Fast Quantized Average Consensus over Static and Dynamic Directed Graphs*

Apostolos I. Rikos\textsuperscript{a}, Christoforos N. Hadjicostis\textsuperscript{b}, Karl H. Johansson\textsuperscript{a}

\textsuperscript{a}Division of Decision and Control Systems, KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden.
\textsuperscript{b}Department of Electrical and Computer Engineering, University of Cyprus, Nicosia, Cyprus.

Abstract

In this paper we study the distributed average consensus problem in multi-agent systems with directed communication links that are subject to quantized information flow. Specifically, we present and analyze a distributed averaging algorithm which operates exclusively with quantized values (i.e., the information stored, processed and exchanged between neighboring agents is subject to deterministic uniform quantization) and relies on event-driven updates (e.g., to reduce energy consumption, communication bandwidth, network congestion, and/or processor usage). The main idea of the proposed algorithm is that each node (i) models its initial state as two quantized fractions which have numerators equal to the node’s initial state and denominators equal to one, and (ii) transmits one fraction randomly while it keeps the other stored. Then, every time it receives one or more fractions, it averages their numerators with the numerator of the fraction it stored, and then transmits them to randomly selected out-neighbors. We characterize the properties of the proposed distributed algorithm and show that its execution, on any static and strongly connected digraph, allows each agent to reach in finite time a fixed state that is equal (within one quantisation level) to the average of the initial states. We extend the operation of the algorithm to achieve finite-time convergence in the presence of a dynamic directed communication topology subject to some connectivity conditions. Finally, we provide examples to illustrate the operation, performance, and potential advantages of the proposed algorithm. We compare against state-of-the-art quantized average consensus algorithms and show that our algorithm’s convergence speed significantly outperforms most existing protocols.

Key words: Quantized average consensus, distributed algorithms, event-triggered, quantization, digraphs, multi-agent systems.

1 Introduction

In recent years, there has been a growing interest for control and coordination of networks consisting of multiple agents, like groups of sensors [45] or mobile autonomous agents [29]. A problem of particular interest in distributed control is the consensus problem where the objective is to develop distributed algorithms that can be used by a group of agents in order to reach agreement to a common decision. The agents start with different initial states/information and are allowed to communicate locally via inter-agent information exchange under some constraints on connectivity. Consensus processes play an important role in many problems, such as leader election [24], motion coordination [5, 29], and clock synchronization [38].

One special case of the consensus problem is the distributed averaging problem, where each agent (initially endowed with a numerical state) can send/receive information to/from other agents in its neighborhood and update its state iteratively, so that eventually, all agents compute the average of the initial states. Average consensus is an important problem and has been studied extensively, primarily in settings where each agent processes and transmits real-valued states with infinite precision [5, 9, 15, 22, 43]. However, most existing average consensus algorithms are only able to guarantee asymptotic convergence, implying that they cannot be readily
applied to real-world distributed control and coordination applications. Furthermore, constraints on the bandwidth of communication links and the capacity of physical memories require both communication and computation to be performed assuming finite precision. For these reasons, researchers have studied the case where network links can only allow messages of limited length to be transmitted between agents, effectively extending techniques for average consensus towards the direction of quantized average consensus [1,6–8,12,18,20]. In addition, the demand for more efficient usage of network resources, has lead to an increasing interest for novel event-triggered algorithms for distributed control [23,28,39].

Distributed algorithms that achieve quantized average consensus in an event-driven fashion have a wide variety of applications. They can be used as the basis for various encoding schemes, such as quantized privacy protocols for guaranteeing additional levels of security without significantly increasing communication overhead [33,37]. Furthermore, in recent years there is a tremendous growth in distributed optimization [14,19,26,30] and machine learning algorithms [16,41]. The distributed operation of these algorithms over directed graphs requires exchange of the agents’ states without any error in order to guarantee convergence to a desired solution. However, as the network size becomes larger, e.g., to speed up the training of deep learning algorithms, the communication overhead of each iteration becomes a major bottleneck. Quantization and event-driven communication are effective approaches to tackle this issue, since reduction of communication and processing costs leads to bandwidth and energy efficient algorithms. The emerging importance of the aforementioned approaches can be further seen in various recent works [10,31,40,42] where researchers present machine learning algorithms, which employ quantization strategies to tackle the large communication overhead and reduce communication payload size, while maintaining fast convergence rates and possible privacy preserving guarantees.

1.1 Literature Review

In recent years, quite a few probabilistic distributed algorithms for averaging under quantized communication, have been proposed. Specifically, the probabilistic quantizer in [1] converges to a common state with a random quantization level for the case where the topology forms a directed graph. In [17] the authors present a distributed algorithm which adds a dither to the agents’ measurements (before the quantization process) and they show that the mean square error can be made arbitrarily small. In [4] the authors present a distributed algorithm which guarantees that all agents reach consensus to a value on the interval in which the average lies after a finite number of time steps. In [20] the authors present a quantized gossip algorithm which deals with the distributed averaging problem over a connected weighted graph, and calculates lower and upper bounds on the expected value of the convergence time, which depend on the principal submatrices of the Laplacian matrix of the weighted graph.

The available literature concerning deterministic distributed algorithms for averaging under quantized communication comprises less publications. In [21], the authors present a distributed averaging algorithm with dynamic encoding and decoding schemes. They show that for a connected undirected dynamic graph, average consensus is achieved asymptotically with as few as one bit of information exchange between each pair of adjacent agents at each time step, and the convergence rate is asymptotic and depends on the number of network nodes, the number of quantization levels and the synchronizability of the network. In [44] the authors present a novel quantization scheme for solving the average consensus problem when sensors exchange quantized state information. The proposed scheme is based on progressive reduction of the range of a uniform quantizer and leads to progressive refinement of the information exchanged by the sensors. In [7] the authors derive bounds on the rate of convergence to average consensus for a team of mobile agents exchanging information over time-invariant or randomly time-varying communication networks with symmetries. Furthermore, they study the control performance when agents also exchange logarithmically quantized data over static communication topologies with symmetries. In [27] the authors study distributed algorithms for the averaging problem over networks with dynamic topologies, with a focus on tight bounds on the convergence time of a general class of averaging algorithms. They consider algorithms for the case where agents can exchange and store continuous or quantized states, establish a tight convergence rate, and show that these algorithms guarantee convergence within some error from the average of the initial states; this error depends on the number of quantization levels.

Recent papers have studied the quantized average consensus problem with the additional constraint that the state of each node is an integer value. In [18] the authors present a probabilistic algorithm which allows every agent to reach quantized consensus almost surely over a static and undirected communication topology, while in [11] and [3] the authors analyze and further improve its convergence rate. In [6] a probabilistic algorithm was proposed to solve the quantized consensus problem for static directed graphs for the case where the agents exchange quantized information and store the changes of their states in an additional (also quantized) variable called “surplus”. The authors of [8] present a deterministic distributed averaging algorithm subject to quantization on the links and show that, depending on initial conditions, the system either converges in finite time to quantized consensus, or the nodes enter into a periodic behaviour with their states oscillating around the average. In [25], the authors present two distributed
algorithms, one for fixed tree graphs with finite time convergence and one for dynamic directed graphs with exponential convergence. The algorithms proposed in this work calculate the initial average as the ratio of two scaled sums obtained by running in parallel two iterations. In [34, 36] the authors present two distributed algorithms, one probabilistic and one deterministic, which calculate the exact quantized average of the initial states (i.e., there is no quantization error) in a finite number of time steps, which is explicitly calculated. In [35] the authors present a distributed randomized algorithm which calculates the quantized average of the initial states with high probability. The algorithm is shown to outperform other algorithms but the states of the nodes exhibit oscillating behavior (between the ceiling and the floor of the real-valued average of the initial states).

1.2 Main Contributions

In this paper, we present a novel distributed average consensus algorithm in which processing, storing, and exchange of information between neighboring agents is event-driven and subject to uniform quantization. The main idea behind the proposed algorithm is that initially each node stores two fractions. Each fraction has numerator equal to the node’s initial quantized state and denominator equal to one. Then, the node transmits one fraction randomly while it keeps the other stored. Every time it receives one or more fractions, it averages their numerators with the numerator of the fraction it keeps stored, and then transmits them to randomly selected out-neighbors. Note here that most work dealing with quantization has concentrated on the scenario where the agents have real-valued states but can only transmit quantized values through limited rate channels (e.g., [7, 8]). By contrast, our setup covers the case where the states are stored in digital memories of finite capacity (as in [6, 18, 27]). Specifically, we assume that states are integer-valued (which comprises a class of quantization effects such as uniform quantization) and the control actuation of each node is event-based, which enables more efficient use of available resources.

The main contribution of this paper is fourfold.
A. We introduce a novel distributed algorithm that allows all agents to reach quantized average consensus in finite time with high probability (Algorithm 1).
B. We show that, unlike existing algorithms in the literature, this algorithm allows every agent to calculate either the ceiling or the floor of the real average of the initial states with no oscillations in finite time (Theorem 1).
C. We present experimental results in which we compare the proposed algorithm against existing schemes and observe that its convergence speed significantly outperforms most finite-time distributed algorithms for average consensus under quantized communication.
D. We present an enhanced version of the algorithm (Algorithm 2) which achieves fast oscillation-free convergence in finite time in the presence of a dynamically changing communication topology (Theorem 2).

1.3 Outline

The remainder of this paper is organized as follows. In Section 2, we introduce the notation used throughout the paper, while in Section 3 we formulate the quantized average consensus problem. In Section 4, we present a probabilistic distributed algorithm which operates over static digraphs and allows the agents to reach consensus to the quantized average of the initial states, in finite time, with probability one. We demonstrate its performance with an illustrative example, and we analyze its operation and establish its finite time termination. In Section 5 we present an enhanced version of our algorithm which allows every agent to reach consensus to the quantized average of the initial states in the presence of a dynamic communication topology. In Section 6, we present simulation results and comparisons against the current state-of-the-art. We conclude in Section 7 with a brief summary and remarks about future work.

2 Preliminaries

2.1 Notation

The sets of real, rational, integer and natural numbers are denoted by \( \mathbb{R}, \mathbb{Q}, \mathbb{Z}, \) and \( \mathbb{N} \), respectively. The symbol \( \mathbb{Z}_+ \) denotes the set of nonnegative integers and the symbol \( \mathbb{N}_0 \) denotes the set of natural numbers that also includes zero. For any real number \( a \in \mathbb{R} \), the floor \( \lfloor a \rfloor \) denotes the greatest integer less than or equal to \( a \) while the ceiling \( \lceil a \rceil \) denotes the least integer greater than or equal to \( a \).

The communication topology is a network of \( n \ (n \geq 2) \) agents communicating only with their immediate neighbors, and can be captured by a directed graph (digraph), called communication digraph, which is either static (i.e., it does not change over time) or dynamic (i.e., it changes over time).

A static digraph is defined as \( \mathcal{G}_d = (\mathcal{V}, \mathcal{E}) \), where \( \mathcal{V} = \{v_1, v_2, \ldots, v_n\} \) is the set of nodes (representing the agents of the multi-agent system) and \( \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} - \{(v_j, v_j) \mid v_j \in \mathcal{V}\} \) is the set of edges (self-edges excluded). A directed edge from node \( v_i \) to node \( v_j \) is denoted by \( m_{ji} \triangleq (v_j, v_i) \in \mathcal{E} \), and captures the fact that node \( v_j \) can receive information from node \( v_i \) (but not the other way around). We adopt the common assumption that the given static digraph \( \mathcal{G}_d = (\mathcal{V}, \mathcal{E}) \) is strongly connected (i.e., 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 \( v_j \)). The subset of nodes that can directly transmit information to node \( v_j \) is called the set of in-neighbors of \( v_j \) and is...
represented by \( N_j^- = \{ v_i \in V \mid (v_j, v_i) \in E \} \), while the subset of nodes that can directly receive information from node \( v_j \) is called the set of out-neighbors of \( v_j \) and is represented by \( N_j^+ = \{ v_i \in V \mid (v_i, v_j) \in E \} \). The cardinality of \( N_j^- \) is called the in-degree of \( v_j \) and is denoted by \( D_j^- \) (i.e., \( D_j^- = |N_j^-| \)), while the cardinality of \( N_j^+ \) is called the out-degree of \( v_j \) and is denoted by \( D_j^+ \) (i.e., \( D_j^+ = |N_j^+| \)).

In dynamic digraphs we assume that the set of nodes is fixed while the set of edges among them might change at various points in time. Specifically, a dynamic digraph is defined as \( G_d[k] = (V, E[k]) \), where \( V = \{ v_1, v_2, \ldots, v_n \} \) is the set of nodes (representing the agents of the multi-agent system) and \( E[k] \subseteq V \times V - \{(v_j, v_j) \mid v_j \in V \} \) is the set of edges at time step \( k \) (self-edges excluded).

A directed edge from node \( v_i \) to node \( v_j \) is denoted by \( m_{ji} \equiv (v_i, v_j) \in E[k] \), and captures the fact that node \( v_j \) can receive information from node \( v_i \) (but not the other way around) at time step \( k \). This means that at each time instant \( k \), each node \( v_j \) has many different sets of in- and out-neighbors, denoted respectively by \( N_j^- \) and \( N_j^+ \) and defined as \( N_j^-[k] = \{ v_i \in V \mid (v_i, v_j) \in E[k] \} \) and \( N_j^+[k] = \{ v_i \in V \mid (v_j, v_i) \in E[k] \} \). Furthermore, the cardinality of \( N_j^- \) at time step \( k \), is called the in-degree of \( v_j \) and is denoted by \( D_j^-[k] = |N_j^-[k]| \), while the cardinality of \( N_j^+ \) at time step \( k \), is called the out-degree of \( v_j \) and is denoted by \( D_j^+[k] = |N_j^+[k]| \).

Given a collection of digraphs \( G_d[k] = (V, E[k]) \) for \( k = 1, 2, \ldots, m \), where \( m \in \mathbb{N} \), the union graph is defined as \( G_d[1:2:...:m] = (V, \cup_{k=1}^{m} E[k]) \). A collection of digraphs is said to be jointly strongly connected, if its corresponding union graph \( G_d[1:2:...:m] \) forms a strongly connected digraph.

2.2 Agent Operation

With respect to quantization of information flow each node \( v_j \in V \) maintains, at time step \( k \), 5 + \( 2D_j^+ \) variables, as follows:

(i) The mass variables \( y_j[k], z_j[k] \), where \( y_j[k] \in \mathbb{Z} \) and \( z_j[k] \in \mathbb{N}_0 \), are used for processing and calculating the average of the initial states.

(ii) The state variables \( y_j^0[k], z_j^0[k], q_j^0[k] \), where \( y_j^0[k] \in \mathbb{Z} \), \( z_j^0[k] \in \mathbb{N} \) and \( q_j^0[k] \in \mathbb{Z} \) (with \( q_j^0[k] = \frac{y_j^0[k]}{z_j^0[k]} \)) or \( q_j^0[k] = \left\lfloor \frac{y_j^0[k]}{z_j^0[k]} \right\rfloor \), are used for storing the values of the received mass variables and for calculating the state variable \( q_j \), which is the variable that becomes equal to the quantized average of the initial states.

(iii) The transmission variables \( c_j^0[k] \) and \( c_j^1[k] \) for each \( v_i \in N_j^+[k] \), where \( c_j^0[k] \in \mathbb{Z} \) and \( c_j^1[k] \in \mathbb{N}_0 \), are used for transmitting \( v_j \)'s mass variables towards its out-neighbors.

2.3 Transmission Strategy

In the static communication topology case, in order to randomly determine which out-neighbor to transmit to, each node \( v_j \) assigns a nonzero probability \( b_{ij} \) to each of its outgoing edges \( m_{ij} \) (including a virtual self-edge), where \( v_i \in N_j^+ \cup \{ v_j \} \). This probability assignment for all nodes can be captured by an \( n \times n \) column stochastic matrix \( B = [b_{ij}] \). A simple choice would be to set these probabilities to be equal, i.e.,

\[
 b_{ij} = \begin{cases} 
 \frac{1}{1 + D_j^-}, & \text{if } v_i \in N_j^+ \cup \{ v_j \}, \\
 0, & \text{otherwise.} 
\end{cases}
\]

Each nonzero entry \( b_{ij} \) of matrix \( B \) represents the probability of node \( v_j \) transmitting towards out-neighbor \( v_i \in N_j^+ \) through the edge \( m_{ij} \), or transmitting to itself (i.e., performing no transmission with probability \( b_{jj} \)). Note here that, for every node \( v_j \), the probability \( b_{jj} \) being positive implies that the matrix \( B \) becomes primitive (recall that the underlying static digraph is assumed to be strongly connected), which is of particular importance for the results presented in this paper.

In the dynamic communication topology case, we still have that each node randomly decides which out-neighbor to transmit to. Specifically, each node \( v_j \) assigns a nonzero probability \( b_{ij}[k] \) to each of its outgoing edges \( m_{ij}[k] \) (including a virtual self-edge) at each time step \( k \), where \( v_i \in N_j^+[k] \cup \{ v_j \} \). This probability assignment for all nodes can be captured, at each time step \( k \), by an \( n \times n \) column stochastic matrix \( B[k] = [b_{ij}[k]] \). Again, a simple choice would be to set these probabilities to be equal, i.e.,

\[
 b_{ij}[k] = \begin{cases} 
 \frac{1}{1 + D_j^-[k]}, & \text{if } v_i \in N_j^+[k] \cup \{ v_j \}, \\
 0, & \text{otherwise.} 
\end{cases}
\]

Each nonzero entry \( b_{ij}[k] \) of matrix \( B[k] \) represents the probability of node \( v_j \) transmitting towards out-neighbor \( v_i \in N_j^+[k] \) through the edge \( m_{ij}[k] \) at time step \( k \), or transmitting to itself (i.e., performing no transmission with probability \( b_{jj}[k] \)). Let us note here that the dynamic nature of the underlying communication topology implies that the matrix \( B[k] \) is not necessarily primitive at each time step \( k \) (whereas for a static strongly connected topology, the corresponding \( B \) will necessarily be primitive).

3 Problem Formulation

Consider a digraph \( G_d = (V, E) \), where each node \( v_j \in V \) has an initial quantized state \( y_j[0] \) (for simplicity, we
take \( y_j[0] \in \mathbb{Z} \) and \( q \) is the real average of the initial states:
\[
q = \frac{1}{n} \sum_{i=1}^{n} y_i[0].
\]
(1)

In this paper, we aim to develop distributed algorithms that address the following problems P1 – P2.

**P1.** When the digraph \( G_d \) is static and strongly connected, the algorithm allows the nodes to obtain, after a finite number of steps, a quantized state \( q^* \) which is equal to the ceiling or the floor of the actual average \( q \) of the initial states in (1). Specifically, we require that there exists \( k_0 \) so that for every \( v_j \in \mathcal{V} \) we have
\[
(q^*_j[k] = \lfloor q \rfloor \text{ for } k \geq k_0) \text{ or } (q^*_j[k] = \lceil q \rceil \text{ for } k \geq k_0). \tag{2}
\]

**P2.** When the digraph \( G_{d}[k] \) is dynamic and jointly strongly connected, we require that there exists \( k_0 \) so that (2) holds.

The quantized average \( q^* \) is defined as the ceiling \( \lfloor q \rfloor \) or the floor \( \lceil q \rceil \) of the true average \( q \) of the initial states in (1). Let \( S = \mathbb{1}^T y[0] \), where \( \mathbb{1} = [1 \ldots 1]^T \) is the vector of all ones, and let \( y[0] = [y_1[0] \ldots y_n[0]]^T \) be the vector of the quantized initial states. We can write \( S \) uniquely as
\[
S = nL + R \tag{3}
\]

where \( L \) and \( R \) are both integers and \( 0 \leq R < n \). Thus, we have that either \( L \) or \( L + 1 \) may be viewed as an integer approximation of the average of the initial states \( q = S/n \) (which may not be integer in general).

**Remark 1** Note here that our definition of quantized average consensus is different than in some literature [6, 8, 18, 35, 36]. We require that all agents states converge to a specific integer (either \( \lfloor q \rfloor \) or \( \lceil q \rceil \) where \( q \) satisfies (1)). Apart from [6], this cannot be achieved in existing finite-time algorithms since they either exhibit oscillating behavior of the agent states between the values \( \lfloor q \rfloor \) or \( \lceil q \rceil \) [8, 18, 35], or calculate the average in the form of a quantized fraction [36].

### 4 Quantized Averaging over Static Digraphs

In this section we propose a probabilistic distributed information exchange algorithm which addresses problem (P1) presented in Section 3. This is detailed as Algorithm 1 below (for the case where \( b_{ij} = 1/(1 + D_j) \) for \( v_i \in \mathcal{N}_j^+ \cup \{v_j\} \) and \( b_{ij} = 0 \) otherwise).

The intuition behind Algorithm 1 is the following. Initially, each node \( v_j \) doubles its mass variables (i.e., it sets \( y_j[0] := 2y_j[0] \) and \( z_j[0] = 2 \)). Note that this change has no effect on the average calculation (since both the sums of the initial \( y \)-values and \( z \)-values are doubled so that \( \sum_{j=1}^{n} y_j[0]/(\sum_{j=1}^{n} z_j[0]) \) remains unchanged and equal to \( q \) in (1)). Then, at each time step \( k \), each node \( v_j \) checks its mass variable \( z_j[k] \) and if \( z_j[k] > 1 \) (i) it updates its state variables to be equal to the mass variables and (ii) it splits \( y_j[k] \) into \( z_j[k] \) equal integer pieces.

**Algorithm 1** Quantized Average Consensus over Static Digraphs

**Input** A strongly connected digraph \( G_d = (\mathcal{V}, \mathcal{E}) \) with \( n = |\mathcal{V}| \) nodes and \( m = |\mathcal{E}| \) edges. Each node \( v_j \in \mathcal{V} \) has an initial state \( y_j[0] \in \mathbb{Z} \).

**Initialization:** Each node \( v_j \in \mathcal{V}_p \) does the following:

1) Assigns a nonzero probability \( b_{ij} \) to each of its outgoing edges \( m_{ij} \), where \( v_i \in \mathcal{N}_j^+ \cup \{v_j\} \), as follows
\[
b_{ij} = \begin{cases} 
\frac{1}{1 + D_j}, & \text{if } l = j \text{ or } v_i \in \mathcal{N}_j^+; \\
0, & \text{if } l \neq j \text{ and } v_i \notin \mathcal{N}_j^+.
\end{cases}
\]

2) Sets \( y_j[0] := 2y_j[0] \), \( z_j[0] = 2 \).

**Iteration:** For \( k = 0, 1, 2, \ldots \), each node \( v_j \in \mathcal{V}_p \), does the following:

1) if \( z_j[k] > 1 \), then

1.1) sets \( z_j[k] = z_j[k-1] \), \( y_j[k] = y_j[k-1] \), \( q_j[k] = \lfloor y_j[k] / z_j[k] \rfloor \);  

1.2) sets (i) \( mas^y[k] = y_j[k], \ mas^z[k] = z_j[k] \); (ii) \( c_{ij}^y[k] = 0, c_{ij}^z[k] = 0 \), for every \( v_i \in \mathcal{N}_j^+ \cup \{v_j\} \); (iii) \( \delta = |mas^y[k]/mas^z[k]|, \ mas^{rem}[k] = y_j[k] - \delta mas^z[k]; \)

1.3) while \( mas^z[k] > 1 \), then

1.3a) chooses \( v_i \in \mathcal{N}_j^+ \cup \{v_j\} \) randomly according to \( b_{ij} \);

1.3b) sets (i) \( c_{ij}^y[k] := c_{ij}^y[k] + 1, c_{ij}^z[k] := c_{ij}^z[k] + \delta \); (ii) \( mas^y[k] := mas^y[k] - 1, mas^z[k] := mas^z[k] - \delta; \)

1.3c) if \( mas^{rem}[k] > 0 \) then sets \( c_{ij}^y[k] := c_{ij}^y[k] + 1, mas^{rem}[k] := mas^{rem}[k] - 1 \).

1.4) sets \( c_{ij}^z[k] := c_{ij}^z[k] + mas^y[k], c_{ij}^y[k] := c_{ij}^y[k] + mas^z[k]; \)

1.5) if \( c_{ij}^z[k] > 0 \) then, transmits \( c_{ij}^y[k], c_{ij}^z[k] \) to out-neighbor \( v_i \), for every \( v_i \in \mathcal{N}_j^+ \).

• else if \( z_j[k] \leq 1 \), then sets \( c_{ij}^y[k] = y_j[k], c_{ij}^z[k] = z_j[k]; \)

2) receives \( c_{ji}^y[k], c_{ji}^z[k] \) from \( v_i \in \mathcal{N}_j^- \) and sets
\[
y_j[k+1] = c_{ij}^y[k] + \sum_{v_i \in \mathcal{N}_j^-} w_{ji}[k] c_{ji}^y[k], \tag{4}
\]
\[
z_j[k+1] = c_{ij}^z[k] + \sum_{v_i \in \mathcal{N}_j^-} w_{ji}[k] c_{ji}^z[k], \tag{5}
\]

where \( w_{ji}[k] = 1 \) if node \( v_j \) receives \( c_{ij}^y[k], c_{ij}^z[k] \) from \( v_i \in \mathcal{N}_j^- \) at iteration \( k \) (otherwise \( w_{ji}[k] = 0 \)).

**Output:** (2) holds for every \( v_j \in \mathcal{V} \).
We next provide an example to illustrate the operation of Algorithm 1.

**Example 1** Consider the strongly connected digraph $G_d = (V, E)$ in Fig. 1 (borrowed from [34]), with $V = \{v_1, v_2, v_3, v_4\}$ and $E = \{m_{21}, m_{31}, m_{42}, m_{13}, m_{23}, m_{34}\}$, where nodes have initial quantized states $y_1[0] = 5$, $y_2[0] = 3$, $y_3[0] = 7$, and $y_4[0] = 2$, respectively. The assignment $q$ of the initial states of the nodes, is equal to $q = 4.25$ which means that the quantized state $q^*$ is equal to $q^* = 4$ or $q^* = 5$ (see Section 3).

Each node $v_j \in V$ follows the initialization steps $1-2$ in Algorithm 1, assigning to its outgoing edges $v_i \in N^+_j \cup \{v_j\}$ a nonzero probability value $b_{ij}$ equal to $b_{ij} = \frac{1}{1 + B^2}$. The assigned values can be seen in the following matrix:

$$B = \begin{bmatrix} \frac{1}{3} & 0 & \frac{1}{3} & 0 \\ \frac{1}{3} & \frac{1}{2} & \frac{1}{3} & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{2} \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix}.$$ 

Furthermore, nodes $v_1$, $v_2$, $v_3$, $v_4$, set $y_1[0] = 10$, $z_1[0] = 2$, $y_2[0] = 6$, $z_2[0] = 2$, $y_3[0] = 14$, $z_3[0] = 2$, $y_4[0] = 4$, $z_4[0] = 2$, respectively.

For the execution of Algorithm 1, at time step $k = 0$, each node $v_j$ calculates its state variables $y_j^0[0]$, $z_j^0[0]$ and $q_j^0[0]$ as shown in Table 1. Then, each node $v_j$ calculates its transmission variables $c_{ij}^y[k]$ and $c_{ij}^z[k]$ for every $v_i \in N^+_j \cup \{v_j\}$. Specifically, each node $v_j$ splits $y_j[0]$ into 2 equal pieces (since $z_j[0] = 2$), and keeps one piece (which has the minimum value) for itself and transmits the other piece to a randomly chosen out-neighbor or itself according to the matrix $B$. For this reason, in the analysis below we have $c_{ij}^z[0] > 0$ for every node $v_j$. Suppose that, following the random choices, nodes $v_1, v_2, v_3, v_4$ set

\[
\begin{align*}
\text{v}_1 & : c_{11}^y[0] = 5, c_{21}^y[0] = 5, c_{31}^y[0] = 0, c_{41}^y[0] = 0, c_{11}^z[0] = 1, c_{21}^z[0] = 1, c_{31}^z[0] = 0, \\
\text{v}_2 & : c_{22}^y[0] = 6, c_{32}^y[0] = 0, c_{22}^z[0] = 2, c_{32}^z[0] = 0, \\
\text{v}_3 & : c_{13}^y[0] = 7, c_{23}^y[0] = 0, c_{33}^y[0] = 7, c_{13}^z[0] = 1, c_{23}^z[0] = 0, c_{33}^z[0] = 1, \\
\text{v}_4 & : c_{34}^y[0] = 2, c_{44}^y[0] = 2, c_{34}^z[0] = 1, c_{44}^z[0] = 1.
\end{align*}
\]

We have that nodes $v_1$, $v_3$ and $v_4$ perform transmissions to nodes $v_2$, $v_1$ and $v_3$, respectively, whereas node $v_2$ transmits to itself.

Then, each node $v_j$ receives from its in-neighbors $v_i \in N^-_j \cup \{v_j\}$ the transmission variables $c_{ij}^y[0]$ and $c_{ij}^z[0]$ and, at time step $k = 1$, it calculates its state variables $y_j^1[1]$, $z_j^1[1]$ and $q_j^1[1]$. The mass and state variables are shown in Table 1 for $k = 1$.

| Nodes | Mass and State Variables for $k = 0$ |
|-------|-----------------------------------|
| $v_j$ | $y_j[0]$ $z_j[0]$ $y_j^0[0]$ $z_j^0[0]$ $q_j^0[0]$ |
| $v_1$ | 10 2 10 2 5 |
| $v_2$ | 6 2 6 2 3 |
| $v_3$ | 14 2 14 2 7 |
| $v_4$ | 4 2 4 2 2 |

| Nodes | Mass and State Variables for $k = 1$ |
|-------|-----------------------------------|
| $v_j$ | $y_j[1]$ $z_j[1]$ $y_j^1[1]$ $z_j^1[1]$ $q_j^1[1]$ |
| $v_1$ | 12 2 12 2 6 |
| $v_2$ | 11 3 11 3 3 |
| $v_3$ | 9 2 9 2 4 |
| $v_4$ | 2 1 2 1 2 |

Then, each node $v_j$ calculates and transmits its transmission variables $c_{ij}^y[k]$ and $c_{ij}^z[k]$ for every $v_i \in N^+_j \cup \{v_j\}$. Specifically, each node $v_j$ splits $y_j[0]$ into 2 equal pieces (since $z_j[0] = 2$), and keeps one piece (which has the minimum value) for itself and transmits the other piece to a randomly chosen out-neighbor or itself according to the matrix $B$. For this reason, in the analysis below we have $c_{ij}^z[0] > 0$ for every node $v_j$. Suppose that, following the random choices, nodes $v_1, v_2, v_3, v_4$ set

\[
\begin{align*}
\text{v}_1 & : c_{11}^y[1] = 12, c_{21}^y[1] = 0, c_{31}^y[1] = 0, c_{11}^z[1] = 2, c_{21}^z[1] = 0, c_{31}^z[1] = 0, \\
\text{v}_2 & : c_{22}^y[1] = 3, c_{32}^y[1] = 8, c_{22}^z[1] = 1, c_{32}^z[1] = 2,
\end{align*}
\]
In the analysis above we can see, for example, that node $v_3$ splits the value $y_3[1] = 9$ into $z_3[1] = 2$ almost equal pieces which are 4 and 5 respectively. Then, it keeps the piece 4 (which has the minimum value) and transmits it to itself, and then it transmits the remaining piece 5 to a randomly selected out-neighbor which happens to be node $v_1$.

In Fig. 2 we plot the resulting state variable $q_s^j[k]$ of every node $v_j \in V$, from which it can be seen that for $k \geq 9$ we have

$$q_s^j[k] = \lfloor q \rfloor \quad \text{or} \quad q_s^j[k] = \lceil q \rceil,$$

for every $v_j \in V$. This means that every node $v_j$ obtains, after a finite number of iterations, a quantized state $q_s^j$, which is equal either to the ceiling or to the floor of the real average $q$ of the initial states of the nodes.

**Remark 2** It is important to note here that Algorithm 1 is significantly different from the algorithm in [35]. In [35] we have that (i) Initialization Step 2 is set to $z_j[0] = 1$, (ii) Iteration Step 1 is set to $z_j[k] > 0$ and (iii) Iteration Step 1.3 is set to $\text{max}^z[k] > 0$. This means that each node splits $y_j[k]$ into $z_j[k]$ equal pieces (or with maximum difference equal to 1) and then transmits every piece to randomly chosen out-neighbors or to itself. Steps (i), (ii), and (iii) allow us to calculate the quantized average in finite time, but they cause the state $q_j$ of each node $v_j$ to oscillate between $\lfloor q \rfloor$ and $\lceil q \rceil$ (where $q$ is defined in (1)) rather than stabilizing to one of the aforementioned values. During the operation of the proposed Algorithm 1, steps (i), (ii) and (iii) are changed to $z_j[0] = 2$, $z_j[k] > 1$, and $\text{max}^z[k] > 1$, respectively. This means that each node splits $y_j[k]$ into $z_j[k]$ equal pieces (or with maximum difference equal to 1), transmits one piece to itself and then transmits the remaining pieces to randomly chosen out-neighbors or to itself. These changes affect significantly the operation of the proposed algorithm and leads to fast convergence while avoiding oscillatory behavior of the agent’s states. Specifically, by changing steps (i), (ii) and (iii), Algorithm 1 (a) achieves quantized average consensus in finite time, (b) maintains similar fast convergence speed as the algorithm in [35] and (c) the state $q_j$ of each node $v_j$ stabilizes to $\lfloor q \rfloor$ or $\lceil q \rceil$.

Furthermore, Algorithm 1 is also significantly different from the algorithm in [36], where the authors presented a distributed deterministic algorithm in which every node $v_j$ “merged” (i.e., added) the incoming mass variables sent by its in-neighbors. No splitting was done and mass variables remained merged during the algorithm execution. The authors of [36] showed that every node $v_j$ calculated, after a finite number of time steps, a quantized fraction which is equal to the actual average $q$ of the initial states of the nodes (i.e., there was zero quantization error), but due to strict accumulation of the states, the proposed algorithm required a significant number of time steps and could also lead to a memory overflow problem if the initial node states were all close to the maximum representable value on the quantized scale (in which case the sum of these states may not be representable using a specific number of (fixed-point) bits). By contrast, during the operation of Algorithm 1, every node $v_j$ “merges” and then “splits” the incoming mass variables, sent by its in-neighbors; splitting of the mass variables allows faster convergence and avoids a memory overflow problem. As we will see in the following sections, this modification allows Algorithm 1 to significantly outperform (in terms of convergence speed) the algorithm presented in [36] and other state-of-the-art algorithms in the available literature.

### 4.1 Convergence of Algorithm 1

We now show that, during the operation of Algorithm 1, each agent $v_j$ addresses problem (P1) presented in Section 3 and reaches, after a finite number of time steps, a consensus state which is equal to either the ceiling or the floor of the actual average $q$ of the initial states of the nodes. We first consider Lemma 1, which is necessary for our subsequent development.

**Lemma 1** ([35]) Consider a strongly connected digraph $G_d = (V, \mathcal{E})$ with $n = |V|$ nodes and $m = |\mathcal{E}|$ edges. Suppose that each node $v_j$ assigns a nonzero probability $b_{ij}$ to each of its outgoing edges $m_{ij}$, where $v_i \in N_j^+ \cup \{v_j\}$, as follows

$$b_{ij} = \begin{cases} 
\frac{1}{1 + \nu_j}, & \text{if } l = j \text{ or } v_l \in N_j^+ \\
0, & \text{if } l \neq j \text{ and } v_l \notin N_j^+.
\end{cases}$$

At time step $k = 0$, node $v_j$ holds a “token” while the other nodes $v_i \in V - \{v_j\}$ do not. Each node $v_j$ transmits the “token” (if it has it, otherwise it performs no
transmission) according to the nonzero probability \(b_{ij}\) it assigned to its outgoing edges \(m_{ij}\). The probability \(P_{T_i}^{n-1}\) that the token is at node \(v_i\) after \(n-1\) time steps satisfies

\[
P_{T_i}^{n-1} \geq (1 + D_+^{\max})^{-(n-1)} > 0,
\]

where \(D_+^{\max} = \max_{v_i \in V} D_j^+\).

**Theorem 1** Consider a strongly connected digraph \(G_d = (V, E)\) with \(n = |V|\) nodes and \(m = |E|\) edges and \(z_j[0] = 1\) and \(y_j[0] \in \mathbb{Z}\) for every node \(v_j \in V\) at time step \(k = 0\). Suppose that each node \(v_j \in V\) follows the Initialization and Iteration steps as described in Algorithm 1. For any \(\epsilon\), where \(0 < \epsilon < 1\), there exists \(K_0 \in \mathbb{Z}_+\), so that with probability \((1 - \epsilon)^{y_{\text{init}} + n}\), we have

\[
(q_j^k[k] = \lfloor q \rfloor \text{ for } k \geq K_0) \text{ or } (q_j^k[k] = \lceil q \rceil \text{ for } k \geq K_0),
\]

for every \(v_j \in V\), where \(q\) fulfills (1) and

\[
y_{\text{init}} = \sum_{v_j: y_j[0] > \lfloor q \rfloor} (y_j[0] - \lfloor q \rfloor) + \sum_{v_j: y_j[0] < \lfloor q \rfloor} (\lceil q \rceil - y_j[0]),
\]

is the total initial state error (i.e., \(y_{\text{init}}\) is the sum of the differences between (i) the value \(\lfloor q \rfloor\) and the initial state \(y_j[0]\) of each node \(v_j\) that has an initial state higher than the ceiling of the real average and (ii) the value \(\lceil q \rceil\) and the initial state \(y_j[0]\) of each node \(v_j\) that has an initial state less than the floor of the real average).

**Proof.** The operation of Algorithm 1 can be interpreted as the “random walk” of \(n\) “tokens” in a Markov chain, where \(n = |V|\). At each time step \(k\), each of these \(n\) tokens contains a pair of values \(y_j[k]\), \(z[k]\), for which \(y_j[k] \in \mathbb{Z}\) and \(z[k] = 1\). Specifically, at time step \(k = 0\), node \(v_j\) holds two “tokens”. One token is \(T_j^{\text{in}}\) and is stationary, whereas the other token is \(T_j^{\text{out}}\) and performs a random walk. Each token \(T_j^{\text{in}}\) and \(T_j^{\text{out}}\) contains a pair of values \(y_j^{\text{in}}[k]\), \(z_j^{\text{in}}[k]\), and \(y_j^{\text{out}}[k]\), \(z_j^{\text{out}}[k]\), respectively. Initially, we have \(y_j^{\text{in}}[0] = y_j^{\text{out}}[0] = y_j[0]\) and \(z_j^{\text{in}}[0] = z_j^{\text{out}}[0] = z[0]\). At each time step \(k\), each node \(v_j\) keeps the token \(T_j^{\text{in}}\) (i.e., it never transmits it) while it transmits the token \(T_j^{\text{out}}\), according to the nonzero probability \(b_{ij}\) it assigned to its outgoing edges \(m_{ij}\) during the Initialization Steps. If \(v_j\) receives one or more tokens \(T_j^{\text{out}}\) from its in-neighbors \(v_i\) the values \(y_j^{\text{out}}[k]\) and \(y_j^{\text{in}}[k]\) become equal (or with maximum difference equal to 1); then \(v_j\) transmits each received token \(T_j^{\text{out}}\) to a randomly selected out-neighbor according to the nonzero probability \(b_{ij}\) it assigned to its outgoing edges \(m_{ij}\). Note here that during the operation of Algorithm 1 we have

\[
\sum_{j=1}^{n} y_j^{\text{out}}[k] + \sum_{j=1}^{n} y_j^{\text{in}}[k] = 2 \sum_{j=1}^{n} y_j[0], \forall k \in \mathbb{Z}_+,
\]

(i.e., the sum of the \(y_j[k]\) values of the tokens at any given \(k\) is equal to twice the initial sum).

Let us now define

\[
Y[k] = Y_1[k] + Y_2[k],
\]

where

\[
Y_1[k] = \sum_{(v_j \in V: |y_j[k]/z[j][k]| < \lfloor q \rfloor)} ([y_j[k]/z[j][k]] - \lfloor q \rfloor),
\]

and

\[
Y_2[k] = \sum_{(v_j \in V: \lfloor y_j[k]/z[j][k]\rfloor < \lfloor q \rfloor)} (\lfloor q \rfloor - [y_j[k]/z[j][k]]),
\]

where \(q\) satisfies (1). We have that \(Y_1[k]\) denotes the sum of the differences between the values \(\lfloor q \rfloor\) and \(y/[k]\) of the tokens that have a \(y\) value higher than the ceiling of the real average \(\lfloor q \rfloor\), and \(Y_2[k]\) denotes the sum of the differences between the values \(\lfloor q \rfloor\) and \(y/[k]\) of the tokens that have \(y\) value less than the floor the real average \(\lfloor q \rfloor\). Clearly, if \(Y_1[k] > 0\) then at least one token has \(y[k]\) value which is greater than the ceiling of the real average, and if \(Y_2[k] > 0\) then at least one token has \(y[k]\) value which is less than the floor of the real average, at time step \(k\). From Iteration Steps 1.3 and 1.4, we have that if two (or more) “tokens” \(T_i^{\text{out}}\), \(T_i^{\text{out}}\) (where \(v_i, v_l \in V\) meet at the same node \(v_j\) with token \(T_j^{\text{out}}\) (which is kept in node \(v_j\)) during time step \(k\), then their values \(y[j][k]\) become equal (or with maximum difference equal to 1). For the scenario \([q] > \lfloor q \rfloor\), we have at time step \(k\) (note that similar arguments hold also for \(\lfloor q \rfloor = \lfloor q \rfloor\)):

Case (i): If \(Y_1[k] > 0\) and a token which has \(y[k] > \lfloor q \rfloor\) meets with a token that has \(y[k] \leq \lfloor q \rfloor\) then we have \(Y_1[k+1] \leq Y_1[k] - 1\).

Case (ii): If \(Y_2[k] > 0\) and a token which has \(y[k] < \lfloor q \rfloor\) meets with a token that has \(y[k] \geq \lfloor q \rfloor\) then we have \(Y_2[k+1] \leq Y_2[k] - 1\).

Case (iii): If \(Y_1[k] > 0\) and \(Y_2[k] > 0\) and a token which has \(y[k] > \lfloor q \rfloor\) meets with a token that has \(y[k] < \lfloor q \rfloor\) then we have \(Y_1[k+1] \leq Y_1[k] - 1\) and \(Y_2[k+1] \leq Y_2[k] - 1\).

Note that for the scenario \([q] = \lfloor q \rfloor\) we have that only Case (iii) above holds. Case (i) and Case (ii) do not hold since the difference between the values \(y[k]\) of the tokens that meet might be equal to one which means that the values of \(Y_1[k]\) and \(Y_2[k]\) will not decrease.
Clearly, we have

\[ 0 \leq Y[k+1] \leq Y[k] \leq y^{\text{init}}, \]

for all time steps \( k \), where \( y^{\text{init}} \) fulfills (6). This means that if cases (i), (ii), (iii) hold for \( y^{\text{init}} \) times the value of \( Y \) becomes equal to zero (where \( Y \) is defined in (8)). As a result, for every token the values \( y \) become equal or have difference equal to one (recall that, during the operation of Algorithm 1, we have that (7) holds for every \( k \)).

In this proof, we focus on the scenario in which one token \( y_i^{\text{out}} \) visits a specific node \( v_j \) (for which it holds \( |y_i^{\text{out}} - y_i^{\text{ins}}| > 1 \)) and obtains equal values (or with maximum difference between them equal to 1) with the token \( y_i^{\text{ins}} \) which is kept in node \( v_j \). We consider and analyze the probability that this specific token visits this specific node in the network after a finite number of time steps. For any \( \epsilon \), where \( 0 < \epsilon < 1 \), we show that (i) \( \exists k_0 \in \mathbb{Z}_+ \) for which with probability \((1-\epsilon)^{y_{i}^{\text{out}}}\), it holds that \( Y[k] = 0 \) for every \( k \geq k_0 \), and (ii) \( \exists k_0 \in \mathbb{Z}_+ \) for which, with probability \((1-\epsilon)^{(y_i^{\text{ins}}+n)}\), (2) holds for every \( k \geq k_0 \). This means that after a finite number of time steps \( k_0 \) the value \( y[k] \) of every token is equal either to \([q]\) or to \([\bar{q}]\), and for the state variable \( q_j^{\text{ins}}[k] \) of every node \( v_j \) we have \( q_j^{\text{ins}}[k] = [q] \) or \( q_j^{\text{ins}}[k] = [\bar{q}] \), respectively.

Let us consider tokens \( T_\lambda^{\text{out}} \) and \( T_\lambda^{\text{ins}} \) for which it holds (i) \( y_1^{\text{out}} \geq [q], y_1^{\text{ins}} < [q], y_i^{\text{out}} > [q], y_i^{\text{ins}} \leq [q], \) or (ii) \( y_1^{\text{out}} < [q], y_1^{\text{ins}} \geq [q], y_i^{\text{out}} \leq [q], y_i^{\text{ins}} > [q] \). During the operation of Algorithm 1, \( n \) “tokens” perform independent random walks, and from Lemma 1 we have that the probability \( P_{T_\lambda^{\text{out}}}^{n-1} \) that “the specific token \( T_\lambda^{\text{out}} \) is at node \( v_i \) after \( n-1 \) time steps” is

\[ P_{T_\lambda^{\text{out}}}^{n-1} \geq (1 + D_{\text{max}}^+)^{-(n-1)}. \] (11)

This means that the probability \( P_{N,T_\lambda^{\text{out}}}^{n-1} \) that “the specific token \( T_\lambda^{\text{out}} \) has not visited node \( v_i \) after \( n-1 \) time steps” is

\[ P_{N,T_\lambda^{\text{out}}}^{n-1} \leq 1 - (1 + D_{\text{max}}^+)^{-(n-1)}. \] (12)

By extending this analysis, we can state that for any \( \epsilon \), where \( 0 < \epsilon < 1 \) and after \( \tau(n-1) \) time steps where

\[ \tau \geq \left\lceil \frac{\log \epsilon}{\log (1 - (1 + D_{\text{max}}^+)^{-(n-1)})} \right\rceil, \] (13)

the probability \( P_{N,T_\lambda^{\text{out}}} \) that “the specific token \( T_\lambda^{\text{out}} \) has not visited node \( v_i \) after \( \tau(n-1) \) time steps” is

\[ P_{N,T_\lambda^{\text{out}}} \leq [P_{N,T_\lambda^{\text{out}}}^{n-1}]^\tau \leq \epsilon. \] (14)

This means that after \( \tau(n-1) \) time steps, where \( \tau \) fulfills (13), the probability that “the specific token \( T_\lambda^{\text{out}} \) has visited node \( v_i \) after \( \tau(n-1) \) time steps” is equal to \( 1 - \epsilon \).

As a result, after \( \tau(n-1) \) time steps, where \( \tau \) fulfills (13), we have that if \( Y_1[k] > 0 \) and/or \( Y_2[k] > 0 \) at time step \( k \), then it holds that \( Y_1[k + \tau(n-1)] \leq Y_1[k] - 1 \) and/or \( Y_2[k + \tau(n-1)] \leq Y_2[k] - 1 \) with probability \( 1 - \epsilon \). By extending this analysis, we have that for \( k \geq y^{\text{init}} \tau(n-1) \) time steps, where \( y^{\text{init}} \) is given by (6), we have \( Y[k] = 0 \) with probability \((1-\epsilon)^{y_{i}^{\text{out}}}\). Therefore for \( k \geq y^{\text{init}} \tau(n-1) \), we have that the value \( y[k] \) of every token is equal to either \([q]\) or \([\bar{q}]\) with probability \((1-\epsilon)^{y_{i}^{\text{out}}}\). Since (7) holds, for \( k \geq y^{\text{init}} \tau(n-1) \) we have

\[ [y_j^{\text{out}}[k]/z_j^{\text{out}}[k]] = [q] \] or \([y_j^{\text{out}}[k]/z_j^{\text{out}}[k]] = [\bar{q}]\),

and

\[ [y_j^{\text{ins}}[k]/z_j^{\text{ins}}[k]] = [q] \] or \([y_j^{\text{ins}}[k]/z_j^{\text{ins}}[k]] = [\bar{q}]\),

for every \( v_j \in \mathcal{V} \) with probability \((1-\epsilon)^{y_{i}^{\text{out}}}\). Furthermore, we have that for \( k \geq y^{\text{init}} \tau(n-1) \) it holds

\[ |\{T_\lambda^{\text{ins}}, v_j \in \mathcal{V} | y_j^{\text{ins}}[k] = [q]\}| + |\{T_\lambda^{\text{out}}, v_j \in \mathcal{V} | y_j^{\text{out}}[k] = [q]\}| = 2n - 2R. \] (17)

Continuing the analysis, let us consider now the following two cases

1. \( 2n - 2R \geq n \) (or \( R < n/2 \)),
2. \( 2n - 2R < n \) (or \( R > n/2 \)),

where \( n \) is the number of nodes and \( R \) is defined in (3).

For the first case, we have that the number of tokens which have value equal to \([q]\) is greater than (or equal to) the number of nodes. This means that by executing Algorithm 1 for an additional number of \( n \tau(n-1) \) time steps, where \( \tau \) fulfills (13), we have that every node will receive at least one token with value \([q]\) with probability \((1-\epsilon)^{n}\). The reason is that during the first \( \tau(n-1) \) steps, one of the tokens with value \([q]\) will reach node \( v_1 \) with probability \( 1 - \epsilon \); during the second \( \tau(n-1) \) steps, one of the tokens with value \([q]\) will reach node \( v_2 \) with probability \( 1 - \epsilon \), and so on. During the last \( \tau(n-1) \) steps, one of the tokens with value \([q]\) will reach node \( v_n \) with probability \( 1 - \epsilon \). From Iteration Steps 1.3 and
1.4 of Algorithm 1 we have that if node $v_j$ receives a token with value $[q]$, then the value of its $y^{ins}_j$ token becomes equal to $[q]$ which means that also the value of its state variable $q^*_j$ becomes equal to $[q]$. As a result, since $2n - 2R \geq n$, for $k \geq (y^{init} + n)\tau(n - 1)$, where $y^{init}$ fulfills (6) and $\tau$ fulfills (13), we have

$$y^{ins}_j[k] = [q], \text{ for every } v_j \in \mathcal{V},$$

and

$$q^*_j[k] = [q], \text{ for every } v_j \in \mathcal{V},$$

with probability $(1 - \varepsilon)^{(y^{init} + n)}$.

For the second case, the number of tokens which have value equal to $[q]$ is less than the number of nodes. Identically to the first case, by executing Algorithm 1 for an additional number of $n\tau(n - 1)$ time steps, where $\tau$ fulfills (13), for $k \geq (y^{init} + n)\tau(n - 1)$ we have with probability $(1 - \varepsilon)^{(y^{init} + n)}$ that

$$y^{ins}_j[k] = [q] \text{ and } q^*_j[k] = [q], \text{ for every } v_j \in \mathcal{V},$$

where $\mathcal{V}' \subset \mathcal{V}$ and $|\mathcal{V}'| = 2n - 2R$, and

$$y^{ins}_j[k] = [q] \text{ and } q^*_j[k] = [q], \text{ for every } v_j \in \mathcal{V}'',$$

where $\mathcal{V}'' \subset \mathcal{V}$ and $|\mathcal{V}''| = 2R - n$.

As a result, during the operation of Algorithm 1, for $k \geq (y^{init} + n)\tau(n - 1)$ we have

$$q^*_j[k] = [q] \text{ or } q^*_j[k] = [q],$$

for every $v_j \in \mathcal{V}$, with probability $(1 - \varepsilon)^{(y^{init} + n)}$. \hfill $\square$

Remark 3 Note that the we can also bound the number of time steps $k_0$ needed for Algorithm 1 to converge according to a desired probability $p_0$. From the proof of Theorem 1, we have that each node $v_j$ can calculate the quantized average of the initial states with probability $p_0 \geq (1 - \varepsilon)^{(y^{init} + n)}$, where $0 < \varepsilon < 1$ and $y^{init}$ fulfills (6), after at least $k_0 \geq (y^{init} + n)\tau(n - 1)$ time steps, where $y^{init}$ fulfills (6) and $\tau$ fulfills (13). This means that if we want to guarantee convergence of Algorithm 1 with probability greater than or equal to $p_0$, we have to choose $\varepsilon$ for which $e = 1 - 2^{\frac{\log_2 p_0}{y^{init} + n}}$. Then, according to the proof of Theorem 1, we need to execute Algorithm 1 for a number of time steps $k_0$ for which it holds

$$k_0 \geq (y^{init} + n) \frac{\log_2 \left( 1 - 2^{\frac{\log_2 p_0}{y^{init} + n}} \right)}{\log_2 \left( 1 - (1 + D^+_{max})^{(n-1)} \right)}(n - 1),$$

where $D^+_{max} = \max_{v_j \in \mathcal{V}} D^+_j$ and $y^{init}$ fulfills (6).

Remark 4 Algorithm 1 possesses attractive features for consensus-based distributed optimization. Apart from operating with quantized states which reduces the communication bottleneck [31, 40], it also allows for fast distributed averaging, which makes it suitable as an intermediate step between the optimization operations [14, 19]. In the latter case, the convergence speed of the averaging algorithm plays a significant role for the overall convergence speed of the optimization procedure (as we will see in subsequent sections, the convergence speed of Algorithm 1 significantly outperforms state-of-the-art algorithms in the available literature). Algorithm 1 can also find various applications on load balancing and voting schemes where each node needs to calculate a specific state rather than oscillate between two different states/decisions.

5 Quantized Averaging over Dynamic Digraphs

In this section, we present a distributed algorithm which addresses problem (P2) presented in Section 3. This means that the results of Section 4 are extended to include directed topologies with time-varying communication links. We assume that, at each time step $k$, the interconnections between components in the multi-component system are captured by a digraph $G_d[k] = (\mathcal{V}, \mathcal{E}[k])$ in which the set of nodes is fixed but the communication links may change.

Assumption 1

A.1. At each time step $k$, each node $v_j$ has knowledge of the set of its out-neighbors $N^+_j[k]$ and the number of its out-neighbors $D^+_j[k]$.

A.2. For any infinite sequence of dynamic digraphs $G_d[1], G_d[2], ..., G_d[k], ...,$, there is a finite window length $l \in \mathbb{N}$ and an infinite sequence of time instants $t_0, t_1, ..., t_m, ...,$ where $t_0 = 0$, such that for any $m \in \mathbb{Z}_+$, we have $0 < t_{m+1} - t_m < l < \infty$ and the union graph $G_d[t_0, t_1, ..., t_{m+1}]$, is equal to the nominal digraph $G_d$ which is assumed to be strongly connected.

A.2. There is a finite collection of digraphs $G_{d,1}, G_{d,2}, ..., G_{d,M}$, such that the union graph is strongly connected and at each time step $k$ one such topology is selected independently in an i.i.d. manner. Specifically, at time step $k$, we have $G_d[k] = G_{d,\theta}$ for some $\theta \in \{1, 2, ..., M\}$ with probability $p_\theta$ where $\sum_{\theta=1}^M p_\theta = 1$.

Assumption A.1 implies that the transmitting node knows the number of nodes it transmits messages to at each time instant. In an undirected graph setting, this is not difficult and can be done straightforwardly; in a directed graph setting, this is challenging but there are ways in which knowledge of the out-degree might be possible. For example, there can be an acknowledgement signal via a distress signal (special tone in a control slot or some separate control channel) sent at higher power.
than normal so that it is received by transmitters in its vicinity [2]. Knowledge of the out-degree is also possible if the nodes periodically perform checks to determine the number of their out-neighbors (e.g., by periodically transmitting the distress signals mentioned above).

Assumption A2 (or $\overline{A_2}$) is sufficient for the existence of at least one directed path between any pair of nodes infinitely often.

Under the above assumptions, during the operation of Algorithm 2, each node $v_j$ is required to calculate the nonzero probabilities $b_{ij}[k]$ for each of its outgoing edges $m_{ij}[k]$ (where $v_i \in N_{ij}^+[k] \cup \{v_j\}$) at each time step $k$. This calculation is due to the dynamic nature of the communication topology $G_d[k]$. Note that since each transmitting node $v_j$ has instant knowledge of its out-degree then it sets the weights $b_{ij}[k]$ to be equal to $b_{ij}[k] = \frac{1}{1 + D_{jj}[k]}$ for $v_i \in N_{ij}^+[k] \cup \{v_j\}$. This choice satisfies $\sum_{i=1}^{n_2} b_{ij}[k] = 1$ for all $v_j \in V$ which means that the transition matrix $B[k] = [b_{ij}[k]]$ is column-stochastic at every time step $k$. Furthermore, unspecified weights in $B[k]$ are set to zero and correspond to pairs of nodes ($v_i, v_j$) that are not connected at time step $k$, i.e., $b_{ij}[k] = 0$ for $v_i \notin N_{ij}^+[k] \cup \{v_j\}$.

**Remark 5** Note here that the operation of Algorithm 1 is simpler than Algorithm 2 since the computation of the nonzero probabilities $b_{ij}$ (for each of its outgoing edges $m_{ij}$) is done by each node $v_j$ only during the initialization steps and is not repeated at each time step $k$. This means that for static directed communication topologies, both algorithms will exhibit the same performance and convergence rate but, due to their operation, Algorithm 1 will consume less computational resources compared to Algorithm 2.

### 5.1 Convergence of Algorithm 2

We now show that, during the operation of Algorithm 2, each agent $v_j$ reaches a consensus state which is equal either to the ceiling or the floor of the actual average $q$ of the initial states of the nodes for the case where the digraph $G_d[k]$ is dynamic (i.e., each agent addresses problem (P2) presented in Section 3). We analyze the operation of Algorithm 2 considering the set of assumptions $A_1, A_2$ (the set of assumptions $A_1, \overline{A_2}$ can be proven identically).

**Theorem 2** Consider a sequence of digraphs $G_d[k] = (V, E[k]), k = 0, 1, 2, \ldots$, with $n = |V|$ nodes, $m[k] = |E[k]|$ edges so that assumptions $A_1, A_2$ hold for $G_d[k]$ over all $k$. Every node $v_j \in V$ has the variables $z_j[0] = 1$ and $y_j[0] \in \mathbb{Z}$ at time step $k = 0$, and it follows the initialization and iteration steps as described in Algorithm 2. For any $\varepsilon$, where $0 < \varepsilon < 1$, there exists $k_0 \in \mathbb{Z}_+$, so that

**Algorithm 2** Quantized Average Consensus Over Dynamic Digraphs

**Input** A set of digraphs $G_d[k] = (V, E[k])$ with $n = |V|$ nodes and $m[k] = |E[k]|$ edges, for each $k = 0, 1, 2, \ldots$. Each node $v_j \in V$ has an initial state $y_j[0] \in \mathbb{Z}$.

**Initialization:** Each node $v_j \in V_p$ sets $y_j[0] := 2y_j[0], \ z_j[0] = 2$.

**Iteration:** For $k = 0, 1, 2, \ldots$, each node $v_j \in V_p$ does the following:
1. assigns a nonzero probability $b_{ij}[k]$ to each of its outgoing edges $m_{ij}[k]$, where $v_i \in N_{ij}^+[k] \cup \{v_j\}$, as follows
   $$b_{ij}[k] = \begin{cases} \frac{1}{1 + D_{jj}[k]}, & \text{if } l = j \text{ or } v_i \in N_{ij}^+[k], \\ 0, & \text{if } l \neq j \text{ and } v_i \notin N_{ij}^+[k]. \end{cases}$$
2. if $z_j[k] > 1$, then
   2.1) sets $z_j^2[k] = z_j[k], y_j^2[k] = y_j[k], q_j^2[k] = \left\lfloor \frac{y_j[k]}{z_j[k]} \right\rfloor$;
   2.2) sets (i) $\mas^y[k] = y_j[k], \mas^z[k] = z_j[k]$; (ii) $c_{ij}^y[k] = 0, c_{ij}^z[k] = 0$, for every $v_i \in \mathbb{N}_{ij}^+[k] \cup \{v_j\}$;
   (iii) $\delta = [\mas^y[k]/\mas^z[k]], \mas^{rem}[k] = y_j[k] - \delta \mas^z[k];$
3. while $\mas^z[k] > 1$, then
   3.1a) chooses $v_i \in N_{ij}^+[k] \cup \{v_j\}$ randomly according to $b_{ij}$;
   3.1b) sets (i) $c_{ij}^y[k] := c_{ij}^y[k] + 1, c_{ij}^z[k] := c_{ij}^z[k] + \delta$; (ii) $\mas^z[k] := \mas^z[k] - 1, \mas^{rem}[k] := \mas^{rem}[k] - \delta$;
   3.2c) if $\mas^{rem}[k] > 0$, then sets $c_{ij}^y[k] := c_{ij}^y[k] + 1, \mas^{rem}[k] := \mas^{rem}[k] - 1$.
4. if $c_{ij}^y[k] > 0$, then transmits $c_{ij}^y[k], c_{ij}^z[k]$ to out-neighbours $v_i$, for every $v_i \in N_{ij}^+[k]$.
   - else if $z_j[k] \leq 1$, then sets $c_{ij}^y[k] = y_j[k], c_{ij}^z[k] = z_j[k];$
5. receives $c_{ij}^y[k], c_{ij}^z[k]$ from $v_i \in N_{ij}^-[k]$ and updates $y_j[k+1], z_j[k+1]$ according to (4), (5).

**Output:** (2) holds for every $v_j \in V$.

with probability $(1 - \varepsilon)^{(y^{init} + n)}$ we have

$$(q_j^y[k] = [q] \text{ for } k \geq k_0) \text{ or } (q_j^z[k] = [q] \text{ for } k \geq k_0),$$

for every $v_j \in V$, where $y^{init}$ is given in (6) and $q$ is given by (1) (i.e., for $k \geq k_0$ every node $v_j$ has calculated the ceiling or the floor of the actual average $q$ of the initial states).

**PROOF.** The operation of Algorithm 2 can be interpreted as the “random walk” of $n$ “tokens” in a dynamic (inhomogeneous) Markov chain (i.e., interconnections change over time) with $n = |V|$ states. Each node
of Algorithm 2, we also have that (7) holds for every k).

In this proof, we consider and analyze the probability that a specific token, with value $y_{init}$, visits a specific node $v_i$, with token value $y_j^{ins}$, in the network after a finite number of time steps and obtains equal values (or with maximum difference between them equal to 1) with the token $y_j^{ins}$, where for tokens $T^+_i$ and $T^-_i$ it holds (i) $y^{out}_i \geq [q_i]$, $y^{ins}_i < [q_i]$, or (ii) $y^{out}_i > [q_i]$, $y^{ins}_i \leq [q_i]$, or (iii) $y^{out}_i < [q_i]$, $y^{ins}_i \geq [q_i]$, or (iv) $y^{out}_i \leq [q_i]$, $y^{ins}_i > [q_i]$. Note that the main difference with the proof of Theorem 1 is that during the operation of Algorithm 2 the underlying communication topology is dynamic (i.e., interconnections change at every time step $k$). For any $\epsilon$, where $0 < \epsilon < 1$, we show that (i) $\exists k' \in \mathbb{Z}_+$ for which with probability $(1-\epsilon)y^{ins}$, it holds that $Y_{\lambda}[0] = 0$ and $Y_{\lambda}[2k_0] = 0$ for every $k \geq k'0$, and (ii) $\exists k_0 \in \mathbb{Z}_+$ for which (2) holds with probability $(1-\epsilon)(y^{init}+1)$, for every $k \geq k_0$. This means that after a finite number of time steps $k_0$ the value of every token is equal either to $[q_i]$ or to $[q_i]$, and for the state variable $q_i^n[k]$ of every node $v_i$ we have $q_i^n[k] = [q_i]$ or $q_i^n[k] = [q_i]$, respectively.

Let us consider tokens $T^+_i$ and $T^-_i$ for which it holds

$$P_{\lambda}(n-1) \geq (1 + d_{max}^-)^{-l(n-1)} > 0.$$  

(18)

This is mainly due to the fact that every $l$ time steps, each edge is active for at least one time step. Since the nominal digraph $G_d$ is strongly connected, it has a path of length at most $n-1$ from each node $v_i$ to each node $v_j$. Thus, at the first $l$ steps, we can select the first edge in this path (at the instant when it is active) and use self loops at the remaining instants; during the next $l$ time steps, we can select the second edge on this path and use self loops at the remaining instants; and so forth.

From (18) we have that the probability $P_{N,DT}^l(n-1)$ that “the specific token $T^+_i$ is not at node $v_i$ after $l(n-1)$ time steps” is

$$P_{N,DT}^l(n-1) \leq 1 - (1 + d_{max}^-)^{-l(n-1)}.$$  

(19)

and from this point onward, we can use steps similar to the analysis in the proof of Theorem 1.

**Remark 6** Note that during the operation of Algo-
Algorithm 2 if we adopt the set of assumptions $A_1, \overline{A_2}$ for $G_d[k]$ during every $k$, then (18) becomes
\[
P_{\theta}^{n-1} \geq [p_{\theta_{\min}}(1 + D_{max}^+)]^{-(n-1)} > 0, \tag{20}
\]
where $p_{\theta_{\min}} = \min_{\theta \in \{1, 2, \ldots, M\}} p_{\theta}$. Since each digraph $G_{\theta}$, for $\theta \in \{1, 2, \ldots, M\}$, is selected in an i.i.d. manner with probability $p_{\theta} > p_{\theta_{\min}}$ and the union graph is strongly connected, we can first select a topology that includes the first edge on the path from node $v_i$ to node $v_j$ (at least one such topology exists) then select a topology that includes the second edge on the path from node $v_i$ to node $v_j$ (at least one such topology exists), and so forth (with self loops included if necessary). Then, (18) is replaced by (20) and the structure of the proof remains the same.

It is important to note here that Algorithm 2 converges in finite time even in the presence of dynamic communication topologies. Compared to Algorithm 1, the main difference is an increase on the required number of time steps for convergence (which will be shown explicitly in the next section). However, in practical applications, there is also a possible increase in the processor usage of every node (due to the calculation of the nonzero probabilities $b_{ij}[k]$ for each of its outgoing edges $m_{ij}[k]$ during each time step $k$ in Iteration Step 1 of Algorithm 2) and a possible increase on the required number of transmissions for convergence. Analysis of the requirements on processor usage and number of transmissions will be considered in the future in order to highlight the proposed algorithms’ operational advantages.

6 Simulation Results

In this section, we illustrate the behavior and the advantages of the proposed distributed algorithms. We execute Algorithm 1 over a static random digraph of 10 nodes, and Algorithm 2 over a set of dynamic digraphs of 10 nodes whose union graph is equal to the nominal digraph after $l = 5$ time steps. The initial quantized states of the nodes were randomly chosen between 1 and 50 (for each node, the initial state was a randomly chosen quantized value between 1 and 50 with probability $\frac{1}{50}$) with the average of the initial states of the nodes turning out to be $q = \frac{368}{10} = 36.8$ which means that $\lfloor q \rfloor = 36$ and $\lceil q \rceil = 37$.

Then, we show the average number of time steps needed for quantized average consensus to be reached over 1000 randomly generated digraphs of 20 nodes each and compare the performance of Algorithm 1 against existing state-of-the-art approaches. The initial quantized states of the nodes were also randomly chosen between 1 and 50 with the average of the initial states of the nodes turning out to be $q = \frac{526}{20} = 26.3$. Furthermore, for convenience, the initial quantized state of each node remained the same for each one of the 1000 randomly generated digraphs, which means that the average of the nodes initial quantized states also remained equal to $q = \frac{526}{20} = 26.3$.

We compare the performance of Algorithm 1 against five other algorithms: (a) the distributed averaging algorithm with quantized communication presented in [35] in which, at each time step $k$, each agent splits its mass variables in equal pieces and then transmits all of the pieces to randomly chosen out-neighbors, (b) the distributed averaging algorithm with quantized communication presented in [36] in which, at each time step $k$, each agent sends its mass variables towards an out-neighbor chosen according to a priority in the form of a quantized fraction, (c) the distributed averaging algorithm with quantized communication presented in [8] in which, at each time step $k$, one edge, say edge $(v_i, v_j)$, is selected at random and, node $v_j$ sends its state information and surplus to node $v_i$, which performs updates over its own state and surplus values, (e) the quantized gossip algorithm presented in [18] in which, at each time step $k$, one edge is selected at random, independently from earlier instants, and the states of the nodes that the selected edge is incident on are updated.

Fig. 3 shows the operation of Algorithm 1 over a static random digraph of 10 nodes and Algorithm 2 over a set of dynamic digraphs of 10 nodes whose union is equal to the nominal digraph after $l = 5$ time steps. In both cases the average of the initial states of the nodes being equal to $q = \frac{368}{10} = 36.8$. On the top of Fig. 3 we can see that during the operation of Algorithm 1 every node is able to reach quantized average consensus after 10 time steps and the states of the nodes stabilize to be equal either to $\lfloor q \rfloor = 36$ or $\lceil q \rceil = 37$ after 15 time steps. At the bottom of Fig. 3 we can see that Algorithm 2 requires more steps to converge than Algorithm 1 due to the dynamic nature of the communication topology (since the union graph of the the set of dynamic digraphs is equal to the nominal digraph after $l = 5$ time steps) and each node’s state is able to stabilize to be equal either to $\lfloor q \rfloor = 36$ or $\lceil q \rceil = 37$ after 47 time steps. This makes Algorithm 2 the first algorithm in the literature to achieve oscillation-free quantized average consensus after a finite number of time steps over dynamic digraphs without any network requirements since (i) in [6] the

\[1\] Note here that the algorithm in [18] requires the underlying graph to be undirected. For this reason, in Fig. 4, for [18], we make the randomly generated underlying digraphs undirected (by enforcing that if $(v_i, v_j) \in \mathcal{E}$ then also $(v_j, v_i) \in \mathcal{E}$) while, for the algorithms in [6, 8, 35, 36], the randomly generated underlying graph is generally directed.
calculation of the quantized average relies on a static threshold that depends on the number of nodes in the network, (ii) in [8] the operation requires a set of weights over the links of the dynamic digraph that form a doubly stochastic matrix which need to be recalculated again during each time step (see [13, 32]) while the states of the nodes exhibit an oscillating behavior, and (iii) in [18] the operation requires bidirectional communication (i.e., undirected graph) and the states of the nodes also exhibit an oscillating behavior.

Fig. 4 shows the average number of time steps needed for quantized average consensus to be reached over 1000 randomly generated digraphs of 20 nodes each, in which the average of the nodes initial states is equal to $q = \frac{526}{20} = 26.3$. In Fig. 4 we can see that Algorithm 1 generally outperforms most finite time algorithms in the current literature. Its convergence speed is equal to [35] and [8], with the difference, however, being that every node’s state is able to stabilize either to $[q] = 26$ or $\lceil q \rceil = 27$ rather than oscillate between these states. This can be seen from (A), (B), (D) in Fig. 4. Specifically, during Algorithm 1 (see (A)) each node’s state becomes equal to $[q] = 26$. However, during the algorithms in [35] and [8] (see (B), (D)) the state of each node does not become equal to a specific value due to oscillations between $|q| = 26$ or $\lceil q \rceil = 27$. The proposed algorithm has no prerequisites regarding the underlying communication topology (e.g., there is no need to obtain a set of weights over the digraph links that form a doubly stochastic matrix [8]).

7 Conclusions

We have considered the quantized average consensus problem and presented a randomized distributed averaging algorithm in which the processing, storing and exchange of information between neighboring agents is subject to uniform quantization. We analyzed its operation, established its correctness and showed that it allows every agent to reach a consensus state equal to either the ceiling or the floor of the real average (thus avoiding oscillating behavior) without any specific requirements regarding the network that describes the underlying communication topology, apart from strong connectedness (see [8]). Furthermore, we presented experimental results and argued that its convergence speed appears to outperform almost every algorithm in the available literature. Finally, we presented an enhanced version of our algorithm in which each agent achieves quantized average consensus (while also avoiding oscillating behavior) in the presence of a dynamically changing communication network.

In the future we plan to investigate stricter bounds on the convergence speed of Algorithms 1 and 2. We also plan to explore their performance in the presence of network unreliability (e.g., delays and packet drops).

References

[1] T. C. Aysal, M. Coates, and M. Rabbat. Distributed average consensus using probabilistic quantization. IEEE/SP Workshop on Statistical Signal Processing, pages 640–644, 2007.

[2] N. Bambos, S. C. Chen, and G. J. Pottie. Channel access algorithms with active link protection for wireless communication networks with power control. IEEE/ACM Transactions on Networking, 8(5):583–597, 2000.

[3] T. Basar, S. Etesami, and A. Olshesvsky. Convergence time of quantized metropolis consensus over time-varying networks. IEEE Transactions on Automatic Control, 61(12):4048–4054, December 2016.

[4] F. Benezit, P. Thiran, and M. Vetterli. The distributed multiple voting problem. IEEE Journal of Selected Topics in Signal Processing, 5(4):791–804, August 2011.

[5] V. D. Blondel, J. M. Hendrickx, A. Olshesvsky, and J. N. Tsitsiklis. Convergence in multiagent cooperation, consensus, and flocking. Proceedings of the IEEE Conference on Decision and Control, pages 2996–3000, 2005.

[6] K. Cai and H. Ishii. Quantized consensus and averaging on gossip digraphs. IEEE Transactions on Automatic Control, 56(9):2087–2100, September 2011.

[7] R. Carli, F. Fagnani, A. Speranzon, and S. Zampieri. Communication constraints in the average consensus problem. Automatica, 44(3):671–684, 2008.

[8] M. E. Chamie, J. Liu, and T. Basar. Design and analysis of distributed averaging with quantized communication. IEEE Transactions on Automatic Control, 61(12):3870–3884, December 2016.

[9] T. Charalambous, Y. Yuan, T. Yang, W. Pan, C. N. Hadjicostis, and M. Johansson. Decentralised minimum-time average consensus in digraphs. Proceedings of the IEEE Conference on Decision and Control, pages 2617–2622, 2013.
| Number of Iterations (k) | Nodes State Variables (qs[j][k]) |
|-------------------------|----------------------------------|
| 0                       |                                   |
| 5                       |                                   |
| 10                      |                                   |
| 15                      |                                   |
| 20                      |                                   |
| 25                      |                                   |
| 30                      |                                   |
| 35                      |                                   |
| 40                      |                                   |
| 45                      |                                   |
| 50                      |                                   |

Fig. 4. Comparison between Algorithm 1 (A), the distributed averaging algorithm with quantized communication in [35] (B), the distributed averaging algorithm with quantized communication in [36] (C), the distributed averaging algorithm with quantized communication in [8] (D), the quantized asymmetric averaging algorithm in [6] (E), and the quantized gossip algorithm in [18] (F), averaged over 1000 randomly generated strongly connected digraphs of 20 nodes each.

[9x516]0 20 40 60 80 100 120 140 160 180 200
Number of Iterations (k)
0
5
10
15
20
25
30
35
40
45
50

Node States Variables (qs[j][k])

| Number of Iterations (k) | Nodes State Variables (qs[j][k]) |
|-------------------------|----------------------------------|
| 50 55 60                | 25.996                           |
|                          | 25.998                           |
|                          | 26                               |
|                          | 26.002                           |
|                          | 26.004                           |

[44x115]

S. Kar and J. M. F. Moura. Distributed consensus algorithms

C. N. Hadjicostis, A. D. Domínguez-García, and

A. Kashyap, T. Basar, and R. Srikant. Quantized consensus.

E. Garcia, Y. Cao, H. Yuc, P. Antsaklis, and D. Casbeer.

S. Etesami and T. Basar. Convergence time for unbiased

A. Elgabli, J. Park, A. S. Bedi, C. B. Issaid, M. Bennis, and V. Aggarwal. Q-GADMM: Quantized group ADMM for

J. Lavaei and R. M. Murray. Quantized consensus by means of

T. Li, M. Fu, L. Xie, and J. F. Zhang. Distributed consensus with limited communication data rate.

J. Liu, S. Mou, A. S. Morse, B. D. O. Anderson, and C. Yu. Deterministic gossiping. Proceedings of the IEEE, 99(9):1505–1524, September 2011.

Z. Liu, Z. Chen, and Z. Yuan. Event-triggered average-consensus of multi-agent systems with weighted and direct
topology. Journal of Systems Science and Complexity, 25(5):845–855, October 2012.

N. Lynch. Distributed Algorithms. CA: Morgan Kaufmann Publishers, San Mateo, 1996.

S. Mou, E. Garcia, and D. W. Casbeer. Distributed algorithms for the average bridge consensus. Proceedings of the IEEE Conference on Control Technology and Applications (CCTA), pages 1710–1715, 2017.

A. Nedic, A. Olshevsky, A. Ozdaglar, and J. N. Tsitsiklis. Distributed subgradient methods and quantization effects. Proceedings of the IEEE Conference on Decision and Control, pages 4177–4184, 2008.

A. Nedic, A. Olshevsky, A. Ozdaglar, and J.N. Tsitsiklis. On distributed averaging algorithms and quantization effects. IEEE Transactions on Automatic Control, 54(11):2506–2517, November 2009.

C. Nowzari and J. Cortés. Distributed event-triggered coordination for average consensus on weight-balanced
digraphs. Automatica, 68:237–244, June 2016.
[29] R. Olfati-Saber and R.M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, September 2004.

[30] M. G. Rabbat and R. D. Nowak. Quantized incremental algorithms for distributed optimization. *IEEE Journal on Selected Areas in Communications*, 23(4):798–808, 2005.

[31] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani. FedPAQ: A communication-efficient federated learning method with periodic averaging and quantization. *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*, pages 2021–2031, 2020.

[32] A. I. Rikos, T. Charalambous, and C. N. Hadjicostis. Distributed weight balancing over digraphs. *IEEE Transactions on Control of Network Systems*, 1(2):190–201, June 2014.

[33] A. I. Rikos, T. Charalambous, K. H. Johansson, and C. N. Hadjicostis. Privacy-preserving event-triggered quantized average consensus. *Proceedings of the IEEE Conference on Decision and Control*, pages 6246–6253, 2020.

[34] A. I. Rikos and C. N. Hadjicostis. Distributed average consensus under quantized communication via event-triggered mass summation. *Proceedings of the IEEE Conference on Decision and Control*, pages 894–899, 2018.

[35] A. I. Rikos and C. N. Hadjicostis. Distributed average consensus under quantized communication via event-triggered mass splitting. *Proceedings of the 20th IFAC World Congress*, pages 3019–3024, 2020.

[36] A. I. Rikos and C. N. Hadjicostis. Event-triggered quantized average consensus via ratios of accumulated values. *IEEE Transactions on Automatic Control*, 66(3):1293–1300, 2021.

[37] M. Ruan, H. Gao, and Y. Wang. Secure and privacy-preserving consensus. *IEEE Transactions on Automatic Control*, 64(10):4035–4049, 2019.

[38] L. Schenato and G. Gamba. A distributed consensus protocol for clock synchronization in wireless sensor network. *Proceedings of the IEEE Conference on Decision and Control*, pages 2289–2294, 2007.

[39] Georg S. Seyboth, Dimos V. Dimarogonas, and Karl H. Johansson. Event-based broadcasting for multi-agent average consensus. *Automatica*, 49(1):245–252, January 2013.

[40] N. Shlezinger, M. Chen, Y. C. Eldar, H. V. Poor, and S. Cui. Federated learning with quantization constraints. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8851–8855, 2020.

[41] J. Sun, T. Chen, G. Giannakis, and Z. Yang. Communication-efficient distributed learning via lazily aggregated quantized gradients. *Advances in Neural Information Processing System*, pages 3370–3380, 2019.

[42] J. Sun, T. Chen, G. B. Giannakis, Q. Yang, and Z. Yang. Lazily aggregated quantized gradient innovation for communication-efficient federated learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020 (Early Access).

[43] S. Sundaram and C. N. Hadjicostis. Distributed function calculation and consensus using linear iterative strategies. *IEEE Journal on Selected Areas in Communications*, 26(4):650–660, May 2008.

[44] D. Thanou, E. Kokiopeoulou, Y. Pu, and P. Frossard. Distributed average consensus with quantization refinement. *IEEE Transactions on Signal Processing*, 61(1):194–205, January 2013.

[45] L. Xiao, S. Boyd, and S. Lall. A scheme for robust distributed sensor fusion based on average consensus. *Proceedings of the International Symposium on Information Processing in Sensor Networks*, pages 63–70, 2005.