A Linear Push-Pull Average Consensus Algorithm for Delay-Prone Networks

Evagoras Makridis and Themistoklis Charalambous

***Abstract*— In this paper, we address the average consensus problem of multi-agent systems for possibly unbalanced and delay-prone networks with directional information flow. We propose a linear distributed algorithm (referred to as RPPAC) that handles asynchronous updates and time-varying heterogeneous information delays. Our proposed distributed algorithm utilizes a surplus-consensus mechanism and information regarding the number of incoming and outgoing links to guarantee state averaging, despite the imbalanced and delayed information flow in directional networks. The convergence of the RPPAC algorithm is examined using key properties of the backward product of time-varying matrices that correspond to different snapshots of the directional augmented network.**

***Index Terms*— distributed algorithms, push-pull consensus, average consensus, directed graphs, time-varying heterogeneous delays.**

I. Introduction

Distributed consensus algorithms have recently gained significant prominence due to their widespread applicability in various domains, including wireless sensor networks, multi-agent systems, and smart power grids. The main objective of these algorithms is to enforce a network of interconnected agents (or nodes) to collectively converge towards a common value, known as the *consensus*, by means of local information exchange (see [1] for an overview of consensus methods). Among the diverse consensus problems, *average consensus* is distinguished as a fundamental challenge, wherein the agents aim to cooperatively compute the average of their initial values held by each individual node. This problem has gained significant attention for its relevance in tasks such as distributed formation control [2]–[4], distributed estimation and filtering [5]–[8], and distributed optimization [9]–[13].

Substantial research on distributed average consensus has been conducted in the context of bidirectional networks where the information between agents flow in both directions forming an undirected graph [14]–[16]. However, in most real-world multi-agent systems, agents communicate over wireless channels in which they have different transmitting power capabilities and experience different levels of interference. As a consequence the communication links become inherently directional forming directed graphs (*digraphs*). Although directional networks provide a more realistic characterization of the underlying communication network, further complexities and challenges arise when agents aim to achieve average consensus in a distributed manner. For instance, the imbalance in the flow of information, due to the directed links, may lead to slower convergence or even incapability in reaching average consensus.

To overcome this issue, the works in [17]–[19] have provided different approaches where agents update their state variables through a linear combination of the incoming (received and owned) information. A necessary condition towards reaching average consensus is to prevent information from becoming trapped at a specific agent and to guarantee that the information flows throughout the network, thus, the underlying digraph should be strongly connected[1]. The *Ratio Consensus* (RC) algorithm proposed in [20], proved to be able to reach average consensus by computing in an iterative way the ratio of two concurrently running linear iterations: one to compute a weighted average of the network's initial values, and one to track the information flow imbalance. Nevertheless, such methods that require agents to send their state variables by weighting them according to the number of their outgoing links (forming a column-stochastic (CS) matrix of weights), require additional computation per iteration for each agent, that is the nonlinear computation of the ratio of the two concurrently running iterations. Another approach towards mitigating the information flow imbalance of digraphs for reaching average consensus has been proposed in [19]. In particular, the authors proposed a surplus-based approach where the information flow imbalance is handled by augmenting the state of each agent with an additional variable (often called *surplus*) that locally tracks individual state updates through the assignment of weights on both the incoming and outgoing links that form both row-stochastic (RS) and CS matrices, respectively.

Taking one step further towards more realistic scenarios, one should take into consideration that the information flows over delay-prone communication links due to network congestion and packet retransmissions requested as a result of decoding errors detected at the receiving agent [21]–[25]. This highlights the necessity of ensuring resilience to delays towards reaching average consensus. Under the conditions on delay-prone directed information exchange between agents, the authors in [22] proposed a robustified version of the ratio consensus algorithm (hereinafter referred to as RRC) which is able to reach asymptotic average consensus in the presence of bounded time-varying delays. However, the nonlinear nature of RRC makes it difficult to find tight bounds on the convergence rate of the algorithm itself [22], but also to

The authors are with the Department of Electrical and Computer Engineering, School of Engineering, University of Cyprus, 1678 Nicosia, Cyprus. E-mails: surname.name@ucy.ac.cy. T. Charalambous is also a Visiting Professor at the Department of Electrical Engineering and Automation, School of Electrical Engineering, Aalto University, 02150 Espoo, Finland.

This work was partly supported by the European Research Council (ERC) Consolidator Grant MINERVA (Grant agreement No. 101044629).

[1] A digraph is called strongly connected if there exists a directed path between any pair of nodes in the network. This ensures that the information propagates and reaches all agents in the network.

various consensus-based distributed optimization algorithms that use RC as its consensus protocols [26]–[30].

This paper aims at developing a linear discrete-time distributed average consensus algorithm which operates over directed networks, and can handle heterogeneous and time-varying information delays. Specifically, we propose a linear surplus-based distributed protocol that enforces nodes in the network to converge to the average of their initial values, although the communication links are directional and prone to time-varying delays. To the best of our knowledge, this is the first linear (push-pull) asynchronous average consensus algorithm that reaches the exact average of the agents' initial values over delay-prone directional links in a network. The characteristics of our proposed algorithm in comparison with main average consensus algorithms that operate in digraphs, are emphasized in Table I.

Algorithm	Linear	RS	CS	Delays
[20]	✗	✗	✓	✗
[22]	✗	✗	✓	✓
[19]	✓	✓	✓	✗
RPPAC	✓	✓	✓	✓

TABLE I: Comparison of the characteristics of the main average consensus algorithms in digraphs.

II. Preliminaries

A. Network Model

Consider a group of $n > 1$ agents communicating over an unreliable time-invariant and directed network. The interconnection topology of the communication network is modeled by a digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each agent v_j is included in the set of digraph nodes $\mathcal{V} = \{v_1, \cdots, v_n\}$. The interactions between agents are included in the set of digraph edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. The total number of edges in the network is denoted by $m = |\mathcal{E}|$. A directed edge $\varepsilon_{ji} \triangleq (v_j, v_i) \in \mathcal{E}$ indicates that node v_j receives information from node v_i, *i.e.,* $v_i \to v_j$. The nodes that transmit information to node v_j directly are called in-neighbors of node v_j, and belong to the set $\mathcal{N}_j^{\text{in}} = \{v_i \in \mathcal{V} | \varepsilon_{ji} \in \mathcal{E}\}$. The number of nodes in the in-neighborhood set is called in-degree and is denoted by $d_j^{\text{in}} = |\mathcal{N}_j^{\text{in}}|$. The nodes that receive information from node v_j directly are called out-neighbors of node v_j, and belong to the set $\mathcal{N}_j^{\text{out}} = \{v_l \in \mathcal{V} | \varepsilon_{lj} \in \mathcal{E}\}$. The number of nodes in the out-neighborhood set is called out-degree and is denoted by $d_j^{\text{out}} = |\mathcal{N}_j^{\text{out}}|$. Each node $v_j \in \mathcal{V}$ has immediate access to its own local state, and thus we assume that the corresponding self-loop is available $\varepsilon_{jj} \in \mathcal{E}$, although it is not included in the nodes' out-neighborhood and in-neighborhood. In $\mathcal{G}$ a node v_i is reachable from a node v_j if there exists a path from v_j to v_i which respects the direction of the edges. The digraph $\mathcal{G}$ is said to be strongly connected if every node is reachable from every other node.

B. Problem Setup

At each time instant $k \geq 0$ each node $v_j \in \mathcal{V}$ maintains a scalar state $x_j(k) \in \mathbb{R}$. For analysis purposes, we define the aggregate state of all nodes by $x(k) = (x_1(k), \ldots, x_n(k))^\top \in \mathbb{R}^n$. The goal of the agents is to collaboratively solve the following discrete-time average consensus (DTAC) problem:

$$\text{Problem 1:} \quad \bar{x} := \frac{1}{n} \sum_{i=1}^{n} x_i(0) \tag{1}$$

where $\bar{x}$ denotes the network-wide average of all agents' initial values, $x_i(0)$. Clearly, in the absence of global knowledge, individual agents are required to execute an iterative distributed algorithm to eventually converge to the initial network-wide average, by updating their states using information received from their neighboring nodes.

C. Average Consensus using Push-Pull Weights (PPAC)

A linear algorithm for reaching average consensus over directed and strongly connected networks, has been proposed in [19]. The main idea behind their proposed algorithm is to maintain a time-invariant state sum $\mathbf{1}^T x$, such that the agents do not lose the track of the initial average $x(0)$, which is the main difficulty when the network is asymmetric (modeled via digraph). To achieve this, at each iteration k, each agent $v_j \in \mathcal{V}$ maintains a local state variable $x_j(k) \in \mathbb{R}$, and an auxiliary variable $s_j(k) \in \mathbb{R}$ (called *surplus*). The surplus variable locally records the state changes of individual nodes such that $\mathbf{1}^\top (x(k) + s(k)) = \mathbf{1}^\top x(0)$ for all time k, where $s(k) = (s_1(k), \ldots, s_n(k))^\top \in \mathbb{R}^n$. Then, each agent v_j iteratively updates its variables at each time step k as

$$x_j(k+1) = \gamma s_j(k) + \sum_{v_i \in \mathcal{N}_j^{\text{in}} \cup \{j\}} r_{ji} x_i(k), \tag{2a}$$

$$s_j(k+1) = x_j(k) - x_j(k+1) + \sum_{v_i \in \mathcal{N}_j^{\text{in}} \cup \{j\}} c_{ji} s_i(k), \tag{2b}$$

initialized at arbitrary $x_j(0) \in \mathbb{R}$, and $s_j(0) = 0$. Each agent v_j assigns the weights for the incoming information based on its in-degree as:

$$r_{ji} = \begin{cases} \dfrac{1}{1 + d_j^{\text{in}}}, & \text{if } v_i \in \mathcal{N}_j^{\text{in}} \text{ or } j = i, \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

where the resulting weights $r_{ji} \geq 0$ (often called *"pull" weights*) are the (j,i)-th entries of the row-stochastic matrix $R = \{r_{ji}\} \in \mathbb{R}_+^{n \times n}$. The assignment of "pull" weights is straightforward since each agent can easily obtain its in-degree by counting the incoming streams of information. Moreover, each agent v_j assigns the weights for the outgoing information based on its out-degree as:

$$c_{lj} = \begin{cases} \dfrac{1}{1 + d_j^{\text{out}}}, & \text{if } v_l \in \mathcal{N}_j^{\text{out}} \text{ or } l = j, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

where the resulting weights $c_{lj} \geq 0$ (often called *"push" weights*) are the (l,j)-th entries of the column-stochastic matrix $C = \{c_{lj}\} \in \mathbb{R}_+^{n\times n}$. Note that, agents are required to have the knowledge of their out-degree to assign the weights c_{lj}, hence it is required to have either an estimate of the out-degree [31], [32] or to compute the out-going streams of information by utilizing 1-bit feedback links [24].

The parameter $\gamma > 0$ (often referred to as *surplus gain*) denotes the gain by which the surplus is amplified, such that each agent regulates its convergence speed to the average consensus value. The selection of parameter γ requires global knowledge of the network size.

Remark 1. *The weight matrices R and C that are formed by assigning the weights as in (3) and (4), preserve row- and column-stochasticity, respectively. This implies that $R\mathbf{1} = \mathbf{1}$ and $\mathbf{1}^\top C = \mathbf{1}^\top$. Moreover, the fact that digraph $\mathcal{G}$ includes self-loops (each agent has access to its own variables), implies that $r_{ii} > 0$ and $c_{ii} > 0$, $\forall v_i \in \mathcal{V}$.*

III. Algorithm Development

In this section we design a linear distributed strategy for each node $v_j \in \mathcal{V}$ in a directed network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to handle information that is received over delay-prone directional links, such that all the nodes converge to the average consensus value in (1). More specifically we assume that, the transmission on the link ε_{ji} at time step k experiences a delay, $\tau_{ji}(k)$, *i.e.,* the (discrete) time interval between the transmitting and receiving time steps. The delays on the transmission links are heterogeneous, time-varying, and bounded, *i.e.,* $0 \leq \tau_{ji}(k) \leq \bar{\tau}_{ji} \leq \bar{\tau} < \infty$, where $\bar{\tau}_{ji}$ denotes the maximum delay over the link ε_{ji}, and $\bar{\tau}$ the maximum delay of the network over all the links *i.e.,* $\bar{\tau} = \max_{\varepsilon_{ji}\in\mathcal{E}}\{\bar{\tau}_{ji}\}$. Note that, each node v_j has immediate access to its own local state x_j, and hence $\tau_{jj}(k) = 0$ for all $v_j \in \mathcal{V}$ and all $k \geq 0$.

A. Robustified Push-Pull Average Consensus (RPPAC)

Inspired by the linear distributed algorithm in [19] for reaching average consensus over reliable directed graphs, we introduce a robustified alternative by which nodes can reach exact average consensus by handling heterogeneous time-varying delays. Although the number of nodes in the network and the interconnecting links are considered fixed, the presence of time-varying and heterogeneous delays affect the way that each node should assign the consensus weights such that the sum of the nodes' values (*i.e., mass*) of the network is preserved at each time step. Hence, to preserve the mass of the network fixed, we devise a new linear algorithm (hereinafter called RPPAC) based on the surplus consensus in [19], where each node updates its information state (at each iteration) via a linear combination of the (possibly delayed) information state received from its neighbors at that iteration. This algorithm converges to the exact average of the nodes' initial values, despite the presence of arbitrary, yet bounded time-delays.

In particular, at each iteration k, each agent $v_j \in \mathcal{V}$ maintains a local state variable $x_j(k) \in \mathbb{R}$, and an auxiliary surplus variable $s_j(k) \in \mathbb{R}$, that is used to preserve the total mass in the network constant at each time step (*i.e.,* $\mathbf{1}^\top(\mathbf{x}(k)+\mathbf{s}(k)) = \mathbf{1}^\top\mathbf{x}(0),\ \forall k \geq 0$). First, each node v_j sets $x_j(0) = V_j$, $s_j(0) = 0$, and $0 < \gamma < 1$. Prior the iterative phase of the RPPAC algorithm, it broadcasts dummy packets and receives an acknowledgment feedback signal from each in-neighbor to acquire its in-degree and out-degree (see [24] for more details on the acquisition of out-degree). Following, at each iteration k, each node v_j performs the following steps:

Broadcasting: It broadcasts its own (unweighted) state variable, $x_j(k)$, and a weighted version of its surplus variable, $c_{lj}s_j(k)$, to its out-neighbors $v_l \in \mathcal{N}_j^{\text{out}}$, over possibly delay-prone links $\varepsilon_{lj} \in \mathcal{E}$. The "push" consensus weights c_{lj} for $l = 1,\ldots,n$ can be assigned offline by each node v_i, given that the network is fixed, using the "push" weight assignment strategy in (4). This ensures that at each time step k the total mass of the surplus variable is fully (and equally) distributed to the out-neighbors of v_j, since $\sum_{v_l\in\mathcal{N}_j^{\text{out}}} c_{lj} = 1$. Notice that, node v_j transmits its local variables without considering that the information sent over its outgoing links ε_{lj} for any $v_l \in \mathcal{N}_j^{\text{out}}$ might experience a delay.

Receiving: It receives the weighted surplus variables $c_{ji}s_i(k-\delta)$ and the state variables $x_i(k-\delta)$ for all $0 \leq \delta \leq \bar{\tau}_{ji}$ from (possibly some of) its in-neighbors $v_i \in \mathcal{N}_j^{\text{in}}$ that arrived over possibly delay-prone links $\varepsilon_{ji} \in \mathcal{E}$. Upon the arrival of these variables, it scales each received state variable $x_i(k-\delta)$ by the "pull" consensus weights $r_{ji}(k)$ as $r_{ji}(k)x_i(k)$, where the weight $r_{ji}(k)$ is assigned by each node v_j depending on the possibly delayed packets arrived exactly at time step k. The assignment of "pull" weights is elaborated later in §IV.

Updating: Upon the reception of (possibly delayed) information from the in-neighbors of v_j, it updates its own state variable $x_j(k+1)$ and its auxiliary surplus variable $s_j(k+1)$, as follows:

$$x_j(k+1) = \gamma s_j(k) + \underbrace{\sum_{v_i\in\mathcal{N}_j^{\text{in}}\cup\{j\}} \sum_{\delta=0}^{\bar{\tau}_{ji}} r_{ji}(k)x_i(k-\delta)\ell_{ji}(k-\delta)}_{\text{possibly delayed incoming states}}, \tag{5a}$$

$$s_j(k+1) = g_j(k+1) + \underbrace{\sum_{v_i\in\mathcal{N}_j^{\text{in}}\cup\{j\}} \sum_{\delta=0}^{\bar{\tau}_{ji}} c_{ji}s_i(k-\delta)\ell_{ji}(k-\delta)}_{\text{possibly delayed incoming surplus}}, \tag{5b}$$

where $g_j(k+1) \triangleq x_j(k) - x_j(k+1)$ and $\ell_{ji}(k-\delta)$ captures the delay on link ε_{ji} at iteration k as:

$$\ell_{ji}(k-\delta) = \begin{cases} 1, & \text{if } \tau_{ji}(k-\delta) = \delta, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

The parameter $\gamma > 0$ is a sufficiently small number depending on the network size and topology, and the length of delays.

IV. Convergence Analysis

In this section, we analyse the convergence of RPPAC, by introducing its vector-matrix augmented form that corresponds to an augmented digraph that models (possibly) time-varying delayed information. To simplify the analysis, we consider that the maximum delay at each link is identical to the maximum delay in the network, *i.e.,* $\bar{\tau}_{ji} = \bar{\tau}$. To model all the possible delayed transmissions consider the following augmentation on the original graph $\mathcal{G}$. For each agent $v_j \in \mathcal{V}$, we add $\bar{\tau}$ extra virtual nodes that represent local buffers which propagate the delayed information to its destined agent after $0 < \delta \leq \bar{\tau}$ iterations. Hence the total number of nodes in the augmented digraph $\mathcal{G}^\alpha = \{\mathcal{V}^\alpha, \mathcal{E}^\alpha\}$ is $\tilde{n} = n(\bar{\tau}+1)$, where the actual agents are indexed by $1, \ldots, n$ and the virtual nodes by $n+1, \ldots, \tilde{n}$. Thus, the virtual nodes $n+1, \ldots, 2n$ model the information delayed by $\delta = 1$ time step, $2n+1, \ldots, 3n$ model the information delayed by $\delta = 2$ time steps, and so on.

Based on the augmented digraph model, we further define the augmented variables $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{s}}$ that hold the (possibly delayed) information in the (virtual buffer) nodes as

$$\tilde{\mathbf{x}}(k) = \left(x^\top(k),\ x^{(1)}(k), \ldots, x^{(\bar{\tau})}(k)\right)^\top, \tag{7}$$

$$\tilde{\mathbf{s}}(k) = \left(s^\top(k),\ s^{(1)}(k), \ldots, s^{(\bar{\tau})}(k)\right)^\top, \tag{8}$$

where $x^{(\delta)}(k) = \left(x_1^{(\delta)}(k), \ldots, x_n^{(\delta)}(k)\right)$, and $s^{(\delta)}(k) = \left(s_1^{(\delta)}(k), \ldots, s_n^{(\delta)}(k)\right)$. Then we can rewrite the update phase of RPPAC in its vector-matrix form as:

$$\tilde{\mathbf{x}}(k+1) = \tilde{R}(k)\tilde{\mathbf{x}}(k) + H\tilde{\mathbf{s}}(k), \tag{9a}$$

$$\tilde{\mathbf{s}}(k+1) = J(k)\tilde{\mathbf{x}}(k) + (\tilde{C}(k) - H)\tilde{\mathbf{s}}(k), \tag{9b}$$

where

$$\tilde{R}(k) \triangleq \begin{pmatrix} R^{(0)}(k) & R^{(1)}(k) & \cdots & R^{(\bar{\tau})}(k) \\ I & 0 & \cdots & 0 \\ 0 & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} \in \mathbb{R}_+^{\tilde{n}\times\tilde{n}},$$

$$H \triangleq \begin{pmatrix} \gamma I & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} \in \mathbb{R}_+^{\tilde{n}\times\tilde{n}},$$

$$J(k) \triangleq \begin{pmatrix} I - R^{(0)}(k) & -R^{(1)}(k) & \cdots & -R^{(\bar{\tau})}(k) \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} \in \mathbb{R}^{\tilde{n}\times\tilde{n}},$$

$$\tilde{C}(k) \triangleq \begin{pmatrix} C^{(0)}(k) & I & \cdots & 0 \\ C^{(1)}(k) & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & I \\ C^{(\bar{\tau})}(k) & 0 & \cdots & 0 \end{pmatrix} \in \mathbb{R}_+^{\tilde{n}\times\tilde{n}}. \tag{10}$$

The element at the j-th row and i-th column of $R^{(\delta)}(k) \in \mathbb{R}_+^{n\times n}$ and $C^{(\delta)}(k) \in \mathbb{R}_+^{n\times n}$, for $\delta = 0, 1, \ldots, \bar{\tau}$, are determined by:

$$r_{ji}^{(\delta)}(k) = \begin{cases} \frac{1}{1+|\mathcal{N}_j^{\alpha,\text{in}}(k)|}, & \text{if } \tau_{ji}(k-\delta) = \delta,\ \varepsilon_{ji} \in \mathcal{E}, \\ 0, & \text{otherwise,} \end{cases} \tag{11}$$

where $|\mathcal{N}_j^{\alpha,\text{in}}(k)|$ is the virtual in-degree of the actual node v_j which denotes the number of (possibly delayed) incoming streams of information that arrived at node v_j exactly at time step k; and

$$c_{ji}^{(\delta)}(k) = \begin{cases} \frac{1}{1+|\mathcal{N}_i^{\text{out}}|}, & \text{if } \tau_{ji}(k-\delta) = \delta,\ \varepsilon_{ji} \in \mathcal{E}, \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

By appending the vectors $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{s}}$ of (9) in a new augmented state, we get the following matrix form representation:

$$\begin{bmatrix} \tilde{\mathbf{x}}(k+1) \\ \tilde{\mathbf{s}}(k+1) \end{bmatrix} = M(k) \begin{bmatrix} \tilde{\mathbf{x}}(k) \\ \tilde{\mathbf{s}}(k) \end{bmatrix}. \tag{13}$$

where

$$M(k) \triangleq \begin{bmatrix} \tilde{R}(k) & H \\ J(k) & \tilde{C}(k) - H \end{bmatrix} \tag{14}$$

The following theorem states the conditions on which the algorithm in (13) achieves average consensus.

Theorem 1. *The algorithm in (13) achieves asymptotic average consensus with the parameter $\gamma > 0$ sufficiently small, if and only if the digraph $\mathcal{G}$ is strongly connected, and the transmission delays are bounded, $\tau_{ji}(k) \leq \bar{\tau}_{ji} \leq \bar{\tau} < \infty$ for all $j, i \in \mathcal{V}$.*

Proof. A sketch of the proof is provided in the Appendix. □

V. Simulation Results

Consider a delay-prone directed network, $\mathcal{G}$, shown in Fig. 1 comprised of ten agents ($n = 10$), with each agent v_j executing the RPPAC algorithm as described in III. In this example, the agents' initial values are set at their unique identification index, (*i.e.,* $x_j(0) = j$ for $j = 1, \ldots, n$), while the auxiliary variables are initialized at $s_j(0) = 0$ for all $v_j \in \mathcal{V}$. Based on this configuration, the average of the agents' initial values is $\bar{x} = 5.5$. At each time step k the packets transmitted over the network's communication links are possibly and uniformly experiencing a time delay $\tau_{ji} \leq \bar{\tau}$ for all $\varepsilon_{ji} \in \mathcal{E}$.

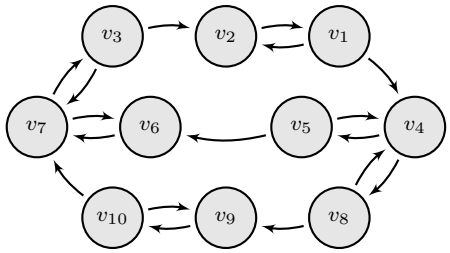

Fig. 1: Delay-prone digraph $\mathcal{G}$ comprised of 10 agents. Self-loops are allowed but not shown for ease of presentation.

As a first step towards validating our theoretical results, we run the RPPAC algorithm over different directional networks with heterogeneous time-varying delays bounded by $\bar{\tau} = \{0, 2, 5\}$, with a fixed surplus gain $\gamma = 0.1$. The state variables x_i at each agent v_i converge to the average consensus value $\bar{x} = 5.5$, as shown in Fig. 2, while the surplus variables s_j, are driven to 0, as shown in Fig. 3. It is worth mentioning that, the agents successfully converge to the average consensus value, although the surplus gain γ is not carefully chosen based on the network topology and size, as well as the length of delays, but it is rather chosen to be relatively small and same for all considered $\bar{\tau}$.

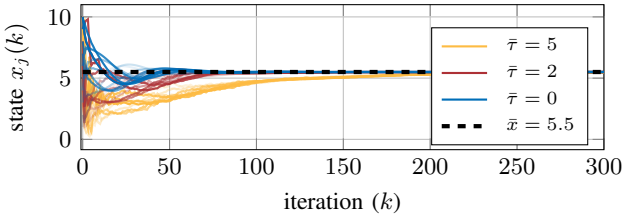

Fig. 2: State variable $x_j(k)$ at each agent.

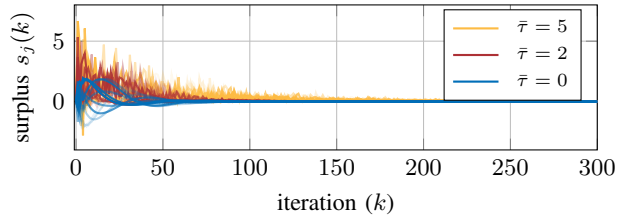

Fig. 3: Surplus variable $s_j(k)$ at each agent.

In Fig. 4, we present the mean consensus error $\frac{1}{n}\mathbf{e}^\top(k)\mathbf{e}(k)$ where $\mathbf{e}(k) \triangleq \mathbf{x}(k) - \mathbf{1}\bar{x}$, achieved by executing the distributed RPPAC algorithm, after averaging over 100 Monte Carlo simulations for three different upper bounds on the delays, *i.e.,* $\bar{\tau} = \{0, 2, 5\}$ and $\gamma = 0.1$. The mean square consensus error for the delay-free network is driven to 0 faster than the delay-prone networks, with the same surplus gain parameter γ.

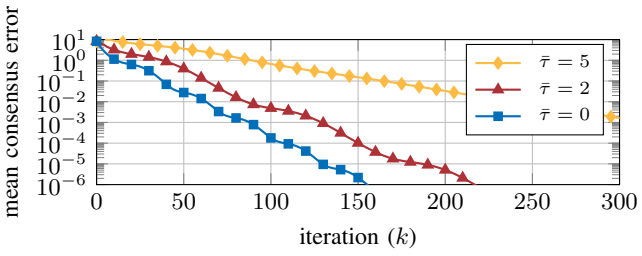

Fig. 4: Mean square consensus error for different length of delays.

In Fig. 5 we present the spectral gap[2] of $M(k)$ for different upper bounds on the delays, *i.e.,* $\bar{\tau} = \{0, 2, 5\}$. The higher the spectral gap, the faster the convergence of the RPPAC algorithm, as the second largest absolute eigenvalue of $M(k)$, λ_2, moves away from the spectral radius $\lambda_1 = \rho(M(k))$. As depicted in Fig. 5, when the length of delays on the links of the network is longer, the spectral gap of the corresponding matrix $M(k)$ becomes smaller, leading to slower convergence. This behavior is shown in Fig. 6 for different upper bounds on the delays, *i.e.,* $\bar{\tau} = \{0, 1, \ldots, 10\}$. However, for a given matrix $M(k)$ that corresponds to a particular snapshot of the interactions in the network, one can choose a surplus gain γ that guarantees the fastest convergence to the average consensus value.

[2]Spectral gap is the difference between the moduli of the two largest eigenvalues of a matrix, *i.e.,* $|\lambda_1| - |\lambda_2|$ where λ_i is the i-th eigenvalue of a matrix $A \in \mathbb{R}^{n\times n}$, with $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$.

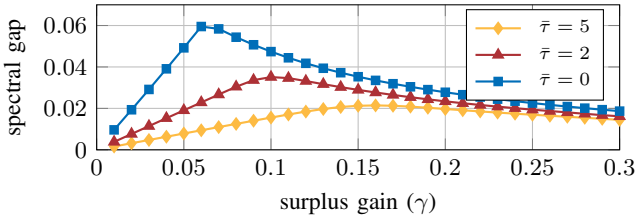

Fig. 5: Mean spectral gap of $M(k)$ that corresponds to different for different upper bounds on the delays $\bar{\tau} = \{0, 2, 5\}$.

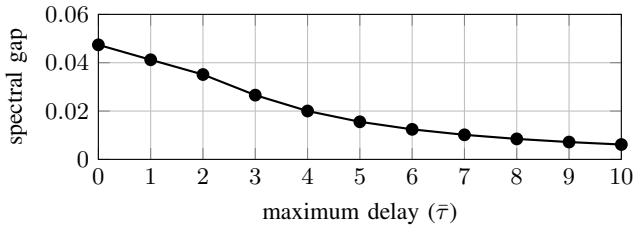

Fig. 6: Mean spectral gap of $M(k)$ with $\gamma = 0.1$.

Following the discussion on the selection of the surplus gain γ, we run 100 Monte Carlo simulations for the network $\mathcal{G}$ with an upper bound on the delays of $\bar{\tau} = 2$, and different surplus gains $\gamma = \{0.01, 0.1, 0.3\}$. The convergence rate in terms of the mean consensus error for this example is shown in Fig. 7. Notice that, the fastest convergence using the RPPAC algorithm for this particular configuration, is with $\gamma = 0.1$, for which the spectral gap is maximized, as previously shown in Fig. 5.

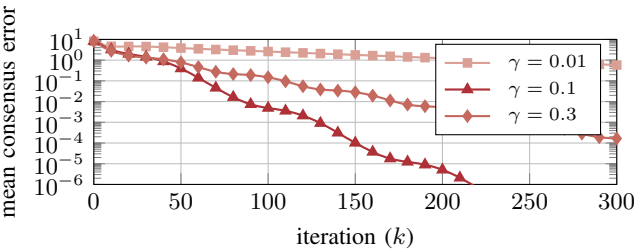

Fig. 7: Mean square consensus error with $\bar{\tau} = 2$ and $\gamma = \{0.01, 0.1, 0.3\}$.

VI. Conclusions and Future Directions

In this paper, we have tackled the discrete-time average consensus in multi-agent systems where the inter-agent com-

munication is directional, potentially unbalanced, and delay-prone. We have introduced a linear distributed algorithm designed to accommodate asynchronous updates and cope with time-varying heterogeneous delays. Our proposed algorithm leverages knowledge about the number of incoming and outgoing links for each agent, thereby ensuring that state averaging is achieved even in the presence of an asymmetric network structure and information flow delays.

RPPAC brings forward several interesting challenges. For example, what is the optimal choice of γ for guaranteeing the fastest convergence? Additionally, it would be interesting to investigate how nodes could choose their own γ, *i.e.,* γ_i, guaranteeing that convergence is reached. A promising direction for future research is to study the proposed asynchronous push-pull average consensus algorithm in the context of distributed optimization.

APPENDIX

Sketch of the Proof of Theorem 1:
We start by partitioning $M(k)$ into the summation of the terms $M_0(k)$ and M_1, *i.e.,*

$$M(k) = M_0(k)+M_1, \tag{15}$$

where

$$M_0(k) := \begin{bmatrix} \tilde{R}(k) & 0 \\ J(k) & \tilde{C}(k)-H \end{bmatrix}, \text{ and } M_1 := \begin{bmatrix} 0 & H \\ 0 & 0 \end{bmatrix}.$$

$M_0(k)$ is lower block triangular, its eigenvalues are the union of eigenvalues of its block diagonal elements, that is, the matrices $\tilde{R}(k)$ and $\tilde{C}(k)-H$, *i.e.,* its spectrum is

$$\sigma(M_0(k)) = \sigma\big(\tilde{R}(k)\big) \cup \sigma\big(\tilde{C}(k)-H\big),$$

where $\sigma(\cdot)$ denotes the spectrum, and $\tilde{R}(k)$ and $\tilde{C}(k)$ are the row- and column-stochastic matrices defined in (10), with spectral radius $\rho(\tilde{R}(k)) = \rho(\tilde{C}(k)) = 1$.

Since $\mathcal{G}$ is strongly connected, the corresponding augmented graph $\mathcal{G}^\alpha$ that models the delayed information is jointly strongly connected after $\bar{\tau}+1$ steps [22]. Therefore, any $\beta-$length word $\bar{E}_{(\beta)} = (\tilde{C}(k+\beta)-H)(\tilde{C}(k+\beta-1)-H)\dots(\tilde{C}(k+1)-H)$ and $\bar{R}_{(\beta)} = \tilde{R}(k+\beta)\tilde{R}(k+\beta-1)\dots\tilde{R}(k+1)$, for any integer $\beta \geq \bar{\tau}+1$, gives graphs that are strongly connected.

We start by defining the augmented state by concatenating the state and surplus variables of all the augmented nodes as $\chi(\cdot) = \big[\tilde{\mathbf{x}}^\top(\cdot),\ \tilde{\mathbf{s}}^\top(\cdot)\big]^\top$. First, consider the state and surplus variables update using the RPPAC in (13) initialized with $\chi(k_0)$, at iteration k_2:

$$\chi(k_2) = M(k_1)\chi(k_1) = M(k_1)M(k_0)\chi(k_0).$$

We examine the backward product of matrices $M(k_1)M(k_0)$. From (15),

$$\begin{aligned} M(k_1)M(k_0) &= (M_0(k_1)+M_1)(M_0(k_0)+M_1) \\ &= M_0(k_1)M_0(k_0)+M_0(k_0)M_1+M_1M_0(k_0)+M_1^2 \\ &\overset{(a)}{=} M_0(k_1)M_0(k_0)+M_0(k_0)M_1+M_1M_0(k_0), \end{aligned} \tag{16}$$

where (a) stems from the fact that $M_1^2 = \mathbf{0}_{\tilde{n}\times\tilde{n}}$ (comes directly from the definition of M_1 in (15)). Next, consider the corresponding backward product of matrices $M(k_2)M(k_1)M(k_0)$, *i.e.,*

$$\begin{aligned} &M(k_2)M(k_1)M(k_0) \\ &\quad= (M_0(k_2)+M_1)\big[M_0(k_1)M_0(k_0) \\ &\qquad +M_0(k_0)M_1+M_1M_0(k_0)\big] \\ &\quad\overset{(b)}{=} M_0(k_2)M_0(k_1)M_0(k_0)+M_0(k_2)M_0(k_0)M_1+ \\ &\qquad M_0(k_2)M_1M_0(k_0)+M_1M_0(k_1)M_0(k_0)+ \\ &\qquad M_1M_0(k_0)M_1, \end{aligned} \tag{17}$$

where (b) stems again from the fact that $M_1^2 = \mathbf{0}_{\tilde{n}\times\tilde{n}}$. Continuing in the same way, we can see that a lot of terms that have M_1 on the left-side of the product will be cancelled out. Additionally, we can deduce the following properties that hold for any integer $\beta \geq \bar{\tau}+1$, for the $\beta-$length word $\bar{M}_{(\beta)} = M(k+\beta)M(k+\beta-1)\dots M(k+1)$.

- Since the product of lower triangular matrices results in a lower triangular matrix, then the product $\bar{M}_{0,(\beta)} \triangleq M_0(k+\beta)M_0(k+\beta-1)\dots M_0(k+1)$ results in a lower-triangular matrix of the form:

$$\bar{M}_{0,(\beta)} = \begin{bmatrix} \bar{R}_{(\beta)} & \mathbf{0} \\ * & \bar{E}_{(\beta)} \end{bmatrix},$$

  where $\rho(\bar{R}_{(\beta)}) = 1$ which is known from the product of row-stochastic matrices. Then, γ should be chosen such that $\rho(\bar{E}_{(\beta)}) < 1$ [33]. This can be secured for $0 < \gamma < \underline{c}$, where $\underline{c} \triangleq \min\{C\}$ is the minimum consensus weight of the original column-stochastic matrix C as formed by the weight assignment in (4). This comes from the fact that the diagonal elements of the top-left block of dimension n of $\tilde{C}(k)-H$ are strictly positive by enforcing $0 < \gamma < \underline{c}$. While this is a conservative bound guaranteeing that $\rho(\tilde{C}(k)-H) < 1\ \forall k$ it gives a simple bound for the values of γ.
- The fact that $M_1^2 = \mathbf{0}_{\tilde{n}\times\tilde{n}}$ makes several terms to be cancelled. The remaining products are such that they make the correction for the consensus to reach the average (as demonstrated in the simulations). The proof of this part is tedious and omitted due to space limitation. We will include it in the extended version of the paper.

Based on the aforementioned properties of the product of $\beta-$length $M(k)$ matrices, it is guaranteed that the state variables of the networked agents executing the RPPAC algorithm converge to the average consensus value and concurrently their surplus variables are driven to 0, since the network is jointly strongly connected after $\bar{\tau}+1$ steps.

Remark 2. *The convergence rate of the RPPAC algorithm is driven by the convergence of the product $\bar{E}_{(\beta)}$ to $\mathbf{0}_{\tilde{n}\times\tilde{n}}$. In other words, the algorithm achieves average consensus when the surplus variables at each agent $s_i(k)$ have been completely released to the corresponding state variable $x_i(k)$.*

References

[1] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and Cooperation in Networked Multi-agent Systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[2] W. Ren, "Multi-Vehicle Consensus with a Time-Varying Reference State," *Systems and Control Letters*, vol. 56, no. 7-8, pp. 474–483, 2007.

[3] J. A. Fax and R. M. Murray, "Information Flow and Cooperative Control of Vehicle Formations," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1465–1476, 2004.

[4] W. Yang, Z. Shi, and Y. Zhong, "Distributed Robust Adaptive Formation Control of Multi-Agent Systems with Heterogeneous Uncertainties and Directed Graphs," *Automatica*, vol. 157, p. 111 275, 2023.

[5] G. Wang, N. Li, and Y. Zhang, "Diffusion Distributed Kalman Filter over Sensor Networks without Exchanging Raw Measurements," *Signal Processing*, vol. 132, pp. 1–7, 2017.

[6] S. P. Talebi and S. Werner, "Distributed Kalman Filtering and Control Through Embedded Average Consensus Information Fusion," *IEEE Transactions on Automatic Control*, vol. 64, no. 10, pp. 4396–4403, 2019.

[7] W. Ren and U. M. Al-Saggaf, "Distributed Kalman–Bucy Filter with Embedded Dynamic Averaging Algorithm," *IEEE Systems Journal*, vol. 12, no. 2, pp. 1722–1730, 2017.

[8] B. Lian, Y. Wan, Y. Zhang, M. Liu, F. L. Lewis, and T. Chai, "Distributed Kalman Consensus Filter for Estimation with Moving Targets," *IEEE Transactions on Cybernetics*, vol. 52, no. 6, pp. 5242–5254, 2020.

[9] R. Xin and U. A. Khan, "A Linear Algorithm for Optimization over Directed Graphs with Geometric Convergence," *IEEE Control System Letters*, vol. 2, no. 3, pp. 315–320, 2018.

[10] S. Pu, W. Shi, J. Xu, and A. Nedić, "Push-Pull Gradient Methods for Distributed Optimization in Networks," *IEEE Transactions on Automatic Control*, vol. 66, no. 1, pp. 1–16, 2020.

[11] A. Nedić, D. T. A. Nguyen, and D. T. Nguyen, "AB/Push-Pull Method for Distributed Optimization in Time-Varying Directed Networks," *Optimization Methods and Software*, pp. 1–28, 2023.

[12] D. Wang, Z. Wang, J. Lian, and W. Wang, "Surplus-Based Accelerated Algorithms for Distributed Optimization over Directed Networks," *Automatica*, vol. 146, p. 110 569, 2022.

[13] V. Khatana, G. Saraswat, S. Patel, and M. V. Salapaka, "GradConsensus: Linearly Convergent Algorithm for Reducing Disagreement in Multi-Agent Optimization," *IEEE Transactions on Network Science and Engineering*, pp. 1–13, 2023.

[14] L. Xiao and S. Boyd, "Fast Linear Iterations for Distributed Averaging," *Systems and Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[15] A.-S. Esteki, H. Moradian, and S. S. Kia, "The Fastest Linearly Converging Discrete-Time Average Consensus using Buffered Information," *arXiv preprint arXiv:2206.09916*, 2022.

[16] E. Sebastián, E. Montijano, C. Sagüés, M. Franceschelli, and A. Gasparri, "Accelerated Multi-Stage Discrete Time Dynamic Average Consensus," *IEEE Control Systems Letters*, vol. 7, pp. 2731–2736, 2023.

[17] A. D. Dominguez-Garcia and C. N. Hadjicostis, "Coordination and Control of Distributed Energy Resources for Provision of Ancillary Services," in *IEEE International Conference on Smart Grid Communications*, 2010, pp. 537–542.

[18] 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, 2010.

[19] K. Cai and H. Ishii, "Average Consensus on General Strongly Connected Digraphs," *Automatica*, vol. 48, no. 11, pp. 2750–2761, 2012.

[20] A. D. Domínguez-García and C. N. Hadjicostis, "Distributed Strategies for Average Consensus in Directed Graphs," in *IEEE Conference on Decision and Control*, 2011, pp. 2124–2129.

[21] F. Xiao and L. Wang, "Consensus Protocols for Discrete-Time Multi-Agent Systems with Time-Varying Delays," *Automatica*, vol. 44, no. 10, pp. 2577–2582, 2008.

[22] C. N. Hadjicostis and T. Charalambous, "Average Consensus in the Presence of Delays in Directed Graph Topologies," *IEEE Transactions on Automatic Control*, vol. 59, no. 3, pp. 763–768, 2013.

[23] C. Zhao, J. He, P. Cheng, and J. Chen, "Performance Analysis of Discrete-Time Average Consensus under Uniform Constant Time Delays," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 11 725–11 730, 2017.

[24] E. Makridis, T. Charalambous, and C. N. Hadjicostis, "Utilizing Feedback Channel Mechanisms for Reaching Average Consensus over Directed Network Topologies," in *American Control Conference*, 2023, pp. 1781–1787.

[25] ——, "Harnessing HARQ Retransmissions for Fast Average Consensus Over Unreliable Communication Channels," in *IEEE Conference on Decision and Control*, 2023, pp. 5437–5443.

[26] Y. Lin and J. Liu, "Subgradient-Push is of the Optimal Convergence Rate," in *IEEE Conference on Decision and Control*, 2022, pp. 5849–5856.

[27] C. Xi and U. A. Khan, "DEXTRA: A Fast Algorithm for Optimization over Directed Graphs," *IEEE Transactions on Automatic Control*, vol. 62, no. 10, pp. 4980–4993, 2017.

[28] A. Nedic, A. Olshevsky, and W. Shi, "Achieving Geometric Convergence for Distributed Optimization over Time-varying Graphs," *SIAM Journal on Optimization*, vol. 27, no. 4, pp. 2597–2633, 2017.

[29] C. Xi, R. Xin, and U. A. Khan, "ADD-OPT: Accelerated Distributed Directed Optimization," *IEEE Transactions on Automatic Control*, vol. 63, no. 5, pp. 1329–1339, 2017.

[30] C. Xi, Q. Wu, and U. A. Khan, "On the Distributed Optimization over Directed Networks," *Neurocomputing*, vol. 267, pp. 508–515, 2017.

[31] C. N. Hadjicostis, N. H. Vaidya, and A. D. Domínguez-García, "Robust Distributed Average Consensus via Exchange of Running Sums," *IEEE Transactions on Automatic Control*, vol. 61, no. 6, pp. 1492–1507, 2015.

[32] T. Charalambous, M. G. Rabbat, M. Johansson, and C. N. Hadjicostis, "Distributed Finite-Time Computation of Digraph Parameters: Left-Eigenvector, Out-Degree and Spectrum," *IEEE Transactions on Control of Network Systems*, vol. 3, no. 2, pp. 137–148, 2015.

[33] A. A. Ahmadi and P. A. Parrilo, "Joint Spectral Radius of Rank One Matrices and the Maximum Cycle Mean Problem," in *IEEE Conference on Decision and Control*, 2012, pp. 731–733.