Robustified Distributed Average Consensus

Themistoklis Charalambous[a,b], Alejandro D. Domínguez-García[c] and Christoforos N. Hadjicostis[a]
[a]University of Cyprus, Department of Electrical and Computer Engineering, 1 Panepistimiou Av., 2109 Aglatzia, P.O. Box 20537, 1678 Nicosia, Cyprus
[b]Aalto University, Department of Electrical Engineering and Automation, P.O. Box 11110, FI-00076 Aalto, Finland
[c]University of Illinois, Department of Electrical and Computer Engineering, 306 N. Wright Street, Urbana, IL 61801, USA

Chapter Article tagline: update of previous edition,, reprint..

Abstract

Distributed average consensus is a key primitive for accomplishing a variety of distributed estimation, control, and optimization tasks in multi-agent systems. This article discusses a popular class of such algorithms, as well as their application to distributed control and optimization settings. The class of algorithms we focus on operate in an iterative fashion: at each iteration, each agent updates a set of locally held variables by linearly combining them with similar values it receives from its immediate neighboring agents; each agent then transmits its updated values to its neighboring agents and the iteration continues. Such linear iterative schemes have become popular due to their simplicity and ease of implementation, however, they are vulnerable to malfunctions or attacks in the commnunication network on which their operation relies, or the agents themselves. The article provides an overview of techniques for robustifying these linear iterative schemes against network abnormalities (e.g., packet delays or drops) and malfunctioning agents (e.g., faulty or malicious agents).

Keywords: Automatic Repeat Request, Distributed Averaging, Distributed Control, Optimization and Learning, Faulty/Malicious Agents, Linear Iteration, Local Invariance, Multi-Agent Systems, Packet Delays, Packet Drops, Push Sum, Ratio Consensus, Running-Sum Ratio Consensus, Weak Ergodicity

Key Points/Objectives

- Distributed averaging algorithms are key for applications in distributed optimization, learning, control and coordination;
- Distributed detection, identification, and correction against faulty/malicious activity is one of the critical challenges in emerging networked/distributed applications;
- Local invariant conditions are instrumental for detecting/identifying faulty/malicious agents;
- Running-sum ratio consensus or running-sum push sum schemes enable resiliency to network abnormalities (packet delays and/or drops) and faulty/malicious agents.

1 Introduction and Motivation

A multi-agent system consists of a set of agents that can share information with neighboring agents via connection links, forming a generally partially connected and possibly directed communication topology. In many applications of multi-agent systems, some of which are elaborated upon later in this introduction, the agents aim to calculate a function of certain parameters that are held at individual nodes. Depending on the underlying application, the parameter at each node could be a measurement of some quantity of interest (e.g., from a sensor in a sensor network), the speed or direction of a mobile robot in a robotic network, or the scheduled output of a generator in a power network. When all agents calculate the average of these parameters, the agents are said to reach average consensus. Average consensus and, more generally, distributed function calculation have been studied by the control community, the communication community, and the computer science community (see, e.g., the works by Tsitsiklis (1984); Lynch (1996); Koetter and Médard (2003); Olfati-Saber and Murray (2004); Hromkovic et al. (2005); Cortés (2008)). For control applications, average consensus has been studied primarily in settings where convergence to the average is asymptotic and each node processes and transmits real-valued states with infinite precision (see, e.g., the works by Jadbabaie et al. (2003); Olfati-Saber and Murray (2004); Cao et al. (2008); Olshevsky and Tsitsiklis (2009)). Various extensions have also been addressed: issues of finite time completion were addressed by Sundaram and Hadjicostis (2007); Wang and Xiao (2010); Yuan et al. (2013); Hendrickx et al. (2014), quantized transmissions by Kashyap et al. (2007); Aysal et al. (2008); Nedic et al. (2009); Lavaei and Murray (2011); Cai and Ishii (2011), and event-triggered operation by Seyboth et al. (2013); Nowzari et al. (2019); Rikos and Hadjicostis (2020a,b).

1.1 Connections to Distributed Optimization and Learning

Distributed optimization constitutes a framework in which agents cooperatively solve a global optimization problem. In the basic setup, each agent has only *local* (and, hence, partial) knowledge of the entire problem; for example, it knows a local part of the objective function and the constraints. A typical goal is to optimize the sum of local objective functions subject to a number of (local and/or global) constraints.

The aim is to iteratively solve the problem locally at each individual agent, which can subsequently communicate and exchange information with other agents. Consensus algorithms are then deployed to combine the local solutions and achieve a globally optimal solution, over multiple iterations. Distributed (average) consensus algorithms play a pivotal role in this context by enabling agents to share information with their neighbors about their local estimates and/or gradients, and iteratively update their local estimates of the optimal solution based on local information and messages received from neighboring agents, in order to solve the optimization problem collaboratively. Among other advantages, this setting eliminates the reliance on the central node, which constitutes a single point of failure and imposes a significant communication load, and makes the scheme more easily adaptable and scalable.

Several distributed optimization algorithms deploy (average) consensus mechanisms to solve distributed optimization problems, including the following:

1. In distributed gradient descent methods, each agent computes a local gradient of the objective function and updates its local estimate by taking a step in the direction of the negative gradient. Consensus algorithms help in aggregating these local updates to reach global consensus (see, e.g., the works by Xi and Khan (2017); Nedić et al. (2017); Xi et al. (2018); Notarstefano et al. (2019), and the references therein).
2. Distributed Alternating Direction Method of Multipliers (ADMM) is a popular algorithm for solving convex distributed optimization problems with distributed constraints. Consensus steps in ADMM ensure that agents reach consensus on the dual variables, guaranteeing the convergence of the method; see, for example, the works by Shi et al. (2014); Khatana and Salapaka (2020); Jiang and Charalambous (2021); Khatana and Salapaka (2023); Rikos et al. (2023), and references therein.

When working with big data, training time exponentially increases, prohibiting scalability and online optimization. Making use of a distributed approach, training on huge amounts of data can be accelerated, facilitating fast decision-making and optimization. Distributed learning algorithms, such as distributed stochastic gradient descent (SGD) or federated learning, deploy consensus protocols to train machine learning models across multiple devices or agents; see, for example, the works by Scaman et al. (2019); Spiridonoff et al. (2020); Verbraeken et al. (2020).

1.2 Some Engineering Applications

Beyond the applications discussed above, distributed average consensus algorithms have found numerous applications in electric power systems (see, e.g., the works by Domínguez-García and Hadjicostis (2010); Domínguez-García and Hadjicostis (2011); Domínguez-García et al. (2012b,a); Cady et al. (2015b,a, 2017), and the references therein). One of the earliest applications of these algorithms appears in the work by Domínguez-García and Hadjicostis (2010), wherein the problem is to coordinate the power outputs of a set of distributed energy resources (DERs) so that the sum of the power outputs is equal to some pre-specified value while respecting individual DER capacity constraints; such problem is important in, e.g., enabling the utilization of the DERs for real-time load balancing in a power network. If there is no cost associated to each of the DERs, the problem reduces to a feasibility problem; then, one solution is to allocate power to each DER as a convex combination of its upper and lower capacity limits, with the parameter specifying such convex combination being a function of the pre-specified (total power) value to be provided by the DERs and their upper and lower capacity limits. The problem faced by the DERs is then to compute said parameter in a distributed fashion. As it turns out, such a problem can be viewed as a distributed weighted average consensus problem, which can be solved using a variant of the algorithm in Section 2.3.

One can generalize the setting discussed above by endowing each DER with a cost function; then, the problem can be cast as an optimization program where the goal is to determine the power outputs of each DER that result in minimum total cost while respecting the DER individual capacity limits, and ensuring the total power output they provide is equal to the pre-specified amount of power the DERs should collectively provide. This problem is referred to as the economic dispatch (ED) problem in the power systems literature, wherein either linear or quadratic costs are assumed (see, e.g., the book by Bergen and Vittal (2000)). One approach to solving the problem is to formulate its Lagrange dual. As it turns out, the structure of the dual problem is such that it can also be solved in a distributed fashion using the algorithm in Section 2.3; see the works by Domínguez-García et al. (2012a) for the quadratic cost case, and by Xu et al. (2017) for the linear cost case.

Another problem where distributed averaging has found application is frequency regulation in microgrids operating in islanded mode (see, e.g., the work by Cady et al. (2015a, 2017)). Specifically, one would like to use feedback control to regulate the power output of the generating units so as to balance the load in real time in order to regulate the system frequency to some nominal value, e.g., 60 Hz. The problem is solved using an integral controller wherein the error is the deviation from said nominal value of the average of the frequency in the microgrid network measured at different points. While such a controller is typically implemented in a centralized fashion with a single processor having access to the aforementioned frequency measurements, in certain applications where fault-tolerance and resilience are key, e.g., military microgrids, it is desirable to implement the controller in a distributed fashion; this can be easily accomplished as follows. The controller of each generating resource can measure frequency locally. Then, assuming the controllers can exchange information with neighboring controllers, they can execute a distributed averaging algorithm to obtain the average frequency. Once each controller has learned the average frequency, they can use it as an input to a local copy of the integral-based controller; the reader is referred to the work by Cady et al. (2017) for further details.

2 Linear Iterative Schemes for Distributed Average Consensus

2.1 Graph Notation and Problem Formulation

We can conveniently capture a multi-agent system via a digraph (directed graph) $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$, $N \geq 2$, is the set of agents (nodes), and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} - \{(v_j, v_j) \mid v_j \in \mathcal{V}\}$ is the set of interconnections among them. More specifically, $(v_j, v_i) \in \mathcal{E}$ if node v_j may receive information from node v_i. We assume that $\mathcal{G}$ is strongly connected, which means that, for each pair of nodes $v_j, v_i \in \mathcal{V}$, $v_j \neq v_i$, there exists a directed *path* from v_i to v_j (i.e., we can find a sequence of nodes $v_i =: v_{l_0}, v_{l_1}, \ldots, v_{l_t} := v_j$ such that $(v_{l_{\tau+1}}, v_{l_\tau}) \in \mathcal{E}$ for $\tau = 0, 1, \ldots, t-1$). Note that the exchange of information between pairs of nodes may be *asymmetric*, i.e., v_j may receive information from node v_i but not vice-versa; in such case, we have that $(v_j, v_i) \in \mathcal{E}$ but $(v_i, v_j) \notin \mathcal{E}$. We define the set of in-neighbors of node v_j as $\mathcal{N}_j^- := \{v_i \in \mathcal{V} \mid (v_j, v_i) \in \mathcal{E}\}$, and the set of out-neighbors of node v_j as $\mathcal{N}_j^+ := \{v_l \in \mathcal{V} \mid (v_l, v_j) \in \mathcal{E}\}$. The cardinalities of these sets are referred to as the in-degree and out-degree of node v_j, denoted respectively as $D_j^- = |\mathcal{N}_j^-|$ and $D_j^+ = |\mathcal{N}_j^+|$.

Problem Formulation: We assume that each node v_j possesses a value V_j and the objective for all the nodes is to compute the average of the V_j's, i.e., compute

$$\overline{V} := \frac{\sum_{\ell=1}^{N} V_\ell}{N} \ . \tag{1}$$

The computation of $\overline{V}$ has to be done following a distributed iterative algorithm that adheres to the communication constraints imposed by $\mathcal{G}$, i.e., at each iteration, each node can only receive values from its in-neighbors and can only send values to its out-neighbors. We assume a simple broadcast model throughout this article: each node v_j transmits a value (or values) in a single packet that is received (at least under nominal conditions) by all of its out-neighbors in the set $\mathcal{N}_j^+$. Furthermore, each packet is associated with a unique identifier that allows receiving nodes to identify the transmitting node.

2.2 Single Linear Iteration

One of the most popular algorithms for distributed average consensus is one in which each node v_j maintains a scalar state variable $x_j[k]$, which it updates at each iteration k based on the values it receives from its in-neighbors. The updated $x_j[k+1]$ is then sent to the out-neighbors of node v_j, and the iteration continues. Specifically, each node v_j uses a linear time-invariant update of the form

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

where $x_j[0] = V_j$ for all $v_j \in \mathcal{V}$, and the p_{ji}'s are constant weights, chosen to satisfy certain constraints that we describe later in this section. If we let $x[k] = [x_1[k], x_2[k], \ldots, x_N[k]]^{\mathrm{T}}$, then the iteration in (2) can be written in matrix form as

$$x[k+1] = Px[k] \ , \quad x[0] = [V_1, V_2, \ldots, V_N]^{\mathrm{T}}, \tag{3}$$

where $P = [p(j,i)]$ is the $(N \times N)$=dimensional weight matrix with its entry $p(j,i)$ at the jth row, ith column satisfying $p(j,i) = p_{ji}$ if $v_i \in \mathcal{N}_j^- \cup \{v_j\}$ (zero otherwise). In other words, the constraint (imposed to adhere to the distributivity requirements) is that $p(j,i) = 0$ if $v_i \notin \mathcal{N}_j^- \cup \{v_j\}$.

The nodes are said to reach asymptotic average consensus if

$$\lim_{k \to \infty} x_j[k] = \frac{\sum_{\ell=1}^{N} x_\ell[0]}{N} = \overline{V} \ , \quad \forall v_j \in \mathcal{V} \ .$$

The necessary and sufficient conditions for the iteration in (3) to asymptotically reach average consensus are: (i) P has a simple eigenvalue at 1, with left eigenvector 1_N^{T} and right eigenvector 1_N (where 1_N denotes the N-dimensional all ones column vector), and (ii) all other eigenvalues of P have magnitude strictly less than 1 (see, e.g., the works by Xiao and Boyd (2004); Olfati-Saber and Murray (2004)). If one focuses on nonnegative weights, these conditions are equivalent to P being a primitive doubly stochastic matrix. The rate of convergence to average consensus of the iteration in (3) is governed by the magnitude of the second largest in magnitude eigenvalue of matrix P (see, e.g., the work by Xiao and Boyd (2004)).

In the case of an undirected[1] graph or, more generally, a balanced digraph,[2] there are very simple ways for the nodes to choose the weights in a distributed manner so that P forms a primitive doubly stochastic matrix (see, e.g., the monograph by Hadjicostis et al. (2018)). Since in these graphs we have $D_j^+ = D_j^- := D_j$ for all $v_j \in \mathcal{V}$, one possibility is the following: each node v_j sets the weights on all of its in-coming links to be $p_{ji} = \frac{1}{N'}$ for all $v_i \in \mathcal{N}_j^-$ and $p_{jj} = 1 - \frac{D_j}{N'}$ (zero otherwise), where N' is an upper bound on the total number of nodes (i.e., $N' \geq N$). It is easy to verify that P will be a doubly stochastic matrix (in fact, P will also be symmetric in the case of an undirected graph). Furthermore, P will be primitive as long as the graph $\mathcal{G}$ is strongly connected. Another possibility for choosing weights in the case of an undirected graph is the so-called Metropolis weights in the work by Xiao et al. (2005) where $p_{ji} = p_{ij} = 1/(1 + \max\{D_i, D_j\})$ when $(v_j, v_i) \in \mathcal{E}$ (and, thus, $(v_i, v_j) \in \mathcal{E}$) and $p_{jj} = 1 - \sum_{v_i \in \mathcal{N}_j} p_{ji}$ where $\mathcal{N}_j = \mathcal{N}_j^+ = \mathcal{N}_j^-$ is the set of neighbors of node v_j (weights are set to zero

[1] An undirected graph is a digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in which $(v_j, v_i) \in \mathcal{E}$ if and only if $(v_i, v_j) \in \mathcal{E}$; thus, in an undirected graph, we trivially have $\mathcal{N}_j^+ = \mathcal{N}_j^- := \mathcal{N}_j$ and $D_j^+ = D_j^- := D_j$ for all $v_j \in \mathcal{V}$.

[2] A balanced digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is one in which $D_j^+ = D_j^- := D_j$ for all $v_j \in \mathcal{V}$.

otherwise).

When the underlying topology $\mathcal{G}$ is not necessarily undirected or balanced, choosing weights that form a primitive doubly stochastic matrix is harder (because, in general, $D_j^+ \neq D_j^-$). The problem reduces to a matrix scaling/balancing problem, centralized solutions for which are summarized in the survey by Schneider and Zenios (1990); distributed solutions to such problems are more challenging (see, e.g., the more recent works by Gharesifard and Cortés (2012); Domínguez-García and Hadjicostis (2013); Rikos et al. (2014)).

2.3 Double Linear Iteration

Consider a distributed system, captured by a digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, in which each node $v_j \in \mathcal{V}$ has an initial value V_j. Ratio consensus is an iterative distributed algorithm that allows the nodes to asymptotically calculate the average $\overline{V}$ in (1) as the ratio of two local variables that they update in a linear iterative manner, *without* having to obtain weights that form a doubly stochastic matrix.

Each node v_j maintains a pair of scalar state variables, $y_j[k]$ and $z_j[k]$, which it updates at each iteration k based on similar pairs of state variables it receives from its in-neighbors. The updated $y_j[k+1]$ and $z_j[k+1]$ are then sent to the out-neighbors of node v_j and the iteration continues. Specifically, each node v_j uses linear time-invariant updates of the form

$$y_j[k+1] = \sum_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} y_i[k]/(1+D_i^+), \tag{4}$$

$$z_j[k+1] = \sum_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} z_i[k]/(1+D_i^+), \tag{5}$$

where $y_j[0] = V_j$, and $z_j[0] = 1$, for all $v_j \in \mathcal{V}$. The protocol assumes that each node v_j is aware of its out-degree D_j^+ and can be implemented by having each node transmit the values $\overline{y}_j[k] := y_j[k]/(1+D_j^+)$ and $\overline{z}_j[k] := z_j[k]/(1+D_j^+)$ in a single broadcast transmission that is received by all of its out-neighbors; each receiving node simply adds the values it receives from its in-neighbors.

It can be shown (see, e.g., the monograph by Hadjicostis et al. (2018)) that, as long as the underlying digraph $\mathcal{G}$ is strongly connected, the ratio $r_j[k] := y_j[k]/z_j[k]$ at each node v_j asymptotically converges to the average of the initial values, i.e.,

$$\lim_{k\to\infty} r_j[k] = \frac{\sum_\ell y_\ell[0]}{\sum_\ell z_\ell[0]} = \overline{V}\ , \quad \forall\ v_j \in \mathcal{V}\ . \tag{6}$$

To see this, let us use matrix-vector notation to write the ratio consensus iterations as

$$y[k+1] = Py[k]\ , \tag{7}$$

$$z[k+1] = Pz[k]\ , \tag{8}$$

where $y[k] = [y_1[k], y_2[k], \ldots, y_N[k]]^{\mathrm{T}}$, $z[k] = [z_1[k], z_2[k], \ldots, z_N[k]]^{\mathrm{T}}$ (with $y[0]$ and $z[0]$ initialized as above), and $P = [p(j,i)]$ is a *column stochastic* matrix such that the entry $p(j,i)$ at the jth row, ith column satisfies $p(j,i) = 1/(1+D_i^+)$ if $v_j \in \mathcal{N}_i^+ \cup \{v_i\}$ (zero otherwise). It is easy to see that P will also be primitive if the underlying digraph $\mathcal{G}$ is strongly connected. In such case, we have $\lim_{k\to\infty} P^k = \pi 1^{\mathrm{T}}$, where π is the right eigenvector of P that corresponds to eigenvalue 1 (by the Perron-Frobenius theorem, π is a strictly positive vector, normalized in this case so that the sum of its entries is equal to unity). Therefore, we have that $\lim_{k\to\infty} y[k] = \lim_{k\to\infty} P^k y[0] = \pi(1^{\mathrm{T}} y[0])$ and $\lim_{k\to\infty} z[k] = \lim_{k\to\infty} P^k z[0] = \pi(1^{\mathrm{T}} z[0])$, from which it follows that

$$\lim_{k\to\infty} r_j[k] = \frac{\pi_j \times \left(\sum_{\ell=1}^N y_\ell[0]\right)}{\pi_j \times \left(\sum_{\ell=1}^N z_\ell[0]\right)} = \overline{V}\ .$$

Notice that π_j, the jth entry of vector π, is strictly positive and need not be known to the nodes.

From the above argument, it is clear that ratio consensus works with any primitive column stochastic matrix P (whose zero/one structure—excluding the diagonal elements—necessarily reflects the constraints captured by the given digraph, so that the ratio consensus protocol conforms to the given distributivity constraints); see, e.g., the monograph by Hadjicostis et al. (2018). In fact, a variation of ratio consensus can also lead to asymptotic average consensus even when the communication topology is changing. Next, we describe the so-called push sum algorithm for the case when the communication topology is time-varying, i.e., at iteration k, the topology is captured by a digraph $\mathcal{G}[k] = (\mathcal{V}, \mathcal{E}[k])$ where $\mathcal{E}[k] \subseteq \mathcal{E}$. Below, we use $\mathcal{N}_j^+[k]$ ($D_j^+[k]$) to denote the set (number) of out-neighbors of node v_j at iteration k, and $\mathcal{N}_j^-[k]$ ($D_j^-[k]$) to denote the set (number) of in-neighbors of node v_j at iteration k. The push-sum iterations, which are essentially a time-varying version of the ratio consensus iterations, are given by

$$y[k+1] = P[k]y[k], \tag{9}$$

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

where $P[k] = [p[k](j,i)]$ is a column stochastic matrix such that the entry $p[k](j,i)$ at its jth row, ith column satisfies $p[k](j,i) = 1/(1+D_i^+[k])$ if $v_j \in \mathcal{N}_i^+[k] \cup \{v_i\}$ (zero otherwise). In terms of implementation, the push-sum protocol assumes that each node v_j is aware of its (instantaneous) out-degree $D_j^+[k]$ at iteration k, and transmits the values $\overline{y}_j[k] := y_j[k]/(1+D_j^+[k])$ and $\overline{z}_j[k] := z_j[k]/(1+D_j^+[k])$ in a single transmission to all of its out-neighbors; each receiving node simply adds the values it receives from its in-neighbors.

Subject to some mild joint properties on the sequence of matrices $P[k]$, convergence in (6) still holds though the proof is more complex (see, e.g., the works by Kempe et al. (2003); Bénézit et al. (2010); Hadjicostis and Charalambous (2014); Hadjicostis et al. (2016)). From

(9)–(10), we can write

$$\begin{aligned} y[k+1] &= T[k]y[0] \ , \\ z[k+1] &= T[k]z[0] \ , \end{aligned}$$

where $T[k] = P[k]P[k-1]...P[1]P[0]$ is a column stochastic matrix that governs convergence. It can be shown that, under the mild assumption stated below, as k goes to infinity, we have

$$T[k] = \pi[k]1_N^\top, \tag{11}$$

where $\pi[k] = [\pi_1[k], \pi_2[k], \ldots, \pi_N[k]]^\top$ is a (nonnegative) stochastic vector (i.e., $0 \leq \pi_i[k] \leq 1$ for all $i = 1, 2, \ldots, N$, and $\sum_{\ell=1}^N \pi_\ell[k] = 1$). In other words, *weak ergodicity* of the product of column stochastic matrices $P[k]$ is obtained according to the terminology by Seneta (2006); this intuitively means that the columns of $T[k]$ equalize asymptotically but they do not necessarily converge (i.e., in general, they will depend on k). By using (11), as k goes to infinity, it is easy to see that

$$y_j[k] = \pi_j[k]\Big(\sum_{\ell=1}^N y_\ell[0]\Big), \tag{12}$$

$$z_j[k] = \pi_j[k]\Big(\sum_{\ell=1}^N z_\ell[0]\Big). \tag{13}$$

which means that the ratio $r_j[k] = y_j[k]/z_j[k]$ asymptotically converges[3] to the average of the initial values, even though $y_j[k]$ and $z_j[k]$ do not converge in general. Again node v_j does not need not know explicitly the values $\pi_j[k]$.

With the particular choice of weights we made in this section,[4] a sufficient condition for reaching asymptotic average consensus using the push-sum algorithm is to be able to find a finite K such that each union digraph

$$\mathcal{G}[\tau K] \cup \mathcal{G}[\tau K+1] \cup \ldots \cup \mathcal{G}[\tau K+K-1] := (\mathcal{V}, \mathcal{E}[\tau K] \cup \mathcal{E}[\tau K+1] \cup \ldots \cup \mathcal{E}[\tau K+K-1])$$

for $\tau = 0, 1, 2, \ldots$, is strongly connected.

3 Robustification against Packet-Dropping Links and Delays

3.1 Running-Sum Ratio Consensus

The *running-sum ratio consensus* algorithm (see, e.g., the monograph by Hadjicostis et al. (2018)) is a variation of the ratio consensus algorithm used to overcome packet drops (e.g., the work by Domínguez-García et al. (2012b)) or other network-induced uncertainties, such as unknown out degrees (e.g., the work by Hadjicostis et al. (2016)), when performing distributed average consensus over an arbitrary (strongly connected) digraph. The main idea in running-sum ratio consensus is that, at time step k, instead of broadcasting $\bar{y}_j[k] := y_j[k]/(1+D_j^+)$ and $\bar{z}_j[k] := z_j[k]/(1+D_j^+)$, node v_j broadcasts its y- and z-running sums, denoted by σ_j and η_j respectively, and defined as follows:

$$\begin{aligned} \sigma_j[k+1] &:= \sum_{t=0}^{k} y_j[t]/(1+D_j^+) \ , \\ \eta_j[k+1] &:= \sum_{t=0}^{k} z_j[t]/(1+D_j^+) \ . \end{aligned}$$

Clearly, each receiving node can recover $\bar{y}_j[k] = y_j[k]/(1+D_j^+)$, $\bar{z}_j[k] = z_j[k]/(1+D_j^+)$, by taking the differences $\sigma_j[k+1] - \sigma_j[k]$ and $\eta_j[k+1] - \eta_j[k]$ (i.e., the difference between two consecutive running sums received from node v_j). Thus, apart from broadcasting/receiving y- and z-running sums, the ratio consensus algorithm could be run in exactly the same way (by first recovering the needed values at each iteration), modulo some additional bookkeeping at each node. More specifically, each node v_j needs to maintain the following additional variables:

1. Node v_j maintains its own *out-going* running sums $\sigma_j[k]$ and $\eta_j[k]$, which it initializes at $\sigma_j[0] = 0$ and $\eta_j[0] = 0$, and updates recursively as

$$\sigma_j[k+1] = \sigma_j[k] + y_j[k]/(1+D_j^+) \ , \tag{14}$$
$$\eta_j[k+1] = \eta_j[k] + z_j[k]/(1+D_j^+) \ . \tag{15}$$

2. Node v_j also stores the pair of *in-coming* running sums it receives from each of its in-neighbors (so that it can use it to take the differences when it receives the updated pair of running sums at the next iteration). More specifically, for each in-neighbor $v_i \in \mathcal{N}_j^-$,

[3] Depending on the instantiation of matrix $P[k-1]$, it is possible that $y_j[k] = z_j[k] = 0$ for one or more nodes $v_j \in \mathcal{V}$; in such case, the ratio $r_j[k]$ is taken to be undefined.
[4] More generally, we need to ensure that nonzero weights in matrices $P[k]$ are bounded away from zero (and, of course, that each $P[k]$ is column stochastic with its entry $p[k](j,i)$ at the jth row, ith column satisfying $p[k](j,i) \neq 0$ if and only if $v_j \in \mathcal{N}_i^+ \cup \{v_i\}$).

node v_j maintains two in-coming running sums, namely $\rho_{ji}[k]$, which keeps track of the y-running sum last broadcast by node v_i, and $\mu_{ji}[k]$, which keeps track of the z-running sum last broadcast by node v_i. Indirectly, this assumes that each node is able to associate an incoming message with the node that sends it (e.g., via a unique identifier).

At each iteration of running-sum ratio consensus, each node v_j updates its variables as

$$y_j[k+1] = \sum_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} (\rho_{ji}[k+1] - \rho_{ji}[k]) , \tag{16}$$

$$z_j[k+1] = \sum_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} (\mu_{ji}[k+1] - \mu_{ji}[k]) . \tag{17}$$

where $\rho_{ji}[k] = \sigma_i[k]$ and $\mu_{ji}[k] = \eta_i[k]$ (both ρ and μ are initialized at zero, i.e., $\rho_{ji}[0] = 0$ and $\mu_{ji}[0] = 0$). Clearly, the iterations above reduce to the ratio consensus iterations in (4)–(5); thus, as long as the underlying digraph is strongly connected, the ratio $r_j[k] := y_j[k]/z_j[k]$, for each node $v_j \in \mathcal{V}$, asymptotically converges to the average $\overline{V}$ of the initial values.

3.2 Handling Packet Drops

The reason running-sum ratio consensus was first proposed was to handle occasional packet drops by the communication links, i.e., handle situations where the pair of values $\sigma_i[k+1]$ and $\eta_i[k+1]$ that are sent by node v_i at iteration k never reach a particular out-neighbor $v_j \in \mathcal{N}_i^+$. To model such situations, one can think of $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as the nominal (strongly connected) digraph that captures the interactions between nodes in a multi-agent system. Then, at each iteration, we can use $\mathcal{E}[k]$, $\mathcal{E}[k] \subseteq \mathcal{E}$, to denote the set of *active* communication links. Specifically, $(v_j, v_i) \in \mathcal{E}[k]$ if the transmission from node v_i to node v_j is successful (i.e., there is no packet drop at iteration k). This results in a sequence of digraphs $\mathcal{G} = (\mathcal{V}, \mathcal{E}[k])$ capturing the set of edges that are active at each iteration k.

The approaches by Domínguez-García et al. (2012b); Hadjicostis et al. (2016) handle packet drops by having each node execute the running-sum ratio consensus iterations in (14)–(17), where each node $v_j \in \mathcal{V}$ initializes all ρ_{ji} and μ_{ji}, for each $v_i \in \mathcal{N}_j^- \cup \{v_j\}$, to zero and updates them at each iteration as

$$\rho_{ji}[k+1] = \begin{cases} \sigma_i[k+1], & \text{if } (v_j, v_i) \in \mathcal{E}[k] \text{ or if } i = j , \\ \rho_{ji}[k], & \text{if } (v_j, v_i) \notin \mathcal{E}[k] \text{ and } i \neq j , \end{cases} \tag{18}$$

$$\mu_{ji}[k+1] = \begin{cases} \eta_i[k+1], & \text{if } (v_j, v_i) \in \mathcal{E}[k] \text{ or if } i = j , \\ \mu_{ji}[k], & \text{if } (v_j, v_i) \notin \mathcal{E}[k] \text{ and } i \neq j . \end{cases} \tag{19}$$

When a packet drop occurs at iteration k on the link (v_j, v_i) (from node v_i to node v_j), the updates of $y_j[k+1]$ and $z_j[k+1]$ do not take into account node v_i, since $\rho_{ji}[k+1] - \rho_{ji}[k] = 0$ and $\mu_{ji}[k+1] - \mu_{ji}[k] = 0$. However, when the transmission on this link is successful at a subsequent step, say at iteration $k+1$, node v_j will be able to incorporate in its update of $y_j[k+2]$ (respectively, $z_j[k+2]$) not only the effects of the most recent transmission from node v_i, *but also the effects of the lost transmission*, by taking the difference $\rho_{ji}[k+2] - \rho_{ji}[k+1] = \sigma_i[k+2] - \sigma_i[k] = \frac{y_i[k+1]}{1+D_i^+} + \frac{y_i[k]}{1+D_i^+}$ (respectively, $\mu_{ji}[k+2] - \mu_{ji}[k+1] = \eta_i[k+2] - \eta_i[k] = \frac{z_i[k+1]}{1+D_i^+} + \frac{z_i[k]}{1+D_i^+}$).

The proof of the following theorem can be established using techniques by Hadjicostis et al. (2016). Note that stochastic versions of the theorem below have also been formulated (see, e.g., the work by Domínguez-García et al. (2012b)).

Theorem 1. *Consider a nominal strongly connected digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where nodes execute the running-sum ratio consensus algorithm in the presence of packet dropping links as described in* (14)–(19). *Let $\mathcal{E}[k]$, $\mathcal{E}[k] \subseteq \mathcal{E}$, be the set of active edges at iteration k (i.e., packets on the links in the set $\mathcal{E} \setminus \mathcal{E}[k]$ are dropped). Assume that there exists a finite integer K such that*[5]

$$\mathcal{E}[\tau K] \cup \mathcal{E}[\tau K + 1] \cup \ldots \cup \mathcal{E}[\tau K + K - 1] = \mathcal{E}$$

for $\tau = 0, 1, 2, \ldots$. Then, for each $v_j \in \mathcal{V}$, we have

$$\lim_{k\to\infty} r_j[k] = \frac{\sum_{\ell=1}^N y_\ell[0]}{\sum_{\ell=1}^N z_\ell[0]} = \overline{V} ,$$

where $r_j[k] := y_j[k]/z_j[k]$.

The main idea for the proof (more details can be found in the works by Domínguez-García et al. (2012b); Hadjicostis et al. (2016)) is to introduce a virtual buffer node b_{ji} for each edge (v_j, v_i) in the given digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Virtual buffer node b_{ji} has node v_i as its only in-neighbor and node v_j as its only out-neighbor. The link from node v_i to virtual buffer node b_{ji} is active when there is a packet drop on link (v_j, v_i), whereas the link from b_{ji} to node v_j is active when there is no packet drop. Essentially, the buffer node b_{ji} is used to store the cumulative y and z values that node v_i meant to send to node v_j but failed to do so due to packet drops. When there are consecutive packet drops on link (v_j, v_i), values keep accumulating in the virtual buffer node b_{ji}; however, as soon as there is a successful transmission, all accumulated values transfer to node v_j. The addition of the virtual buffer nodes allows us to model the execution of the running-sum ratio consensus algorithm under packet drops as the execution of the push-sum iterations (under no packet drops) in (9)–(10) over an augmented

[5]The condition essentially requires that on each link there is at least one successful transmission every K iterations.

digraph $\mathcal{G}_a = (\mathcal{V}_a, \mathcal{E}_a)$, where

$$\begin{aligned} \mathcal{V}_a &= \mathcal{V} \cup \{b_{ji} \mid (v_j, v_i) \in \mathcal{E}\} , \\ \mathcal{E}_a &= \mathcal{E} \cup \{(b_{ji}, v_i),\ (v_j, b_{ji}) \mid (v_j, v_i) \in \mathcal{E}\} . \end{aligned}$$

More specifically, for appropriately chosen augmented y_a and z_a vectors, and appropriately chosen column stochastic matrices $P_a[k]$, we can write the evolution of the state variables involved in the execution of the algorithm as

$$y_a[k+1] = P_a[k] y_a[k] , \tag{20}$$

$$z_a[k+1] = P_a[k] z_a[k] , \tag{21}$$

where y_a and z_a are N_a-dimensional column vectors with $N_a = |\mathcal{V}| + |\mathcal{E}|$. Each $P_a[k]$ is an $N_a \times N_a$ column stochastic matrix that depends on $\mathcal{E}[k]$ (i.e., there are $2^{|\mathcal{E}|}$ different matrix choices at each k). The final argument to establish the above theorem relies on weak ergodicity techniques but has to also take into account the fact that some of the virtual buffer nodes may have undefined ratios (immediately after a successful transmission on link (v_j, v_i), both the y and z values held at virtual buffer node b_{ji} are zero).

3.3 Handling Retransmissions and Packet Drops in Automatic Repeat Request (ARQ) Error Control Protocol

An average consensus mechanism that can overcome packet retransmissions (information delays) and packet-dropping links (information loss) over directed networks is introduced in the work by Makridis et al. (2023b) by incorporating an Automatic Repeat reQuest (ARQ) error control protocol into the ratio consensus algorithm. ARQ uses error-detection codes, acknowledgment (ACK) or negative acknowledgment (NACK) messages, and retransmissions to maintain the reliability of data transmissions over unreliable communication channels. Under the ARQ protocol, a packet is retransmitted after each NACK (sent by the receiver), until it is successfully received (without errors). The packet can be retransmitted for up to a predefined number of retransmission trials before it is dropped. Different approaches can be considered depending on how the retransmissions take place (e.g., unicast or broadcast). Here, we assume the Multiple Transmissions Multiple Feedback (MTMF) scheme, in which each new packet is transmitted individually along with possibly retransmitted (delayed) packets.

We describe the ARQ-based Ratio Consensus algorithm by considering a topology captured by a digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each node transmits (receives) data packets to (from) its out-neighbors (in-neighbors) utilizing the ARQ protocol. Specifically, each node v_j executes the following steps:

1. During the initialization phase of the algorithm, each node broadcasts a few dummy packets and acquires the number of its out-neighbors by obtaining the sum of its incoming ARQ feedback signals (ACK/NACK), which equals the number of its out-neighbors.
2. Given its value V_j, each node v_j initializes the two variables it maintains to $y_j[0] = V_j$ and $z_j[0] = 1$. To keep track of possibly lost information due to packet drops, each node v_j maintains the accumulated y_j and z_j masses that it wants to transmit to each of its out-neighbors, denoted by $\sigma_j[k]$ and $\eta_j[k]$ in (14)–(15), and initialized to $\sigma_j[0] = 0$ and $\eta_j[0] = 0$, respectively. Similarly, each node maintains the variables $\rho_{ji}[k]$ and $\mu_{ji}[k]$ that keep track of the accumulated y and z masses received at node v_j from each in-neighbor $v_i \in \mathcal{N}_j^-$; these variables are initialized at $\rho_{ji}[0] = 0$ and $\mu_{ji}[0] = 0$.
3. Next, node v_j updates the variables $y_j[k+1]$ and $z_j[k+1]$ it maintains using the equations

$$y_j[k+1] = \sum_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} \sum_{r=0}^{\bar{\tau}_{ji}} \iota_{ji}[k-r] \bar{y}_i[k-r], \tag{22}$$

$$z_j[k+1] = \sum_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} \sum_{r=0}^{\bar{\tau}_{ji}} \iota_{ji}[k-r] \bar{z}_i[k-r], \tag{23}$$

whenever the actual packet retransmissions over link $(v_j, v_i) \in \mathcal{E}$ have *not* reached the maximum retransmission limit, i.e., as long as $\tau_{ji}[k] < \bar{\tau}_{ji}$, where $\bar{\tau}_{ji}$ is the maximum retransmission limit (i.e., the maximum number of times a packet is retransmitted, imposed by the ARQ protocol) on link (v_j, v_i) and $\tau_{ji}[k]$ is the transmission delay on the packet transmitted (for the first time) on link (v_j, v_i) at iteration k. Here, $\bar{y}_i[k-r] = p_{ji} y_i[k-1]$ and $\bar{z}_i[k-r] = p_{ji} z_i[k-1]$, where $p_{ji} = P(j, i)$ is the entry at the jth row, ith column of the column stochastic matrix P in (9)–(10). The indicator function, $\iota_{ji}[k-r]$, indicates whether the bounded delay $\tau_{ji}[k-r] \le \bar{\tau}_{ji}$ on link (v_j, v_i) at iteration $k-r$, equals r (which means that the transmission on link (v_j, v_i) at iteration $k-r$ successfully reaches node v_j at iteration k) and is defined as

$$\iota_{ji}[k-r] = \begin{cases} 1, & \text{if } \tau_{ji}[k-r] = r, \\ 0, & \text{otherwise.} \end{cases} \tag{24}$$

It is important to note that node v_j processes (delayed) packets *as soon as they arrive successfully*. Clearly, in the absence of delays, this strategy reduces to the ratio consensus algorithm (4)–(5).

4. If the retransmission limit of a packet sent by node v_j is reached, then the running-sum mechanism is activated. At this stage, the running-sum variables $\sigma_j[k+1]$ and $\eta_j[k+1]$ are updated according to (14) and (15). The accumulated information in the running-sum variables of node v_j is transmitted to its out-neighbors. If the packet arrives at the receiving node v_l without errors, then variables $\rho_{lj}[k+1]$ and $\mu_{lj}[k+1]$ are updated with the accumulated masses $\sigma_j[k+1]$ and $\eta_j[k+1]$ according to (18) and (19), respectively. In other words, with the next (possibly delayed) successful transmission, the updated running sums reach their destination node v_l, which updates its state

variables accordingly.

Remark 1. The ARQ-based consensus mechanism has been enhanced further in the work by Makridis et al. (2023a) by incorporating a hybrid ARQ (HARQ) mechanism. It has been shown that the decoding error when HARQ is invoked reduces exponentially with the number of retransmissions (see, e.g., the work by Lagrange (2010)). In particular, if at every time step the packet is retransmitted, the following model can be adopted from the work by Ceran et al. (2019):

$$q_{ji}[k - \bar{\tau}_{ji} + r] = q_{ji}[k - \bar{\tau}_{ji}]\lambda^r, \tag{25}$$

where the probability that a packet initially transmitted over the link (v_j, v_i) at time slot $k - \bar{\tau}_{ji}$ is erroneous is denoted by $q_{ji}[k - \bar{\tau}_{ji}]$ (similarly, at time slot $k - \bar{\tau}_{ji} + r$, $r \leq \bar{\tau}_{ji}$, that probability is denoted by $q_{ji}[k - \bar{\tau}_{ji} + r]$), and the rate of packet error probability varies with parameter $\lambda \in (0, 1]$. This implies that, for small λ, the error probability decreases faster with retransmissions. Setting $\lambda = 1$, the HARQ protocol reduces to the ARQ-based scheme. More details about the adaptation of the ratio consensus algorithm for operation under the HARQ error control protocol and its proof of convergence can be found in the work by Makridis et al. (2023a).

4 Robustification against Faulty/Malicious Nodes

4.1 Running Sums over Time-Varying Topology

In this section we assume a situation where we have a time-varying topology and execute a variation of the running-sum ratio consensus algorithm. More specifically, we are given a nominal strongly connected digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a sequence of digraphs $\mathcal{G}[k] = (\mathcal{V}[k], \mathcal{E}[k])$, $k = 0, 1, 2, \ldots$, where $\mathcal{V}[k] \subseteq \mathcal{V}$ is the set of normal (i.e., non-faulty or non-malicious) nodes at iteration k and $\mathcal{E}[k] = \mathcal{E} \cap (\mathcal{V}[k] \times \mathcal{V}[k])$ is the set of edges among them. As earlier, we use $\mathcal{N}_j^- := \{v_i \in \mathcal{V} \mid (v_j, v_i) \in \mathcal{E}\}$ to denote the set of nominal in-neighbors of node v_j and $\mathcal{N}_j^+ := \{v_l \in \mathcal{V} \mid (v_l, v_j) \in \mathcal{E}\}$ to denote the set of nominal out-neighbors of node v_j; and we use $\mathcal{N}_j^-[k] := \{v_i \in \mathcal{V} \mid (v_j, v_i) \in \mathcal{E}[k]\} = \mathcal{V}[k] \cap \mathcal{N}_j^-$ to denote the set of normal in-neighbors of node v_j at iteration k and $\mathcal{N}_j^+[k] := \{v_l \in \mathcal{V} \mid (v_l, v_j) \in \mathcal{E}[k]\} = \mathcal{V}[k] \cap \mathcal{N}_j^+$ to denote the set of normal out-neighbors of node v_j at iteration k. The cardinalities of these sets are referred to as the (nominal) in- and out-degree of node v_j and the (instantaneous) in- and out-degree of node v_j at iteration k, defined respectively as $D_j^- = |\mathcal{N}_j^-|$, $D_j^+ = |\mathcal{N}_j^+|$, $D_j^-[k] = |\mathcal{N}_j^-[k]|$ and $D_j^+[k] = |\mathcal{N}_j^+[k]|$.

For simplicity, we assume that there is a mechanism that allows normal nodes to determine whether other nodes are faulty/malicious. This also implies that at each iteration k, each normal node is able to identify which of its in- or out-neighbors are normal. Furthermore, once a node is declared faulty/malicious, it remains as such, i.e., $\mathcal{V}[k+1] \subseteq \mathcal{V}[k]$ for $k = 0, 1, 2, \ldots$. Finally, we assume that there exists a finite k_0 such that $\mathcal{V}[k] = \mathcal{V}_N$ for all $k \geq k_0$. In other words, eventually the set of normal nodes stabilizes to the set $\mathcal{V}_N$.

The variation of running-sum ratio consensus we are interested in operates as follows: each node v_j uses the number of its (normal) out-neighbors at iteration k (given by $D_j^+[k] = |\mathcal{N}_j^+[k]|$) to update its y- and z-running sums as follows:

$$\sigma_j[k+1] \quad := \quad \sum_{t=0}^{k} y_j[t]/(1 + D_j^+[t]) \,, \tag{26}$$

$$\eta_j[k+1] \quad := \quad \sum_{t=0}^{k} z_j[t]/(1 + D_j^+[t]) \,. \tag{27}$$

Note that these sums can be easily updated iteratively as

$$\sigma_j[k+1] \quad = \quad \sigma_j[k] + y_j[k]/(1 + D_j^+[k]) \,, \tag{28}$$

$$\eta_j[k+1] \quad = \quad \eta_j[k] + z_j[k]/(1 + D_j^+[k]) \,. \tag{29}$$

As in the case of the running-sum ratio consensus presented in the previous section, this variation requires each node v_j to (i) track the running-sum values $\sigma_j[k]$ and $\eta_j[k]$ it broadcasts (updated as described in (28)–(29)), and (ii) maintain, for each in-neighbor $v_i \in \mathcal{N}_j^-$, two in-coming running sums, namely, $\rho_{ji}[k]$, which keeps track of the y-running sum broadcast by node v_i, with $\rho_{ji}[0] = 0$, and $\mu_{ji}[k]$, which keeps track of the z-running sum broadcast by node v_i, with $\mu_{ji}[0] = 0$. Of course, this assumes that each node is able to associate an incoming message with the node that sends it (e.g., via a unique identifier).

With these modifications, at iteration k, node v_j updates its y and z values based on the running sums it receives from its (nominal) in-neighbors as follows:

$$y_j[k+1] \quad = \quad \sum_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} (\rho_{ji}[k+1] - \rho_{ji}[k]) \,, \tag{30}$$

$$z_j[k+1] \quad = \quad \sum_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} (\mu_{ji}[k+1] - \mu_{ji}[k]) \,, \tag{31}$$

with the caveat that ρ and μ are updated at iteration k as

$$\rho_{ji}[k+1] \quad = \quad \begin{cases} \sigma_i[k+1], & \text{if } v_i \in \mathcal{N}_j^-[k] \text{ or if } i = j \ , \\ 0, & \text{if } v_i \notin \mathcal{N}_j^-[k] \text{ and } i \neq j \ , \end{cases} \tag{32}$$

$$\mu_{ji}[k+1] \quad = \quad \begin{cases} \eta_i[k+1], & \text{if } v_i \in \mathcal{N}_j^-[k] \text{ or if } i = j \ , \\ 0, & \text{if } v_i \notin \mathcal{N}_j^-[k] \text{ and } i \neq j \ . \end{cases} \tag{33}$$

In terms of implementation, the above iterations could be executed easily, as long as each node v_j is aware at each iteration k of $\mathcal{N}_j^-[k]$ (i.e., the subset of its in-neighbors that are normal at iteration k) in order to properly update the ρ's and μ's it maintains. In addition, each node v_j is aware of $D_j^+[k]$ (i.e., the number of its out-neighbors that are normal) in order to properly update its running sums σ and η.

At this point, it should be clear that, in the long run, the nodes that are not in $\mathcal{V}_N$ are excluded from the distributed computation (since they do not belong in the set $\mathcal{V}_N$ after iteration k_0, their ρ's and μ's in (32)–(33) are set to zero by all normal nodes and do not have any further effect on the updates of the values maintained at normal nodes). In fact, after iteration k_0, the distributed computation is performed over the digraph $\mathcal{G}_N = (\mathcal{V}_N, \mathcal{E} \cap (\mathcal{V}_N \times \mathcal{V}_N))$. Clearly, if $k_0 = 0$, then it follows from the discussion on running-sum ratio consensus that the algorithm will converge to the average

$$\overline{V}_N := \frac{\sum_{v_\ell \in \mathcal{V}_N} V_\ell}{|\mathcal{V}_N|} \ , \tag{34}$$

as long as the digraph $\mathcal{G}_N$ is strongly connected. The question at this point is what happens if k_0 is greater than zero. Can faulty/malicious nodes affect the outcome of the computation since they are allowed to interact with normal nodes for a finite number of time steps before their status is revealed to normal nodes? The following theorem clarifies that this is *not* the case and that normal nodes asymptotically converge to $\overline{V}_N$.

Theorem 2. *Consider a nominal strongly connected digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a sequence of digraphs $\mathcal{G}[k] = (\mathcal{V}[k], \mathcal{E}[k])$, $k = 0, 1, 2, \ldots$, where $\mathcal{V}[k] \subseteq \mathcal{V}$ is a set of normal nodes at iteration k and $\mathcal{E}[k] = \mathcal{E} \cap (\mathcal{V}[k] \times \mathcal{V}[k])$ is the set of edges among them. Assume that $\mathcal{V}[k+1] \subseteq \mathcal{V}[k]$ for all k and that there exists a finite k_0 such that $\mathcal{V}[k] = \mathcal{V}_N$ for all $k \geq k_0$ and that the digraph $\mathcal{G}_N = (\mathcal{V}_N, \mathcal{E} \cap (\mathcal{V}_N \times \mathcal{V}_N))$ is strongly connected. Then, if normal nodes in $\mathcal{V}_N$ execute the running-sum ratio consensus algorithm over a time-varying topology as described in* (28)–(33)*, they asymptotically converge to the average of their initial values: for each $v_j \in \mathcal{V}_N$, we have*

$$\lim_{k \to \infty} r_j[k] = \overline{V}_N \ ,$$

where $r_j[k] := y_j[k]/z_j[k]$ and $\overline{V}_N$ was defined in (34)*.*

Remark 2. Notice that the theorem above does not state what is the behavior of faulty/malicious nodes, which are allowed to behave arbitrarily (i.e., update their variables in an arbitrary manner). The reason is that the updates in (28)–(33) essentially ensure that when node v_i is identified as faulty or malicious, say at iteration k', then $\rho_{ji}[k'+1] = 0$ ($\mu_{ji}[k'+1] = 0$) for all nodes v_j that are out-neighbors of node v_i. This means that $y_j[k'+1]$ ($z_j[k'+1]$) will subtract $\rho_{ji}[k]$ ($\mu_{ji}[k]$) in (30) (in (31)), which effectively removes all effect that node v_i had on its out-neighboring node v_j up to iteration k'. Note that we still assume a broadcast communication model, which implies that malicious nodes cannot behave in a Byzantine manner (i.e., the same, possibly corrupted value is received by all of the out-neighbors of a malicious node). This assumption can potentially be relaxed, but we adopt it at this point for ease of exposition.

The proof of Theorem 2 hinges on the following theorem, the proof of which can be established by induction using techniques similar to the ones by Hadjicostis and Domínguez-García (2024) and is omitted.

Theorem 3. *Consider the setting in Theorem 2. At each iteration $k = 0, 1, 2, \ldots$, it holds for each normal node $v_j \in \mathcal{V}_N$*

$$y_j[k] + D_j^+[k-1]\sigma_j[k] - \sum_{v_i \in \mathcal{N}_j^-[k-1]} \sigma_i[k] \quad = \quad V_j \ , \tag{35}$$

$$z_j[k] + D_j^+[k-1]\eta_j[k] - \sum_{v_i \in \mathcal{N}_j^-[k-1]} \eta_i[k] \quad = \quad 1 \ , \tag{36}$$

where $\sigma_j[0] = \eta_j[0] = 0$ for all $v_j \in \mathcal{V}$ (notice that $\mathcal{N}_j^-[-1]$ and $D_j^+[-1]$ are not critical here but can be taken to satisfy $\mathcal{N}_j^-[-1] = \mathcal{N}_j^-$ and $D_j^+[-1] = D_j^+$, respectively).

Using the invariant properties in Theorem 3 we can prove Theorem 2 as follows.

Proof. For $k \geq k_0$, the normal nodes execute the running-sum ratio consensus algorithm over digraph $\mathcal{G}_N$. Since $\mathcal{G}_N$ is assumed to be strongly connected, the ratios $r_j[k] = y_j[k]/z_j[k]$ of all normal nodes will converge to

$$\lim_{k \to \infty} r_j[k] = \frac{\sum_{v_\ell \in \mathcal{V}_N} y_\ell[k_0+1]}{\sum_{v_\ell \in \mathcal{V}_N} z_\ell[k_0+1]} \ , \quad \forall \ v_j \in \mathcal{V}_N \ . \tag{37}$$

Using the invariant in Theorem 3 at time $k_0 + 1$, we have

$$y_j[k_0+1] + D_j^+[k_0]\sigma_j[k_0+1] - \sum_{v_i \in \mathcal{N}_j^-[k_0]} \sigma_i[k_0+1] = V_j \ ,$$

where $D_j^+[k_0]$ ($\mathcal{N}_j^-[k_0]$) is the out-degree (set of in-neighbors) of node v_j in the digraph $\mathcal{G}_N$. If we sum all such equations over all v_j in $\mathcal{V}_N$, on the right, we have the sum $\sum_{v_j \in \mathcal{V}_N} V_j$, whereas on the left we have

$$\textstyle\sum_{v_j \in \mathcal{V}_N} y_j[k_0+1] + \underbrace{\sum_{v_j \in \mathcal{V}_N} (D_j^+[k_0]\sigma_j[k_0+1]) - \sum_{v_j \in \mathcal{V}_N} \left(\sum_{v_i \in \mathcal{N}_j^-[k_0]} \sigma_i[k_0+1] \right)}_{=0} = \textstyle\sum_{v_j \in \mathcal{V}_N} y_j[k_0'+1] \ .$$

The simplification occurs because, for each link (v_j, v_i) of $\mathcal{G}_N$, we have σ_j appearing twice, once with a positive sign and once with a negative sign. This establishes that $\sum_{v_l \in \mathcal{V}_N} y_l[k_0+1] = \sum_{v_l \in \mathcal{V}_N} V_l$. One can similarly establish that $\sum_{v_l \in \mathcal{V}_N} z_l[k_0+1] = |\mathcal{V}_N|$. Therefore, all ratios $r_j[k]$ in (37) converge to $\overline{V}_N$. □

4.2 Handling Faulty/Malicious Nodes

Consider the setting described in the previous section, where we are given a nominal strongly connected digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each node $v_j \in \mathcal{V}$ has an initial value V_j and the goal is to calculate the average $\overline{V}$. However, during this process, certain nodes may be discovered to be faulty or malicious. If eventually the set of normal (non-faulty/non-malicious) nodes is $\mathcal{V}_N$, then we would like to calculate the average of the normal nodes $\overline{V}_N$. If we let $V[k]$ denote the set of normal nodes at iteration k, this situation can be modeled by the sequence of digraphs $\mathcal{G}[k] = (\mathcal{V}[k], \mathcal{E}[k])$, $k = 0, 1, 2, \ldots$, where $\mathcal{V}[k] \subseteq \mathcal{V}$ is a set of normal nodes at iteration k and $\mathcal{E}[k] = \mathcal{E} \cap (\mathcal{V}[k] \times \mathcal{V}[k])$ is the set of edges among them. The discussion in the previous section essentially establishes that by following the running-sum average consensus algorithm over the time-varying topology, the normal nodes will converge to $\overline{V}_N$.

Notice that what is interesting is that it does not matter *when* a faulty/malicious node is actually discovered. As long as all such nodes are eventually discovered, the end result for the normal nodes remains they same: they asymptotically converge to the average $\overline{V}_N$. Importantly, the effect of the past interactions that faulty/malicious nodes had up to the point when they were discovered to be faulty/malicious is collectively removed by their neighboring normal nodes.

The ability to remove the effects of the past interactions with faulty/malicious nodes when they are discovered as such was taken advantage of in the work by Hadjicostis and Domínguez-García (2022) in order to handle malicious nodes. The setting there assumed an initially unknown set of malicious nodes $\mathcal{V}_M \subseteq \mathcal{V}$ (that may also depend on their behavior during the execution of the distributed algorithm). At each time step k, each normal node $v_j \in \mathcal{V}_N$ (note that $\mathcal{V}_N = \mathcal{V} \setminus \mathcal{V}_M$) is given a binary indicator $t_{ij}[k]$ ($t_{lj}[k]$) that captures the trustworthiness of each in-neighbor $v_i \in \mathcal{N}_j^-$ (each out-neighbor $v_l \in \mathcal{N}_j^+$) from the perspective of node v_j. More specifically, $t_{ij}[k] = 1$ ($t_{lj}[k] = 1$) if in-neighbor v_i (out-neighbor v_l) is perceived trustworthy by node v_j; otherwise, $t_{ij}[k] = 0$ ($t_{lj}[k] = 0$). Note that the techniques by Hadjicostis and Domínguez-García (2022) only required $t_{ji}[k]$ and $t_{lj}[k]$ to correctly reflect the trustworthiness of neighbors of node v_j asymptotically. More specifically, there exists some k_0 such that for $k \geq k_0$, we have for each $v_j \in \mathcal{V}_N$

$$\begin{aligned} \{v_i \in \mathcal{N}_j^- \mid t_{ij}[k] = 1\} &= \mathcal{N}_j^- \cap \mathcal{V}_N \ , \\ \{v_l \in \mathcal{N}_j^+ \mid t_{lj}[k] = 1\} &= \mathcal{N}_j^+ \cap \mathcal{V}_N \ . \end{aligned}$$

However, for $k < k_0$ the t_{ij}'s and t_{lj}'s could be incorrect or flactuate.

The idea in the work by Hadjicostis and Domínguez-García (2022) was to run a modified version of the running-sum ratio consensus algorithm over a time-varying topology (as presented in (28)–(33) in the previous section) by having each node $v_j \in \mathcal{V}_N$ use, at each iteration k, the following sets of in-neighbors and out-neighbors:

$$\begin{aligned} \mathcal{N}_j^-[k] &= \{v_i \in \mathcal{N}_j^- \mid t_{ij}[k] = 1\} \ , \\ \mathcal{N}_j^+[k] &= \{v_l \in \mathcal{N}_j^+ \mid t_{lj}[k] = 1\} \ . \end{aligned}$$

The problem is, of course, the fact that the above sets of instantaneous in- and out-neighbors of node v_j do *not* necessarily satisfy $\mathcal{N}_j^-[k] = \mathcal{V}_N \cap \mathcal{N}_j^-$ or $\mathcal{N}_j^+[k] = \mathcal{V}_N \cap \mathcal{N}_j^+$ (since the t_{ji}'s and the t_{lj}'s may not correctly reflect the status of node v_j's in- and out-neighbors for iterations before k_0). To handle this problem, the approach by Hadjicostis and Domínguez-García (2022) imposes the following additional requirements (on top of the algorithm in the previous section):

- Case 1: Previously thought trustworthy out-neighbor v_l is now thought to be untrustworthy ($t_{lj}[k] = 0$ and $t_{lj}[k-1] = 1$). In this case, node v_j needs to adjust for all transmissions that it sent to node v_l while it was considered trustworthy. It does so by adding to its y and z values its own y-running sum and z-running sum that it sent to node v_l (note that the y-running sum and z-running sum of node v_j is the same for all out-neighbors):

$$\begin{aligned} y_j[k] &:= y_j[k] + \sigma_j[k] \ , \\ z_j[k] &:= z_j[k] + \eta_j[k] \ . \end{aligned}$$

This adjustment essentially "diverts" all communication to node v_l back to node v_j and it has to be done for every previously thought trustworthy out-neighbor v_l that is now thought to be untrustworthy.

- Case 2: Previously thought untrustworthy out-neighbor v_l is now thought to be trustworthy ($t_{lj}[k] = 1$ and $t_{lj}[k-1] = 0$). In this case, node v_j needs to adjust for all transmissions that node v_l received from v_j while it was considered untrustworthy by node v_j (note that node v_l was receiving transmissions from node v_j even though node v_j did not consider node v_l in its computations). Node v_j can achieve this adjustment by subtracting from its own y and z values, its corresponding y-running sum and z-running sum (note that the y-running sum and z-running sum of node v_j is the same for all out-neighbors):

$$\begin{aligned} y_j[k] &:= y_j[k] - \sigma_j[k] \,, \\ z_j[k] &:= z_j[k] - \eta_j[k] \,. \end{aligned}$$

  The reason is that $\sigma_j[k]$ (respectively, $\eta_j[k]$) represents the cumulative y (respectively, z) values that the previously untrustworthy out-neighbor v_l has been receiving. Since these values will be incorporated in the computation by node v_l who is now considered trustworthy, node v_j can account for this by making an appropriate adjustment on its own values. This adjustment has to be done for every previously thought untrustworthy out-neighbor v_l that is now thought to be trustworthy.

Overall, Cases 1 and 2 can be combined as follows. Letting $\Delta D_j^+[k] := D_j^+[k] - D_j^+[k-1]$ denote the change in the number of trustworthy out-neighbors of node v_j at iteration k, node v_j can compensate by performing the following adjustments to $y_j[k]$ and $z_j[k]$:

$$\begin{aligned} y_j[k] &:= y_j[k] - \Delta D_j^+[k]\sigma_j[k] \,, \\ z_j[k] &:= z_j[k] - \Delta D_j^+[k]\eta_j[k] \,, \end{aligned}$$

and then proceed with the running-sum ratio consensus algorithm over a time-varying topology as in (28)–(33).

Remark 3. The main difference from the previous section is that the set of non-faulty/non-malicious nodes $\mathcal{V}[k]$ (called normal nodes in the previous section) is not assumed precisely known by all normal nodes at each iteration. In fact, different non-faulty/non-malicious nodes in $\mathcal{V}[k]$ could have different views on who is faulty/malicious and who is not. However, as long as eventually (after iteration step k_0), all non-faulty/non-malicious nodes have precise knowledge[6] of $\mathcal{V}_N$, then the nodes will converge to the average $\overline{V}_N$.

One issue that has not been addressed in this section is *how* to detect and identify faulty/malicious nodes (e.g., how to obtain the $t_{lj}[k]$, $t_{lj}[k]$ for each node v_j). Ideally, this needs to be achieved in a distributed manner, e.g., by having each node v_j test its in-neighbors and/or out-neighbors. One such approach is outlined by Hadjicostis and Domínguez-García (2024) by taking advantage of the invariants in Theorem 3. The idea is that any entity (e.g., a neighbor of node v_j that has access to the variables V_j, $y_j[k]$, $D_j^+[k-1]$, $\sigma_j[k]$, and $\sigma_i[k]$ for $v_i \in \mathcal{N}_j^-[k-1]$ (or $z_j[k]$, $D_j^+[k-1]$, $\eta_j[k]$, and $\eta_i[k]$ for $v_i \in \mathcal{N}_j^-[k-1]$) can check whether the invariant holds for node v_j. If the invariant does not hold at any iteration k, then the checking entity can declare node v_j faulty or malicious.

Note that invariant-based checking can be performed *concurrently* at each iteration k (assuming the checking entity has access to the needed variables at each iteration k). To relax the communication requirements, the work by Hadjicostis and Domínguez-García (2024) also discusses *any-time* schemes where the checking entity performs the checks only when it has access (or requests) the needed information for verifying the invariants. Again, if the invariant does not hold, then the checking entity can declare node v_j faulty or malicious. Note that in the case of any-time checks, it is possible for the invariant to hold even though node v_j has been malfunctioning (e.g., if errors that occurred at different points in time cancel themselves out). In this case, this is not necessarily a problem, because we know from Theorem 2 that, as long as the two invariant conditions hold for each non-faulty/non-malicious node when the checks are performed, then these nodes will converge to the average $\overline{V}_N$ (at least if no more errors are introduced in the computation).

5 Conclusions

This article has provided an overview of popular distributed algorithms for reaching average consensus in multi-agent systems whose communication topology is captured by undirected or directed graphs. The distributed computation of the average of quantities held at different agents is key to several control applications of multi-agent systems (e.g., sensor, robotic, or electrical networks), as well as distributed optimization and learning. The focus of the article was on describing some recent work that allows the robustification of these popular distributed algorithms for reaching average consensus using the concept of running sums. Different ways of utilizing running-sum transmissions in ways that allow the handling of packet drops and delays (over unreliable communication links) or the handling of faulty/malicious nodes (that perform arbitrary, erroneous updates of their values) were also discussed.

Acknowledgments

The work of Themistoklis Charalambous was partly supported by the European Research Council (ERC) Consolidator Grant MINERVA (Grant agreement No. 101044629).

[6]In fact, node $v_j \in \mathcal{V}_N$ only needs precise knowledge of who is faulty/malicious among its in-neighbors and out-neighbors.

See Also: C. N. Hadjicostis, A. D. Domínguez-García, and T. Charalambous, "Distributed Averaging and Balancing in Network Systems, with Applications to Coordination and Control." Foundations and Trends® in Systems and Control, vol. 5, no. 3-4, Now Publishers, 2018.
What about other chapters in the Encyclopedia?

References

Aysal TC, Coates MJ and Rabbat MG (2008). Distributed average consensus with dithered quantization. *IEEE Trans. on Signal Processing* 56 (10): 4905–4918.

Bénézit F, Blondel V, Thiran P, Tsitsiklis J and Vetterli M (2010), Weighted gossip: Distributed averaging using non-doubly stochastic matrices, Proc. of IEEE International Symposium on Information Theory (ISIT), 1753–1757.

Bergen A and Vittal V (2000). Power System Analysis, Prentice Hall, Upper Saddle River, NJ.

Cady ST, Domínguez-García AD and Hadjicostis CN (2015a). A distributed generation control architecture for islanded ac microgrids. *IEEE Trans. on Control Systems Technology* 23 (5): 1717–1735.

Cady ST, Domínguez-García AD and Hadjicostis CN (2015b), Finite-time approximate consensus and its application to distributed frequency regulation in islanded AC microgrids, Proc. of Hawaii International Conference on System Sciences (HICSS), 2664–2670.

Cady ST, Zholbaryssov M, Domínguez-García AD and Hadjicostis CN (2017). A distributed frequency regulation architecture for islanded inertialess ac microgrids. *IEEE Trans. on Control Systems Technology* 25 (6): 1961–1977.

Cai K and Ishii H (2011). Quantized consensus and averaging on gossip digraphs. *IEEE Trans. on Automatic Control* 56 (9): 2087–2100.

Cao M, Morse AS and Anderson BDO (2008). Reaching a consensus in a dynamically changing environment: A graphical approach. *SIAM Journal on Control and Optimization* 47 (2): 575–600.

Ceran ET, Gündüz D and György A (2019). Average age of information with hybrid ARQ under a resource constraint. *IEEE Trans. on Wireless Communications* 18 (3): 1900–1913.

Cortés J (2008). Distributed algorithms for reaching consensus on general functions. *Automatica* 44 (3): 726–737.

Domínguez-García AD and Hadjicostis CN (2010), Coordination and control of distributed energy resources for provision of ancillary services, Proc. of IEEE International Conference on Smart Grid Communications (SmartGridComm), 537–542.

Domínguez-García AD and Hadjicostis CN (2011), Distributed algorithms for control of demand response and distributed energy resources, Proc. of 50th IEEE Conference on Decision and Control (CDC), 27–32.

Domínguez-García AD and Hadjicostis CN (2013). Distributed matrix scaling and application to average consensus in directed graphs. *IEEE Trans. on Automatic Control* 58 (3): 667–681.

Domínguez-García AD, Cady ST and Hadjicostis CN (2012a), Decentralized optimal dispatch of distributed energy resources, Proc. of 51st IEEE Conference on Decision and Control (CDC), 3688–3693.

Domínguez-García AD, Hadjicostis CN and Vaidya NH (2012b). Resilient networked control of distributed energy resources. *IEEE Journal on Selected Areas in Communications* 30 (6): 1137–1148.

Gharesifard B and Cortés J (2012). Distributed strategies for generating weight-balanced and doubly stochastic digraphs. *European Journal of Control* 18 (6): 539–557.

Hadjicostis CN and Charalambous T (2014). Average consensus in the presence of delays in directed graph topologies. *IEEE Trans. on Automatic Control* 59 (3): 763–768.

Hadjicostis CN and Domínguez-García AD (2022), Trustworthy distributed average consensus, Proc. of 61st IEEE Conference on Decision and Control (CDC), 7403–7408.

Hadjicostis CN and Domínguez-García A (2024), Invariant properties of linear-iterative distributed averaging algorithms and application to error detection, Proc. of European Control Conference (ECC).

Hadjicostis CN, Vaidya NH and Domínguez-García AD (2016). Robust distributed average consensus via exchange of running sums. *IEEE Trans. on Automatic Control* 61 (6): 1492–1507.

Hadjicostis CN, Domínguez-García AD and Charalambous T (2018). Distributed averaging and balancing in network systems, with applications to coordination and control. *Foundations and Trends® in Systems and Control* 5 (3–4).

Hendrickx JM, Shi G and Johansson KH (2014). Finite-time consensus using stochastic matrices with positive diagonals. *IEEE Trans. on Automatic Control* 60 (4): 1070–1073.

Hromkovic J, Klasing R, Pelc A, Ruzicka P and Unger W (2005). Dissemination of Information in Communication Networks, Springer-Verlag.

Jadbabaie A, Lin J and Morse A (2003). Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Trans. on Automatic Control* 48 (6): 988–1001.

Jiang W and Charalambous T (2021), Distributed alternating direction method of multipliers using finite-time exact ratio consensus in digraphs, Proc. of European Control Conference (ECC), 2205–2212.

Kashyap A, Başar T and Srikant R (2007). Quantized consensus. *Automatica* 43 (7): 1192–1203.

Kempe D, Dobra A and Gehrke J (2003), Gossip-based computation of aggregate information, Proc. of Annual IEEE Symposium on Foundations of Computer Science (FOCS), 482–491.

Khatana V and Salapaka MV (2020), D-DistADMM: A O(1/k) distributed ADMM for distributed optimization in directed graph topologies, Proc. of 59th IEEE Conference on Decision and Control (CDC), 2992–2997.

Khatana V and Salapaka MV (2023). DC-DistADMM: ADMM algorithm for constrained optimization over directed graphs. *IEEE Trans. on Automatic Control* 68 (9): 5365–5380.

Koetter R and Médard M (2003). An algebraic approach to network coding. *IEEE/ACM Trans. on Networking* 11 (5): 782–795.

Lagrange X (2010). Throughput of HARQ protocols on a block fading channel. *IEEE Communications Letters* 14 (3): 257–259. doi:10.1109/LCOMM.2010.03.092310.

Lavaei J and Murray RM (2011). Quantized consensus by means of gossip algorithm. *IEEE Trans. on Automatic Control* 57 (1): 19–32.

Lynch NA (1996). Distributed Algorithms, CA: Morgan Kaufmann Publishers, San Mateo.

Makridis E, Charalambous T and Hadjicostis CN (2023a), Harnessing HARQ retransmissions for fast average consensus over unreliable communication channels, Proc. of 62nd IEEE Conference on Decision and Control (CDC), 5437–5443.

Makridis E, Charalambous T and Hadjicostis CN (2023b), Utilizing feedback channel mechanisms for reaching average consensus over directed network topologies, Proc. of American Control Conference (ACC), 1781–1787.

Nedic A, Olshevsky A, Ozdaglar A and Tsitsiklis J (2009). On distributed averaging algorithms and quantization effects. *IEEE Trans. on Automatic*

Control 54 (11): 2506–2517.
Nedić A, Olshevsky A and Shi W (2017). Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM Journal on Optimization* 27 (4): 2597–2633.
Notarstefano G, Notarnicola I and Camisa A (2019). Distributed optimization for smart cyber-physical networks. *Foundations and Trends® in Systems and Control* 7 (3): 253–383.
Nowzari C, Garcia E and Cortés J (2019). Event-triggered communication and control of networked systems for multi-agent consensus. *Automatica* 105: 1–27.
Olfati-Saber R and Murray R (2004). Consensus problems in networks of agents with switching topology and time-delays. *IEEE Trans. on Automatic Control* 49 (9): 1520–1533.
Olshevsky A and Tsitsiklis JN (2009). Convergence speed in distributed consensus and averaging. *SIAM Journal on Control and Optimization* 48 (1): 33–55.
Rikos AI and Hadjicostis CN (2020a). Event-triggered quantized average consensus via mass summation. *arXiv preprint arXiv:2003.14183* .
Rikos AI and Hadjicostis CN (2020b). Event-triggered quantized average consensus via ratios of accumulated values. *IEEE Trans. on Automatic Control* 66 (3): 1293–1300.
Rikos A, Charalambous T and Hadjicostis CN (2014). Distributed weight balancing over digraphs. *IEEE Trans. on Control of Network Systems* 1 (2): 190–201.
Rikos AI, Jiang W, Charalambous T and Johansson KH (2023), Asynchronous distributed optimization via ADMM with efficient communication, Proc. of 62nd IEEE Conference on Decision and Control (CDC), 7002–7008.
Scaman K, Bach F, Bubeck S, Lee YT and Massoulié L (2019). Optimal convergence rates for convex distributed optimization in networks. *Journal of Machine Learning Research* 20 (159): 1–31. `http://jmlr.org/papers/v20/19-543.html`.
Schneider MH and Zenios SA (1990). A comparative study of algorithms for matrix balancing. *Operations Research* 38 (3): 439–455.
Seneta E (2006). Non-Negative Matrices and Markov Chains, revised printing from 1973 ed., Springer, New York, NY.
Seyboth GS, Dimarogonas DV and Johansson KH (2013). Event-based broadcasting for multi-agent average consensus. *Automatica* 49 (1): 245–252.
Shi W, Ling Q, Yuan K, Wu G and Yin W (2014). On the linear convergence of the ADMM in decentralized consensus optimization. *IEEE Trans. on Signal Processing* 62 (7): 1750–1761.
Spiridonoff A, Olshevsky A and Paschalidis IC (2020). Robust asynchronous stochastic gradient-push: Asymptotically optimal and network-independent performance for strongly convex functions. *Journal of Machine Learning Research* 21 (58): 1–47. `http://jmlr.org/papers/v21/18-813.html`.
Sundaram S and Hadjicostis CN (2007), Finite-time distributed consensus in graphs with time-invariant topologies, Proc. of American Control Conference (ACC), 711–716.
Tsitsiklis J (1984). Problems in Decentralized Decision Making and Computation. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA.
Verbraeken J, Wolting M, Katzy J, Kloppenburg J, Verbelen T and Rellermeyer JS (2020). A survey on distributed machine learning. *ACM Computing Surveys* 53 (2): 1–33.
Wang L and Xiao F (2010). Finite-time consensus problems for networks of dynamic agents. *IEEE Trans. on Automatic Control* 55 (4): 950–955.
Xi C and Khan UA (2017). DEXTRA: A fast algorithm for optimization over directed graphs. *IEEE Trans. on Automatic Control* 62 (10): 4980–4993.
Xi C, Xin R and Khan UA (2018). ADD-OPT: Accelerated distributed directed optimization. *IEEE Trans. on Automatic Control* 63 (5): 1329–1339.
Xiao L and Boyd S (2004). Fast linear iterations for distributed averaging. *Systems and Control Letters* 53 (1): 65–78.
Xiao L, Boyd S and Lall S (2005), A scheme for robust distributed sensor fusion based on average consensus, Proc. of Fourth International Symposium on Information Processing in Sensor Networks (IPSN), 63–70.
Xu H, Utomi S, Domínguez-García AD and Sauer P (2017), Coordination of distributed energy resources in lossy networks for providing frequency regulation, Proc. of X Bulk Power Systems Dynamics and Control Symposium.
Yuan Y, Stan GB, Shi L, Barahona M and Goncalves J (2013). Decentralised minimum-time consensus. *Automatica* 49 (5): 1227–1235.