Distributed Matrix Scaling and Application to Average Consensus in Directed Graphs

Alejandro D. Domínguez-García, *Member, IEEE*, Christoforos N. Hadjicostis, *Senior Member, IEEE*

***Abstract*—We propose a class of distributed iterative algorithms that enable the asymptotic scaling of a primitive column stochastic matrix, with a given sparsity structure, to a doubly stochastic form. We also demonstrate the application of these algorithms to the average consensus problem in networked multi-component systems. More specifically, we consider a setting where each node is in charge of assigning weights on its outgoing edges based on the weights on its incoming edges. We establish that, as long as the (generally directed) graph that describes the communication links between components is strongly connected, each of the proposed matrix scaling algorithms allows the system components to asymptotically assign, in a distributed fashion, weights that comprise a primitive doubly stochastic matrix. We also show that the nodes can asymptotically reach average consensus by executing a linear iteration that uses the time-varying weights (as they result at the end of each iteration of the chosen matrix scaling algorithm).**

***Index Terms*—Average consensus, distributed algorithms, directed graph, nonnegative matrix, matrix scaling.**

I. Introduction and Background

Over the past few decades, the design of protocols and algorithms for distributed function calculation and control/decision tasks has attracted significant attention by the computer science, communication, and control communities (e.g., [1], [2], [3], [4], [5], [6], [7], and references therein). For example, given a set of interconnected nodes (which could be sensors in a sensor network, routers in a communication network, or unmanned vehicles in a multi-agent system), the nodes may be interested in averaging their measurements; transmitting data from one/multiple sources to one/multiple sinks; coordinating their speed or direction; or electing a leader.

The distributed consensus (or agreement) problem is a special case of distributed function calculation where each node in the network possesses an initial value and the nodes need to agree on the same value, typically by calculating (via some iterative algorithm) the same function of their initial values [1], [8], [9]. A prototypical application of consensus is a network of sensors with noisy measurements of the same quantity (e.g., the temperature in a room) that try to average (or, more generally, obtain a weighted linear combination of) their initial measurements so as to have a more accurate estimate of the measured quantity. The consensus problem has received extensive attention from the control community due to its applicability to topics such as cooperative control and multi-agent systems (see, e.g., [9], [10], [11], [12], [13], and references therein). These works and others have demonstrated that consensus can be reached under exceedingly weak conditions on interaction (e.g., nonlinear updates, time-varying graphs with relatively few connections at any given time, etc.).

Asymptotic average consensus is a special case of the distributed consensus problem where all nodes converge asymptotically to the average of their initial values. One approach to asymptotic average consensus is to use a linear iteration, where each node repeatedly updates its value as a weighted linear combination of its own previous value and the previous values of its neighbors. In such case, one deals with an autonomous discrete-time linear system with a transition matrix P, also referred to as *weight matrix*, that is defined by the coefficients (weights) used in the linear updates. It is well-known that if the weight matrix P is primitive doubly stochastic, then the nodes will asymptotically converge to the average of their initial values (see, e.g., [12] and references therein).

In this paper, we develop algorithms that allow the nodes to choose their weights, in a distributed manner and subject to the sparsity structure imposed by a given *directed* graph, so that the resulting weight matrix P is primitive doubly stochastic. We also demonstrate that the output of these algorithms at each iteration, i.e., the set of nonnegative (time-varying) weights, can be used to conduct a time-varying linear algorithm that allows the nodes to asymptotically reach average consensus. When the graph is undirected both problems mentioned above (namely, the problems of forming a primitive doubly stochastic weight matrix and reaching asymptotic average consensus) are rather trivial; however, these problems are more challenging for the case of a directed graph. The extension to directed graphs is important because it allows us to adopt a very general model for the communication modality between nodes (if node i can transmit information to another node j, it is not required that node j can transmit information to node i).

The problem of obtaining a doubly stochastic matrix and the distributed solutions proposed in this paper are related to matrix scaling problems, which have been addressed in the context of nonnegative matrices [14], [15], [16] (the comparison of these works with ours is deferred to Section III-B). One of the early motivations for this problem was the desire to start from the stochastic matrix of a Markov chain and obtain a scaled version of it that is doubly stochastic and adheres to the

A. D. Domínguez-García is with the ECE Department at the University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA. E-mail: aledan@ILLINOIS.EDU.

C. N. Hadjicostis is with the ECE Department at the University of Cyprus, Nicosia, Cyprus, and also with the ECE Department at the University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA. E-mail: chadjic@UCY.AC.CY.

The work of A. D. Domínguez-García was supported in part by the National Science Foundation (NSF) under Career Award ECCS-CAR-0954420. The work of C. N. Hadjicostis was supported in part by the European Commission (EC) Seventh Framework Programme (FP7/2007-2013), under grant agreements INFSO-ICT-223844 and PIRG02-GA-2007-224877.

sparsity structure of the original one [17]. Many applications of matrix scaling can also be found in economics, urban planning, statistics, and demography—see, for instance, the discussions in [18]. There are also many applications where distributed matrix scaling is useful, including estimation problems with some underlying averaging operation [19], distributed optimization problems [20], distributed power control in wireless ad-hoc networks [21], and others.

The techniques in [22], [23], [24] investigated conditions and developed (centralized and distributed) algorithms to assign nonnegative weights to the edges of a directed graph (without self-loops) so as to balance it (i.e., make the sum of the weights of the incoming edges at each node equal to the sum of the weights of the outgoing edges). These techniques are related to matrix scaling problems like the ones we study here but are distinct from the algorithms we propose. The major differences between our algorithms and the algorithms in [22], [23], [24] include (i) the matrix parametrization, (ii) the communication modality of the network (we assume a broadcast model instead of the direct point-to-point communication required in [24]), and (iii) the fact that our approach obtains the doubly stochastic matrix in an asymptotic fashion—not in finite time. In Section IV-E, we elaborate further on the differences between our work and the work in [22], [23], [24], once we have the opportunity to describe our algorithms.

Recent work that has looked at average consensus problems in directed graphs via discrete iterations with weights that are time varying is also related to our developments here. For instance, the work in [25] proposed a broadcast-based gossip algorithm that relies on each node knowing its out-degree and performing an iteration that involves two variables (that are coupled); experiments using this approach indicate that average consensus is reached but a formal proof of convergence is still open. The authors of [26] use a similar approach and prove convergence for certain small "gain" values. Also, some related recent work has addressed the average-consensus problem in directed graphs (via discrete iterations) in the presence of unreliable communication links [27], [28]. Specifically, the goal of these works is to deal with packet drops in the communication links; however the underlying assumption is that the nodes already possess a set of weights that form a doubly stochastic matrix.

The remainder of this paper is organized as follows. Section II introduces relevant notation, graph theoretic notions used to describe the communication model, and ancillary results needed throughout the paper. Section III considers average consensus and proposes a class of linear time-varying algorithms that rely on scaling a nonnegative matrix to doubly stochastic form, while Section IV proposes and illustrates via examples a family of distributed algorithms to solve this problem. Concluding remarks are presented in Section V.

II. Preliminaries

In this section, we introduce relevant notation, some graph theoretic notions used to describe the communication model of the networked multi-component system, and ancillary results used throughout the paper.

A. Notation

Real vectors are denoted by small letters, e.g., $x = [x_1, x_2, \ldots, x_n]'$, $x_j \in \mathbb{R}$, where the symbol " $'$ " denotes matrix/vector transposition. Matrices are defined over the field of real numbers and are denoted by capital letters, i.e., $A = [a_{ji}] \in \mathbb{R}^{n\times n}$, and $A' = [a_{ij}]$. The symbol I denotes the $n \times n$ identity matrix. We reserve the symbol k to index time and explicitly show the time dependence of a vector or a matrix as $x[k]$ or $A[k]$, respectively. The symbol P typically denotes a primitive doubly stochastic matrix, whereas P_c typically denotes a primitive column stochastic matrix. We use the notation $\text{diag}(a_1, a_2, \ldots, a_n)$ to denote a diagonal matrix, with a_j as the (j, j) diagonal entry. The cardinality of a set $\mathcal{S}$ is denoted by $|\mathcal{S}|$.

B. Graph Theoretic Notions

The exchange of information between components (nodes) of a multi-component system can be conveniently described by a *directed graph* $\mathcal{G}_d = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{1, 2, \ldots, n\}$ is the vertex set (each vertex corresponds to a component/node) and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of *directed edges*. Edge $(j, i) \in \mathcal{E}$ if node j can receive information from node i. By convention, we assume no self-loops in $\mathcal{G}_d$ (i.e., $(j, j) \notin \mathcal{E}$ for all $j \in \mathcal{V}$). In an *undirected graph*, which we denote by $\mathcal{G}_u = \{\mathcal{V}, \mathcal{E}\}$, we have $(j, i) \in \mathcal{E}$ if and only if $(i, j) \in \mathcal{E}$ (i.e., if node j can receive information from node i, then node i can also receive information from node j). In subsequent developments, unless explicitly stated, we always consider directed graphs. All nodes that can transmit information to node j are said to be in-neighbors of node j and are represented by the set $\mathcal{N}_j^- = \{i \in \mathcal{V} \mid (j, i) \in \mathcal{E}\}$. The number of in-neighbors of j is called the in-degree of j and is denoted by $\mathcal{D}_j^-$ (i.e., $\mathcal{D}_j^- = |\mathcal{N}_j^-|$). The nodes that have j as an in-neighbor are called its out-neighbors and are denoted by $\mathcal{N}_j^+ = \{l \in \mathcal{V} : (l, j) \in \mathcal{E}\}$; the out-degree of node j is $\mathcal{D}_j^+ = |\mathcal{N}_j^+|$. The $n \times n$ matrix A, with $a_{ji} = 1$ if $(j, i) \in \mathcal{E}$ and $a_{ji} = 0$ otherwise, is called the graph adjacency matrix. We clearly have $\mathcal{D}_j^+ = \sum_l a_{lj}$ and $\mathcal{D}_j^- = \sum_i a_{ji}$. A directed (undirected) graph is *strongly connected* (*connected*) if for any pair of vertices i and j there exists a path that starts in i and ends in j, i.e., a sequence of edges $(i_1, i), (i_2, i_1), \ldots, (i_t, i_{t-1}), (j, i_t)$ all of which belong in $\mathcal{E}$.

C. Ancillary Results

The results in the following two lemmas are used in Section IV for analyzing the convergence properties of the proposed distributed weight matrix scaling algorithms. While these results apply to general primitive matrices, the reader should keep in mind that, in our setting, these matrices are always associated with a strongly connected directed graph.

Lemma 1: Let P_c be a primitive column stochastic matrix with diagonal entries $P_c(j, j) = p_{jj} \neq 1,\ \forall j$, and let $v = \pi = [\pi_1, \pi_2, \ldots, \pi_n]' > 0$ be the unique solution to $v = P_c v$, normalized so that $\sum_{j=1}^n \pi_j = 1$. Let $\Delta = \text{diag}\big(\delta_1, \delta_2, \ldots, \delta_n\big)$ be a diagonal matrix with $0 < \delta_j \leq \frac{1}{1-P_c(j,j)},\ \forall j$, with at least one $i \in \{1, 2, \ldots, n\}$ such that $0 < \delta_i < \frac{1}{1-P_c(i,i)}$,

and define $\hat{P}_c = P_c\Delta + (I - \Delta)$, where I is the identity matrix. Then, (i) $\hat{P}_c$ is a primitive column stochastic matrix, and (ii) if $\hat{v} = \hat{\pi} = [\hat{\pi}_1,\ \hat{\pi}_2, \ldots, \hat{\pi}_n]' > 0$ is the unique solution to $\hat{v} = \hat{P}_c\hat{v}$, normalized so that $\sum_{j=1}^n \hat{\pi}_j = 1$, we have $\delta_j\hat{\pi}_j = \alpha\pi_j, \forall j$, for some constant $\alpha > 0$.

Proof: The fact that $\hat{P}_c$ is column stochastic is obvious from the definition of $\hat{P}_c$ and the choice of the range of δ_j's (all entries of $\hat{P}_c$ are nonnegative and the sum of each column is one). Furthermore, since P_c is primitive, the underlying directed graph (that has an edge from node i to node j, $i \neq j$, if $P_c(j,i)$ is nonzero) is strongly connected. The underlying directed graph for matrix $\hat{P}_c$ is identical to the one of P_c (because all the δ_j's are strictly greater than zero so that each nonzero $P_c(j,i)$, $j \neq i$, is in an one-to-one correspondence with a nonzero $\hat{P}_c(j,i)$) and thus, also strongly connected. Primitivity of $\hat{P}_c$ follows then from the fact that at least one diagonal entry (namely $\hat{P}_c(i,i)$) is nonzero [29].

Let $\hat{v} = \hat{\pi} = [\hat{\pi}_1, \hat{\pi}_2, \ldots, \hat{\pi}_n]'$ be the solution of $\hat{v} = \hat{P}_c\hat{v}$ normalized so that $\sum_{j=1}^n \hat{\pi}_j = 1$. Then, $\hat{\pi} = \hat{P}_c\hat{\pi} = [P_c\Delta + (I - \Delta)]\hat{\pi}$, which implies that $P_c\Delta\hat{\pi} = \Delta\hat{\pi}$. Since both $\hat{P}_c$ and P_c are primitive column stochastic matrices, the Perron-Frobenius theorem ensures that there is a unique (up to scalar multiplication) positive solution of $\hat{v} = \hat{P}_c\hat{v}$ with $\hat{v}_j > 0, \forall j$, and a unique (up to scalar multiplication) positive solution of $v = P_cv$ with $v_j > 0, \forall j$. Thus, $\Delta\hat{\pi} = \alpha\pi$ or, equivalently, $\delta_j\hat{\pi}_j = \alpha\pi_j,\ \forall j$, for some $\alpha > 0$ (because $0 < \delta_j,\ \forall j = 1, 2, \ldots, n$). ■

The following lemma describes the limiting behavior of a product of primitive column stochastic matrices when the sequence of matrices in the product converges to a limiting matrix P_c; the result follows from [17, Thm. 4.14].

Lemma 2: Let $P[0], P[1], \ldots$ be a sequence of column stochastic matrices such that $\lim_{k\to\infty} P[k] = P_c$ elementwise, where the matrix P_c is column stochastic and primitive. Then $\lim_{k\to\infty} P[k]\ldots P[1]P[0] = vw'$, where v is the unique solution to $v = P_cv$, normalized so that $\sum_{j=1}^n v_j = 1$, and $w = [1, 1, \ldots, 1]'$.

Proof: Let $T_k = (P[k]\ldots P[1]P[0])' = P'[0]P'[1]\ldots P'[k]$, where, $P'[k]$, $\forall k \geq 0$, is a row stochastic matrix, and $\lim_{k\to\infty} P'[k] = P'_c$, where P'_c is primitive and row stochastic, and $v' = v'P'_c$. By Theorem 4.14 in [17], $T := \lim_{k\to\infty} T_k = wv'$. Therefore $\lim_{k\to\infty} P[k]\ldots P[1]P[0] = \lim_{k\to\infty}(P'[0]P'[1]\ldots P'[k])' = \lim_{k\to\infty} T'_k = T' = vw'$. ■

III. Distributed Average Consensus via Nonnegative Weight Matrix Scaling

In this section, we propose a linear time-varying iterative distributed algorithm for average consensus over directed graphs. This algorithm relies on another distributed iterative algorithm that (runs in parallel and) scales the nonnegative weight matrix in a way that keeps it column stochastic at each iteration and eventually drives it to a primitive doubly stochastic weight matrix. This weight matrix scaling algorithm belongs to a class of distributed nonnegative matrix scaling algorithms, the formulation and convergence of which is presented in Section IV; here, we provide the intuition behind their construction, and their application to average consensus.

A. Average Consensus via a Linear Time-Varying Algorithm

Consider a strongly connected directed graph $\mathcal{G}_d = \{\mathcal{V}, \mathcal{E}\}$, and denote by u_j some value that node j possesses. In the average-consensus problem, the objective is for the nodes in $\mathcal{G}_d$ to calculate the average of the u_j's, i.e., $\overline{u} := \frac{\sum_{j=1}^n u_j}{n}$. Assume that there is a weight $0 < p_{ji} \leq 1$ associated to each $(j,i) \in \mathcal{E}$, and a self-weight $0 \leq p_{jj} < 1$ for each $j \in \mathcal{V}$, and define the corresponding weight matrix $P = [p_{ji}]$ ($p_{ji} = 0$ for all $j \neq i$ such that $(j,i) \notin \mathcal{E}$). It is well known (see, e.g., [11], [12], [30]) that if $\sum_{i\in\mathcal{N}_j^-\cup\{j\}} p_{ji} = \sum_{i=1}^n p_{ji} = 1$ and $\sum_{l\in\mathcal{N}_j^+\cup\{j\}} p_{lj} = \sum_{l=1}^n p_{lj} = 1,\ \forall j$, i.e, the weight matrix P is primitive and doubly stochastic, then the nodes can asymptotically reach average consensus by executing a linear-iterative algorithm of the form

$$x_j[k+1] = p_{jj}x_j[k] + \sum_{i\in\mathcal{N}_j^-} p_{ji}x_i[k],\ k = 0, 1, 2, \ldots, \quad (1)$$

with $x_j[0] = u_j$, for all j.

Now, consider the time-varying version of (1) given in Algorithm 0, with weights that, for all k, satisfy: (i) $0 < p_{ji}[k] \leq 1,\ (j,i) \in \mathcal{E}$; (ii) $p_{ji}[k] = 0,\ (j,i) \notin \mathcal{E},\ j \neq i$; (iii) $0 \leq p_{jj}[k] < 1,\ \forall j$; and (iv) $\sum_{l=1}^n p_{lj}[k] = 1$. These time-varying weights are inputs to the algorithm and nodes determine them by executing (in parallel with Algorithm 0) any of four distributed *auxiliary algorithms*—completely independent of Algorithm 0—that we introduce later in the paper. With respect to this, if we let $x[k] = [x_1[k], x_2[k], \ldots, x_n[k]]'$, we can rewrite line 5 of Algorithm 0 in matrix form as

$$x[k+1] = P[k]x[k], \qquad x[0] = [u_1, u_2, \ldots, u_n]', \quad (2)$$

where $P[k],\ \forall k$, is column stochastic but not necessarily doubly stochastic. Then, any of the four aforementioned distributed auxiliary algorithms enables the nodes in the network to choose the $p_{ji}[k]$'s in a distributed fashion and ensure the convergence of (2) to average consensus. These distributed auxiliary algorithms iteratively scale the weight matrix $P[k]$ to a doubly stochastic form, while in the process, they ensure that all the $P[k]$'s are column stochastic and primitive. We argue later that under these conditions the nodes reach average consensus.

Algorithm 0

1 **Set:** $x_j[0] = u_j$ for all j

for $k \geq 0$ **each node** j **does:**

2 **Output**: $x_j[k]$

Receive:

3 From auxiliary algorithm:
$p_{jj}[k+1]$ and $p_{lj}[k+1]$ for all $l \in \mathcal{N}_j^+$

4 From all $i \in \mathcal{N}_j^-$:
$p_{ji}[k]x_i[k]$

Compute:

5 $x_j[k+1] = p_{jj}[k]x_j[k] + \sum_{i\in\mathcal{N}_j^-} p_{ji}[k]x_i[k]$

6 **Transmit:** $p_{lj}[k]x_j[k]$ to each $l \in \mathcal{N}_j^+$

Remark 1: When one is interested in reaching asymptotic consensus (but not necessarily to the average of some initial values), convergence is guaranteed under a variety of relatively weak conditions. When convergence to the average is required, things remain rather simple in undirected graphs: assuming that each node j knows the total number of nodes n or an upper bound $n' \geq n$, it can easily choose fixed (nonnegative) weights on its out-neighbor edges so that $\sum_{i=1}^{n} p_{ji} = \sum_{l=1}^{n} p_{lj} = 1, \forall j$: for instance, node j can set $p_{jj} = 1 - \frac{\mathcal{D}_j}{n'}$ and $p_{lj} = \frac{1}{n'}$ if $(l,j) \in \mathcal{E}$ (of course, $p_{lj} = 0$ if $l \neq j,\ (l,j) \notin \mathcal{E},\ l \neq j$), where $\mathcal{D}_j = \mathcal{D}_j^+ = \mathcal{D}_j^-$. On the other hand, in a directed graph, a given node j may not necessarily have $\mathcal{D}_j^+ = \mathcal{D}_j^-$, and it is not as straightforward for nodes to determine weights that satisfy $\sum_{l=1}^{n} p_{lj} = \sum_{i=1}^{n} p_{ji} = 1, \forall j$. □

B. Distributed Matrix Scaling to Doubly Stochastic Form

In Section IV, we propose four algorithms for distributed weight matrix scaling. All algorithms assume that each node j can choose its self-weight $p_{jj}[k]$ and set the weights $p_{lj}[k],\ l \in \mathcal{N}_j^+$, on its out-neighbor edges. We assume that each node can observe but cannot control the (likely different) values on each of its in-neighbor edges, and cannot necessarily identify the sender node associated with each value. These assumptions hold naturally for most interconnection topologies that form a directed graph (in fact, in many practical situations additional information may be available to each node). In all four algorithms, each node uses the *same* weight for all of its out-neighbor edges, which is a particularly attractive feature in the case of wireless networks (because a single broadcast transmission by a node ensures that all of its out-neighbors obtain the transmitted value and/or weight).

Let $k = 0, 1, 2, \ldots$ index the iteration steps, and denote by $p_{lj}[k],\ l \in \mathcal{N}_j^+ \cup \{j\}$, the choice of weights made by node j at step k. In all proposed algorithms, this choice satisfies the following constraints for all k: (i) $p_{jj}[k] > 0$; (ii) $p_{lj}[k] > 0,\ \forall l \in \mathcal{N}_j^+$; (iii) $p_{lj}[k] = 0$ for all l such that $l \notin \mathcal{N}_j^+ \cup \{j\}$; and (iv) $\sum_{l=1}^{n} p_{lj}[k] = 1,\ \forall j$. At $k = 0$, node j sets its self-weight and the weights on its out-neighbor edges to be $p_{lj}[0] = 1/(1 + \mathcal{D}_j^+),\ l \in \mathcal{N}_j^+ \cup \{j\}$; hence $\sum_{l \in \mathcal{N}_j^+ \cup \{j\}} p_{lj}[0] = 1$ but in general $\sum_{i \in \mathcal{N}_j^- \cup \{j\}} p_{ji}[0] \neq 1$. Thus, the initial weight matrix, which we denote by P_c, is column stochastic but not row stochastic. At each subsequent time step k, each node j calculates a quantity $0 < \delta_j[k] \leq 1$ and uses it to scale its own weight and the weights on its out-neighbor edges; since the out-neighbor edge weights are equal, the resulting $P[k]$ can be parametrized as

$$P[k] = \overline{P}\Delta[k] + (I - \Delta[k]), \qquad P[0] = P_c, \qquad (3)$$

where $\Delta[k] = \text{diag}\big(\delta_1[k], \delta_2[k], \ldots, \delta_n[k]\big)$ is a diagonal matrix, and $\overline{P} = [\overline{p}_{ji}]$ is a matrix with all its diagonal entries being zero and, for each $j \in \mathcal{V}$, $\overline{p}_{lj} = 1/\mathcal{D}_j^+,\ \forall l \in \mathcal{N}_j^+$, and zero otherwise. Note that $P_c = P[0] = \overline{P}\Delta[0] + (I - \Delta[0])$, where $\Delta[0]$ is a diagonal matrix with $\delta_j[0] = \mathcal{D}_j^+/(1 + \mathcal{D}_j^+)$. Since $\overline{P}$ is column stochastic, from (3) it immediately follows that, as long as $0 < \delta_j[k] \leq 1,\ \forall j$, $P[k]$, is also column stochastic, i.e., each node j enforces $\sum_{l \in \mathcal{N}_j^+ \cup \{j\}} p_{lj}[k] = 1$. The rules for updating the diagonal entries of $\Delta[k]$ (which differ for each of the proposed algorithms) must be such that $\lim_{k\to\infty} P[k] = P$, where P is a primitive doubly stochastic matrix (not necessarily the same for each algorithm). In order to reach a limiting doubly stochastic matrix, each node j performs several computations (to be defined for each algorithm) that rely on information obtained locally by j. Note that the particular parameterization that is chosen for $P[k]$ implies that during the iteration step of Algorithm 0, the separate algorithm that computes the weights in line 3 of Algorithm 0 only needs to provide $p_{jj}[k+1]$ to node j.

Remark 2: In order to initialize the iteration in (3) via $\Delta[0]$, each node j sets $\delta_j[0] = \frac{\mathcal{D}_j^+}{1+\mathcal{D}_j^+}$, i.e., each node needs to know the total number of its out-neighbors $\mathcal{D}_j^+ = |\mathcal{N}_j^+|$ (in all four proposed algorithms, $P[0] = P_c$). In practice, there are many ways in which nodes can discover their out-degree in a directed topology setting. For example, one could envision each node transmitting a *hello* packet at a chosen power level (possibly multiple times); any node receiving one of these *hello* packets responds by sending an *acknowledgement* message (possibly multiple times, and possibly at a higher power level than normal transmissions). When node j receives an *acknowledgement* message from node l, node j knows that l can hear j's transmissions with nonzero probability. Following initialization and during the operation of the system, nodes do not necessarily transmit at high power; the transmission level that each node chooses (perhaps taking into account factors such as its battery lifetime, etc.) determines how many nodes will be able to receive its transmission. Since each node is aware of its neighborhood (due to the neighbor discovery phase), then each node can determine its out-degree (for the transmission level it chooses). □

Earlier work on matrix scaling (see, e.g., [14], [15], [16]) developed and analyzed centralized solutions to (variations of) the following problem: given a nonnegative matrix A, find a pair of positive diagonal matrices D and E such that $P := DAE$ is doubly stochastic. This, among other things, requires the all-ones vector to be the right eigenvector of DAE that corresponds to eigenvalue 1. Our proposed algorithms differ from the ones in [16] in the following aspects. First, in the algorithms we propose, for every $k \geq 0$, the matrix $P[k]$ to be scaled is parametrized as $P[k] = \overline{P}\Delta[k] + (I - \Delta[k])$, where $P[0] = P_c$ is a column stochastic matrix, and the nonzero entries of the diagonal matrix $\Delta[k]$ are calculated (using only locally available information) in a manner that makes the $P[k]$'s column stochastic. [Note that column stochasticity is key for ensuring convergence of Algorithm 0 to average consensus (see, Theorem 1).] It is worth pointing out that there is no *a priori* guarantee that the particular parametrization we use will converge to a doubly stochastic matrix; this, however, follows as a result of our analysis. Also, while enforcing column stochasticity seems restrictive, it is quite easy to do in directed graph settings by having each node j choose its own weight $p_{jj}[k]$ and the weights $p_{lj}[k],\ l \in \mathcal{N}_j^+$, on its out-neighbor edges. By contrast, in the scaling algorithms proposed in [16] and related works, the matrix $P[k]$ to be iteratively scaled to doubly stochastic form is parametrized by

the recurrence $P[k+1] = D[k]P[k]E[k]$, where $P[0] = A$, and $D[k]$ and $E[k]$ are positive diagonal matrices defined by each particular algorithm. Thus, the algorithms in [14], [15], [16] do not enforce column stochasticty. Finally, while our algorithms are designed for a distributed implementation that adheres to the underlying communication topology, it is not clear that the algorithms in [16] are immediately amenable to a distributed implementation.

C. Convergence Analysis of Average Consensus Algorithm

The key in establishing the convergence of (2) to average consensus with a $P[k]$ of the form in (3) is the choice of the $\delta_j[k]$'s. In all four algorithms proposed, this choice ensures that, for all $k \geq 0$, all existing edges in the directed graph $\mathcal{G}_d$ are assigned a (strictly) positive weight, and at least one self-weight is (strictly) positive; since $\mathcal{G}_d$ is strongly connected, this implies that the matrices $P[0], P[1], \ldots, P[k], \ldots$ are primitive column stochastic. This observation, together with the fact that $\lim_{k\rightarrow} P[k] = P$, where P is primitive doubly stochastic, can be used to show that (2) converges to the average of the initial conditions, therefore establishing that the nodes can employ Algorithm 0 to reach average consensus.

Theorem 1: Let $x[k+1] = P[k]x[k]$, with $x[0] = [u_1, u_2, \ldots, u_n]'$ such that $|u_j| < \infty,\ \forall j$, and $P[k],\ k \geq 0$, being column stochastic and primitive with $\lim_{k\rightarrow\infty} P[k] = P$ elementwise, where P is a primitive doubly stochastic matrix. Then, $\lim_{k\rightarrow\infty} x_j[k] = \frac{\sum_{l=1}^{n} u_l}{n},\ \forall j$. Furthermore, for all j, $|x_j[k]| \leq \infty,\ \forall k \geq 0$.

Proof: From Lemma 2, it follows that, as $k \rightarrow \infty$, the product $P[k] \ldots P[1]P[0]$ converges to a matrix $T = vw'$, where v is the unique solution to $v = Pv$ normalized so that $\sum_{j=1}^{n} v_j = 1$, and $w = [1, 1 \ldots, 1]'$. Moreover, since P is doubly stochastic, $v = 1/n[1, 1 \ldots, 1]'$. Finally, $\lim_{k\rightarrow\infty} x[k] = P[k] \ldots P[1]P[0]x[0] = Tx[0] = vw'x[0]$, therefore $\lim_{k\rightarrow\infty} x_j[k] = \sum_{l=1}^{n} x_l[0]/n = \sum_{l=1}^{n} u_l/n = \overline{u},\ \forall j$.

Let $x_j[0] = u_j$, where $|u_j| < \infty$; if all $x_j[0]$'s are non-negative (non-positive), it immediately follows (from the fact that all $P[k]$'s are column stochastic) that all $x_j[k]$'s are non-negative (non-positive) and, thus, $|\sum_j x_j[k]| = |\sum_j x_j[0]| < \infty,\ \forall k$, and $|x_j[k]| < \infty,\ \forall k$. If $x[0]$ has positive, negative and zero entries, we can rewrite $x[0] = x^-[0] + x^+[0]$, where $x^-[0]$ is the portion of $x[0]$ corresponding to negative entries (all other entries are set to zero) and $x^+[0]$ is the portion of $x[0]$ corresponding to zero or positive entries (all other entries are set to zero). Then, by linearity and the fact that all entries in $P[k]$ are nonnegative, it can be seen that the evolution of $x[k+1] = P[k]x[k],\ x[0] = x^-[0] + x^+[0]$, is determined by $x^+[k+1] = P[k]x^+[k],\ x^+[0]$ and $x^-[k+1] = P[k]x^-[k],\ x^-[0]$. As already argued, $|x^+[k]| < \infty$, and $|x^-[k]| < \infty,\ \forall k$, i.e., their sum (and thus $x[k]$) is bounded. ■

IV. Distributed Weight Matrix Scaling to Doubly Stochastic Form

We propose four distributed algorithms that allow the nodes of a multi-component system (whose interconnection topology is described by a strongly connected directed graph) to asymptotically obtain a set of weights that forms a doubly stochastic matrix. We also discuss how the nodes can use these time-varying weights in conjunction with the iterative Algorithm 0 described in Section III-A to asymptotically reach average consensus.

A. Algorithm 1

At every iteration step k, each node j adjusts the weights it controls using two auxiliary variables, $\pi_j[k]$ and $\eta_j[k]$. More specifically, node j updates the first auxiliary variable $\pi_j[k+1]$ to be a weighted linear combination of its own previous value $\pi_j[k]$ and the previous values of its in-neighbors, i.e., the $\pi_i[k]$'s for each node i that sends information to j. The weights $p_{ji},\ i \in \mathcal{N}_j^- \cup \{j\}$, used to update $\pi_j[k]$ remain constant as k evolves. The second auxiliary variable, $\eta_j[k]$, basically allows each node j to asymptotically track the maximum value among $\pi_i[k],\ i \in \mathcal{N}_j^-,\ k \geq 0$. Then, each node updates its $\delta_j[k]$ as a function of the ratio of $\pi_j[k]$ and $\eta_j[k]$.

1) Algorithm Formulation: Consider a strongly connected graph $\mathcal{G}_d = \{\mathcal{V}, \mathcal{E}\}$. Each node $j \in \mathcal{V}$ initializes $\pi_j[0] = \eta_j[0] = 1$, and sets $\delta_j[0] = \frac{\mathcal{D}_j^+}{1+\mathcal{D}_j^+}$ so that $p_{lj}[0] = \frac{\delta_j[0]}{\mathcal{D}_j^+},\ \forall l \in \mathcal{N}_j^+ \cup \{j\}$ ($p_{lj}[0] = 0$ otherwise). At each time step $k \geq 0$, node j updates the weights over which it has control, i.e., $p_{lj}[k],\ l \in \mathcal{N}_j^+ \cup \{j\}$, by executing Algorithm 1. The output of Algorithm 1 is used by the nodes as the input to Algorithm 0, which they execute in parallel, allowing them to reach consensus to the average of the u_j's (initial values).

Algorithm 1

1 **Input**: $\pi_j[0] = 1$, $\eta_j[0] = 1$, $\delta_j[0] = \frac{\mathcal{D}_j^+}{1+\mathcal{D}_j^+}$

for $k \geq 0$ **each node** j **does:**

2 **Output**: $p_{jj}[k] = 1 - \delta_j[k]$

3 $p_{lj}[k] = \frac{1}{\mathcal{D}_j^+}\delta_j[k],\ \forall l \in \mathcal{N}_j^+$

4 **Receive:** $\frac{\mathcal{D}_i^+}{1+\mathcal{D}_i^+}\pi_i[k]$ and $\eta_i[k]$ from all $i \in \mathcal{N}_j^-$

Compute:

5 $\pi_j[k+1] = \frac{1}{1+\mathcal{D}_j^+}\pi_j[k] + \sum_{i\in\mathcal{N}_j^-} \frac{1}{1+\mathcal{D}_i^+}\pi_i[k]$

6 $\eta_j[k+1] = \max\{\pi_j[k], \max_{i\in\mathcal{N}_j^-\cup\{j\}}\{\eta_i[k]\}\}$

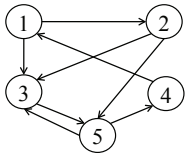
7 $\delta_j[k+1] = \frac{\mathcal{D}_j^+}{1+\mathcal{D}_j^+}\frac{\pi_j[k+1]}{\eta_j[k+1]}$

8 **Broadcast:** $\frac{\mathcal{D}_j^+}{1+\mathcal{D}_j^+}\pi_j[k+1],\ \eta_j[k+1]$ to all $l \in \mathcal{N}_j^+$

Letting $\pi[k] = [\pi_1[k], \pi_2[k], \ldots, \pi_n[k]]'$, line 5 of Algorithm 1 can be rewritten in matrix form as

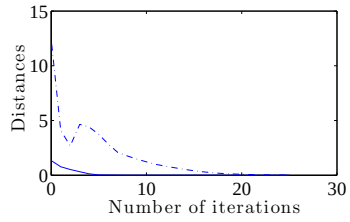
$$\pi[k+1] = P_c\pi[k]\ , \quad \pi[0] = [1,\ 1, \ldots, 1]', \tag{4}$$

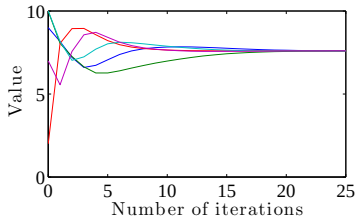
with P_c as defined in (3). [Since $\pi_j[0] > 0,\ \forall j$, then $\pi_j[k] > 0$ and $\eta_j[k] > 0,\ \forall j,\ \forall k$, thus $\delta_j[k] = \frac{\mathcal{D}_j^+}{1+\mathcal{D}_j^+}\frac{\pi_j[k]}{\eta_j[k]},\ \forall j$, is well defined.]



$$\overline{P} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 1/2 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 1/2 \\ 0 & 1/2 & 1 & 0 & 0 \end{bmatrix} \qquad P_c = \begin{bmatrix} 1/3 & 0 & 0 & 1/2 & 0 \\ 1/3 & 1/3 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/2 & 0 & 1/3 \\ 0 & 0 & 0 & 1/2 & 1/3 \\ 0 & 1/3 & 1/2 & 0 & 1/3 \end{bmatrix}$$

Fig. 1. Small directed graph used for illustration of the various algorithms, and associated weight matrices $\overline{P}$ and P_c.





$$P_{Alg1} = \begin{bmatrix} 0.72 & 0 & 0 & 0.28 & 0 \\ 0.14 & 0.86 & 0 & 0 & 0 \\ 0.14 & 0.07 & 0.51 & 0 & 0.28 \\ 0 & 0 & 0 & 0.72 & 0.28 \\ 0 & 0.07 & 0.49 & 0 & 0.44 \end{bmatrix}$$

Fig. 2. (left) Distance from a doubly stochastic matrix (solid line) and distance from average consensus (dash-dotted line) vs. number of iterations; (middle) node values vs. number of iterations; (right) limiting matrix P_{Alg1}.

2) Convergence Analysis: As a result of Algorithm 1, we obtain a sequence of column stochastic weight matrices $P[k] = \overline{P}\Delta[k] + (I - \Delta[k])$, where $\Delta[k] = \text{diag}\big(\delta_1[k], \delta_2[k], \ldots, \delta_n[k]\big)$. [$\overline{P}$ was defined below equation (3).] We next argue that $\lim_{k\to\infty} \delta_j[k]$, $\forall j$, exists and that $\lim_{k\to\infty} P[k] = P$ is doubly stochastic and primitive; hence, from Theorem 1, it follows that (2) with the $P[k]$'s as defined by Algorithm 1 reaches average consensus. We also show that the rate of convergence of $P[k]$ to P is geometric.

Theorem 2: Let $P[k] = \overline{P}\Delta[k]+(I-\Delta[k])$, where $\Delta[k] = \text{diag}(\delta_1[k], \delta_2[k], \ldots, \delta_n[k])$ with $\delta_j[k]$, $\forall j$, as defined in line 7 of Algorithm 1. Then $P[k]$, $\forall k \geq 0$, is primitive and column stochastic, and $\lim_{k\to\infty} P[k] = P$ elementwise, where P is a primitive doubly stochastic matrix.

Proof: From line 7 of Algorithm 1, it is easy to see that $0 < \delta_j[k] \leq \delta_j[0] < 1$, $\forall k \geq 0$, and $P[k] = \overline{P}\Delta[k] + (I - \Delta[k]) = P_c\hat{\Delta}[k] + (I - \hat{\Delta}[k])$, where $\hat{\Delta}[k] = \text{diag}\big(\frac{\pi_1[k]}{\eta_1[k]}, \frac{\pi_2[k]}{\eta_2[k]}, \ldots, \frac{\pi_n[k]}{\eta_n[k]}\big)$. Since P_c is column stochastic and primitive, and from lines 5–6 of Algorithm 1 we have $0 < \frac{\pi_j[k]}{\eta_j[k]} \leq 1$, $\forall j, k$, then $P[k]$, $\forall k \geq 0$, is also column stochastic and primitive (because $P[k]$, just like P_c, corresponds to the given underlying strongly connected directed graph and has nonzero diagonals). Now consider (4) and let $v = \pi$ be the unique solution to $v = P_c v$ normalized so that $\sum_{j=1}^n \pi_j = n$; then, $\lim_{k\to\infty} \pi_j[k] = \pi_j$, and from line 6 of Algorithm 1 it follows that $\lim_{k\to\infty} \eta_j[k] = \max_{l,m}\{\pi_l[m]\} =: \eta \leq n$ for all j, where $m = 1, 2, \ldots$, and $l = 1, 2, \ldots, n$. Then, $0 < \delta_j := \lim_{k\to\infty} \delta_j[k] = \frac{\mathcal{D}_j^+}{1+\mathcal{D}_j^+}\frac{\pi_j}{\eta} \leq \delta_j[0] < 1$, $\forall j = 1, 2, \ldots, n$, and therefore $P = \lim_{k\to\infty} P[k] = \lim_{k\to\infty}(\overline{P}\Delta[k] + \big(I - \Delta[k]\big)) = \overline{P}\Delta + \big(I - \Delta\big) = P_c\hat{\Delta} + \big(I - \hat{\Delta}\big)$, where $\hat{\Delta} = \text{diag}\big(\frac{\pi_1}{\eta}, \frac{\pi_2}{\eta}, \ldots, \frac{\pi_n}{\eta}\big)$. Let $\hat{v} = \hat{\pi}$ be the unique solution to $\hat{v} = P\hat{v}$, normalized so that $\sum_{j=1}^n \hat{\pi}_j = 1$. Then, by Lemma 1, P is a primitive column stochastic matrix, and $\hat{\pi}_j = \frac{\alpha}{\delta_j}\pi_j = \alpha\eta$. Thus, $\hat{\pi} = \alpha\eta[1, 1, \ldots, 1]'$, and since $\sum_{j=1}^n \hat{\pi}_j = 1$, we have that $\alpha = \frac{1}{n\eta}$. Thus, $\hat{v} = \hat{\pi} = \frac{1}{n}[1, 1, \ldots, 1]'$ is a solution of $\hat{v} = P\hat{v}$, consequently P is also row stochastic. ■

Corollary 1: Consider (2) with $P[k] = \overline{P}\Delta[k]+(I-\Delta[k])$, with $\Delta[k] = \text{diag}(\delta_1[k], \delta_2[k], \ldots, \delta_n[k])$ as defined in line 7 of Algorithm 1. Then, $\lim_{k\to\infty} x_j[k] = \frac{\sum_{l=1}^n u_l}{n}$, $\forall j$. Furthermore, for all j, $|x_j[k]| \leq \infty$, $\forall k \geq 0$.

Proof: Theorem 2 established that $P[0], \ldots, P[k], \ldots$ are primitive and column stochastic, and that $\lim_{k\to\infty} P[k] = P$ elementwise, where P is primitive doubly stochastic. The result then follows immediately from Theorem 1. ■

Lemma 3: Denote by λ_2 the eigenvalue with the second largest magnitude of matrix P_c (as defined in (3)) that governs the evolution of (4). Then, $\|P - P[k]\|_\infty \leq Ck^{m_2-1}|\lambda_2|^k$ where m_2 is the algebraic multiplicity of eigenvalue λ_2 and the constant C only depends on the matrix P_c.

Proof: We can write $\|P - P[k]\|_\infty = \|(P_c - I)(\hat{\Delta} - \hat{\Delta}[k])\|_\infty \leq \|(P_c - I)\|_\infty\|(\hat{\Delta} - \hat{\Delta}[k])\|_\infty$, where (as defined earlier) $\hat{\Delta}[k] = \text{diag}\big(\frac{\pi_1[k]}{\eta_1[k]}, \frac{\pi_2[k]}{\eta_2[k]}, \ldots, \frac{\pi_n[k]}{\eta_n[k]}\big)$ and the last inequality holds because $\hat{\Delta} - \hat{\Delta}[k]$ is a diagonal matrix. But, $\|(\hat{\Delta}[k] - \hat{\Delta})\|_\infty = \|\hat{\delta} - \hat{\delta}[k]\|_\infty$, where $\hat{\delta}[k] = [\frac{\pi_1[k]}{\eta_1[k]}, \frac{\pi_2[k]}{\eta_2[k]}, \ldots, \frac{\pi_n[k]}{\eta_n[k]}]'$ and $\hat{\delta} = [\frac{\pi_1}{\eta}, \frac{\pi_2}{\eta}, \ldots, \frac{\pi_n}{\eta}]'$. Define $H[k] = \text{diag}(\frac{1}{\eta_1[k]}, \frac{1}{\eta_2[k]}, \ldots, \frac{1}{\eta_n[k]})$, $H = \text{diag}(\frac{1}{\eta_1}, \frac{1}{\eta_2}, \ldots, \frac{1}{\eta_n})$, and $L = \lim_{k\to\infty} P_c^k$; then $\|\delta - \delta[k]\|_\infty = \|H\pi - H[k]\pi[k]\|_\infty = \|HL\pi[0] - H[k]P_c^k\pi[0]\|_\infty \leq n\|HL - H[k]P_c^k\|_\infty\|\pi[0]\|_\infty \leq n\|HL - H[k]P_c^k\|_\infty$. Note that $1 \geq 1/\eta_j \geq 1/\eta_j[k] \geq 1/n$, $\forall j, k$. Depending on the sign of the entry in $HL - H[k]P_c^k$ that results in $\|HL - H[k]P_c^k\|_\infty$, either $\|HL - H[k]P_c^k\|_\infty \leq \frac{1}{\eta_j[k]}\|L - P_c^k\|_\infty \leq \|L - P_c^k\|_\infty$, for some j, or $\|HL - H[k]P_c^k\|_\infty \leq \frac{1}{\eta}\|L - P_c^k\|_\infty \leq \|L - P_c^k\|_\infty$; in either case, $\|HL - H[k]P_c^k\|_\infty \leq \|L - P_c^k\|_\infty$. It is well-known (see, e.g., [29, Thm. 8.5.1]) that $\|L - P_c^k\|_\infty \leq C'k^{m_2-1}|\lambda_2|^k$, for some constant C' that only depends on matrix P_c, and the result follows. ■

Example 1: Consider the directed graph on the left of Fig. 1 with corresponding weight matrices $\overline{P}$ and P_c as shown on the right of the same figure, and assume that the initial values of the five nodes are $u = [9, 10, 2, 10, 7]'$, with average $\overline{u} = 7.6$. We simultaneously run Algorithm 1 and Algorithm 0. The plot on the left of Fig. 2 shows the evolution of two

quantities of interest against the iteration step k: (i) the solid line shows the distance of the resulting $P[k]$ from a doubly stochastic matrix as measured by $\sum_{j=1}^{5} |\sum_{i=1}^{5} p_{ji}[k]-1|$, and (ii) the dash-dotted line shows the distance of the resulting $x[k]$ in iteration (2) from average consensus as measured by $\sum_{j=1}^{5} |x_j[k] - \overline{u}|$. Note that Algorithm 1 converges rather quickly (within 10 iterations or so) to a doubly stochastic matrix. The plot in the middle of Fig. 2 shows how the value $x_j[k]$ of each node j in iteration (2) converges (within 25 iterations or so) to the average consensus value $\overline{u} = 7.6$. The entries of the limiting matrix P, denoted by P_{Alg1}, that the nodes converge to (refer to Theorem 2) are shown on the right of Fig. 2. The matrix P_{Alg1} is easily verified to be primitive doubly stochastic. □

B. Algorithm 2

In this algorithm, the nodes choose the weights on the out-neighbor edges so that they collectively maintain a weight matrix of the form $P[k] = \overline{P}\Delta[k]+(I-\Delta[k])$ as in (3). This is done by having each node j adjust its $\delta_j[k]$ periodically, once every k_0 steps for a sufficiently large k_0. More specifically, each node j will adjust its $\delta_j^r := \delta_j[rk_0] = \delta_j[rk_0+1] = \delta_j[rk_0+2] = ... = \delta_j[(r+1)k_0-1]$, and therefore all weights over which it has control (i.e, $p_{lj}^r := p_{lj}[rk_0] = p_{lj}[rk_0+1] = \cdots = p_{lj}[(r+1)k_0-1],\ \forall l \in \mathcal{N}_j^+ \cup \{j\}$).

Since the δ_j's are updated once every k_0 steps, we refer to the interval between consecutive changes of δ's as super-iterations (super-iteration r extends from steps rk_0 to $(r+1)k_0-1$). In order to update δ_j^r to δ_j^{r+1}, node j uses an auxiliary variable $\pi_j[k]$ which is updated by having the nodes run a parallel linear iteration as follows: $\pi[rk_0] = [1,1,...,1]'$ and $\pi[k+1] = P_r\pi[k]$ for $k = rk_0, rk_0+1, ..., (r+1)k_0-1$, where the $(j,i)^{th}$ entry of P_r is given by $P_r(j,i) = p_{ji}^r$. [Effectively, this means that each node j sets $\pi_j[rk_0] = 1$ and runs an update of the same form as the one in line 5 of Algorithm 1 with weights $p_{ji} = P_r(j,i)$.]

The constant k_0 is assumed to be large enough[1] so that $\pi[(r+1)k_0] = \pi^r$, where π^r is the unique eigenvector that satisfies $\pi^r = P_r\pi^r$. If $\pi_j^r := \pi_j[(r+1)k_0] \leq 1$, node j will increase, in a manner to be described, the value of δ_j^{r+1} (with respect to δ_j^r), and will decrease it otherwise.

1) Algorithm Formulation: Consider a strongly connected graph $\mathcal{G}_d = \{\mathcal{V}, \mathcal{E}\}$ and its corresponding matrix $\overline{P}$ defined below equation (3). Initially, each node starts with a set of weights as in Algorithm 1, i.e., $P_0 = \overline{P}\Delta_0 + (I-\Delta_0) = P_c$ where $\Delta_0 = \text{diag}(\delta_1^0, \delta_2^0, ..., \delta_n^0)$ is a diagonal matrix with $\delta_j^0 = \mathcal{D}_j^+/(1+\mathcal{D}_j^+)$ (so that $P_0 = P_c$). [Strong connectivity implies that each node can send its value to at least one other node, thus $\mathcal{D}_j^+ \geq 1$, and therefore $\delta_j^0 \geq \frac{1}{2}, \forall j$; this observation will be useful in our analysis later on.] For each $r = 0, 1, 2, ...$, node j updates $p_{lj}[rk_0],\ l \in \mathcal{N}_j^+ \cup \{j\}$, by executing Algorithm 2. The output of Algorithm 2 is used by the nodes as the input to Algorithm 0, which they execute in parallel, allowing them to reach consensus to the average of the u_j's (initial values).

[1]Selecting a large enough k_0 might not be an easy decision for the nodes to make in a distributed manner; however, as we will see (refer to the discussions in the examples for Algorithm 2 and in Section V), the choice of k_0 is not critical in practice.

Algorithm 2

1 **Input**: $\delta_j^0 = \mathcal{D}_j^+/(1+\mathcal{D}_j^+)$, k_0

for $r \geq 0$ **each node** j **does:**

2 **Output**: $P_r(j,j) = 1 - \delta_j^r$

3 $P_r(l,j) = \frac{\delta_j^r}{\mathcal{D}_j^+},\ \forall l \in \mathcal{N}_j^+$

4 **Set:** $\pi_j[rk_0] = 1$

for $k = rk_0, rk_0+1, ..., (r+1)k_0-1$ **each node** j **does:**

5 **Receive:** $P_r(j,i)\pi_i[k]$ from all $i \in \mathcal{N}_j^-$

Compute:

6 $\pi_j[k+1] = P_r(j,j)\pi_j[k] + \sum_{i\in\mathcal{N}_j^-} P_r(j,i)\pi_i[k]$

7 **Broadcast:** $\frac{\delta_j^r}{\mathcal{D}_j^+}\pi_j[k+1]$

Update:

8 $$\delta_j^{r+1} = \begin{cases} \pi_j^r\delta_j^r, & \text{if } \pi_j^r \leq 1, \\ 1 - \frac{1}{\pi_j^r}\big(1-\delta_j^r\big), & \text{if } \pi_j^r > 1, \end{cases}$$

where $\pi_j^r = \pi_j[(r+1)k_0]$

Note that line 6 of Algorithm 2 can be rewritten in matrix form as

$$\pi[k+1] = P_r\pi[k]\ , \quad \pi[rk_0] = [1,\ 1, \ldots, 1]',$$

where $k = rk_0, rk_0+1, ..., (r+1)k_0-1$, $P_r = \overline{P}\Delta_r+(I-\Delta_r)$ and $\Delta_r = \text{diag}\big(\delta_1^r, \delta_2^r, \ldots, \delta_j^r, \ldots, \delta_n^r\big)$.

2) Convergence Analysis: As a result of the execution of Algorithm 2, a sequence of column stochastic weight matrices, $P_0, P_1 \ldots, P_r, \ldots,$ is obtained. We next argue that (i) the matrices $P_0, P_1 \ldots, P_r, \ldots$ that result from the above process are all column stochastic and primitive, (ii) $\lim_{r\to\infty} \delta_j^r,\ \forall j$, exists, and (iii) $\lim_{k\to\infty} P_r = P$ is primitive and doubly stochastic. In the process, we also establish some monotonic properties of π^r and bound the rate at which P_r converges to P (in terms of the number of super-iterations).

Theorem 3: Let π^0 be the solution to $v = P_0 v$ normalized so that $\sum_{j=1}^n \pi_j^0 = n$, where $P_0 = \overline{P}\Delta_0 + (I-\Delta_0)$ with $\Delta_0 = \text{diag}(\delta_1^0, \delta_2^0, ..., \delta_n^0)$ and $\delta_j^0 = \mathcal{D}_j^+/(1+\mathcal{D}_j^+)$. Let $v = \pi^r$ be the solution to $v = P_r v$ normalized so that $\sum_{j=1}^n \pi_j^r = n$, and let $P_{r+1} = \overline{P}\Delta_{r+1} + (I-\Delta_{r+1})$, where $\Delta_{r+1} = \text{diag}\big(\delta_1^{r+1}, \delta_2^{r+1}, \ldots, \delta_n^{r+1}\big)$ is recursively obtained as described in line 8 of Algorithm 2. Then, the following hold:

1) If $\pi_j^r > 1$ for some $r \geq 0$, then $\pi_j^{r+1} < \pi_j^r$ and $\delta_j^{r+1} > \delta_j^r \geq 1/2$;
2) If $\pi_j^r \leq 1$ for some $r \geq 0$, then $\pi_j^{r+l} \leq 1, \forall l > 0$.

Proof: By induction. First, we consider the case $r = 0$ and analyze what happens at the end of the 0^{th} super-iteration. We partition the set $\mathcal{I} = \{1, 2, \ldots, n\}$ that indexes the nodes into

two sets as follows: $\mathcal{I} = \mathcal{A}_0 \bigcup \mathcal{B}_0$, where $\pi_j^0 > 1, \forall j \in \mathcal{A}_0$, and $\pi_j^0 \leq 1, \forall j \in \mathcal{B}_0$. We will establish that after the weight matrix is updated according to line 8 of Algorithm 2, it results in $\pi_j^1 < \pi_j^0, \forall j \in \mathcal{A}_0$, and $\pi_j^1 \leq 1, \forall j \in \mathcal{B}_0$. In other words, the new steady-state values of nodes whose steady-state value was below unity remain below unity; the new steady-state value of nodes whose steady-state value was above unity is reduced.

Let $P_0 = \overline{P}\Delta_0 + (I - \Delta_0) = P_c$ and $P_1 = \overline{P}\Delta_1 + (I - \Delta_1)$; then, we can write $P_1 = P_0\Delta_0^{-1}\Delta_1 + (I - \Delta_0^{-1}\Delta_1)$, and it follows from Lemma 1 that $\pi^1 = \alpha^1\Delta_1^{-1}\Delta_0\pi^0$ for some $\alpha^1 > 0$ (note that the requirements for Lemma 1 are satisfied: $0 < \dfrac{\delta_j^1}{\delta_j^0} \leq 1 < \dfrac{1}{1 - P_0(j,j)}$ for $j \in \mathcal{B}_0$, and $0 < \dfrac{\delta_j^1}{\delta_j^0} < \dfrac{1}{\delta_j^0} = \dfrac{1}{1 - (1 - \delta_j^0)} = \dfrac{1}{1 - P_0(j,j)}$ for $j \in \mathcal{A}_0$). Since P_0 is column stochastic, P_1 is also column stochastic and thus $\sum_{j=1}^n \pi_j^1 = n$. Then, since $\sum_{j=1}^n \pi_j^1 = \alpha^1 \sum_{j=1}^n \frac{\delta_j^0}{\delta_j^1}\pi_j^0$, we obtain that

$$\alpha^1 = \frac{n}{\sum_{j=1}^n \dfrac{\delta_j^0}{\delta_j^1}\pi_j^0}. \tag{5}$$

Now, we show that α^1 is smaller than or equal to one. Recall from line 8 of Algorithm 2 that $\delta_j^1/\delta_j^0 = \pi_j^0 \leq 1,\ \forall j \in \mathcal{B}_0$, and $(1 - \delta_j^1)/(1 - \delta_j^0) = 1/\pi_j^0 < 1,\ \forall j \in \mathcal{A}_0$. We can rewrite the denominator of the fraction in (5) as

$$\sum_{j=1}^n \frac{\delta_j^0}{\delta_j^1}\pi_j^0 = |\mathcal{B}_0| + \sum_{j \in \mathcal{A}_0} \frac{\delta_j^0(1 - \delta_j^0)}{\delta_j^1(1 - \delta_j^1)}. \tag{6}$$

Since $\delta_j^0 \geq 1/2$ and $\delta_j^1 > \delta_j^0,\ \forall j \in \mathcal{A}_0$, it can be easily checked that $\dfrac{\delta_j^0(1 - \delta_j^0)}{\delta_j^1(1 - \delta_j^1)} > 1,\ \forall j \in \mathcal{A}_0$, so that $\alpha^1 \leq \frac{n}{|\mathcal{A}_0| + |\mathcal{B}_0|} = 1$, where the equality is possible only if $\mathcal{A}_0 = \emptyset$. Now, we have the following two cases:

1) For $j \in \mathcal{A}_0$, $\pi_j^1 = \alpha^1 \dfrac{\delta_j^0}{\delta_j^1}\pi_j^0$, and since $\alpha^1 \leq 1$ and $\frac{\delta_j^0}{\delta_j^1} < 1$, we obtain that $\pi_j^1 < \pi_j^0$.
2) For $j \in \mathcal{B}_0$, $\pi_j^1 = \alpha^1 \dfrac{\delta_j^0}{\delta_j^1}\pi_j^0 = \alpha^1$, and since $\alpha^1 \leq 1$, we obtain that $\pi_j^1 \leq 1$.

Now we proceed with the inductive step. At super-iterations $r-1$ and r, we have that $P_{r-1} = \overline{P}\Delta_{r-1} + (I - \Delta_{r-1})$ and $P_r = \overline{P}\Delta_r + (I - \Delta_r)$, with both P_{r-1} and P_r being column stochastic matrices. As above, we can write $P_r = P_{r-1}\Delta_{r-1}^{-1}\Delta_r + (I - \Delta_{r-1}^{-1}\Delta_r)$, and it follows from Lemma 1 that $\pi^r = \alpha^r\Delta_r^{-1}\Delta_{r-1}\pi^{r-1}$ for some $\alpha^r > 0$. Now assume that (i) $\mathcal{I} = \mathcal{A}_{r-1} \bigcup \mathcal{B}_{r-1}$ (with $\mathcal{A}_{r-1} \bigcap \mathcal{B}_{r-1} = \emptyset$) where $\pi_j^{r-1} > 1, \forall j \in \mathcal{A}_{r-1}$ and $\pi_j^{r-1} \leq 1, \forall j \in \mathcal{B}_{r-1}$; (ii) $\delta_j^{r-1} \geq 1/2,\ \forall j \in \mathcal{A}_{r-1}$; and (iii) $\alpha^r \leq 1$. Then, from the induction hypothesis we have

$$\begin{aligned} \pi_j^r < \pi_j^{r-1},\ \delta_j^r > \delta_j^{r-1} \geq 1/2, \qquad & \forall j \in \mathcal{A}_{r-1}, \\ \pi_j^r \leq 1, \qquad & \forall j \in \mathcal{B}_{r-1}. \end{aligned} \tag{7}$$

At super-iteration $r+1$, we have that $\pi^{r+1} = \alpha^{r+1}\Delta_{r+1}^{-1}\Delta_r\pi^r$, for some $\alpha^{r+1} > 0$, where the diagonal entries of Δ_{r+1} are obtained from line 8 of Algorithm 2 as follows:

$$\delta_j^{r+1} = \begin{cases} \pi_j^r\delta_j^r, & \forall j \in \mathcal{B}_r, \\ 1 - \frac{1}{\pi_j^r}\big(1 - \delta_j^r\big), & \forall j \in \mathcal{A}_r, \end{cases}$$

where $\mathcal{I} = \mathcal{A}_r \bigcup \mathcal{B}_r$ (with $\mathcal{A}_r \bigcap \mathcal{B}_r = \emptyset$), and $\mathcal{A}_r \subseteq \mathcal{A}_{r-1}$, i.e., some of the nodes whose steady-state value was above unity at super-iteration $r-1$ might have gone at or below unity at super-iteration r. Since P_r is a primitive column stochastic matrix, then P_{r+1} is also a primitive column stochastic matrix. Thus, since $\sum_{j=1}^n \pi_j^{r+1} = n$, and (from Lemma 1) $\sum_{j=1}^n \pi^{r+1} = \alpha^{r+1}\sum_{j=1}^n \dfrac{\delta_j^r}{\delta_j^{r+1}}\pi_j^r$, we obtain that

$$\alpha^{r+1} = \frac{n}{\sum_{j=1}^n \dfrac{\delta_j^r}{\delta_j^{r+1}}\pi_j^r}. \tag{8}$$

As before, since $\delta_j^{r+1} > \delta_j^r \geq 1/2,\ \forall j \in \mathcal{A}_r$, we have $\alpha^{r+1} \leq \dfrac{n}{|\mathcal{A}_r| + |\mathcal{B}_r|} = 1$, where the equality holds only if $\mathcal{A}_r = \emptyset$. To show that $\pi_j^{r+1} < \pi_j^r$ and $\delta_j^{r+1} > \delta_j^r \geq 1/2,\ \forall j \in \mathcal{A}_r$, and $\pi_j^{r+1} \leq 1,\ \forall j \in \mathcal{B}_r$, one can follow the steps used to obtain the corresponding inequalities for the case when $r = 0$. ■

The following corollary establishes convergence rates for super-iteration evolution. It is important to note that the convergence rate established in this corollary does not take into account the convergence rate within each super-iteration, i.e., it does not take into account how fast $[1, 1, ..., 1]'$ converges to π^r for every $P_r,\ r = 0, 1, \ldots$.

Corollary 2: Let $P_r = \overline{P}\Delta_r + (I - \Delta_r)$ be the weight matrix at the r^{th} super-iteration that results from Algorithm 2, and let $v = \pi^r$ be the steady-state solution of $v = P_r v$ normalized so that $\sum_{j=}^n \pi_j^r = n$. If $\pi_j^0 > 1$, then π_j^r either becomes smaller than one for some finite r (and subsequently remains below one) or asymptotically converges to one with worst-case convergence given by

$$\pi_j^{r+1} \leq \frac{1}{\big(\frac{n-1}{n}\big)^{r+1} + \big(1 - \big(\frac{n-1}{n}\big)^{r+1}\big)\pi_j^0}\pi_j^0.$$

Proof: For every $j \in \mathcal{A}_r$, the denominator of (8) can be written as $\sum_{i=1}^n \dfrac{\delta_i^r}{\delta_i^{r+1}}\pi_i^r = |\mathcal{B}_r| + \sum_{i \in \mathcal{A}_r, i \neq j} \dfrac{\delta_i^r(1 - \delta_i^r)}{\delta_i^{r+1}(1 - \delta_i^{r+1})} + \dfrac{\delta_j^r}{\delta_j^{r+1}}\pi_j^r$. As we have seen in the proof of Theorem 3, $\dfrac{\delta_i^r(1 - \delta_i^r)}{\delta_i^{r+1}(1 - \delta_i^{r+1})} \geq 1,\ \forall i \in \mathcal{A}_r$; therefore, $|\mathcal{B}_r| + \sum_{i \in \mathcal{A}_r, i \neq j} \dfrac{\delta_i^r(1 - \delta_i^r)}{\delta_i^{r+1}(1 - \delta_i^{r+1})} + \dfrac{\delta_j^r}{\delta_j^{r+1}}\pi_j^r \geq |\mathcal{A}_r| + |\mathcal{B}_r| + \dfrac{\delta_j^r}{\delta_j^{r+1}}\pi_j^r - 1 = n + \dfrac{\delta_j^r}{\delta_j^{r+1}}\pi_j^r - 1$ (with equality if $\mathcal{A}_r = \{j\}$). Then, it follows from (8) that

$$\alpha^{r+1} \leq \frac{n}{n + \dfrac{\delta_j^r}{\delta_j^{r+1}}\pi_j^r - 1},\ \forall j \in \mathcal{A}_r.$$

Since $\pi_j^{r+1} = \alpha^{r+1}\frac{\delta_j^r}{\delta_j^{r+1}}\pi_j^r$, we establish that

$$\pi_j^{r+1} \leq \frac{n}{\big(n-1\big)\frac{\delta_j^{r+1}}{\delta_j^r} + \pi_j^r}\pi_j^r, \ \forall j \in \mathcal{A}_r, \tag{9}$$

and since $\frac{\delta_j^{r+1}}{\delta_j^r} > 1, \forall j \in \mathcal{A}_r$, it follows from (9) that

$$\pi_j^{r+1} \leq \frac{n}{\big(n-1\big) + \pi_j^r}\pi_j^r, \ \forall j \in \mathcal{A}_r. \tag{10}$$

It is easy to see that the right side of inequality (10) monotonically decreases; in fact, we can use this recurrence relation to bound π_j^{r+1} in terms of π_j^0 as

$$\pi_j^{r+1} \leq \frac{1}{\big(\frac{n-1}{n}\big)^{r+1} + \frac{1}{n}\frac{1-\big(\frac{n-1}{n}\big)^{r+1}}{1-\frac{n-1}{n}}\pi_j^0}\pi_j^0, \ \forall j \in \mathcal{A}_0, \tag{11}$$

from where the result follows. ■

The next theorem establishes that Algorithm 2 converges to a solution where all the nodes reach a set of weights that form a primitive doubly stochastic weight matrix P.

Theorem 4: Let $P_r = \overline{P}\Delta_r + (I - \Delta_r)$ be the weight matrix that results from the r^{th} super-iteration of Algorithm 2, and let $v = \pi^r$ be the steady-state solution of $v = P_r v$ normalized so that $\sum_{j=1}^n \pi_j^r = n$. Then, the following hold:

1) $\lim_{r\to\infty} \pi_j^r = 1, \ \forall j = 1, 2, \ldots, n$;
2) $0 < \delta_j^r < 1, \forall r \geq 0, \ \forall j = 1, 2, \ldots, n$;
3) $\Delta = \text{diag}(\delta_1, \delta_2, \ldots, \delta_n)$, where $\delta_j = \lim_{r\to\infty} \delta_j^r, \ \forall j = 1, 2, \ldots, n$;
4) $0 < \delta_j \leq 1, \ \forall j = 1, 2, \ldots, n$, and $\delta_i < 1$ for some i.
5) $\lim_{r\to\infty} P_r = P$ elementwise, where P is a primitive doubly stochastic matrix.

Proof: Conclusion (1) can be proved as follows. For every $j \in \mathcal{A}_0$, we take the limit of (11) as $r \to \infty$ to obtain that

$$\lim_{r\to\infty} \pi_j^{r+1} \leq \lim_{r\to\infty} \frac{1}{\big(\frac{n-1}{n}\big)^{r+1} + \big(1 - \big(\frac{n-1}{n}\big)^{r+1}\big)\pi_j^0}\pi_j^0 = 1. \tag{12}$$

Now, we know that either (i) (12) holds in the limit for all $j \in \mathcal{A}_0$ or (ii) at some finite r the value of π_j^r becomes smaller or equal to one (and remains such). Either situation ensures that $\lim_{r\to\infty} \sum_{j\in\mathcal{A}_0} \pi_j^r \leq |\mathcal{A}_0|$. This observation together with the fact that P_r is column stochastic for every $r = 1, 2, \ldots$, which ensures that $\sum_{j=1}^n \pi_j^r = n$ for every $r = 1, 2, \ldots$, implies that $\lim_{r\to\infty} \sum_{j\in\mathcal{B}_0} \pi_j^r \geq |\mathcal{B}_0|$. However, Theorem 3 established that if $\pi_j^0 \leq 1$, then $\pi_j^r \leq 1, \forall r > 0$; thus, $\lim_{r\to\infty} \sum_{j\in\mathcal{B}_0} \pi_j^r = |\mathcal{B}_0|$ and also $\lim_{r\to\infty} \sum_{j\in\mathcal{A}_0} \pi_j^r = |\mathcal{A}_0|$. The above lead to the conclusion that $\lim_{r\to\infty} \pi_j^r = 1, \ \forall j = 1, 2, \ldots, n$.

Conclusion (2) can be proved by induction. Since $0 < \delta_j^0 < 1, \forall j = 1, 2, \ldots, n$, and $\pi_j^0 > 0, \forall j = 1, 2, \ldots, n$, it follows from line 8 of Algorithm 2 that $0 < \delta_j^1 < 1, \ \forall j = 1, 2, \ldots, n$. Assuming that $0 < \delta_j^{r-1} < 1, \forall j = 1, 2, \ldots, n$, and since $\pi_j^{r-1} > 0, \forall j = 1, 2, \ldots, n$, it follows from line 8 of Algorithm 2 that $0 < \delta_j^r < 1, \ \forall j = 1, 2, \ldots, n$.

Conclusion (3) can be proved as follows. The existence of $\delta_j, \ \forall j = 1, 2, \ldots, n$, follows from Conclusions (1) and (2), and the update rule in line 8 of Algorithm 2. Since $\lim_{r\to\infty} \pi_j^r = 1, \ \forall j = 1, 2, \ldots, n$; $\delta_j^r \neq 0, \forall j, \forall r \geq 0$; and $1 - \delta_j^r \neq 0, \forall j, \forall r \geq 0$, then, for some $r = r_0 > 0$

$$\begin{aligned} \lim_{r\to\infty} \frac{\delta_j^{r_0+r+1}}{\delta_j^{r_0+r}} &= \lim_{r\to\infty} \pi_j^{r_0+r} = 1, \ \forall j \in \mathcal{B}_{r_0}, \\ \lim_{r\to\infty} \frac{1-\delta_j^{r_0+r}}{1-\delta_j^{r_0+r+1}} &= \lim_{r\to\infty} \pi_j^{r_0+r} = 1, \ \forall j \in \mathcal{A}_{r_0}, \end{aligned}$$

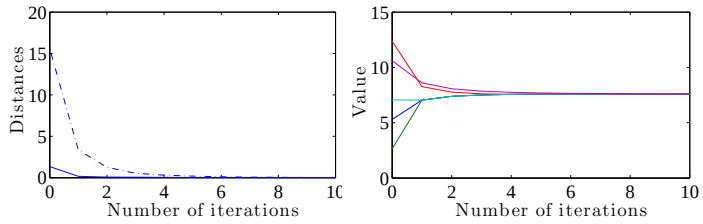
consequently, $\lim_{r\to\infty} \delta_j^r = \delta_j$ for some δ_j for all $j = 1, 2, \ldots, n$.

The first inequality in Conclusion (4) can be shown as follows. If $\delta_j = 1, \forall j = 1, 2, \ldots, n$, it means that after some $r = r_0 > 0$, all nodes update their δ_j^r's according to the second condition in line 8 of Algorithm 2, which suggests that $\pi_j^{r_0+k} > 1, \forall j = 1, 2, \ldots, n, \ \forall k > 0$. But this is a contradiction, as it violates the fact that $\sum_{j=1}^n \pi_j^r = n, \ \forall r \geq 0$; thus, there has to exist at least one node i such that $\delta_i < 1$. The second inequality in Conclusion (4) can be shown as follows. If $\delta_j = 0$ for some j, this means that all the nodes i that send values to j (i.e., $i \in \mathcal{N}_j^-$) must be in the same situation as j, i.e., $\delta_i = 0, \forall i \in \mathcal{N}_j^-$; otherwise, $\pi_j > 1$ or $\pi_i = 0, \forall i \in \mathcal{N}_j^-$, which would contradict the fact that $\pi_j = 1, \forall j = 1, 2, \ldots, n$. Following the same argument, for every node $i \in \mathcal{N}_j^-$, it must also hold that all the nodes l in the in-neighborhood of i, i.e., $l \in \mathcal{N}_i^-$ must be in the same situation as i, i.e., $\delta_l = 0, \forall l \in \mathcal{N}_i^-$. Since the number of nodes is finite and the graph is strongly connected, by repeatedly using this argument, we conclude that $\delta_j = 0, \forall j = 1, 2, \ldots, n$. The δ_j^r's are updated according to line 8 of Algorithm 2 and the only way that all δ_j's become 0 is if after some $r = r_0 > 0$, all nodes update their δ_j^r's according to the first condition in line 8 of Algorithm 2. This, however, would suggest that $\pi_j^{r_0+k} \leq 1, \ \forall j = 1, 2, \ldots, n, \ \forall k > 0$, and the only way this is possible is if $\pi_j^{r_0+k} = 1, \ \forall j = 1, 2, \ldots, n, \ \forall k > 0$ (because we have $\sum_{j=1}^n \pi_j^r = n, \ \forall r \geq 0$). If, however, $\pi_j^{r_0} = 1$ for all j, that would imply that $\delta_j^{r_0+k} = \delta_j^{r_0} > 0$ (by Conclusion (2)) for all $j = 1, 2, ..., n$ and all $k > 0$. This is a contradiction and thus $\delta_j > 0, \ \forall j$.

Conclusion (5) immediately follows from Conclusions (1) and (4). ■

Corollary 3: Consider (2) with $P[k] = \overline{P}\Delta[k] + (I - \Delta[k])$, where $\Delta[k] = \text{diag}(\delta_1[k], \delta_2[k], \ldots, \delta_n[k])$ are obtained as in line 8 of Algorithm 2. Then, $\lim_{k\to\infty} x_j[k] = \frac{\sum_{l=1}^n u_l}{n}, \ \forall j$. Furthermore, for all j, $|x_j[k]| \leq \infty, \ \forall k \geq 0$.

Proof: Theorem 4 established that the matrices in the sequence $P_0, P_1, \ldots, P_r, \ldots$, where $P_r = P[rk_0] = P[rk_0 + 1] = \cdots = P[(r+1)k_0 - 1]$, for some large enough k_0, are column stochastic and primitive, and that $\lim_{r\to\infty} P_r = P$ elementwise, where P is primitive doubly stochastic. The result then follows immediately from Theorem 1. ■



$$P_{Alg2} = \begin{bmatrix} 0.5913 & 0 & 0 & 0.4087 & 0 \\ 0.2043 & 0.7957 & 0 & 0 & 0 \\ 0.2044 & 0.1021 & 0.2848 & 0 & 0.4087 \\ 0 & 0 & 0 & 0.5913 & 0.4087 \\ 0 & 0.1022 & 0.7152 & 0 & 0.1826 \end{bmatrix}$$

Fig. 3. (left) Distance from a doubly stochastic matrix (solid line) and distance from average consensus (dash-dotted line) vs. number of super-iterations (r); (middle) node values π_j^r vs. number of super-iterations (r); (right) limiting matrix P_{Alg2}.

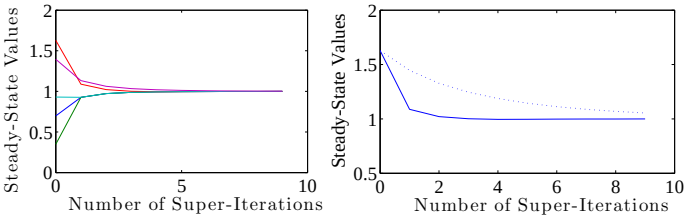


Fig. 4. (left) Values of π_j^r for $j = 1, 2, 3, 4, 5$ vs. number of super-iterations r; (right) Values of π_3^r together with its bound in Theorem 4 vs. number of super-iterations r.

Example 2: Consider again the directed graph in Fig. 1, and assume that the initial values of the five nodes are $u = [9, 10, 2, 10, 7]'$, with average $\overline{u} = 7.6$. Algorithm 2 operates by running super-iterations to asymptotically obtain a doubly stochastic matrix: at the end of super-iteration r, the weight matrix is updated to $P_r = \overline{P}\Delta_r + (I - \Delta_r)$, where $\Delta_r = \text{diag}(\delta_1^r, \delta_2^r, \ldots, \delta_n^r)$ is iteratively obtained (at the end of each super-iteration) according to line 8 of Algorithm 2. At the same time, we simultaneously run Algorithm 0 with $x[0] = u$ and $P[k] = P_{\lfloor k/k_0 \rfloor}$ (where k_0 is the number of steps within each super-iteration) in order to reach average consensus.

We allow each super-iteration to take $k_0 = 100$ iterations (though similar[2] behavior is observed for much smaller k_0). The plot on the left of Fig. 3 shows two quantities of interest against the super-iteration step r: (i) the solid line shows the distance of the weight matrix P_r from a doubly stochastic matrix at the end of super-iteration r, and (ii) the dash-dotted line shows the distance of the resulting $x[rk_0]$ from average consensus at the end of super-iteration r. Note that Algorithm 2 converges rather quickly (within 5 super-iterations or so) to a doubly stochastic matrix, while Algorithm 0 converges to the average consensus value $\overline{u} = 7.6$ within 10 super-iterations or so (see the plot in the middle of Fig. 3). The limiting doubly stochastic matrix P, denoted by P_{Alg2}, that the nodes converge to is displayed on the right of Fig. 3.

The plot on the left of Fig. 4 shows the steady-state values π_j^r, for $j = 1, 2, 3, 4, 5$, at the end of each super-iteration r. The plot verifies that indeed nodes with $\pi_j^{r-1} > 1$ decrease in value (i.e., $\pi_j^r < \pi_j^{r-1}$) and nodes with $\pi_j^{r-1} \leq 1$ remain below 1 (i.e., $\pi_j^r \leq 1$). The plot on the right of Fig. 4 focuses on node 3 (whose π_3^0 is initially above 1) and plots π_3^r as a function of the number of super-iterations r; π_3^r converges to 1 and is bounded above by the bound in Corollary 2. □

[2]It will be evident from the proof of Algorithm 3 in the next subsection that even $k_0 = 1$ will lead to convergence; however, for the monotonic properties of π_j^r (as discussed in the proof of convergence for Algorithm 2 in this subsection) to hold, one needs large enough k_0.

C. Algorithm 3

Each node j initially chooses its self-weight and the weights on its out-neighbor edges as in Algorithms 1 and 2, i.e., $p_{lj}[0] = 1/(1 + \mathcal{D}_j^+)$ for all $l \in \mathcal{N}_j^+ \cup \{j\}$. However, unlike Algorithm 2, each node j updates its $\delta_j[k]$ at every iteration k and, along with it, all the weights it controls, i.e, all $p_{lj}[k],\ \forall l \in \mathcal{N}_j^+ \cup \{j\}$. In order for node j to update its $\delta_j[k+1]$, each node $i \in \mathcal{N}_j^-$ needs to send its weight $p_{ji}[k]$ to node j. Node j then computes $\sum_{i \in \mathcal{N}_j^- \cup \{j\}} p_{ji}[k]$, and if this sum is greater than one, node j increases the value of $\delta_j[k+1]$ (with respect to $\delta_j[k]$) and decreases it otherwise.

1) Algorithm Formulation: Consider a strongly connected graph $\mathcal{G}_d = \{\mathcal{V}, \mathcal{E}\}$. Initially, each node j sets $\delta_j[0] = \frac{\mathcal{D}_j^+}{1+\mathcal{D}_j^+}$ and $p_{lj}[0] = \frac{\delta_j[0]}{\mathcal{D}_j^+},\ \forall l \in \mathcal{N}_j^+ \cup \{j\}$ ($p_{lj}[0] = 0$ otherwise). At each time step $k \geq 0$, each node j updates $p_{lj}[k],\ l \in \mathcal{N}_j^+ \cup \{j\}$, by executing Algorithm 3. The output of Algorithm 3 is used by the nodes as the input to Algorithm 0, which they execute in parallel, allowing them to reach consensus to the average of the u_j's (initial values).

Algorithm 3

1 **Input**: $\delta_j[0] = \frac{\mathcal{D}_j^+}{1+\mathcal{D}_j^+}$

for $k \geq 0$ **each node** j **does:**

2 **Output**: $p_{jj}[k] = 1 - \delta_j[k]$

3 $p_{lj}[k] = \frac{1}{\mathcal{D}_j^+}\delta_j[k],\ \forall l \in \mathcal{N}_j^+$

4 **Receive:** $p_{ji}[k]$ from all $i \in \mathcal{N}_j^-$

Compute:

5 $e_j[k] = \sum_{i \in \mathcal{N}_j^- \cup \{j\}} p_{ji}[k]$

6 $\delta_j[k+1] = \begin{cases} \delta_j[k]e_j[k], & \text{if } e_j[k] \leq 1 \\ 1 - \frac{1}{e_j[k]}\big(1 - \delta_j[k]\big), & \text{if } e_j[k] > 1 \end{cases}$

7 **Broadcast:** $p_{lj}[k+1] = \frac{1}{\mathcal{D}_j^+}\delta_j[k+1]$ to all $l \in \mathcal{N}_j^+$

2) Convergence Analysis: The execution of Algorithm 3 results in a sequence of column stochastic weight matrices $P[0], P[1], \ldots, P[k], \ldots$, where $P[k] = \overline{P}\Delta[k] + (I - \Delta[k])$, with $\Delta[k] = \text{diag}\big(\delta_1[k], \delta_2[k], \ldots, \delta_n[k]\big)$. Through the weight update in lines 5–6 of Algorihm 3, each node tries to make its corresponding row-sum in $P[k]$ closer to one, but in the

process it might make other row-sums worse in terms of their closeness to one. We will next argue, however, that $\lim_{k\to\infty} P[k] = P$, where P is primitive doubly stochastic. The following lemmas are needed for establishing our convergence results.

Lemma 4: Let $P[k-1] = \overline{P}\Delta[k-1] + (I - \Delta[k-1])$ and $P[k] = \overline{P}\Delta[k] + (I - \Delta[k])$ be the weight matrices at steps $k-1$ and k respectively, where $\Delta[k-1] = \text{diag}\big(\delta_1[k-1], \delta_2[k-1], \ldots, \delta_n[k-1]\big)$ and $\Delta[k] = \text{diag}\big(\delta_1[k], \delta_2[k], \ldots, \delta_n[k]\big)$, with $0 < \delta_j[k-1] < 1$ and $0 < \delta_j[k] < 1$, $\forall j$. Assume that for a specific node l, the relation between $\delta_l[k-1]$ and $\delta_l[k]$ is given by line 6 of Algorithm 3 whereas for all $i \neq l$, we have $\delta_i[k-1] = \delta_i[k]$. Then, the following hold:

1) If $\sum_i p_{li}[k-1] \leq 1$, then $\sum_i p_{li}[k-1] \leq \sum_i p_{li}[k] \leq 1$;
2) If $\sum_i p_{li}[k-1] > 1$, then $\sum_i p_{li}[k-1] > \sum_i p_{li}[k] > 1$.

Proof: If $\sum_i p_{li}[k-1] \leq 1$, then $\delta_l[k] = \delta_l[k-1] \sum_i p_{li}[k-1]$, and since $p_{ll}[k] = 1 - \delta_l[k]$, it follows that $p_{ll}[k] = 1 - (1 - p_{ll}[k-1]) \sum_i p_{li}[k-1]$. Since $\delta_i[k-1] = \delta_i[k], \forall i \neq l$, we have $p_{li}[k-1] = p_{li}[k]$, $\forall i \neq l$. Then, $\sum_i p_{li}[k] = p_{ll}[k] + \sum_{i\in\mathcal{N}_l^-} p_{li}[k]$ can be shown to satisfy $\sum_i p_{li}[k] = 1 + p_{ll}[k-1]\big(\sum_i p_{li}[k-1] - 1\big) \leq 1$ (since $\sum_i p_{li}[k-1] - 1 \leq 0$). Also, since $p_{li}[k-1] = p_{li}[k]$, $\forall i \neq l$, and $p_{ll}[k] \geq p_{ll}[k-1]$, we have $\sum_i p_{li}[k-1] \leq \sum_i p_{li}[k]$.

Similarly, if $\sum_i p_{li}[k-1] > 1$, then $\delta_l[k] = 1 - (1 - \delta_l[k-1])\frac{1}{\sum_i p_{li}[k-1]}$, and since $p_{ll}[k] = 1 - \delta_l[k]$, it follows that $p_{ll}[k] = p_{ll}[k-1]\frac{1}{\sum_i p_{li}[k-1]}$. Then, $\sum_i p_{li}[k] = p_{ll}[k] + \sum_{i\in\mathcal{N}_l^-} p_{li}[k]$ can be shown to satisfy $\sum_i p_{li}[k] = 1 + \sum_{i\in\mathcal{N}_l^-} p_{li}[k-1]\left(1 - \frac{1}{\sum_i p_{li}[k-1]}\right) > 1$ (since $\frac{1}{\sum_i p_{li}[k-1]} < 1$). Also, since $p_{li}[k-1] = p_{li}[k]$, $\forall i \neq l$, and $p_{ll}[k] < p_{ll}[k-1]$, we have $\sum_i p_{li}[k-1] > \sum_i p_{li}[k]$. ■

Define $\varepsilon[k] = \sum_{l=1}^n \varepsilon_l[k]$, with $\varepsilon_l[k] = |\sum_i p_{li}[k] - 1|$ (note that $\varepsilon_l[\cdot] \geq 0$ and $\varepsilon[\cdot] \geq 0$). Under the conditions of Lemma 4, the next lemma establishes that $\varepsilon[k]$ is non-increasing.

Lemma 5: Let $P[k-1] = \overline{P}\Delta[k-1] + (I - \Delta[k-1])$ and $P[k] = \overline{P}\Delta[k] + (I - \Delta[k])$ be the weight matrices at steps $k-1$ and k respectively, where $\Delta[k-1] = \text{diag}\big(\delta_1[k-1], \delta_2[k-1], \ldots, \delta_n[k-1]\big)$ and $\Delta[k] = \text{diag}\big(\delta_1[k], \delta_2[k], \ldots, \delta_n[k]\big)$, with $0 < \delta_j[k-1] < 1$ and $0 < \delta_j[k] < 1$, $\forall j$. Assume that for a specific node l, the relation between $\delta_l[k-1]$ and $\delta_l[k]$ is given by line 6 of Algorithm 3, whereas for all $i \neq l$, we have $\delta_i[k-1] = \delta_i[k]$. Then, $\varepsilon[k] \leq \varepsilon[k-1]$ holds for every $k \geq 1$.

Proof: Assume that $\sum_i p_{li}[k-1] \leq 1$; it follows from Lemma 4 that $\sum_i p_{li}[k] \leq 1$; and from line 6 of Algorithm 3 that $p_{ll}[k] \geq p_{ll}[k-1]$. Then,

$$\begin{aligned}\varepsilon_l[k] &= 1 - \sum_i p_{li}[k] = 1 - p_{ll}[k] + \sum_{i\in\mathcal{N}_l^-} p_{li}[k-1] \\ &\leq 1 - p_{ll}[k-1] + \sum_{i\in\mathcal{N}_l^-} p_{li}[k-1] = \varepsilon_l[k-1]. \quad (13)\end{aligned}$$

If we let $p_{ll}[k] = p_{ll}[k-1] + \Delta p_{ll}[k]$ for some positive $\Delta p_{ll}[k]$, it follows from (13) that $\varepsilon_l[k] - \varepsilon_l[k-1] = -\Delta p_{ll}[k]$ and also that $p_{ml}[k] = p_{ml}[k-1] - \frac{1}{\mathcal{D}_l^+}\Delta p_{ll}[k]$ for all $m \in \mathcal{N}_l^+$; this can be used to establish that

$$|\varepsilon_m[k] - \varepsilon_m[k-1]| \leq \frac{1}{\mathcal{D}_l^+}\Delta p_{ll}[k], \ \forall m \in \mathcal{N}_l^+.$$

Note that $p_{ll}[k] + \sum_{i\neq l} p_{li}[k] = p_{ll}[k-1] + \sum_{i\neq l} p_{li}[k-1] - \frac{1}{\mathcal{D}_l^+}\Delta p_{ll}[k]$. Then, $|(p_{ll}[k] + \sum_{i\neq l} p_{li}[k] - 1) - (p_{ll}[k-1] + \sum_{i\neq l} p_{li}[k-1] - 1)| = \frac{1}{\mathcal{D}_l^+}\Delta p_{ll}[k]$, but

$$\begin{aligned}&\Big|(p_{ll}[k] + \sum_{i\neq l} p_{li}[k] - 1) - (p_{ll}[k-1] + \sum_{i\neq l} p_{li}[k-1] - 1)\Big| \geq \\ &\Big||p_{ll}[k] + \sum_{i\neq l} p_{li}[k] - 1| - |p_{ll}[k-1] + \sum_{i\neq l} p_{li}[k-1] - 1|\Big| = \\ &|\varepsilon_l[k] - \varepsilon_l[k-1]|.\end{aligned}$$

Thus, the worst case occurs when node l can only send information to nodes $m \in \mathcal{N}_l^+$ that also satisfy $\sum_{i\in\mathcal{N}_m^-\cup\{m\}} p_{mi}[k-1] \leq 1$, which means that $\varepsilon_m[k] - \varepsilon_m[k-1] = \frac{1}{\mathcal{D}_l^+}\Delta p_{ll}[k]$ and thus, in the worst case, $\varepsilon[k] - \varepsilon[k-1] = \sum_{l=1}^n \varepsilon_l[k] - \sum_{l=1}^n \varepsilon_l[k-1] = 0$. Otherwise $\varepsilon[k] - \varepsilon[k-1] < 0$. A similar argument can be made when $\sum_i p_{li}[k-1] > 1$. ■

By showing that the error $\varepsilon[k]$ converges to zero, the following theorem establishes convergence to doubly stochastic form, even when nodes simultaneously update their self-weights and the weights on their out-neighbor edges.

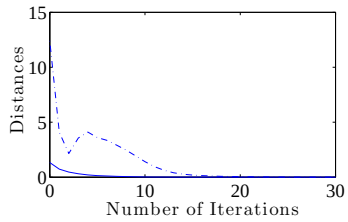
Theorem 5: Let $P[k] = \overline{P}\Delta[k] + (I - \Delta[k])$ be the weight matrix that results after the k^{th} step of the execution of Algorithm 3. Then, $\lim_{k\to\infty} P[k]$ exists and it is a doubly stochastic and primitive matrix $P = \overline{P}\Delta + (I - \Delta)$, where $\Delta = \text{diag}(\delta_1, \delta_2, \ldots, \delta_n)$ with $\delta_j = \lim_{k\to\infty} \delta_j[k]$ satisfying $0 < \delta_j \leq 1$, $\forall j = 1, 2, \ldots, n$, and $\delta_i < 1$ for at least one $i \in \{1, 2, \ldots, n\}$.

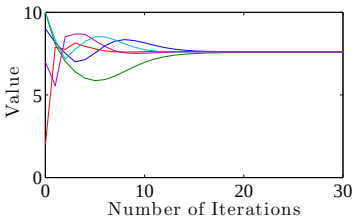
Proof: We will first show that $\varepsilon[k] = \sum_{j=1}^n \varepsilon_j[k] = \sum_{j=1}^n |\sum_{i\in\mathcal{N}_j^-\cup\{j\}} p_{ji}[k] - 1|$ decreases monotonically with k. To see this, we will prove that $\varepsilon[k] - \varepsilon[k-1] < 0$, unless $\varepsilon[k-1] = 0$ in which case $\varepsilon[k] = \varepsilon[k-1] = 0$. As in the proof of Theorem 3, we define the sets $\mathcal{A}_k$ and $\mathcal{B}_k$ as a partition of the index set $\mathcal{I} = \{1, 2, \ldots, n\}$ (i.e., $\mathcal{I} = \mathcal{A}_k \bigcup \mathcal{B}_k$ and $\mathcal{I} = \mathcal{A}_k \bigcap \mathcal{B}_k = \emptyset$), where $\sum_{i\in\mathcal{N}_j^-\cup\{j\}} p_{ji}[k] > 1, \forall j \in \mathcal{A}_k$, and $\sum_{i\in\mathcal{N}_j^-\cup\{j\}} p_{ji}[k] \leq 1, \forall j \in \mathcal{B}_k$. Unlike that proof, we will see that it is not necessarily true that $\mathcal{A}_k \subseteq \mathcal{A}_{k-1}$.

We can rewrite $\varepsilon[k-1]$ as $\varepsilon[k-1] = \sum_{j\in\mathcal{A}_{k-1}} \varepsilon_j[k-1] + \sum_{j\in\mathcal{B}_{k-1}} \varepsilon_j[k-1]$. For all $j \in \mathcal{A}_{k-1}$, it follows from (13) that $\varepsilon_j[k] - \varepsilon_j[k-1] = |\sum_i p_{ji}[k] - 1| - |\sum_i p_{ji}[k-1] - 1| = \Big|\sum_i p_{ji}[k-1] + \Delta p_{jj}[k] + \sum_{i\neq j} \Delta p_{ji}[k] - 1\Big| - (\sum_i p_{ji}[k-1] - 1) \leq |\sum_i p_{ji}[k-1] + \Delta p_{jj}[k] - 1| + \Big|\sum_{i\neq j} \Delta p_{ji}[k]\Big| - (\sum_i p_{ji}[k-1] - 1)$. Then, since $\sum_{i\in\mathcal{N}_j^-\cup\{j\}} p_{ji}[k-1] + \Delta p_{jj}[k] > 1$, we obtain that

$$\begin{aligned}\varepsilon_j[k] - \varepsilon_j[k-1] &\leq \Delta p_{jj}[k] + |\sum_{i\neq j} \Delta p_{ji}[k]| \\ &\leq \Delta p_{jj}[k] + \sum_{i\neq j} |\Delta p_{ji}[k]|,\end{aligned}$$

where $\Delta p_{jj}[k] < 0$. For all $j \in \mathcal{B}_{k-1}$, it follows from a similar argument that $\varepsilon_j[k] - \varepsilon_j[k-1] \leq -\Delta p_{jj}[k] + |\sum_{i\neq j} \Delta p_{ji}[k]| \leq -\Delta p_{jj}[k] + \sum_{i\neq j} |\Delta p_{ji}[k]|$, where $\Delta p_{jj}[k] > 0$. Thus, without loss of generality, we can establish





$$P_{Alg3} = \begin{bmatrix} 0.5756 & 0 & 0 & 0.4244 & 0 \\ 0.2122 & 0.7878 & 0 & 0 & 0 \\ 0.2122 & 0.1061 & 0.2572 & 0 & 0.4245 \\ 0 & 0 & 0 & 0.5756 & 0.4244 \\ 0 & 0.1061 & 0.7428 & 0 & 0.1511 \end{bmatrix}$$

Fig. 5. (left) Distance from a doubly stochastic matrix (solid line) and distance from average consensus (dash-dotted line) vs. number of iterations; (middle) node values vs. number of iterations; (right) limiting matrix P_{Alg3}.

that for all j

$$\begin{aligned} \varepsilon_j[k] - \varepsilon_j[k-1] &\leq -|\Delta p_{jj}[k]| + |\sum_{i \neq j} \Delta p_{ji}[k]| \\ &\leq -|\Delta p_{jj}[k]| + \sum_{i \neq j} |\Delta p_{ji}[k]|. \end{aligned} \tag{14}$$

Then, since $\varepsilon[k] = \sum_{l=1}^{n} \varepsilon_l[k]$, it follows that $\varepsilon[k] - \varepsilon[k-1] = \sum_{j=1}^{n} \varepsilon_j[k] - \sum_{j=1}^{n} \varepsilon_j[k-1] \leq \sum_{j=1}^{n}(-|\Delta p_{jj}[k]| + |\sum_{i \neq j} \Delta p_{ji}[k]|)$, and, by rearranging the summations, we obtain $\varepsilon[k] - \varepsilon[k-1] \leq \sum_{i=1}^{n}(-|\Delta p_{ii}[k]| + \sum_{j \neq i} |\Delta p_{ji}[k]|)$. Lemma 5 established that $-|\Delta p_{ii}[k]| + \sum_{j \neq i} |\Delta p_{ji}[k]| \leq 0, \ \forall i$, so that $\varepsilon[k] - \varepsilon[k-1] \leq 0$, where, as shown next, the inequality is strict, unless $\varepsilon[k-1] = 0$. Assume that $\varepsilon[k] - \varepsilon[k-1] = 0$ with $\varepsilon[k-1] > 0$ for (some) finite k. Following a similar argument as in the proof of Theorem 4 (which we do not repeat here for brevity), we know that $0 < \delta_j[k-1] < 1$ for all j and thus all out-neighbor weights and the self-weights have nonzero values. Since $\varepsilon[k-1] = \sum_{l=1}^{n} \varepsilon_l[k-1] > 0$, it follows that there exists at least one node j for which $\varepsilon_j[k-1] > 0$ and, by the fact that the matrix $P[k-1]$ is column stochastic (with nonnegative entries that sum to n), we conclude that both $\mathcal{A}_{k-1}$ and $\mathcal{B}_{k-1}$ are nonempty sets. From (14), we also have that $-|\Delta p_{jj}[k]| + |\sum_{i \neq j} \Delta p_{ji}[k]| = -|\Delta p_{jj}[k]| + \sum_{i \neq j} |\Delta p_{ji}[k]| = 0, \ \forall j \in \mathcal{A}_{k-1} \cup \mathcal{B}_{k-1}$, which means that all the nodes in $\mathcal{A}_{k-1}$ form a single recurrent class, and all the nodes in $\mathcal{B}_{k-1}$ form another single recurrent class (recall that all edges have nonzero weights). This implies that the graph is not strongly connected which is a contradiction.

Since the sequence $\varepsilon[k]$ is monotonically decreasing for all k, and it is bounded from below by 0, it follows that $\lim_{k\to\infty} \varepsilon[k] = 0$, and that $\lim_{k\to\infty} \sum_{i=1}^{n} p_{ji}[k] = 1, \ \forall j$. [Technically, $\varepsilon := \lim_{k\to\infty} \varepsilon[k]$ could be some nonzero (positive) constant c, but the analysis after (14) clearly implies that ε will strictly decrease at the next iteration which is a contradiction ($c > 0$ can clearly not be the limit $\lim_{k\to\infty} \varepsilon[k]$).] Therefore, $\lim_{k\to\infty} \delta_j[k] = \delta_j$ exists for all j and thus $\lim_{k\to\infty} P[k] = P$ exists and it is a doubly stochastic matrix. The fact that the matrix P is primitive follows from the fact that $0 < \delta_j \leq 1, \ \forall j = 1, 2, \ldots, n$, and $\delta_i < 1$ for at least one $i \in \{1, 2, \ldots, n\}$ (the proof of the last statement is identical to the proof in Theorem 4, with r replaced by k and π_j^r replaced by $\sum_{i \in \mathcal{N}_j^- \cup \{j\}} p_{ji}[k]$, and is omitted). ■

Corollary 4: Consider (2) with $P[k] = \overline{P}\Delta[k] + (I - \Delta[k])$, where $\Delta[k] = \text{diag}(\delta_1[k], \delta_2[k], \ldots, \delta_n[k])$ as defined in line 6 of Algorithm 3. Then, $\lim_{k\to\infty} x_j[k] = \frac{\sum_{l=1}^{n} u_l}{n}, \ \forall j$. Furthermore, for all j, $|x_j[k]| \leq \infty, \ \forall k \geq 0$.

Proof: Theorem 5 established that $P[0], P[1], \ldots, P[k], \ldots$ are primitive column stochastic matrices, and that $\lim_{k\to\infty} P[k] = P$ elementwise, where P is primitive doubly stochastic. The result then follows immediately from Theorem 1. ■

As suggested by the following example and other numerical experiments we have conducted for different size networks, the convergence speed of this algorithm seems to be geometric. However, we have not been able to establish this analytically, and we leave it as an open question.

Example 3: Consider again the directed graph in Fig. 1, and assume that the initial values of the five nodes are $u = [9, 10, 2, 10, 7]'$, with average $\overline{u} = 7.6$. Algorithm 3 is simultaneously executed with Algorithm 0. The plot on the left of Fig. 5 shows two quantities of interest against the number of iterations: (i) the solid line shows the distance of the weight matrix $P[k]$ from a doubly stochastic matrix (i.e., $\varepsilon[k]$) at the end of iteration k, and (ii) the dash-dotted line shows the distance of the resulting $x[k]$ in (2) from average consensus (i.e., $\sum_{j=1}^{n} |x_j[k] - \overline{u}|$) at the end of iteration k. It is worth pointing out that the distance from a doubly stochastic matrix (solid line) decreases monotonically (as expected from the proof of Theorem 5); however, this is not the case for the distance from average consensus (dash-dotted line). The plot in the middle of Fig. 5 shows how the value $x_j[k]$ of each node j, as determined by Algorithm 0, converges to the average consensus value $\overline{u} = 7.6$. The limiting matrix P, denoted by P_{Alg3}, that the nodes converge to is displayed on the right of Fig. 5. □

D. Algorithm 4

We discuss a variation of Algorithm 3 where, at each iteration k, the nodes perform the weight update asynchronously. Thus, at iteration k, each node j will update its weights according to line 6 of Algorithm 3, but instead of using its in-neighbor edge weights $p_{ji}[k], \ i \in \mathcal{N}_j^-$, as calculated during iteration k, if some in-neighbor i has already updated its weights before j, then j will use the updated weight $p_{ji}[k+1]$ instead. The order in which the nodes update their weight and the value within each iteration might affect the weight values and convergence speed, but the algorithm is shown to converge regardless of the particular ordering. In practice, a random ordering could be enforced in a variety of ways (for instance, each node could randomly choose an update time by sampling a random variable uniformly distributed between 0 and some Δt, where $\Delta t > 0$ is some fixed time interval which captures

the time devoted for all the nodes to complete iteration k). To simplify notation and without any loss of generality, we assume that within each iteration k, the order in which the nodes update their values follows the node index, i.e., node 1 will update first, node 2 will update second, and so on.

1) Algorithm Formulation: Consider a strongly connected graph $\mathcal{G}_d = \{\mathcal{V}, \mathcal{E}\}$. Initially, each node j sets $\delta_j[0] = \frac{\mathcal{D}_j^+}{1+\mathcal{D}_j^+}$ and $p_{lj}[0] = \frac{\delta_j[0]}{\mathcal{D}_j^+}$ for all $l \in \mathcal{N}_j^+ \cup \{j\}$ ($p_{lj}[0] = 0$ otherwise). At each time step $k \geq 0$, nodes will sequentially update their weights following the same order as the node indices. Thus, each node j updates $p_{lj}[k],\ l \in \mathcal{N}_j^+ \cup \{j\}$, by executing Algorithm 4. Note that nodes only need to broadcast their weights when they are updated. The output of Algorithm 4 is used by the nodes as the input to Algorithm 0, which they execute in parallel, allowing them to reach consensus the average of the u_j's (initial values).

Algorithm 4

Input: $\delta_j[0] = \frac{\mathcal{D}_j^+}{1+\mathcal{D}_j^+}$

for $k \geq 0$ **each node** j **does:**

Output: $p_{jj}[k] = 1 - \delta_j[k]$

$p_{lj}[k] = \frac{1}{\mathcal{D}_j^+}\delta_j[k],\ \forall l \in \mathcal{N}_j^+$

Set: $p_{lj}[k,1] = p_{lj}[k],\ \forall l \in \mathcal{N}_j^+$

for $m = 1 \ldots n$:

Receive: $p_{ji}[k,m]$ from all $i \in \mathcal{N}_j^-$ (if any)

if $m = j$:

Compute:

$e_j[k] = \sum_{i \in \mathcal{N}_j^- \cup \{j\}} p_{ji}[k,m]$

$\delta_j[k+1] =$
$\begin{cases} \delta_j[k]e_j[k], & \text{if } e_j[k] \leq 1 \\ 1 - \frac{1}{e_j[k]}\big(1 - \delta_j[k]\big), & \text{if } e_j[k] > 1 \end{cases}$

Broadcast: $p_{lj}[k,m+1] = \frac{1}{\mathcal{D}_j^+}\delta_j[k],\ \forall l \in \mathcal{N}_j^+$

else:

Broadcast:
$p_{lj}[k,m+1] = p_{lj}[k,m],\ \forall l \in \mathcal{N}_j^+$

2) Convergence Analysis: The execution of Algorithm 4 results in a sequence of column stochastic matrices $P[0,1], P[0,2], \ldots, P[0,n], P[1,1], P[1,2], \ldots, P[1,n], \ldots, P[k,1], P[k,2], \ldots, P[k,n], \ldots$, where $P[k,j] = \overline{P}\Delta[k,j] + (I - \Delta[k,j])$, with $\Delta[k,j] = \text{diag}\big(\delta_1[k], \ldots, \delta_j[k], \delta_{j+1}[k-1], \ldots, \delta_n[k-1]\big)$; we refer to $j = 1, 2, ..., n$ as sub-iterations within iteration k. We next argue about the convergence of this sequence as $k \to \infty$ to a matrix P that is primitive doubly stochastic.

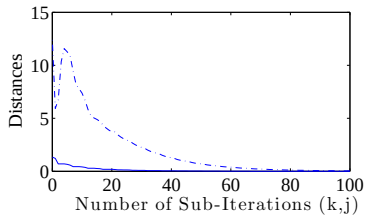
Theorem 6: Let $P[k,j] = \overline{P}\Delta[k,j] + (I - \Delta[k,j])$ be the weight matrix at sub-iteration j within time step k that results from the execution of Algorithm 4. Then, $\lim_{k\to\infty} P[k,j]$ exists and is a primitive doubly stochastic matrix $P = \overline{P}\Delta + (I - \Delta)$, where $\Delta = \text{diag}(\delta_1, \delta_2, \ldots, \delta_n)$ with $\delta_j = \lim_{k\to\infty} \delta_j[k]$ satisfying $0 < \delta_j \leq 1,\ \forall j = 1, 2, \ldots, n$, and $\delta_i < 1$ for at least one $i \in \{1, 2, \ldots, n\}$.

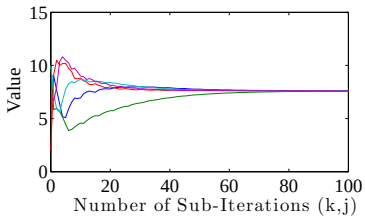
Proof: In order to capture the total distance of matrix $P[k,j]$ from a doubly stochastic matrix, we define $\varepsilon[k,j] = \sum_{l=1}^{n} \varepsilon_l[k,j]|$ where $\varepsilon_l[k,j] = |\sum_{i \in \mathcal{N}_l^- \cup \{l\}} p_{li}[k,j] - 1|$. Thus, for a fixed k, $\varepsilon[k,j]$ measures the progress of the algorithm with weight matrix $P[k,j] = \overline{P}\Delta[k,j] + (I - \Delta[k,j]),\ j = 1, 2, \ldots, n$, where $\Delta[k,j] = \text{diag}\big(\delta_1[k], \delta_2[k], \ldots, \delta_j[k], \delta_{j+1}[k-1], \ldots, \delta_n[k-1]\big)$ (recall that the $\delta_j[k]$'s are obtained according to line 7 of Algorithm 4). It follows from Lemma 5 that $\varepsilon[k, j+1] \leq \varepsilon[k,j],\ \forall j = 0, 1, 2 \ldots, n-1$. Then, unless $\varepsilon[k,n] = 0$, it is easy to show that $\varepsilon[k+1,n] < \varepsilon[k,n]$, otherwise, following a similar argument to the one used in the proof of Theorem 5, the assumption that the graph is strongly connected would be violated. Finally, we can define the sequence $z_k = \varepsilon[k,n]$ which by construction is strictly monotonically decreasing (unless $z_k = 0$), and is bounded below by zero. Thus, we establish that $\lim_{k\to\infty} z_k = 0$, from where we conclude that $\lim_{k\to\infty} \sum_{i=1}^{n} p_{ji}[k,n] = 1,\ \forall j$. Similarly, we can prove that $\delta_j = \lim_{k\to\infty} \delta_j[k,n]$ exists and satisfies $0 < \delta_j \leq 1,\ \forall j = 1, 2, \ldots, n$, and $\delta_i < 1$ for at least one $i \in \{1, 2, \ldots, n\}$ (one can follow a similar argument as the one used in the proof of Theorem 4, and therefore this step is omitted). Note that convergence of the $\delta_j[k,n]$'s also implies convergence of the $\delta_j[k,m]$'s within the sub-iterations (in fact, $\lim_{k\to\infty} \delta_j[k,m] = \delta_j$ for all $m = 1, 2, ..., n$); consequently, the weight matrix converges to a doubly stochastic matrix. ■

Corollary 5: Consider (2) with $P[k] = \overline{P}\Delta[k] + (I - \Delta[k])$, where $\Delta[k] = \text{diag}(\delta_1[k], \delta_2[k], \ldots, \delta_n[k])$ as defined in line 7 of Algorithm 4. Then, $\lim_{k\to\infty} x_j[k] = \frac{\sum_{l=1}^{n} u_l}{n},\ \forall j$. Furthermore, for all $j,\ |x_j[k]| \leq \infty,\ \forall k \geq 0$.

Proof: Theorem 6 established that the matrices $P[0], P[1], \ldots, P[k], \ldots$ are primitive and column stochastic, and that $\lim_{k\to\infty} P[k] = P$ elementwise, where P is primitive doubly stochastic. The result then follows immediately from Theorem 1. ■

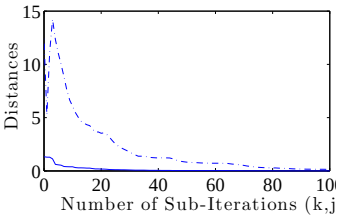
Example 4: Consider again the directed graph in Fig. 1, and assume that the initial values of the five nodes are $u = [9, 10, 2, 10, 7]'$, with average $\overline{u} = 7.6$. Algorithm 4 operates iteratively by having the nodes perform updates of the weights on their out-neighbor edges, one at a time in a prescribed order. Simultaneously, the nodes run Algorithm 0. The plot on the left of Fig. 6 shows the evolution of two quantities of interest as the iterations are performed: (i) the solid line shows the distance of the weight matrix $P[k,j]$ from a doubly stochastic matrix at the end of sub-iteration j of iteration k (indicated by index $5k + j$ in the figure), and (ii) the dash-dotted line shows the distance of the resulting $x[k,j]$ in (2) from average consensus at the end of sub-iteration j of iteration k. The plot in the middle of Fig. 6 shows how the value $x_i[k,j]$ evolves at each node i towards average consensus. The plots in Fig. 7 are identical to the corresponding ones in Fig. 6 for a variation of Algorithm 4 where the node that is supposed to perform

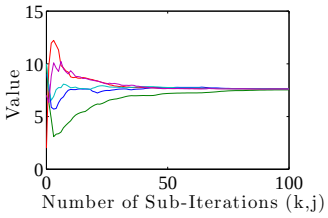




$$P_{Alg4} = \begin{bmatrix} 0.5930 & 0 & 0 & 0.4070 & 0 \\ 0.2035 & 0.7965 & 0 & 0 & 0 \\ 0.2035 & 0.1017 & 0.2878 & 0 & 0.4070 \\ 0 & 0 & 0 & 0.5930 & 0.4070 \\ 0 & 0.1018 & 0.7132 & 0 & 0.1860 \end{bmatrix}$$

Fig. 6. (left) Distance from a doubly stochastic matrix (solid line) and distance from average consensus (dash-dotted line) vs. number of sub-iterations (sequential order); (middle) node values vs. number of sub-iterations (sequential order); (right) limiting matrix P_{Alg4}.





$$P_{Alg4rnd} = \begin{bmatrix} 0.5596 & 0 & 0 & 0.4404 & 0 \\ 0.2202 & 0.7798 & 0 & 0 & 0 \\ 0.2202 & 0.1101 & 0.2292 & 0 & 0.4405 \\ 0 & 0 & 0 & 0.5596 & 0.4404 \\ 0 & 0.1101 & 0.7708 & 0 & 0.1191 \end{bmatrix}$$

Fig. 7. (left) Distance from a doubly stochastic matrix (solid line) and distance from average consensus (dash-dotted line) vs. number of sub-iterations (random order); (middle) node values vs. number of sub-iterations (random order); (right) limiting matrix $P_{Alg4rnd}$.

the update is chosen randomly (possibly with repetition) at each iteration. The corresponding limiting doubly stochastic matrices P, denoted by P_{Alg4} and $P_{Alg4rnd}$, that the nodes converge to (refer to Theorem 6) are displayed on the right of Figs. 6 and 7 respectively. □

Example 5: In this example we present a comparison of convergence to average consensus for Algorithms 1, 3 and 4 with sequential ordering (the comparison with Algorithm 2 is not straightforward due to the existence of super-iterations). We consider randomly generated graphs with $n = 50$, $n = 100$, $n = 200$, and $n = 400$ nodes, and focus on the evolution of the distance from average consensus defined as $\sum_{j=1}^{n} |x_j[k] - \overline{u}|$, where $x[k]$ is the value vector in iteration (2) and $\overline{u}$ is the average of the initial values of the nodes $x[0] = u$. Recall that in Algorithm 3, each node takes a snapshot of each row sum and subsequently all nodes simultaneously update the weights on their out-neighbor edges based on this snapshot, whereas in Algorithm 4 each node update is completed before the next node is allowed to update (in the simulations shown, node updates are in order of their indices but other orderings have similar behavior). To be fair, for Algorithm 4 we do not plot the distance from average consensus at each sub-iteration but only once every n sub-iterations.

The results are shown in Fig. 8 and, as expected, Algorithm 4 does better than Algorithm 3 (because each node update takes into account previous updates by other nodes within the same iteration). Algorithm 1 appears to be the fastest one to reach average consensus (though not shown, Algorithm 1 also appears to consistently be the first one to reach a doubly stochastic weight matrix). □

E. Comparison to Existing Work

The techniques in [22], [23], [24], which discuss conditions and (centralized or distributed) algorithms that allow us to assign nonnegative weights to the edges of a directed graph (without self-loops) so as to (weight) balance it (i.e., make at each node the sum of weights at the in-neighbor edges equal to the sum of weights at the out-neighbor edges), are related to our algorithms. Since the focus of our paper is to obtain a doubly stochastic matrix on a strongly connected graph with self-weights that can be nonzero, we are guaranteed the existence of a doubly stochastic matrix that satisfies the given

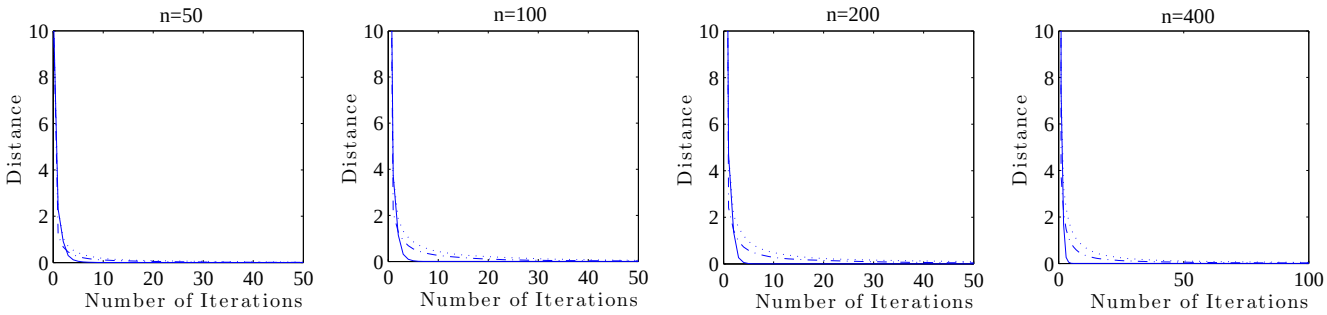


Fig. 8. Distance from average consensus vs. number of iterations for Algorithm 1 (solid), Algorithm 3 (dotted) and the sequential version of Algorithm 4 (dash-dotted) for graph size: (a) $n = 50$, (b) $n = 100$, (c) $n = 200$, and (d) $n = 400$.

sparsity structure in the underlying communication topology. Although the existence of a doubly stochastic matrix P of the form in (3) is not evident from previous work, this becomes apparent by Lemma 1 and its role in the proposed matrix scaling algorithms, which mainly depend on the manipulation of the diagonal elements and *not* on the adjustment of weights on individual edges. Another major difference between our work and the work in [22], [23], [24] is the communication modality, which in our case allows each node to simply broadcast its value (and/or weight) without requiring nodes to send specific messages to different nodes (also, the receiving nodes do not need to know which node has transmitted the in-neighbor message). This is a major departure from the distributed algorithms proposed in [24] for obtaining a doubly stochastic (or weight balanced) matrix because in that setting each node needs to be able to directly communicate with each neighbor separately (the algorithm operates by having each node that is not balanced modify the weight of a particular out-neighbor). Another difference is that our algorithms asymptotically reach weights that correspond to a doubly stochastic matrix whereas the algorithm in [24] does so in a finite number of steps. This should generally be considered a disadvantage of the proposed matrix scaling algorithms, but in the application to average consensus with time-varying weights (see Algorithm 0), it actually allows us to immediately start averaging without having to first wait for a finite but potentially large number of steps in order to obtain a doubly stochastic matrix. In general, however, comparing our algorithms with the ones in [24] is difficult/unfair due to the different assumptions.

V. Concluding Remarks

We studied the problem of scaling a primitive column stochastic matrix to a doubly stochastic form (and its application to average consensus) through distributed computation over a multi-component system with an arbitrary (possibly directed) underlying communication graph. Starting from a very general setup where nodes (i) can assign their self-weights and the weights on their out-neighbor edges, and (ii) observe (but do not assign) the weights on their in-neighbor edges, we developed four different algorithms that allow the nodes to iteratively assign their weights, in a distributed fashion, so that eventually they asymptotically converge to a set of weights that form a doubly stochastic matrix. The only requirement for our algorithms is that the underlying communication graph is strongly connected.

All four algorithms share similar computational/communication requirements, i.e., each node needs to know the number of its out-neighbors, be able to maintain and broadcast a small number of values, and be able to linearly combine (or simply add) the values it receives from its in-neighbors. Nodes also need to be able to perform the operations of division, subtraction, and maximization. The complexity of the computation at each node depends on its in-degree (i.e., the number of values that it receives): at each iteration, node j performs $O(\mathcal{D}_j^-)$ computations. With the exception of Algorithm 4, all other algorithms are synchronized in the sense that all nodes update their weights at the same time. There are some implementation differences among the algorithms; for instance, Algorithm 2 updates the weights used in the iteration more infrequently (once every k_0), Algorithm 3 updates them (synchronously for all nodes) at each iteration, whereas Algorithm 4 updates (asynchronously) the weights of a single node at each sub-iteration.

Apart from establishing the correctness of the algorithms, we also discussed implementation and convergence speed aspects for each of the algorithms, and provided numerical examples to illustrate their performance. In future work, we plan to study in more detail the adaptation of these techniques to cases where the underlying communication graph is allowed to change over time. We also plan to relax the nonnegative weight constraint, in an effort to develop schemes that potentially allow for faster convergence.

References

[1] N. Lynch, *Distributed Algorithms*. San Mateo, CA: Morgan Kaufmann Publishers, 1996.

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

[3] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782–795, Oct. 2003.

[4] M. Rabbat and R. D. Nowak, "Distributed optimization in sensor networks," in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN)*, 2004, pp. 20–27.

[5] A. Giridhar and P. R. Kumar, "Computing and communicating functions over sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 755–764, Apr. 2005.

[6] J. Hromkovic, R. Klasing, A. Pelc, P. Ruzicka, and W. Unger, *Dissemination of Information in Communication Networks*. Springer-Verlag, 2005.

[7] J. Cortés, "Distributed algorithms for reaching consensus on general functions," *Automatica*, vol. 44, no. 3, pp. 726–737, Mar. 2008.

[8] S. Chatterjee and E. Seneta, "Towards consensus: some convergence theorems on repeated averaging," *Journal of Applied Probability*, vol. 14, no. 1, pp. 89–97, Mar. 1977.

[9] J. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, 1984.

[10] A. Jadbabaie, J. Lin, and A. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, Jun. 2003.

[11] S. Sundaram and C. N. Hadjicostis, "Distributed function calculation and consensus using linear iterative strategies," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 4, pp. 650–660, May 2008.

[12] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, no. 1, pp. 65–78, Sep. 2004.

[13] L. Moreau, "Stability of multiagent systems with time-dependent communication links," *IEEE Transactions on Automatic Control*, vol. 50, no. 2, pp. 169–182, Feb. 2005.

[14] A. W. Marshall and I. Olkin, "Scaling of matrices to achieve specified row and column sums," *Numerische Mathematik*, vol. 12, no. 1, pp. 83–90, 1968.

[15] D. Z. Djokovic, "Note on nonnegative matrices," *Proc. of the American Mathematical Society*, vol. 25, no. 1, pp. pp. 80–82, 1970.

[16] B. N. Parlett and T. L. Landis, "Methods for scaling to doubly stochastic form," *Linear Algebra and its Applications*, vol. 48, pp. 53–79, 1982.

[17] E. Seneta, *Nonnegative Matrices and Markov Chains*, revised printing ed. New York, NY: Springer, 2006.

[18] M. H. Schneider and S. A. Zenios, "A comparative study of algorithms for matrix balancing," *Operations Research*, vol. 38, no. 3, pp. 439–455, May–Jun. 1990.

[19] M. Mesbahi and M. Egerstedt, *Graph-theoretic Methods in Multiagent Networks*, ser. Applied Mathematics Series. Princeton University Press, 2010.

[20] A. Nedic, A. Ozdaglar, and P. A. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, Apr. 2010.
[21] G. J. Foschini, "A simple distributed autonomous power control algorithm and its convergence," *IEEE Transactions on Vehicular Technology*, vol. 42, no. 4, pp. 641–646, Nov. 1993.
[22] B. Gharesifard and J. Cortés, "Distributed strategies for making a digraph weight-balanced," in *Proc. of Annual Allerton Conference on Communication, Control, and Computing*, Oct. 2009, pp. 771–777.
[23] ——, "When does a digraph admit a doubly stochastic adjacency matrix?" in *Proc. of American Control Conference*, Jun. 2010, pp. 2440–2445.
[24] ——, "Distributed strategies for generating weight-balanced and doubly stochastic digraphs," *European Journal of Control*, to appear, 2012.
[25] M. Franceschelli, A. Giua, and C. Seatzu, "Distributed averaging in sensor networks based on broadcast gossip algorithms," *IEEE Sensors Journal*, vol. 11, no. 3, pp. 808–817, Mar. 2011.
[26] K. Cai and H. Ishii, "Average consensus on general digraphs," in *Proc. of IEEE Conference on Decision and Control and European Control Conference*, Dec. 2011, pp. 1956–1961.
[27] F. Fagnani and S. Zampieri, "Average consensus with packet drop communication," *SIAM Journal on Control and Optimization*, vol. 48, pp. 102–133, 2009.
[28] Y. Chen, R. Tron, A. Terzis, and R. Vidal, "Corrective consensus: Converging to the exact average," in *Proc. IEEE Conference on Decision and Control*, Dec. 2010, pp. 1221–1228.
[29] R. Horn and C. Johnson, *Matrix Analysis*. New York, NY: Cambridge University Press, 1985.
[30] R. Olfati-Saber and R. M. Murray, "Agreement problems in networks with directed graphs and switching topology," in *Proc. of IEEE Conference on Decision and Control*, vol. 4, Jun. 2003, pp. 4123–4132.

Alejandro D. Domínguez-García (S'02–M'07) is an Assistant Professor in the Department of Electrical and Computer Engineering of the University of Illinois at Urbana-Champaign, where he is affiliated with the Power and Energy Systems area. His research interests are in the areas of system reliability theory and control, and their application to electric power systems, power electronics, and embedded electronic systems for safety-critical/fault-tolerant aircraft, aerospace, and automotive applications. He received the Ph.D. degree in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology, Cambridge, MA, in 2007 and the degree of Electrical Engineer from the University of Oviedo (Spain) in 2001. After finishing his Ph.D., he spent some time as a post-doctoral research associate at the Laboratory for Electromagnetic and Electronic Systems of the Massachusetts Institute of Technology. Dr. Domínguez-García received the NSF CAREER Award in 2010, and the Young Engineer Award from the IEEE Power and Energy Society in 2012. He is also a Grainger Associate since 2011. He currently serves as an Associate Editor for the IEEE Transactions on Power Systems.

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.