# Distributed Strategies for Average Consensus in Directed Graphs

Alejandro D. Domínguez-García and Christoforos N. Hadjicostis

*Abstract*— **We address the average consensus problem for a distributed system whose components (nodes) can exchange information via interconnections (links) that form an arbitrary, strongly connected but possibly directed, topology (graph). Specifically, we discuss how the nodes can asymptotically reach average consensus (i.e., obtain the average of their initial values) with linear-iterative algorithms in which each node updates its value using a weighted linear combination of its own value and the values of neighboring nodes. In the process, the strategies we develop allow the nodes to adapt their weights in a distributed fashion, so that asymptotically they obtain a doubly stochastic weight matrix, which is useful for many algorithms that utilize linear- or nonlinear-iterative schemes to perform various estimation and optimization tasks.**

## I. Introduction and Background

Over the past few decades, the design of protocols and algorithms for distributed computation and control/decision tasks has attracted significant attention by the computer science, communication, and control communities (e.g., [1], [2], [3] and references therein). In a generic consensus problem, each node possesses an initial value and the nodes need to follow a distributed strategy to agree on some function of these initial values. The consensus problem has received extensive attention from the control community due to its applicability to topics such as cooperative control, multi-agent systems, and modeling of flocking behavior in biological and physical systems (e.g., [3], [4], [5], [6], [7], [8] and references therein). In these works, the approach to consensus is to use a linear iteration, where each node in the network repeatedly updates its value as a weighted linear combination of its own previous value and the previous values of its neighbors. In the context of this class of linear iterative algorithms for consensus, a variety of pertinent issues, including convergence properties [9], [10], and weight choices that achieve faster convergence [8], have been investigated. Asymptotic *average* consensus is reached if the nodes (following the linear iterative strategy described above) asymptotically converge to the average of their initial values.

The work of A. D. Domínguez-García was supported in part by the National Science Foundation (NSF) under grant ECCS-CAR-0954420. The work of C. N. Hadjicostis was supported in part by the European Community (EC) 7th Framework Programme (FP7/2007-2013), under grants INFSO-ICT-223844 and PIRG02-GA-2007-224877. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of NSF or EC.

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

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

Most existing studies on (asymptotic) average consensus have assumed that the underlying interconnection topology is described by an *undirected* graph (which implies that if node $j$ receives information from node $i$, then node $i$ necessarily receives information from node $j$). This paper relaxes this assumption and focuses on the average-consensus problem when the interconnection topology is described by a possibly *directed* graph. This situation can arise in a variety of realistic scenarios (e.g., if nodes transmit at different power strengths or if interference levels are not uniform at each node). In such topologies, our previous work in [11] proposed a class of algorithms that solve the average-consensus problem by having each node appropriately choose positive weights on its out-going links in an iterative and distributed fashion, so that they asymptotically obtain a weight matrix that is primitive doubly stochastic. The approach in [11] treats all out-going links in the same manner (which is ideal when broadcasting is possible, e.g., in wireless networks) and only requires that each node knows the number of nodes it can send information to. In this paper, we generalize the class of algorithms developed in [11] by establishing that there exists significant flexibility in the weight updates performed by the nodes in their effort to reach a doubly stochastic matrix. This flexibility could be important when additional information about the topology is available at each node (e.g., the number of out-going links of each of their neighbors); as we will see, such information can be used to improve the rate with which the nodes converge to a weight matrix that is doubly stochastic—at the cost of additional setup time.

The techniques in [12], [13], [14] are related to (but are quite distinct from) the class of algorithms we propose. These works investigate conditions and algorithms that allow us to assign nonnegative weights to the edges of a directed graph (without self-loops) so as to achieve weight balance. A major difference between our work and the work in [12], [13], [14] is the communication modality, which in our case allows each node to simply broadcast a single value without requiring it to send specific messages to specific neighboring nodes (also, the receiving nodes do not need to know which node has transmitted the incoming message). This is a major departure from the distributed algorithms proposed in [14] 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.

It is worth pointing out that several authors have investigated methods for reaching average consensus using techniques that not require the nodes to obtain a set of weights that forms a doubly stochastic matrix. For example, [15] proposes gossip algorithms based on broadcasts that appear to converge to the exact average on arbitrary strongly connected digraphs. Similarly, [16] considers the effect of random and asymmetric packet losses during the linear iteration process and handles them by introducing additional variables at each node and by using periodic corrective operations. Finally, [17] reaches average consensus by having the nodes simultaneously run two gossip-based iterations and taking the ratio of the values obtained in each iteration. Note, however, that [15], [16], [17] implicitly require acknowledgments or the existence of a reverse direction for each link (which does not necessarily imply bi-directional links—because the acknowledgment could occur via a different path—but nevertheless imposes some overhead). It is difficult to directly compare our approach in this paper with the approaches in [15], [16], [17] due to the different underlying assumptions.

The remainder of this paper is organized as follows. Section II provides necessary background on graph theory and discusses the general form of the class of iterative distributed algorithms that we study. In Section III, we present a class of algorithms that allow the nodes to follow a distributed strategy and reach a set of nonnegative weights that solves the average-consensus problem. In Section IV, we discuss and compare various choices of weight updates, all of which asymptotically lead to weights that form a doubly stochastic matrix. Concluding remarks are presented in Section V.

## II. Preliminaries

The exchange of information between components (nodes) of a distributed system can be conveniently described by a directed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, 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, where $(j, i) \in \mathcal{E}$ if node $j$ can receive information from node $i$. By convention, we assume no self-loops in $\mathcal{G}$ (i.e., $(j, j) \notin \mathcal{E}$ for all $j \in \mathcal{V}$). The graph is undirected if and only if whenever $(j, i) \in \mathcal{E}$, then also $(i, j) \in \mathcal{E}$, i.e., if node $j$ can receive information from node $i$, then node $i$ can also receive information from node $j$. All nodes that can transmit information to node $j$ are said to be 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 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 number of nodes that have $j$ as neighbor, i.e., the number of nodes that receive information from $j$, is called the out-degree of $j$ and is denoted by $\mathcal{D}_j^+$. We will assume that the graph remains invariant throughout the process.

Let $x_j$ be the initial value of node $j$. The objective of average-consensus is to have all the nodes calculate the average of these initial values, which we denote by $\mu$, i.e.,

$$\mu = \frac{\sum_{j=1}^n x_j}{n}. \tag{1}$$

Depending on the assumptions, nodes may or may not know $n$, and they will presumably require several rounds of message exchanges in order to obtain $\mu$ (perhaps obtaining the values of $x_j,\ j = 1, 2, \ldots, n$, in the process).

In the algorithms we consider, in order to obtain $\mu$, each node $j$ maintains some value $\pi_j[k]$ at round $k$, and performs a linear iteration of the form

$$\pi_j[k+1] = p_{jj}[k]\pi_j[k] + \sum_{i \in \mathcal{N}_j} p_{ji}[k]\pi_i[k] \ , \tag{2}$$

where the $p_{ji}[k]$, $i \in \mathcal{N}_j \cup \{j\}$, are time-varying weights (that will be described in detail soon). In other words, each node $j$ updates its value to be a linear combination of its own previous value and the values of its neighbors. If we let $\pi[k] = [\pi_1[k], \pi_2[k], \ldots, \pi_j[k], \ldots, \pi_n[k]]'$, then for analysis purposes (2) can be written in matrix form as

$$\pi[k+1] = P[k]\pi[k],\ \pi[0] = \pi_0, \tag{3}$$

where the weight matrix $P[k] = [p_{ji}[k]]$ (with $p_{ji}[k]$ as the entry at the $j$th row-$i$th column of matrix $P[k]$) and $\pi_0 = [x_1, x_2, \ldots, x_j, \ldots, x_n]'$.

In (2), the $p_{ji}[k]$'s are a set of (time-varying) weights that will be chosen so that all $\pi_j[k]$ converge for large $k$ to $\mu$. Node $j$ can only choose its self-weight and the weights on its out-going links, i.e., node $j$ can choose values for $\{p_{ij}[k] \mid i = 1, 2, \ldots, n\}$, with the constraint that $p_{ij}[k] = 0$ for all $i$ such that $j \notin \mathcal{N}_i$. It is assumed that each node can observe but cannot control the (likely different) values on each of its incoming links, 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 at each node). In the work in [11] each node uses the *same* weight for all of its out-going links, but in this paper we show that a node can use fairly arbitrary weights on its out-going links (as long as these weights are strictly positive). In particular, we will see that if additional information about the in- and out-degrees of nodes is available, then it might be beneficial to incorporate this information in the choice of weights.

*Remark 1:* For the case when the weights $p_{ji}[k]$'s are fixed for all $k \geq 0$, as stated in [7], [8], [18] in various forms, the necessary and sufficient conditions for the iteration in (3) (with $P[k] = P$) to asymptotically reach average-consensus are: (i) $P$ has a simple eigenvalue at 1, with left eigenvector $[1, 1, 1, ..., 1]$ and right eigenvector $[1, 1, 1, ..., 1]'$, and (ii) all other eigenvalues of $P$ have magnitude strictly less than 1. If one focuses on nonnegative weights, these conditions are equivalent to the weight matrix $P$ being a primitive doubly stochastic matrix; in general, however, this condition on $P$ would only be sufficient for ensuring that the nodes asymptotically reach average-consensus. □

## III. Distributed Weight Adjustment

The class of algorithms developed in this section assumes that each node updates its self-weight and the weights on its out-going links at every iteration based on the sum of the weights on its incoming links. Specifically, at each iteration $k$, node $j$ executes two tasks: (i) it first chooses a particular value, $p_{jj}[k]$, $0 < p_{jj}[k] < 1$ (in a manner that will be described shortly), for its self-weight, and (ii) it sets the weights on its out-going links to be $p_{ij} = c_{ij}(1 - p_{jj}[k])$, $i \neq j$. The constants $c_{ij}$, $i \neq j$, are time-invariant and chosen so that: (i) $\sum_{i, i\neq j} c_{ij} = 1$, (ii) $c_{ij} > 0$ if node $i$ can receive information from node $j$ (i.e., if $(i,j) \in \mathcal{E}$), and (iii) $c_{ij} = 0$ if node $i$ cannot receive information from node $j$ (i.e., if $(i,j) \notin \mathcal{E}$). For notational convenience, we will take $c_{jj} = 0$ for $j = 1, 2, \ldots, n$, so that $\sum_{i,i\neq j} c_{ij} = \sum_i c_{ij} = 1$.

With the above choices, it can be easily verified that $P[k]$ will be a nonnegative column-stochastic matrix (i.e., $p_{ij}[k] \geq 0$ and $\sum_{i=1}^{n} p_{ij}[k] = 1$, $j = 1, 2, \ldots, n$). In fact, one can also check that as long as the diagonal entries of $P[k]$ are strictly smaller than one and at least one diagonal entry is strictly positive, then $P[k]$ will be primitive. This is due to the fact that $P[k]$ corresponds to a graph that is strongly connected (each edge of the graph has a strictly positive weight), and the diagonal entries of $P[k]$ are nonzero [19]. The update of the diagonal entries $p_{jj}[k]$ will therefore lead to a sequence of primitive column-stochastic matrices $P[0]$, $P[1]$, ..., $P[k]$, ..., which will be shown to converge as $k \to \infty$ to a primitive doubly stochastic matrix $P_{ss}$. Note that, at each iteration $k$, each node $j$ is supposed to perform two tasks: an update of its self-weight and the weights on its out-going links, followed by an update of its value $\pi_j[k]$. The update of the weights (which is governed solely by the update of the diagonal terms $p_{jj}[k]$ and the choice of coefficients $c_{ij}$) is essentially done independently of the update of the $\pi_j[k]$'s and our focus in this section is mostly on describing and establishing the properties of the weight update process.

Initially, for $k = 0$, each node $j$ chooses its self weight to be $p_{jj}[0] = \frac{1}{1+\mathcal{D}_j^+}$, where $\mathcal{D}_j^+$ is the number of nodes that node $j$ sends information to. This results in each node updating its value to $\pi_j[1]$ according to

$$\pi_j[1] = \frac{1}{1+\mathcal{D}_j^+}\pi_j[0] + \sum_{i\in\mathcal{N}_j} \frac{c_{ji}\cdot\mathcal{D}_i^+}{1+\mathcal{D}_i^+}\pi_i[0] \; . \tag{4}$$

We can rewrite (4) in matrix form as follows

$$\pi[1] = P[0]\pi[0], \tag{5}$$

where $\pi[0] = [x_1, x_2, \ldots, x_n]'$ and the matrix $P[0]$ is a column stochastic matrix that can also be parameterized as

$$P[0] = \overline{P}\Delta[0] + (I - \Delta[0]), \tag{6}$$

where $I$ is the $n \times n$ identity matrix, $\Delta[0]$ is a diagonal matrix with nonzero entries $\delta_j[0] = \frac{\mathcal{D}_j^+}{1+\mathcal{D}_j^+} = 1 - p_{jj}[0]$, $j = 1, 2, \ldots, n$, and $\overline{P} = [c_{ij}]$ is the (column stochastic) weight matrix corresponding to the algorithm where each node distributes its value among its neighbors (according to the weights $c_{ij}$) without keeping anything for itself, i.e.

$$\overline{P} = \begin{bmatrix} 0 & c_{12} & c_{13} & \ldots & c_{1n} \\ c_{21} & 0 & c_{23} & \ldots & c_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & c_{n3} & \ldots & 0 \end{bmatrix} . \tag{7}$$

After the first round of exchanges, $k = 1$, each node $j$ will perform two actions: (i) it will first update the weights that it uses to distribute its value among itself and the nodes that $j$ sends information to and, (ii) it will then update its value to $\pi_j[2]$ based on the new weights (chosen by the nodes that sent values to it), which are collectively captured by the weight matrix $P[1]$. More specifically, each node will update the weights on its out-going links so as to try to make its row-sum in the weight matrix $P[1]$ be closer to one. It is important to note that a node cannot modify the weights on its incoming links, and in the process of trying to improve its own row-sum, it might make other row-sums worse in terms of their closeness to one.

Let $P[1] = \overline{P}\Delta[1] + (I - \Delta[1])$ be the matrix associated to the new weights, where $\Delta[1] = \text{diag}\big(\delta_1[1], \delta_2[1], \ldots, \delta_j[1], \ldots, \delta_n[1]\big)$ is a diagonal matrix chosen as follows: let $\rho_j[0] = \sum_{i\in\mathcal{N}_j\cup\{j\}} p_{ji}[0]$ and[1] set $\delta_j[1]$ as follows:

$$\delta_j[1] = \begin{cases} \delta_j[0]\rho_j[0], & \text{if } \rho_j[0] \leq 1 \; , \\ 1 - \frac{1}{\rho_j[0]}\big(1 - \delta_j[0]\big), & \text{if } \rho_j[0] > 1 \; . \end{cases} \tag{8}$$

By construction, $P[1]$ is a primitive column stochastic matrix (because it has a strictly positive entry $p_{ji}[1]$ for each $(j,i) \in \mathcal{E}$ and also has diagonal entries strictly greater than zero and strictly smaller than one). Note that the update of the weights from $P[0]$ to $P[1]$ only involves the weights in $P[0]$ and is independent of the values $\pi[0]$ or $\pi[1]$. Finally, updated node values can be written in matrix form as

$$\pi[2] = P[1]\pi[1] \; . \tag{9}$$

This process continues and, for any $k \geq 0$, the nodes will update their values according to

$$\pi[k+1] = P[k]\pi[k] \; , \tag{10}$$

where $P[k] = \overline{P}\Delta[k] + (I - \Delta[k])$, and $\Delta[k] = \text{diag}\big(\delta_1[k], \delta_2[k], \ldots, \delta_j[k], \ldots, \delta_n[k]\big)$ with

$$\delta_j[k] = \begin{cases} \delta_j[k-1]\rho_j[k-1], & \text{if } \rho_j[k-1] \leq 1 \; , \\ 1 - \frac{1}{\rho_j[k-1]}\big(1 - \delta_j[k-1]\big), & \text{if } \rho_j[k-1] > 1 \; , \end{cases} \tag{11}$$

where $\rho_j[k-1] = \sum_i p_{ji}[k-1]$. [Note that the above update implies that in order for node $j$ to update its weights (including its self-weight and the weights on its out-going links) at iteration $k$, it needs to have access to the weight $p_{ji}[k-1]$ on incoming links from each neighboring node $i$. In addition, in order to perform the value update $\pi_j[k+1]$, node $j$ needs access to the values $\pi_i[k]$ on the incoming link from each neighboring node $i$.]

The idea behind the weight update described in (10) is to make the weight matrix doubly stochastic and primitive as $k$ goes to infinity. Then, taking into account the fact that

[1] Note that the quantity $\sum_{i\in\mathcal{N}_j\cup\{j\}} p_{ji}[0]$ can actually be replaced by $\sum_{i=1}^{n} p_{ji}[0]$ (since $p_{ji}[0] = 0$ for $i \notin \mathcal{N}_j \cup \{j\}$) but it was written in this fashion to emphasize the fact that the update of node $j$ is based purely on locally available information. This is true for all iterations and, from now on, we will use these two expressions interchangeably.

the sequence $P[0]$, $P[1]$, $\ldots$ consists of column stochastic and primitive matrices that converge to a limiting doubly stochastic and primitive matrix $P_{ss}$, the steady-state solution of (10), with initial conditions $\pi_j[0] = x_j$, $\forall j$, denoted by $\pi^{ss}$, is such that

$$\pi_j^{ss} = \frac{\sum_{l=1}^{n} x_l}{n} = \mu, \ \forall j = 1, 2, \ldots, n \ . \tag{12}$$

The above statement follows easily from Theorem 4.14 in Chapter 4 of [20]. To argue that the limiting matrix $P_{ss}$ is a doubly stochastic and primitive matrix, we first need the following two lemmas.

*Lemma 1:* 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 update matrices in (10) at steps $k-1$ and $k$ respectively, where $\Delta[k-1] = \text{diag}\big(\delta_1[k-1], \delta_2[k-1], \ldots, \delta_j[k-1], \ldots, \delta_n[k-1]\big)$ and $\Delta[k] = \text{diag}\big(\delta_1[k], \delta_2[k], \ldots, \delta_j[k], \ldots, \delta_n[k]\big)$ are diagonal matrices with diagonal entries satisfying $0 < \delta_j[k-1] < 1$ and $0 < \delta_j[k] < 1$ for all $j = 1, 2, \ldots, n$. Assume that the underlying connectivity graph associated with $P[k-1]$ (the same graph is associated with $P[k]$) is strongly connected. Furthermore, assume that for a specific node $j$, the relation between $\delta_j[k-1]$ and $\delta_j[k]$ is given by (11), whereas for all $i \neq j$, we have $\delta_i[k-1] = \delta_i[k]$. Then, the following hold:

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

*Proof:* If $\rho_j[k-1] \leq 1$, then $\delta_j[k] = \delta_j[k-1]\rho_j[k-1]$, from where it follows that $p_{jj}[k] = 1 - (1 - p_{jj}[k-1])\rho_j[k-1]$. Since $\delta_i[k-1] = \delta_i[k], \forall i \neq j$, we have $p_{ji}[k-1] = p_{ji}[k], \ \forall i \neq j$. Then, $\rho_j[k] = p_{jj}[k] + \sum_{i \in \mathcal{N}_j} p_{ji}[k]$ can be shown to satisfy $\rho_j[k] = 1 + p_{jj}[k-1]\,(\rho_j[k-1] - 1) \leq 1$ (since $\rho_j[k-1] - 1 \leq 0$). Also, since $p_{ji}[k-1] = p_{ji}[k], \ \forall i \neq j$, and $p_{jj}[k] \geq p_{jj}[k-1]$, we have $\rho_j[k-1] \leq \rho_j[k]$.

Similarly, if $\rho_j[k-1] > 1$, then $\delta_j[k] = 1 - (1 - \delta_j[k-1])\frac{1}{\rho_j[k-1]}$, from where it follows that $p_{jj}[k] = p_{jj}[k-1]\frac{1}{\rho_j[k-1]}$. Then, $\rho_j[k] = p_{jj}[k] + \sum_{i \in \mathcal{N}_j} p_{ji}[k]$ can be shown to satisfy $\rho_j[k] = 1 + \sum_{i \in \mathcal{N}_j} p_{ji}[k-1]\left(1 - \frac{1}{\rho_j[k-1]}\right) > 1$ (since $\frac{1}{\rho_j[k-1]} < 1$). Also, since $p_{ji}[k-1] = p_{ji}[k], \ \forall i \neq j$ and $p_{jj}[k] < p_{jj}[k-1]$, we have $\rho_j[k-1] > \rho_j[k]$. ■

*Lemma 2:* Consider the setting in Lemma 1, and let $\varepsilon[k] = \sum_{l=1}^{n} \varepsilon_l[k]$, where $\varepsilon_l[k] \equiv |\sum_i p_{li}[k] - 1| = |\rho_l[k] - 1|$. It follows that $\varepsilon[k] \leq \varepsilon[k-1]$.

*Proof:* Assume that $\rho_j[k-1] \leq 1$; it follows from Lemma 1 that $\rho_j[k] \leq 1$ and from (11) that $p_{jj}[k] \geq p_{jj}[k-1]$. Then,

$$\begin{aligned} \varepsilon_j[k] &= 1 - \rho_j[k] = 1 - p_{jj}[k] + \sum_{i \in \mathcal{N}_j} p_{ji}[k-1] \\ &\leq 1 - p_{jj}[k-1] + \sum_{i \in \mathcal{N}_j} p_{ji}[k-1] = \varepsilon_j[k-1] \ . \end{aligned}$$

If we let $p_{jj}[k] = p_{jj}[k-1] + \Delta p_{jj}[k]$ for some positive $\Delta p_{jj}[k]$, it follows that $\varepsilon_j[k] - \varepsilon_j[k-1] = -\Delta p_{jj}[k]$ and also that $p_{lj}[k] = p_{lj}[k-1] - c_{lj}\Delta p_{jj}[k]$ for all $l$ such that $p_{lj}[k-1] \neq 0$; this can be used to establish that

$$|\varepsilon_l[k] - \varepsilon_l[k-1]| \leq c_{lj}\Delta p_{jj}[k], \ \forall l \text{ such that } l \in \mathcal{N}_j \ .$$

Thus, the worst case occurs when node $j$ can only send information to nodes $l \neq j$ that also satisfy $\rho_l[k-1] \leq 1$, which means that $\varepsilon_l[k] - \varepsilon_l[k-1] = c_{lj}\Delta p_{jj}[k]$ and thus, in the worst case, $\varepsilon[k] - \varepsilon[k-1] = \sum_{l=1}^{n} \varepsilon_l[k] - \sum_{l=1}^{n} \varepsilon_l[k-1] = 0$ (recall that $\sum_l c_{lj} = 1$). Otherwise, $\varepsilon[k] - \varepsilon[k-1] < 0$. A similar argument can be made when $\rho_j[k-1] > 1$. ■

The following theorem essentially establishes that the error $\varepsilon[k]$ converges to zero even when nodes simultaneously update their self-weights and the weights on their out-going links. The proof is not included due to space limitations; it is based on a related proof in [11] which is available online.

*Theorem 1:* Let $\overline{P}$ be the matrix in (7) and define $P[0] = \overline{P}\Delta[0] + (I - \Delta[0])$ where $I$ is the $n \times n$ identity matrix and $\Delta[0] = \text{diag}\big(\delta_1[0], \delta_2[0], \ldots, \delta_j[0], \ldots, \delta_n[0]\big)$ is a diagonal matrix with nonzero entries $\delta_j[0] = \frac{\mathcal{D}_j^+}{1 + \mathcal{D}_j^+}, \ j = 1, 2, \ldots, n$. Let $P[k] = \overline{P}\Delta[k] + (I - \Delta[k])$ be the update matrix at iteration $k$ with $\Delta[k] = \text{diag}\big(\delta_1[k], \delta_2[k], \ldots, \delta_j[k], \ldots, \delta_n[k]\big)$ obtained from $\Delta[k-1]$ according to the update in (11). Then, $\lim_{k \to \infty} P[k]$ exists and it is a doubly stochastic and primitive matrix $P_{ss} = \overline{P}\Delta_{ss} + (I - \Delta_{ss})$, where $\Delta_{ss} = \text{diag}(\delta_1^{ss}, \delta_2^{ss}, \ldots, \delta_n^{ss})$ with $\delta_j^{ss} = \lim_{k \to \infty} \delta_j[k]$ satisfying $0 < \delta_j^{ss} \leq 1, \ \forall j = 1, 2, \ldots, n$, and $\delta_i^{ss} < 1$ for at least one $i \in \{1, 2, \ldots, n\}$.

## IV. Weight Adjustment Strategies and Examples

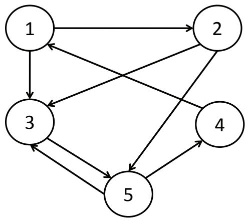
Note that the discussion in the previous section holds for any matrix $\overline{P}$ in (7) as long as the nonnegative coefficients $c_{ij}$ satisfy the conditions stated in the beginning of this section ($\sum_i c_{ij} = 1$, $c_{ij} > 0$ if $(i, j) \in \mathcal{E}$, and $c_{ij} = 0$ otherwise). Note that it is important that $c_{ij} > 0$ for all links $(i, j) \in \mathcal{E}$ in the graph to ensure that the underlying graph remains strongly connected. We now discuss some particular choices for these coefficients.

- *Equal:* In this case, $c_{ij} = 1/\mathcal{D}_j^+$ if and only if $(i, j) \in \mathcal{E}$ (one can easily verify that $\sum_i c_{ij} = 1$). This is essentially the choice studied in [11].
- *Random:* In this case, $c_{ij}$ are chosen as realizations of independent uniform random variables in $(0, 1)$ if and only if $(i, j) \in \mathcal{E}$ (and are zero otherwise), and are then normalized so that $\sum_i c_{ij} = 1$.
- *(Inverse) In-Degree Weighted:* In this case, $c_{ij} = \frac{\mathcal{D}_i^-}{C_j^-}$ (respectively, $c_{ij} = \frac{1/\mathcal{D}_i^-}{C_j^-}$) if and only if $(i, j) \in \mathcal{E}$; otherwise, $c_{ij} = 0$. The constant $C_j^-$ is chosen so that $\sum_i c_{ij} = 1$, i.e., $C_j^- = \sum_{i: j \in \mathcal{N}_i} \mathcal{D}_i^-$ (respectively, $C_j^- = \sum_{i: j \in \mathcal{N}_i} \frac{1}{\mathcal{D}_i^-}$) is the sum of the (inverses of the) in-degrees of the nodes that have $j$ as a neighbor.
- *(Inverse) Out-Degree Weighted:* In this case, $c_{ij} = \frac{\mathcal{D}_i^+}{C_j^+}$ (respectively, $c_{ij} = \frac{1/\mathcal{D}_i^+}{C_j^+}$) if and only if $(i, j) \in \mathcal{E}$; otherwise, $c_{ij} = 0$. The constant $C_j^+$ is chosen so

that $\sum_i c_{ij} = 1$, i.e., $C_j^+ = \sum_{i:j\in\mathcal{N}_i} \mathcal{D}_i^+$ (respectively, $C_j^+ = \sum_{i:j\in\mathcal{N}_i} \frac{1}{\mathcal{D}_i^+}$) is the sum of the (inverses of the) out-degrees of the nodes that have $j$ as a neighbor.

*Remark 2:* Since in a wireless setting, node $j$ can broadcast the same information to all of its neighbors via a single transmission, the easiest way to implement the equally weighted update is to have node $j$ at iteration $k$ broadcast the weight value which is identical for all the nodes receiving its value (i.e., node $j$ can broadcast the value $\frac{1-p_{jj}[k]}{\mathcal{D}_j^+}$). This does not appear to be the case when the weight values are unequal, however, under certain conditions, one can easily ensure that nodes can obtain the proper weights following a single broadcast by node $j$. For instance, if the normalization constant $C_j^-$ or $C_j^+$ is obtained by node $j$ during the initialization phase of the (inverse) in-degree or the (inverse) out-degree weighted update, then node $j$ at iteration $k$ only needs to broadcast the value $\frac{1-p_{jj}[k]}{C_j^-}$ (or $\frac{1-p_{jj}[k]}{C_j^+}$); each node $i$ can then recover its weight by multiplying by (the inverse of) its own in- (out-) degree. □

Consider the directed graph shown at the top of Fig. 1 and assume that the initial values of the five nodes are $x = [4, 5, 6, 3, 2]'$, with average $\mu = 4$. We run the iteration in (4)–(11) and plot the error $\varepsilon[k] = \sum_{j=1}^5 \left|\sum_{i=1}^5 p_{ji}[k] - 1\right|$ as a function of the number of iterations $k$ for two different weighting strategies. In particular, we focus on the equal weighting strategy (whose weight matrix $\overline{P}_e$ is shown at the bottom left of Fig. 1) and the strategy that chooses weights according to the out-degrees of the neighbors (whose weight matrix $\overline{P}_o$ is shown at the bottom right of Fig. 1). To understand how these weights are obtained, we illustrate the situation for node 1: this node has two out-going links to nodes 2 and 3, so the equal weight strategy assigns a coefficient of .5 to the links that go to each one of them; however, if we weight according to the out-degrees, the weight for node 2 becomes $2/3$ and the weight for node 3 becomes $1/3$ (because the out-degrees for node 2 and 3 are $\mathcal{D}_2^+ = 2$ and $\mathcal{D}_3^+ = 1$ respectively).



$$\overline{P}_e = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ .5 & 0 & 0 & 0 & 0 \\ .5 & .5 & 0 & 0 & .5 \\ 0 & 0 & 0 & 0 & .5 \\ 0 & .5 & 1 & 0 & 0 \end{bmatrix} \quad \overline{P}_o = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ .67 & 0 & 0 & 0 & 0 \\ .33 & .33 & 0 & 0 & .5 \\ 0 & 0 & 0 & 0 & .5 \\ 0 & .67 & 1 & 0 & 0 \end{bmatrix}$$

Fig. 1. Small directed graph (top) used for illustration of the class of algorithms and its associated weight matrices (bottom), $\overline{P}_e$ for equal weighting and $\overline{P}_o$ for weighting according to out-degrees.

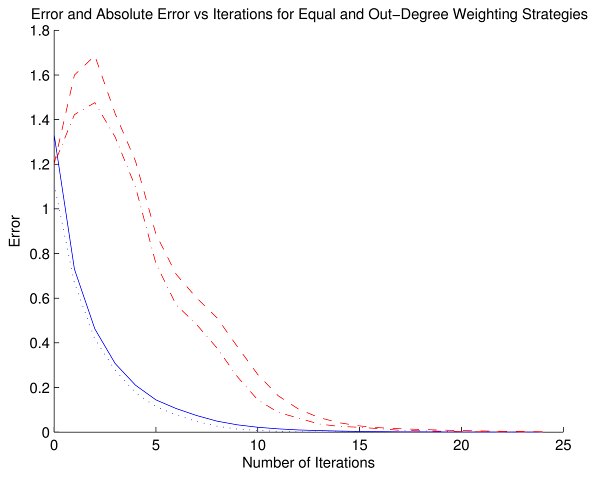


Fig. 2. Error and absolute error, plotted against the number of iterations, for the strategies that use equal and out-degree weighting.

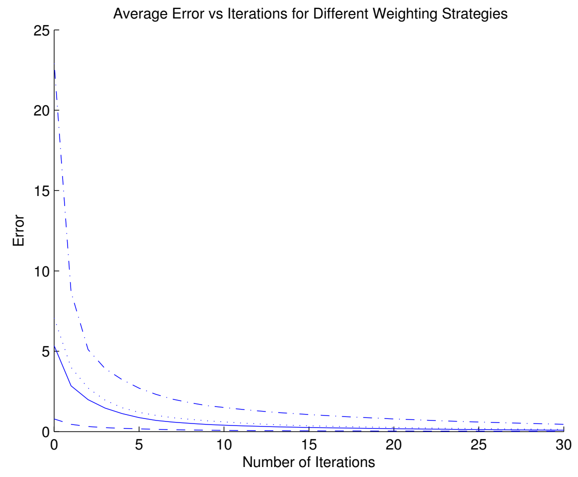


Fig. 3. Error plotted against the number of iterations for four different weighting strategies: equal (solid line), random (dashed-dotted line), *inverse* in-degree (dashed line), and *inverse* out-degree (dotted line).

Fig. 2 plots the error $\varepsilon[k] = \sum_{j=1}^5 \left|\sum_{i=1}^5 p_{ji}[k] - 1\right|$ as a function of the number of iterations $k$ for equal weighting (solid line) and out-degree weighting (dotted line). As we can see, the error goes to zero regardless of the weighting, though the strategy that uses out-degree weighting appears to be slightly faster. The plot in Fig. 2 also shows the converge of the absolute error $\frac{1}{5}\sum_{j=1}^5 |\pi_j[k] - \mu|$ to zero as a function of the number of iterations $k$ for equal weighting (dashed line) and for out-degree weighting (dashed-dotted line). As expected both strategies reach average consensus, with the strategy that updates weights according to out-degrees having a minor advantage in terms of speed of convergence. Note that the two strategies converge to different primitive doubly stochastic matrices.

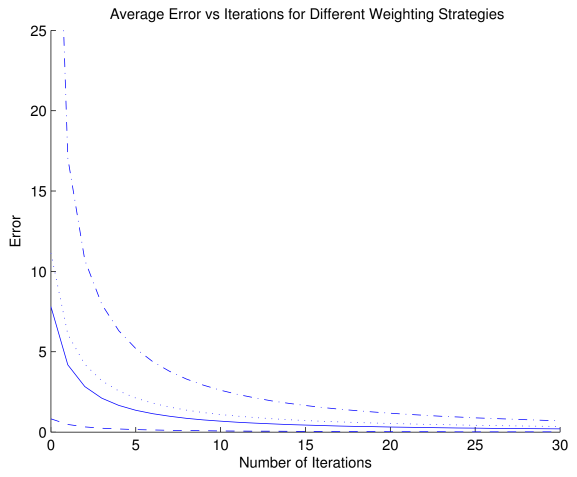


Fig. 4. Average error plotted against the number of iterations for four different weighting strategies: equal (solid line), random (dashed-dotted line), *inverse* in-degree (dashed line), and *inverse* out-degree (dotted line).

Fig. 3 shows what happens in the case of a randomly created graph of 50 nodes. In particular we plot the error $\varepsilon[k] = \sum_{j=1}^{50} \left| \sum_{i=1}^{50} p_{ji}[k] - 1 \right|$ as a function of the number of iterations $k$ for equal weighting (solid line), random weighting (dashed-dotted line), *inverse* in-degree weighting (dashed line), and *inverse* out-degree weighting (dotted line). The plots suggest that inverse in-degree weighting results in slightly faster convergence whereas random weighting leads to much worse convergence. This is also the case in Fig. 4 where we plot the average error $\varepsilon[k] = \frac{1}{100} \sum_{l=1}^{100} \sum_{j=1}^{100} \left| \sum_{i=1}^{100} p_{ji}^{(l)}[k] - 1 \right|$ for the four strategies, averaged over 100 randomly created graphs (index by $l = 1, 2, ..., 100$) of 100 nodes each.[2]

## V. Conclusions and Future Work

In this paper we study the problem of average-consensus in arbitrary (possibly directed) graphs. We start from a very general setup where nodes (i) can choose their self-weights and the weights on their out-going links, and (ii) observe (but do not choose) the weights on their incoming links. Then, we develop a class of distributed algorithms that allow the nodes to iteratively choose their weights so that eventually the set of weights they converge to is a doubly stochastic primitive matrix, which allows them to reach average-consensus. The only requirement is that the underlying communication graph is strongly connected. Apart from establishing the correctness of the algorithms, we also discussed implementation aspects of each of the algorithms and provided numerical examples to illustrate their performance and convergence speed. The main difference between this work and our previous work [11] is the generalization of the choice of weight updates, something that led to different update strategies that may provide faster convergence when additional information is available. Future work will study the adaptation of these techniques to cases where the underlying communication graph is allowed to change over time. In this regard, an important feature of the proposed strategies is that they do not rely on a fixed interconnection topology because nodes that sense changes on their out-going links (e.g., the appearance or disappearance of a link) can easily adjust (e.g., reset) the weights on their out-going links and their self-weight, without affecting the asymptotic outcome of the iteration.

[2]The random graphs were created by choosing a directed edge from node $i$ to node $j$, $1 \leq i, j \leq n$, $i \neq j$, independently with probability $1/2$, and ensuring that the resulting graph is strongly connected.

## References

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

[2] 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.

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

[4] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation*. Athena Scientific, 1997.

[5] 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.

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

[7] 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.

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

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

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

[11] A. D. Domínguez-García and C. N. Hadjicostis, “Average consensus on directed graphs,” under review.

[12] B. Gharesifard and J. Cortés, “Distributed strategies for making a digraph weight-balanced,” in *Proceedings of Annual Allerton Conference on Communication, Control, and Computing*, October 2009, pp. 771–777.

[13] ——, “When does a digraph admit a doubly stochastic adjacency matrix?” in *Proceedings of American Control Conference*, June 2010, pp. 2440–2445.

[14] ——, “Distributed strategies for generating weight-balanced and doubly stochastic digraphs,” 2010. [Online]. Available: http://arxiv.org/abs/0911.0232

[15] 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.

[16] Y. Chen, R. Tron, A. Terzis, and R. Vidal, “Corrective consensus: Converging to the exact average,” in *Proceedings of IEEE Conference on Decision and Control (CDC)*, Dec. 2010, pp. 1221–1228.

[17] F. Benezit, V. Blondel, P. Thiran, J. Tsitsiklis, and M. Vetterli, “Weighted gossip: Distributed averaging using non-doubly stochastic matrices,” in *Proceedings of IEEE International Symposium on Information Theory (ISIT)*, Jun. 2010, pp. 1753–1757.

[18] R. Olfati-Saber and R. M. Murray, “Agreement problems in networks with directed graphs and switching topology,” in *Proceedings of American Control Conference*, vol. 4, 2003, pp. 4123–4132.

[19] R. Horn and C. Johnson, *Matrix Analysis*. New York, NY: Cambridge University Press, 1985.

[20] E. Seneta, *Non-negative Matrices and Markov Chains*. New York, NY: Springer, 2006.