where  $\phi_y(j)$  and  $\phi_x(j)$  are given, respectively, by (8a) and (8b), and  $\beta_j$  is the vector of coefficients, as defined in (7).

**Proof:** The consensus value of node  $v_j$  for each of the iterations (3a) (with initial condition  $y[0] = y_0$ ) and (3b) (with initial condition  $x[0] = \mathbb{1}$ ) is found by (8a)–(8b). Note that the vector  $\beta_j$  does not depend on the initial conditions and, hence, it is the same for each node for every initial condition (except for initial conditions in a set of Lebesgue measure zero). Hence

$$\lim_{k \rightarrow \infty} \frac{y_j[k]}{x_j[k]} = \frac{\phi_y(j)}{\phi_x(j)} = \frac{y_{M_j}^T \beta_j}{x_{M_j}^T \beta_j}.$$

But from ratio consensus and Proposition 1, we already know that  $\lim_{k \rightarrow \infty} \mu_j[k] = \lim_{k \rightarrow \infty} y_j[k]/x_j[k] = \sum_{v_j \in \mathcal{V}} y_0(j)/|\mathcal{V}|$ . Hence  $y_{M_j}^T \beta_j / x_{M_j}^T \beta_j = \sum_{v_j \in \mathcal{V}} y_0(j)/|\mathcal{V}|$ . ■

**Remark 2:** Theorem 1 states that the average consensus value in a strongly connected digraph can be computed as the ratio of the final values obtained for each iteration (3a) with initial condition  $y[0] = y_0$  and iteration (3b) with initial condition  $x[0] = \mathbb{1}$ .

**Remark 3:** It has been shown in [31] that the number of steps required for predicting  $y$  and  $x$  are less than  $2n$ , where  $n$  is the number of nodes in the network. Thus, an upper bound on the network size immediately yields an upper bound on the storage requirements and the convergence time of the algorithm. However, in many cases, we do not need to store these many values: as soon as the square Hankel matrices lose rank,<sup>3</sup> the nodes can stop storing information. Throughout this paper, we assume that nodes can store at least as many values as necessary to obtain the resulting defective matrix.

We now introduce an algorithm, herein called *Algorithm 1*, in which the nodes distributively compute the exact average of the initial values in a finite number of steps.

---

**Algorithm 1** Decentralized minimum-time average consensus in digraphs

---

**Input:** A strongly connected digraph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$  with  $n = |\mathcal{V}|$  nodes and  $m = |\mathcal{E}|$  edges.

**Data:** Successive observations for  $y_j[k]$  and  $x_j[k]$ ,  $\forall v_j \in \mathcal{V}$ ,  $k = 0, 1, 2, \dots$ , using iterations (3a) and (3b), with initial conditions  $y[0] = y_0$  and  $x[0] = \mathbb{1}$ , respectively.

**Step 1:** For  $k = 0, 1, 2, \dots$ , each node  $v_j \in \mathcal{V}$  runs the ratio consensus algorithm (3) and stores the vectors of differences  $\bar{y}_{M_j}^T$  and  $\bar{x}_{M_j}^T$  between successive values of  $y_j[k]$  and  $x_j[k]$ , respectively.

<sup>3</sup>The special structure of Hankel matrices allows for efficient methods to compute their rank. For example, in [35], a modular method is proposed with complexity  $\mathcal{O}(r^2)$ , where  $r$  is the order of the matrix.

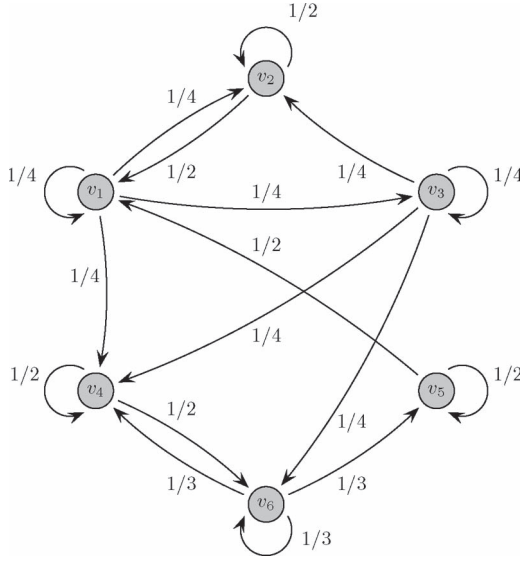


Fig. 1. Digraph consisting of six nodes.

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

**Step 3:** The kernel  $\beta_j = (\beta_0, \dots, \beta_{M_j-1}, 1)^T$  of the first defective matrix gives the values  $\phi_y$  and  $\phi_x$ , via (8a) and (8b), respectively.

**Step 4:** The average consensus value is computed as

$$\mu_j = \frac{\phi_y(j)}{\phi_x(j)} = \frac{y_{M_j}^T \beta_j}{x_{M_j}^T \beta_j}.$$

*Remark 4:* Note that the nodes cannot necessarily stop iterating as soon as they compute the exact average, because other nodes may require more steps to compute their Hankel matrices. If the nodes had knowledge of the number of nodes  $n$  in the network (or an upper bound  $n'$ ), then each node could stop iterations after  $2n$  ( $2n'$ ) steps from the time it started the iterations.

*Example 1:* Consider the directed network shown in Fig. 1 where each node  $v_j$  chooses its weight and the weight of its outgoing links to be  $(1 + \mathcal{D}_j^+)^{-1}$  (such that the sum of all weights assigned by each node  $v_j$  is equal to 1). When each node updates its information state  $w_j[k]$  using (1), the information state for the entire network is given by  $w[k+1] = Pw[k]$  [as in (2)]. Each node  $v_j \in \mathcal{V}$  has an initial value  $y_0(j)$  and runs ratio consensus, that is, the iteration in (3a) with initial value  $y_0(j)$ , and the iteration in (3b) with initial value  $x_0(j) = 1$ , that is, we use the update formula (3) to simultaneously run two iterations with initial conditions  $y[0] = [-1 \ 0 \ 1 \ 2 \ 3 \ 4]^T$  and  $x[0] = \mathbf{1}$ . Since the update matrix is not doubly stochastic, the iteration (3) for this network does converge, but not necessarily to the average (as shown in Fig. 2 for the case when the initial condition is  $y[0]$ ). The final consensus vectors  $\phi_y$  and  $\phi_x$ , for initial conditions  $y[0]$  and  $x[0]$ , respectively, are given by

$$\phi_y = [1.6119 \ 1.0746 \ 0.5373 \ 2.4179 \ 1.3433 \ 2.0149]^T$$

$$\phi_x = [1.0746 \ 0.7164 \ 0.3582 \ 1.6119 \ 0.8955 \ 1.3433]^T.$$

Authorized licensed use limited to: Rutgers University. Downloaded on September 18, 2023 at 16:54:56 UTC from IEEE Xplore. Restrictions apply.

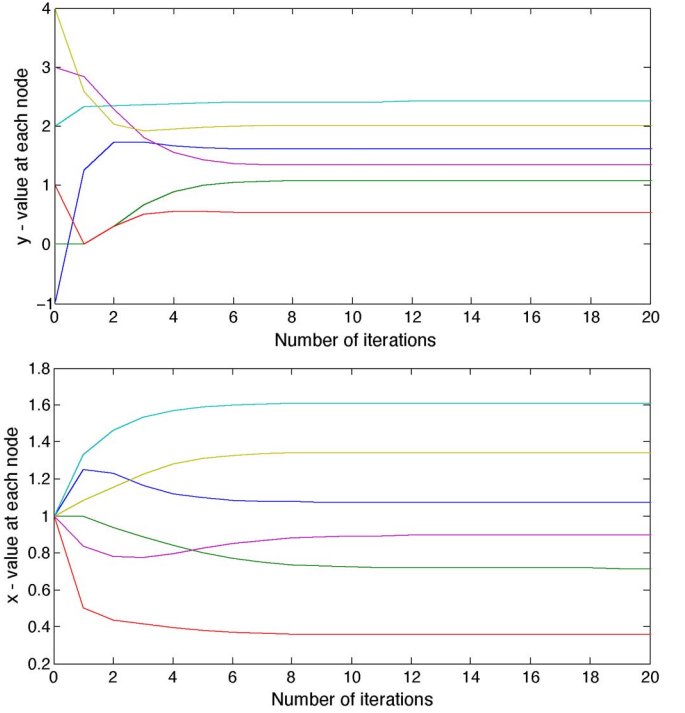


Fig. 2. Iteration (3a) with initial condition  $y[0] = [-1 \ 0 \ 1 \ 2 \ 3 \ 4]^T$  (top) and iteration (3b) with initial condition  $x[0] = \mathbf{1}$  (bottom), for the network in Fig. 1, do not converge to the average (the average in this case is 1.5).

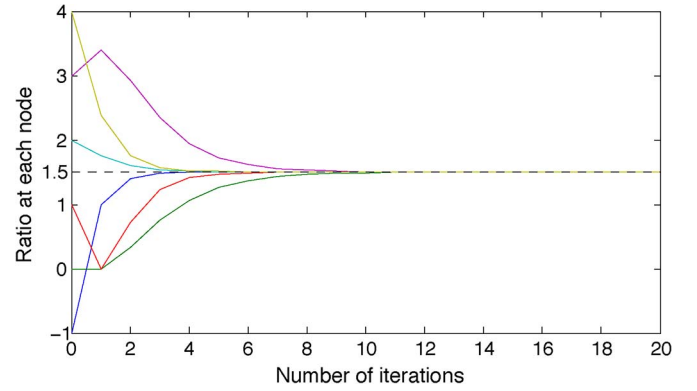


Fig. 3. By running two iterations  $y[k]$  and  $x[k]$  as in (3a)–(3b) (using the weight matrix  $P$  and initial conditions  $y[0] \triangleq y_0$  and  $x[0] = \mathbf{1}$ , respectively), average consensus is asymptotically reached for the ratio  $y_j[k]/x_j[k]$ .

Then, each node can compute the exact average as  $\mu_j = \phi_y(j)/\phi_x(j)$ . For example, for node  $v_1$ , we have

$$\mu_1 = \frac{\phi_y(1)}{\phi_x(1)} = \frac{1.6119}{1.0746} = 1.5.$$

The exact average for  $v_1$  is computed in 12 steps (i.e.,  $2(M_1 + 1) = 2 \times 6 = 12$  steps). See [32] for more details on how to compute the minimum number of steps required.

The exact average is also justified by running the ratio consensus algorithm by showing asymptotic convergence to the exact average (see Fig. 3).

From the simulations, we observe that the ratio consensus algorithm after 12 steps has almost converged; more specifically, the values of the ratio at time step 12 are given by

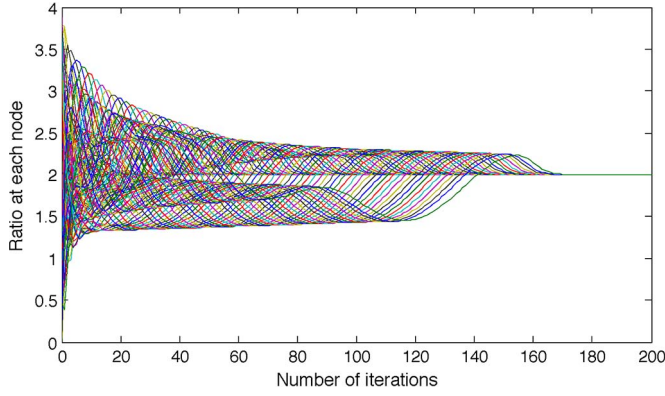


Fig. 4. Average consensus is asymptotically reached for the ratio  $y_j[k]/x_j[k]$  of each node  $v_j$  for a Leslie matrix of size 100.

$z[12] = [1.4998 \ 1.4946 \ 1.4995 \ 1.5001 \ 1.5043 \ 1.5002]^T$ , illustrating that the minimum time needed for decentralized computation of the exact average is comparable with the asymptotic time needed for approximate convergence by the ratio consensus algorithm.

*Example 2:* Next, we consider an example of a larger network, for which asymptotic convergence to the average takes a considerable amount of time. More specifically, we consider a Leslie (a discrete, age-structured model of population growth) matrix consisting of 100 nodes. The adjacency matrix  $P \in \mathbb{R}_+^{100 \times 100}$  is as follows:

$$P = \begin{pmatrix} 1/2 & 1/3 & 1/3 & 1/3 & \cdots & 1/3 & 1/2 \\ 1/2 & 1/3 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1/3 & 1/3 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1/3 & 1/3 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1/3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1/3 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 1/3 & 1/2 \end{pmatrix}.$$

Due to the structure of the network, the asymptotic convergence time is considerable (see Fig. 4). By running our algorithm, each node can compute the exact average value in a minimum number of steps; for example, node  $v_{100}$  computes the average in 34 steps (i.e.,  $2(M_{100} + 1) = 2 \times 17 = 34$  steps), while the (asymptotic) ratio-consensus algorithm seems to need more than 160 steps for the error to be less than 0.1.

#### IV. DELAYED CASE

In this section, we address the finite-time average consensus problem over directed graphs in the presence of bounded delays in the communication links, arising mainly due to propagation and computational delays. We now characterize how time delays affect the number of steps required to compute the final value at each node in the network. Toward this end, we postulate an asynchronous operation of *Algorithm 1*, where each agent updates its own value by using delayed information from neighboring nodes.

We use the integer  $\tau_{ji}[k] \geq 0$  to represent the delay of a message sent from node  $v_i$  to node  $v_j$  at time instant  $k$ . We require that  $0 \leq \tau_{ji}[k] \leq \bar{\tau}_{ji} \leq \bar{\tau}$  for all  $k \geq 0$  for some finite

$\bar{\tau} = \max\{\bar{\tau}_{ji}\}$ ,  $\bar{\tau} \in \mathbb{Z}_+$ . We make the reasonable assumption that  $\tau_{jj}[k] = 0$ ,  $\forall v_j \in \mathcal{V}$ , at all time instances  $k$  (i.e., the own value of a node is always available without delay). A protocol is employed where each node updates its information state  $w_j[k+1]$  by combining the available (possibly delayed) information received by its neighbors  $w_i[s]$  ( $s \in \mathbb{Z}_+$ ,  $s \leq k$ ,  $v_i \in \mathcal{N}_j^-$ ) using constant positive weights  $p_{ji}$ . Proposition 2 presents the decentralized algorithm proposed in [12] with which the exact average is *asymptotically* reached in the presence of bounded delays, even if the digraph is not balanced.

*Proposition 2 [12]:* Consider a strongly connected digraph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$ , where each node  $v_j \in \mathcal{V}$  has some initial value  $y_0(j)$ . Let  $y_j[k]$  and  $x_j[k]$  (for all  $v_j \in \mathcal{V}$  and  $k = 0, 1, 2, \dots$ ) be the result of the iterations

$$y_j[k+1] = p_{jj}y_j[k] + \sum_{v_i \in \mathcal{N}_j^-} \sum_{r=0}^{\bar{\tau}} p_{ji}y_i[k-r]I_{k-r,ji}[r] \quad (10a)$$

$$x_j[k+1] = p_{jj}x_j[k] + \sum_{v_i \in \mathcal{N}_j^-} \sum_{r=0}^{\bar{\tau}} p_{ji}x_i[k-r]I_{k-r,ji}[r] \quad (10b)$$

where  $y[0] = (y_0(1) \ y_0(2) \ \dots \ y_0(|\mathcal{V}|))^T \triangleq y_0$  and  $x[0] = \mathbb{1}$ , and  $I_{k,ji}$  is an indicator function that captures the bounded delay  $\tau_{ji}[k] \leq \bar{\tau}$  ( $\bar{\tau} < \infty$ ) on link  $(v_j, v_i)$  at iteration  $k$ , defined as

$$I_{k,ji}[\tau] = \begin{cases} 1, & \text{if } \tau_{ji}[k] = \tau, \\ 0, & \text{otherwise.} \end{cases} \quad (11)$$

(Note that  $y[k]$  and  $x[k]$  are taken to be zero for negative  $k$ .) Then, we have  $\lim_{k \rightarrow \infty} \mu_j[k] = \sum_{v_i \in \mathcal{V}} y_0(i)/|\mathcal{V}|$ ,  $\forall v_j \in \mathcal{V}$ , where  $\mu_j[k] = y_j[k]/x_j[k]$ .

An augmented graph representation is employed in [12] by adding extra “virtual” nodes and using them to model the delays. The maximum number of “virtual” nodes for each original node is bounded by the maximum delay  $\bar{\tau}$ . In particular, for each node  $v_j \in \mathcal{V}$ , we introduce  $\bar{\tau}$  “virtual” nodes  $v_j^{(1)}, v_j^{(2)}, \dots, v_j^{(\bar{\tau})}$  (node  $v_j^{(d)}$  holds information that is destined to arrive at node  $v_j$  after  $d$  steps). The augmented digraph has  $(\bar{\tau} + 1)|\mathcal{V}|$  nodes and  $(1 + 2\bar{\tau})|\mathcal{E}|$  edges (an example is given in Fig. 5). In the general case, in a network of  $n = |\mathcal{V}|$  nodes, we introduce  $\bar{\tau}n$  nodes (for a total of  $(\bar{\tau} + 1)n$  nodes) so that  $\psi[k+1] = \Xi[k]\psi[k]$ , where

$$\Xi[k] \triangleq \begin{pmatrix} P_0[k] & I_{n \times n} & 0 & \cdots & 0 \\ P_1[k] & 0 & I_{n \times n} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ P_{\bar{\tau}-1}[k] & 0 & 0 & \cdots & I_{n \times n} \\ P_{\bar{\tau}}[k] & 0 & 0 & \cdots & 0 \end{pmatrix} \quad (12)$$

with  $\psi[k] = (w^T[k] \ w^{(1)}[k] \ \dots \ w^{(\bar{\tau})}[k])^T$  and  $w^{(r)}[k] = (w_1^{(r)}[k] \ \dots \ w_n^{(r)}[k])$ ,  $r = 1, 2, \dots, \bar{\tau}$ . Note that  $P_0[k]$ ,  $P_1[k]$ ,  $\dots$ ,  $P_{\bar{\tau}}[k]$  are appropriately defined non-negative matrices that depend on the link delays that are experienced by messages sent

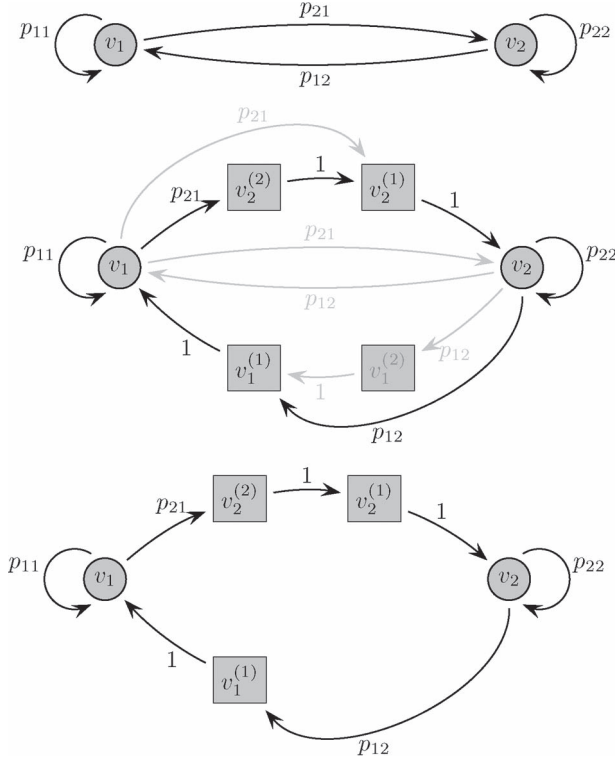


Fig. 5. Simple example with two nodes. When the nodes experience no delays, the graph is as shown at the top figure. When the nodes experience time-varying delays, then at each time instant  $k$ , there will be a path from  $v_1$  to  $v_2$ , either directly or through the “virtual” nodes of the augmented graph, as shown in the middle figure. For time-invariant graphs, the augmented figure remains fixed (e.g., bottom figure for  $\tau_{21} = 2$  and  $\tau_{12} = 1$ ) for all  $k$ .

at time  $k$ . Specifically,  $P_r[k]$  is a matrix associated only with the links of the graph for which the message is being delayed by  $r$  steps at time step  $k$ , and satisfies

$$P_r[k](j, i) = \begin{cases} P(j, i), & \text{if } \tau_{ji}[k] = r, (j, i) \in \mathcal{E}, \\ 0, & \text{otherwise.} \end{cases}$$

Note that for each  $(j, i) \in \mathcal{E}$ , only one of  $P_0[k](j, i), P_1[k](j, i), \dots, P_{\bar{\tau}}[k](j, i)$  is nonzero and is equal to  $P(j, i)$ . It is easily deduced that  $\Xi[\cdot]$  is a column-stochastic, non-negative matrix and, hence, its spectral radius is equal to 1 [33].

In what follows, we consider two cases:

- 1) In the first case, the delay is assumed to be time-invariant. We show that the algorithm for computing the ratio consensus value is actually the same as Algorithm 1, however, requiring a larger number of observations.
- 2) In the second case, the delay is time-varying. By assuming knowledge of an upper bound of the time-varying delays, we can show that the algorithm is the same again as Algorithm 1.

#### A. Time-Invariant Delay

When the delay is time-invariant  $\Xi[k] = \Xi$  and the state-space equation is given by

$$\psi[k+1] = \Xi\psi[k], \quad (13a)$$

where

$$\Xi \triangleq \begin{pmatrix} P_0 & I_{n \times n} & 0 & \cdots & 0 \\ P_1 & 0 & I_{n \times n} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ P_{\bar{\tau}-1} & 0 & 0 & \cdots & I_{n \times n} \\ P_{\bar{\tau}} & 0 & 0 & \cdots & 0 \end{pmatrix}. \quad (13b)$$

Let  $g(z)$  be the minimal polynomial of  $\Xi$ . As proven in [33], the minimal polynomial of a matrix divides its characteristic polynomial. Therefore, by direct calculation

$$\det(zI - \Xi) = \det(zI - P_0 z^{\bar{\tau}} - \dots - P_{\bar{\tau}-1} z - P_{\bar{\tau}}).$$

As a consequence, the degree of  $g(z)$  is, at most,  $(\bar{\tau} + 1)n$  (this can be clearly seen by the size of the matrix describing the augmented graph), where  $n$  is the number of nodes in the network. Then, we have the following corollary, which follows from the fact that the maximum number of steps required from a node to compute its final value is  $2n$ .

**Corollary 1:** Consider the system in (13). Any arbitrarily chosen component  $v_j$  can compute its corresponding final value (provided that the initial value does not belong to the Lebesgue measure zero set) in finite time using, at most,  $2(\bar{\tau} + 1)n$  successive values of its own state.

**Remark 5:** From Corollary 1, it can be deduced that the algorithm for computing the average value in the presence of time-invariant delays is actually the same as Algorithm 1, however, requiring a larger number of observations.

**Remark 6:** In the time-invariant case, the augmented graph is fixed and, hence, apart from the upper bound given in Corollary 1, we can find the exact number of steps required. For example, consider the network of two agents exchanging information as shown in Fig. 5. Note that the weights  $p_{11}, p_{12}, p_{21}$ , and  $p_{22}$  are all strictly positive, and satisfy  $p_{11} + p_{21} = 1$  and  $p_{22} + p_{12} = 1$ . Suppose the agents experience delays that are bounded by 2 ( $\bar{\tau} = 2$ ). Therefore, two extra “virtual” nodes will be added for each node (see Fig. 5 middle figure), depicting the states at which the delayed messages reside before reaching their destination. However, if a node does not experience the maximum delay, then some of the “virtual” nodes (and, therefore, links) can be removed.

In the case of two nodes only, the graph representation of the network for which node  $v_2$  sends information to node  $v_1$  with delay  $\tau_{12} = 1$ , while node  $v_1$  sends information to node  $v_2$  with delay  $\tau_{21} = 2$  is given in Fig. 5 (bottom figure). Therefore, the degree of  $g(z)$  is less than  $(\bar{\tau} + 1)n$ .

**Example 3:** Consider again the network in Fig. 1, for a delay profile that is given by

$$\tau = \begin{pmatrix} 0 & 4 & - & - & 1 & - \\ 5 & 0 & 0 & - & - & - \\ 1 & - & 0 & - & - & - \\ 5 & - & 1 & 0 & - & 4 \\ - & - & - & - & 0 & 0 \\ - & - & 1 & 1 & - & 0 \end{pmatrix}$$

where  $\tau_{ji} \triangleq \tau(j, i)$  is the delay with which node  $v_i$  sends information to  $v_j$ . A “-” is inserted when there does not exist



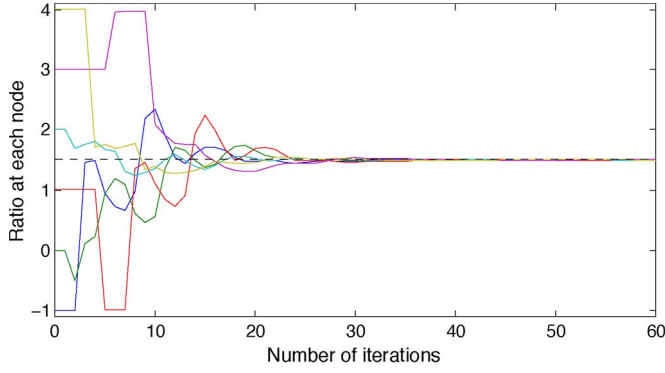


Fig. 6. Ratio  $y_j[k]/x_j[k]$  of each node  $v_j$  converges to the exact average.

a directed link between two nodes. Fig. 6 shows the evolution of the ratio at each node.

The final consensus vectors  $\phi_y$  and  $\phi_x$ , for initial conditions  $y[0]$  and  $x[0]$ , respectively, are calculated using (8a)–(8b) and are given by

$$\phi_y = [0.7105 \ 0.4737 \ 0.2368 \ 1.0658 \ 0.5921 \ 0.8882]^T$$

$$\phi_x = [0.4737 \ 0.3158 \ 0.1579 \ 0.7105 \ 0.3947 \ 0.5921]^T.$$

Each node can compute the exact average  $\mu_j = \phi_y(j)/\phi_x(j)$ , for example, for node  $v_1$ ,  $\mu_1 = \phi_y(1)/\phi_x(1) = 0.7105/0.4737 = 1.5$ . For this example, node  $v_1$  computes the exact average value in 40 steps.

### B. Time-Varying Delay

It can be observed that the iterations (10a)–(10b) themselves do not necessarily converge in the presence of time-varying delays (see Fig. 7, for the network in Fig. 1). However, the ratio of the iterations (10a)–(10b) does converge to the exact average (see Fig. 8). Thus, if one attempts to apply Algorithm 1 in this case for iterations (10a)–(10b), the method will fail, since neither iteration converges to a final value. In addition, if someone attempts to apply the final value approach [32] directly to the ratio value  $\mu_j[k]$ , even though the ratio itself converges, the approach initially fails to converge to a final value due to the nonlinearity of the ratio. However, when the value of the ratio is very close to the equilibrium point (exact average), the system can then be well represented by a linear approximation and, therefore, the approach converges to an approximate value.

Here, we use an alternative approach: if it is possible for each node  $v_j$  to know an upper bound  $\tau_{ji}$  on the delay  $\tau_{ji}[k]$  (i.e.,  $\tau_{ji}[k] \leq \tau_{ji}$ ) for all the incoming links from the in-neighboring nodes  $v_i \in \mathcal{N}_j^-$ , then by having the nodes update their value  $\tau_{ij} + 1$  steps after  $k$  using iterations (10a)–(10b), it can be easily deduced that the problem reduces to the case of time-invariant delays, studied in Section IV-A. In many cases, it is not possible to know an upper bound on the delay of each incoming link of the in-neighbors  $v_i \in \mathcal{N}_j^-$ , but instead an upper bound  $\tau_j$  of all the in-neighboring links or a global upper bound  $\bar{\tau}$  may be available. Both of these cases constitute special cases of the problem with time-invariant delays. When

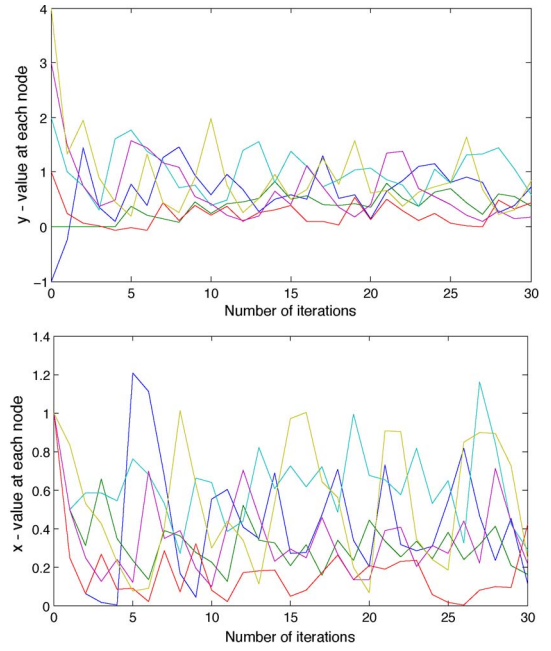


Fig. 7. Iterations (10a) with initial condition  $y[0] = [-1 \ 0 \ 1 \ 2 \ 3 \ 4]^T$  (above) and iteration (10b) with initial condition  $x[0] = \mathbb{1}$  (below), for the network in Fig. 1 do not converge.

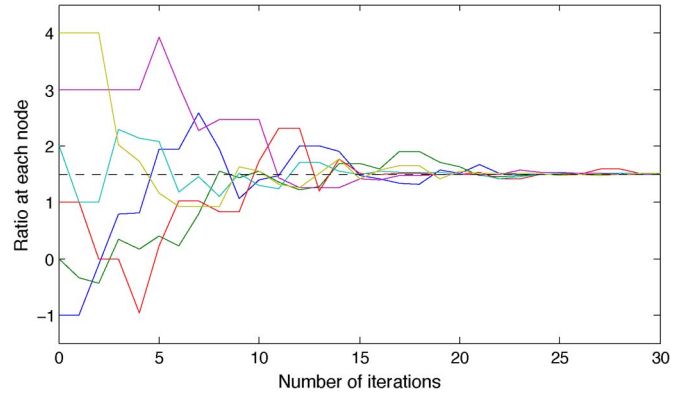


Fig. 8. Ratio  $y_j[k]/x_j[k]$  of each node  $v_j$  of the iterations (10a)–(10b) converges to the exact average.

the nodes have knowledge of a global upper bound  $\bar{\tau}$ , they can update their states at time instants  $k + \bar{\tau}$  for time instants  $k$ ,  $k = 0, 1, 2, \dots$  (i.e., once all (delayed) packets for time instant  $k$  have arrived) and use the iteration (13) (for each initial value). This is illustrated for the network in Fig. 1, where nodes update their states at time instants  $k + \bar{\tau}$  steps (see Fig. 9); the ratio of the iteration converges to the exact average (see Fig. 10).

This is stated in Theorem 2.

**Theorem 2:** Consider a strongly connected graph  $\mathcal{G}(\mathcal{V}, \mathcal{E}, P)$ , 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. Let  $y_j[k]$  and  $x_j[k]$  be the result of the iterations (10a)–(10b) for all  $v_j \in \mathcal{V}$ . By choosing to update at time instants  $h = k + \bar{\tau}$  for time instants  $k$ ,  $k = 0, 1, 2, \dots$  (i.e., once all (delayed) packets for time instant  $k$  have arrived), and by letting  $\hat{y}_j[h]$  and  $\hat{x}_j[h]$  be the result of the



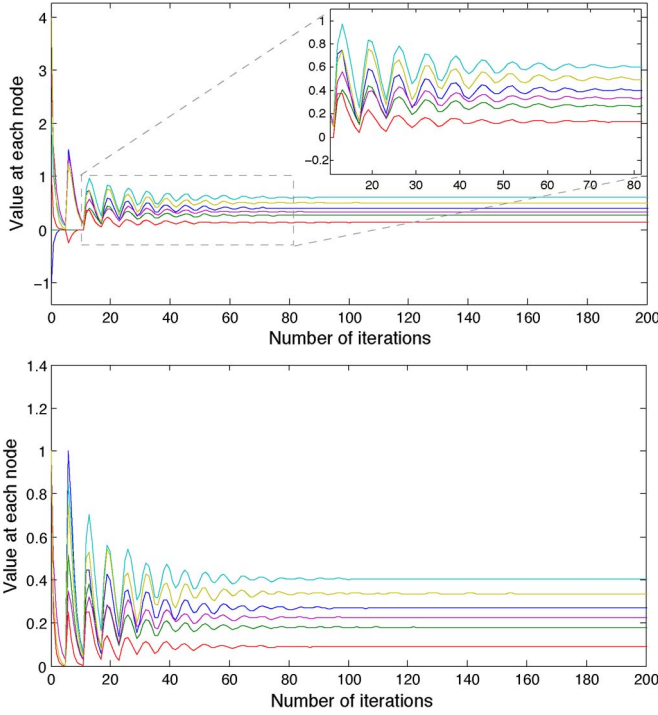


Fig. 9. Iterations (10a) with initial condition  $y[0] = [-1 \ 0 \ 1 \ 2 \ 3 \ 4]^T$  (above) and iteration (10b) with initial condition  $x[0] = \mathbb{1}$  (below) with maximum delay  $\bar{\tau} = 5$ , for the network in Fig. 1 do converge if the updates of the states  $\hat{y}_j[h]$  and  $\hat{x}_j[h]$  occur every step after the first  $\bar{\tau}$  steps (using iterations (3a)–(3b) with  $h = k + \bar{\tau}$ ,  $k = 0, 1, 2, \dots$ ).

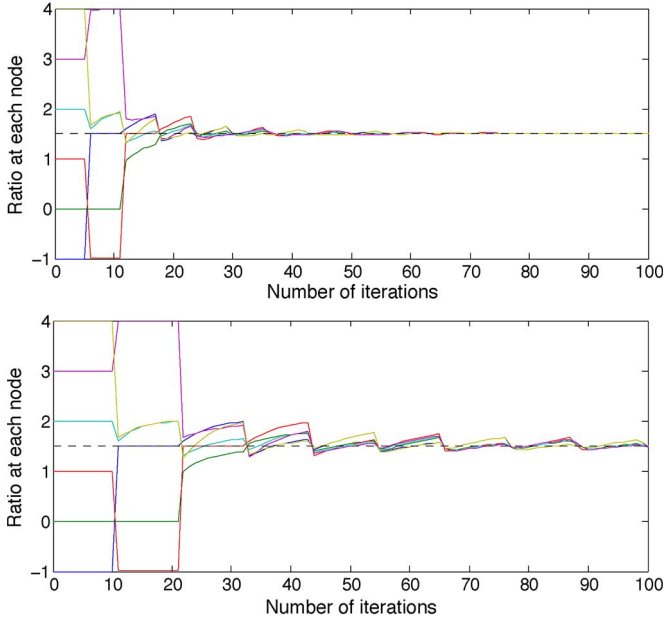


Fig. 10. Ratio  $y_j[k]/x_j[k]$  of each node  $v_j$  of the iterations (10a)–(10b) when updated every  $h = k + \bar{\tau}$  steps,  $k = 0, 1, 2, \dots$ , (a) for  $\bar{\tau} = 5$ , and (b) when the assumed maximum delay  $\bar{\tau}$  is considered as double (i.e.,  $\bar{\tau} = 2\bar{\tau} = 10$ ) the steps, due to an overestimated maximum delay. The ratio converges for both scenarios to the exact average, but with different convergence rates.

iterations (10a)–(10b), the solution to the average consensus is the same, i.e.,

$$\mu_j = \lim_{k \rightarrow \infty} \frac{y_j[k]}{x_j[k]} = \lim_{h \rightarrow \infty} \frac{\hat{y}_j[h]}{\hat{x}_j[h]}. \quad (14)$$

Hence, the solution to the average consensus can be distributively obtained in finite time at each node  $v_j$ , by computing

$$\mu_j \triangleq \lim_{h \rightarrow \infty} \frac{\hat{y}_j[h]}{\hat{x}_j[h]} = \frac{\phi_y(j)}{\phi_x(j)} = \frac{\hat{y}_{M_j}^T \beta_j}{\hat{x}_{M_j}^T \beta_j} \quad (15)$$

where  $\phi_y(j)$  and  $\phi_x(j)$  are given by (8a) and (8b), respectively, and  $\beta_j$  is the vector of coefficients defined in (7).

*Proof:* Each node updates at time instant  $k + \bar{\tau}$  for time instants  $k$ ,  $k = 0, 1, 2, \dots$ , time by which all of the delayed packets for time instant  $k$  have arrived. As a result, this is equivalent to the time-invariant delay case (13) where all the delays are set to the maximum delay, that is,  $P_j = 0_{n \times n}$  for all  $j = 1, 2, \dots, \bar{\tau} - 1$ . Then, the proof proceeds similarly to that of Theorem 1.

The following algorithm, herein called *Algorithm 2*, provides a finite-time distributed algorithm for computing the exact average when delays in the network are arbitrary and time-varying but bounded, and all of the nodes have knowledge of the upper bound of the delay.

---

**Algorithm 2** Decentralized finite-time average consensus in digraphs with bounded time-varying delays

---

**Input:** A strongly connected digraph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$  with  $n = |\mathcal{V}|$  nodes and  $m = |\mathcal{E}|$  edges.

**Data:** Observations for  $\hat{y}_j[h]$  and  $\hat{x}_j[h]$ ,  $\forall v_j \in \mathcal{V}$ ,  $h = k + \bar{\tau}$ ,  $k = 0, 1, 2, \dots$ , using iterations (3a)–(3b), with initial conditions  $y[0] = y_0$  and  $x[0] = \mathbb{1}$ , respectively.

**Step 1:** For  $h = k + \bar{\tau}$ ,  $k = 0, 1, 2, \dots$ , each node  $v_j \in \mathcal{V}$  runs the ratio consensus algorithm (10) for fixed delays and stores the vectors of differences  $\bar{y}_{M_j}^T$  and  $\bar{x}_{M_j}^T$  between successive values of  $\hat{y}_j[h]$  and  $\hat{x}_j[h]$ , respectively.

**Step 2:** Increase the dimension  $h$  of the square Hankel matrices  $\Gamma\{\bar{y}_{M_j}^T\}$  and  $\Gamma\{\bar{x}_{M_j}^T\}$  for each iteration, until they both lose rank; store their first defective matrix.

**Step 3:** The kernel  $\beta_j = (\beta_0, \dots, \beta_{M_j-1}, 1)^T$  of the first defective matrix gives the values  $\phi_y$  and  $\phi_x$ , via (8a) and (8b), respectively.

**Step 4:** The average consensus value is computed as

$$\mu_j = \lim_{h \rightarrow \infty} \frac{\hat{y}_j[h]}{\hat{x}_j[h]} = \frac{\phi_y(j)}{\phi_x(j)} = \frac{\hat{y}_{M_j}^T \beta_j}{\hat{x}_{M_j}^T \beta_j}.$$


---

*Example 4:* Considering the network in Fig. 1 and using Theorem 2, the final consensus vectors  $\phi_y$  and  $\phi_x$ , for initial conditions  $y[0]$  and  $x[0]$ , respectively, are given by

$$\phi_y = [0.4045 \ 0.2697 \ 0.1348 \ 0.6067 \ 0.3371 \ 0.5056]^T$$

$$\phi_x = [0.2697 \ 0.1798 \ 0.0899 \ 0.4045 \ 0.2247 \ 0.3371]^T.$$

Then, each node can compute the exact average  $\mu_j = \phi_y(j)/\phi_x(j)$  (e.g.,  $\mu_1 = \phi_y(1)/\phi_x(1) = 1.5$ ). For this example, the maximum number of steps that a node requires to compute the exact average value is 64 steps, while the values of the ratios for the algorithm described in Proposition 2 using the upper bound  $\bar{\tau}$  are  $z[64] = [1.4906 \ 1.4993 \ 1.4953 \ 1.5016 \ 1.5034 \ 1.4905]^T$

(see the top of Fig. 10). If the delay is overestimated, then our proposed algorithm needs more time to converge to the final value; for this specific example, nodes assume that the upper bound on the delay is 10 (i.e.,  $\bar{\tau} = 10$ ) and the maximum number of steps required to compute the exact average value is 114 steps. This observation suggests that while we have further assumptions on the upper bound of the delay, the proposed algorithm performs well even in the presence of time-varying delays and even if the delays are overestimated. The algorithm described in Proposition 2 using the overestimated upper bound  $\bar{\tau} = 10$  is shown at the bottom of Fig. 10.

## V. CONCLUSIONS AND FUTURE DIRECTIONS

This paper proposes a discrete-time algorithm that allows a directed networked system to distributively compute the exact average consensus in a finite number of steps, using only local observations at each component of a multicomponent system. The average consensus value can be obtained by running a protocol that requires each component to have knowledge of the number of its outgoing links (i.e., the number of components to which it sends information) and its own history of values. For the case of time-invariant delays, we show that the algorithm for computing the average value is similar to the algorithm when no delays are present; however, a larger number of observations are required for convergence. For the case of time-varying delays, it is assumed that nodes have knowledge of an upper bound of the delays in order to apply the proposed methodology. To the best of the authors' knowledge, this is the first algorithm that is guaranteed to reach average consensus in a digraph in a finite number of steps.

It would be interesting to investigate the following issues:

- Can we reach consensus in finite time when agents have no knowledge of the upper bound of the time-varying delays?
- The computation of the rank of the Hankel matrix assumes that its block elements are given exactly. How is the method affected when we have uncertainties in the measurements?
- It is shown in [36] that ratio consensus reaches average consensus even in the presence of switching topologies and time-varying delays. However, switching topologies imply that the system is no longer linear time invariant and, hence, the approach proposed here is not valid. How can one compute the average in networks with switching topologies?

## APPENDIX

We show that there exists a set of initial conditions  $y[0] = y_0$  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 matrices lose ranks. We then establish that this set of initial values belongs to a set of Lebesgue measure zero. The analysis below is not complete since we do not completely characterize the set of Lebesgue measure zero. Our goal is mostly to provide intuition on this issue.

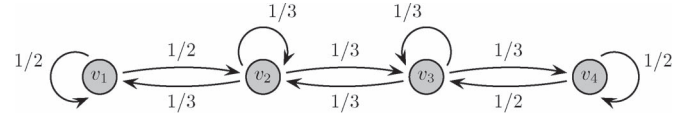


Fig. 11. Simple example with four nodes. Whatever the value of node  $v_4$ , if  $y_1[0] = y_2[0] = y_3[0]$ , node  $v_1$  loses rank in two steps and, hence, it computes the wrong final value.

Usually, the main reason for which the algorithm fails is because the Hankel matrix of some node loses rank for the first time too early. To gain some insight, consider, for example, the case when the Hankel matrix<sup>4</sup> for node  $v_i$  loses rank after just two steps. That means

$$\det \begin{pmatrix} y_i[0] & y_i[1] \\ y_i[1] & y_i[2] \end{pmatrix} = 0. \quad (16)$$

As a result,  $y_i^2[1] = y_i[0]y_i[2]$ . Alternatively, this can be written as  $(P_{i,\bullet}y_0)(P_{i,\bullet}y_0) = y_i[0]P_{i,\bullet}y[1] = y_i[0]P_{i,\bullet}Py_0$ , where  $P_{i,\bullet}$  is the  $i$ th row of matrix  $P$ . As a result

$$P_{i,\bullet}(y_i[0]Py_0 - y_i[1]y_0) = 0. \quad (17)$$

There are two cases for which (17) holds

- 1)  $y_i[0]Py_0 = y_i[1]y_0$ ;
  - a) if  $y_i[0] = 0$ , then from (16),  $y_i^2[1] = 0$ , meaning that  $P_{i,\bullet}y_0 = 0$  (i.e.,  $P_{i,\bullet} \perp y_0$ ) and the state of node  $v_i$  does not change. Note that if  $y_0 = 0$ , then the Hankel matrix will lose rank immediately and the final value will be zero, so there is no problem with  $y_0 = 0$ .
  - b) if  $y_i[0] \neq 0$ , then  $y_0$  corresponds to an eigenvector  $P$ , with eigenvalue  $\lambda_j(P) = y_i[1]/y_i[0]$ , that is,

$$Py_0 = \frac{y_i[1]}{y_i[0]}y_0, \quad \lambda_j(P) \in \mathbb{R}.$$

When this occurs, and  $\lambda_j(P) < 1$ , the final value at which the nodes will converge to is zero (for all nodes) and, as a result, this value is found distributively in two steps. Also, if  $\lambda_j(P) = 1$ , then the value of each node has reached the final value.

In this case, there exists a Lebesgue measure zero set (satisfying  $P_{i,\bullet}y_0 = 0$  and  $y_i[0] = 0$ ) for which the node loses rank before the states have converged to the consensus value.

- 2)  $P_{i,\bullet} \perp (y_i[0]Py_0 - y_i[1]y_0)$ ; it can be easily shown that  $P_{i,\bullet} \perp (P - (y_i[1]/y_i[0])I)$  is not possible unless some special circumstances hold for which the final values have already been reached. As a result,  $P_{i,\bullet}(P - (y_i[1]/y_i[0])I) \perp y_0$ , which implies that node  $v_i$  was not “excited” (i.e., no state difference) by any node of at least 2-hops away; hence, node  $v_i$  is already in a *local* consensus, thus causing its Hankel matrix to lose rank early.

Fig. 11 shows an illustrative example of the aforementioned analysis. Here, nodes  $v_1$  to  $v_3$  have the same value, while node

<sup>4</sup>We use the Hankel matrix formed by the values and not the differences between the values for illustration purposes, but it does not make a difference.

$v_4$  (three hops away from  $v_1$ ) has a different value. Thus,  $v_1$  loses rank before it obtains information from node  $v_4$ .

**Remark 7:** This problem can be alleviated if the nodes have knowledge of the network diameter  $d$  or at least an upper bound of the network diameter  $\bar{d}$  (e.g., the number of nodes  $n$  in the network, or, even an upper bound  $\bar{n}$  of the number of nodes). If such knowledge is available, nodes do not stop neither the iterations nor 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). More specifically, after  $d$  steps, every node will have received information from every other node in the network. The algorithm requires, at most,  $2n$  steps to converge; hence, once the necessary number of steps  $s$  has been exceeded (i.e.,  $s \geq d$  steps), if there was a case for which the Hankel matrix of a node lost rank earlier, then the node should continue the iterations and storage until  $d$  steps have been exceeded and if nothing has changed (i.e., no extra information received), then the true final value is the one computed already. Otherwise, if no rank loss occurred yet and the rank of the Hankel matrix still increases, the node should continue storing values until the next rank loss occurs to the Hankel matrix that gives the true final value. If no information can be assumed/inferred about the size of the diameter or the network, then a node can reduce the probability of losing rank early by continuing to store information until the buffer reaches its capacity. We have not considered in this work the case for which the capacity of the nodes does not suffice for the computation. Note that in the ratio consensus case, we have two concurrent iterations. As a result, a node has the ability to decide that the final value can be computed based on when the iterations lose rank. More specifically, if one of the iterations (say *iteration 1*) loses rank early, while the other iteration (say *iteration 2*) does not, then the node should seek for the next time *iteration 1* loses rank before concluding on the final value. As a result, while the issue may not be resolved for all cases, ratio consensus has the chance to check two initial conditions and, hence, it is a more robust approach than a single iteration.

**Remark 8:** Even though a node can stop storing values as soon as its final value is computed, it should only terminate iterations after  $2n + d$  steps, time by which all nodes are guaranteed to have computed their final value, provided that knowledge of  $n$  or  $\bar{n}$  is available. Note that this is a worst case scenario for which the first information to node  $v_j$  arrives after  $d$  steps and then the value  $x_j$  starts to change. This is consistent with the upper bound on the number of steps required for convergence, as stated in Remark 3.

## REFERENCES

- [1] M. H. DeGroot, "Reaching a consensus," *J. Amer. Stat. Assoc.*, vol. 69, no. 345, pp. 118–121, 1974.
- [2] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Trans. Autom. Control*, vol. 31, no. 9, pp. 803–812, Sep. 1986.
- [3] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1989.
- [4] A. Giridhar and P. R. Kumar, "Toward a theory of in-network computation in wireless sensor networks," *IEEE Commun. Mag.*, vol. 44, no. 4, pp. 98–107, Apr. 2006.
- [5] J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart, and P. Beardsley, "Image and animation display with multiple mobile robots," *Int. J. Robot. Res.*, vol. 31, no. 6, pp. 753–773, 2012.
- [6] I. D. Couzin, N. R. Franks, and S. A. Levin, "Effective leadership and decision-making in animal groups on the move," *Nature*, vol. 433, no. 7025, pp. 513–516, Feb. 2005.
- [7] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Trans. Autom. Control*, vol. 51, no. 3, pp. 401–420, Mar. 2006.
- [8] D. Acemoglu, G. Como, F. Fagnani, and A. Ozdaglar, "Opinion fluctuations and disagreement in social networks," *Math. Oper. Res.*, vol. 38, no. 1, pp. 1–27, 2013.
- [9] A. Jadbabaie, J. Lin, and A. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Trans. Autom. Control*, vol. 48, no. 6, pp. 988–1001, Jun. 2003.
- [10] L. Schenato and F. Fiorentin, "Average TimeSynch: A consensus-based protocol for time synchronization in wireless sensor networks," *Automatica*, vol. 47, no. 9, pp. 1878–1886, 2011.
- [11] G. Vogiatzis, I. MacGillivray, and M. Chli, "A probabilistic model for trust and reputation," in *Proc. 9th Int. Conf. Auton. Agents Multiagent Syst.*, 2010, pp. 225–232.
- [12] C. N. Hadjicostis and T. Charalambous, "Asynchronous coordination of distributed energy resources for the provisioning of ancillary services," in *Proc. 49th Annu. Allerton Conf. Commun., Control, Comput.*, Sep. 2011, pp. 1500–1507.
- [13] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1520–1533, Sep. 2004.
- [14] A. Nedic, A. Olshevsky, A. Ozdaglar, and J. Tsitsiklis, "On distributed averaging algorithms and quantization effects," *IEEE Trans. Autom. Control*, vol. 54, no. 11, pp. 2506–2517, Nov. 2009.
- [15] B. Johansson, M. Rabi, and M. Johansson, "A randomized incremental subgradient method for distributed optimization in networked systems," *SIAM J. Optimiz.*, vol. 20, no. 3, pp. 1157–1170, Aug. 2009.
- [16] F. Zanella, D. Varagnolo, A. Cenedese, G. Pillonetto, and L. Schenato, "Newton-Raphson consensus for distributed convex optimization," in *Proc. 50th IEEE Conf. Dec. Control Eur. Control Conf.*, Dec. 2011, pp. 5917–5922.
- [17] F. Zanella, D. Varagnolo, A. Cenedese, G. Pillonetto, and L. Schenato, "Asynchronous Newton-Raphson consensus for distributed convex optimization," in *Proc. 3rd IFAC Workshop Distrib. Estimation Control NecSys*, Sep. 2012, pp. 133–138.
- [18] A. D. Domínguez-García and C. N. Hadjicostis, "Coordination and control of distributed energy resources for provision of ancillary services," in *Proc. 1st IEEE Int. Conf. Smart Grid Commun.*, Oct. 2010, pp. 537–542.
- [19] M. Franceschelli, A. Giua, and C. Seatzu, "Distributed averaging in sensor networks based on broadcast gossip algorithms," *IEEE Sens. J.*, vol. 11, no. 3, pp. 808–817, Mar. 2011.
- [20] K. Cai and H. Ishii, "Average consensus on general strongly connected digraphs," *Automatica*, vol. 48, no. 11, pp. 2750–2761, Nov. 2012.
- [21] C. N. Hadjicostis and T. Charalambous, "Average consensus in the presence of delays in directed graph topologies," *IEEE Trans. Autom. Control*, vol. 59, no. 3, pp. 763–768, Mar. 2014.
- [22] S. Bhat and D. Bernstein, "Finite-time stability of continuous autonomous systems," *SIAM J. Control Optimiz.*, vol. 38, no. 3, pp. 751–766, 2000.
- [23] J. Cortés, "Finite-time convergent gradient flows with applications to network consensus," *Automatica*, vol. 42, no. 11, pp. 1993–2000, 2006.
- [24] Q. Hui, M. Haddad, and S. Bhat, "Finite-time semistability and consensus for nonlinear dynamical networks," *IEEE Trans. Autom. Control*, vol. 53, no. 8, pp. 1887–1900, Sep. 2008.
- [25] L. Wang and F. Xiao, "Finite-time consensus problems for networks of dynamic agents," *IEEE Trans. Autom. Control*, vol. 55, no. 4, pp. 950–955, Apr. 2010.
- [26] C.-K. Ko, "On matrix factorization and scheduling for finite-time average-consensus," Ph.D. dissertation, Dept. Comput. Math. Sci., California Inst. Technol., Pasadena, CA, USA, 2010.
- [27] L. Georgopoulos, "Definitive consensus for distributed data inference," Ph.D. dissertation, School Comput. Commun. Sci., École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 2011.
- [28] A. Kibangou, "Graph laplacian based matrix design for finite-time distributed average consensus," in *Proc. Amer. Control Conf.*, Jun. 2012, pp. 1901–1906.
- [29] J. M. Hendrickx, R. M. Jungers, A. Olshevsky, and G. Vankeerberghen, "Graph diameter, eigenvalues, minimum-time consensus," *Automatica*, vol. 50, no. 2, pp. 635–640, 2014.



- [30] S. Sundaram and C. N. Hadjicostis, "Finite-time distributed consensus in graphs with time-invariant topologies," in *Proc. Amer. Control Conf.*, Jul. 2007, pp. 711–716.
- [31] 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," *Proc. IEEE 48th Dec. Control Conf.*, pp. 2664–2669, Dec. 2009.
- [32] Y. Yuan, G.-B. Stan, M. Barahona, L. Shi, and J. Gonçalves, "Decentralised minimum-time consensus," *Automatica*, vol. 49, no. 5, pp. 1227–1235, May 2013.
- [33] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge, U.K.: Cambridge University Press, 1985.
- [34] A. I. Rikos, T. Charalambous, and C. N. Hadjicostis, "Distributed weight balancing over digraphs," *IEEE Trans. Control Netw. Syst.*, vol. 1, no. 2, pp. 190–201, Jun. 2014.
- [35] J. Sendra and J. Llovet, "Rank of a hankel matrix over  $\mathbb{Z}[x_1, \dots, x_r]$ ," *Applicable Algebra Eng. Commun. Comput.*, vol. 3, no. 4, pp. 245–256, 1992.
- [36] T. Charalambous and C. N. Hadjicostis, "Average consensus in the presence of dynamically changing directed graph topologies and time delays," *Proc. IEEE Conf. Dec. Control*, pp. 709–714, Dec. 2014.



**Themistoklis Charalambous** (M'14) received the M.Eng. and Ph.D. degrees in electrical and information sciences from Cambridge University, Cambridge, U.K., in 2005 and 2010, respectively.

He was a Research Associate at Imperial College London, London, U.K. and a Visiting Lecturer with the Department of Electrical and Computer Engineering, University of Cyprus, Nicosia, Cyprus. Currently, he is a Research Associate with the Automatic Control Lab of the School of Electrical Engineering at the Royal Institute of Technology (KTH),

Stockholm, Sweden. His research involves distributed coordination and control, distributed decision making, and control to various resource allocation problems in complex and networked systems.

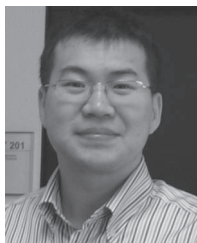


**Ye Yuan** (M'15) received the B.Eng. degree (Hons.) in automation from Shanghai Jiao Tong University, Shanghai, China, in 2008 and the M.Phil. and Ph.D. degrees in engineering from Cambridge University, Cambridge, U.K., in 2009 and 2012, respectively.

Currently, he is a Junior Research Fellow with Darwin College, University of Cambridge, and has been holding Visiting Researcher positions with the California Institute of Technology, Pasadena, CA, USA; Massachusetts Institute of Technology, Cambridge, MA, USA; and Imperial College

London, London, U.K. His research interests include the unification system identification and machine learning with applications in natural and engineering systems.

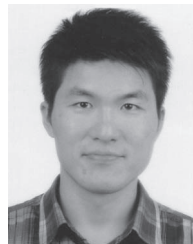
Dr. Yuan is the recipient of Dorothy Hodgkin Postgraduate Awards, Microsoft Research PhD Scholarship, Cambridge Overseas Scholarship, Chinese Government Award for Outstanding Students Abroad, Henry Lester Scholarship, and Best Paper Finalist in IEEE ICIA.



**Tao Yang** (M'12) received the Ph.D. degree in electrical engineering from Washington State University, Pullman, WA, USA, in 2012 and the M.S. degree (Hons.) in control engineering from City University, London, U.K., in 2004.

Currently, he is with the Electricity Infrastructure Group, Pacific Northwest National Laboratory (PNNL), Richland, WA, USA. Prior to joining PNNL, He was an ACCESS Postdoctoral Researcher with the ACCESS Linnaeus Centre, Royal Institute of Technology, Stockholm, Sweden. His research

interests are networked systems, cyber-physical systems, and distributed control and optimization with application to power networks.



**Wei Pan** (S'14) received the B.Sc. degree in automation from Harbin Institute of Technology, Harbin, China, the M.Sc. degree in biomedical engineering from the University of Science and Technology of China, Hefei, Anhui, China, and is currently pursuing the Ph.D. degree in bioengineering from Imperial College London, London, U.K.

His research interests include nonlinear time-series modeling.



**Christoforos N. Hadjicostis** (M'99–SM'05) received the B.S. degrees in electrical engineering, computer science and engineering, and mathematics, and the M.Eng. and Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 1993, 1995, and 1999, respectively.

In 1999, he joined the Faculty at the University of Illinois at Urbana-Champaign, Urbana, IL, USA, where he served as Assistant and then Associate Professor with the Department of Electrical and Com-

puter 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, Nicosia, 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.



**Mikael Johansson** (M'15) received the M.Sc. and Ph.D. degrees in electrical engineering from the University of Lund, Lund, Sweden, in 1994 and 1999, respectively.

He held postdoctoral positions at Stanford University, Stanford, CA, USA, and the University of California Berkeley, Berkeley, CA, USA, before joining the Royal Institute of Technology (KTH), Stockholm, Sweden, in 2002, where he is Full Professor. He has published one book and more than 100 papers, several of which are ISI-highly cited.

His research interests are in distributed optimization, wireless networking, and control.

Dr. Johansson 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 on control and communications.