European Journal of Control 74 (2023) 100848

Contents lists available at ScienceDirect

European Journal of Control

journal homepage: www.elsevier.com/locate/ejcon

Distributed computation of exact average degree and network size in finite time under quantized communication

Apostolos I. Rikos[a], Themistoklis Charalambous[b,*], Christoforos N. Hadjicostis[b], Karl H. Johansson[a]

[a] Division of Decision and Control Systems, KTH Royal Institute of Technology, Stockholm SE-100 44, Sweden
[b] Department of Electrical and Computer Engineering, University of Cyprus, Nicosia 1678, Cyprus

A R T I C L E I N F O

Article history:
Received 14 May 2023
Accepted 8 June 2023
Available online 21 June 2023

Recommended by Prof. T Parisini

Keywords:
Distributed networks
Quantized communication
Size estimation
Averge degree calculation
Leader election
Finite time
Markov chains

A B S T R A C T

We consider the problems of computing the average degree and the size of a given network in a distributed fashion and under quantized communication. More specifically, we present two distributed algorithms, which rely on quantized operation (i.e., nodes process and transmit quantized messages) and are able to obtain the exact solutions in a finite number of steps. During the operation of our algorithms, each node can determine in a distributed manner whether convergence has been achieved and correspondingly terminate its operation. For terminating the operation of our algorithms, we assume a known bound for the network diameter. To the best of the authors' knowledge, these algorithms are the first to find exact solutions (i.e., with no error in the final result) under quantized communication. Note that our network size calculation algorithm is the first in the literature to calculate the exact size of a network in a finite number of steps without introducing a final error; in other algorithms, this error can be either due to quantization or asymptotic convergence. In our case, no error is introduced since the desired result is calculated in the form of a fraction involving an integer numerator and an integer denominator. We demonstrate the operation of our algorithms and their potential advantages through simulations.

1. Introduction

In multi-agent systems, many distributed algorithms require knowledge of the network's parameters, such as the average degree and/or the size of the network. Various applications rely on knowledge of the average node degree or network size, including consensus-based distributed optimization [22], infection propagation strategies [17], antidote distribution to control epidemics [4], and the networked prisoner's dilemma game [32]. Additionally, knowing the network's parameters is crucial for detecting (i) topological changes [19], (ii) node criticality/importance [8], (iii) certain types of network attacks such as node or link insertion attacks[31], and (iv) communities [12]. Moreover, knowledge of network parameters can help to estimate the maximum and the minimum of the initial measurements in the presence of noise via a soft-max operation [34], enable distributed clustering [23], and facilitate the control renewable energy resources while maintaining an average degree that fulfils structural properties [7].

Various methods have been proposed in the literature for calculating the size of a given network. Current approaches rely on statistical methods that require the exchange of excessive information between nodes, random walk strategies, random sampling, and capture-recapture strategies [9,14,15,18,21,25,33]. Furthermore, the problem of calculating the average degree of a given network has been analyzed in [6,11,26]; however, finding its solution in a distributed fashion has received limited attention. Specifically, only [30] presents an asymptotic distributed algorithm for computing the average degree of a network. A byproduct of our proposed size calculation algorithm is a novel method for distributively electing a leader, which is a fundamental problem in distributed computing [20]. Recent related works focus on population protocols, Byzantine leader election strategies, complexity analysis, and election with minimum failure rate [1–3,16]. To the best of the authors' knowledge, existing algorithms to calculate the size, or the average degree of a network, or to elect a leader node, operate with real values and/or exhibit asymptotic convergence. When working with real numbers, high bandwidth channels are needed because these numbers require a large number of bits to be represented accurately. In practice, this can be a significant challenge, particularly when dealing with large-scale networks. The necessity for high-bandwidth communication can also create operational bottlenecks

* Corresponding author.
E-mail addresses: rikos@kth.se (A.I. Rikos), charalambous.themistoklis@ucy.ac.cy (T. Charalambous), hadjicostis.christoforos@ucy.ac.cy (C.N. Hadjicostis), kallej@kth.se (K.H. Johansson).

https://doi.org/10.1016/j.ejcon.2023.100848
0947-3580/© 2023 European Control Association. Published by Elsevier Ltd. All rights reserved.

and increase the communication overhead of each node. Furthermore, asymptotic convergence introduces a final error on the calculated result because most algorithms need to be terminated after a pre-defined finite number of iterations. This error leads to imprecise calculation of the desired quantity, which may be of significant magnitude in the case of large scale networks. In contrast, calculating network parameters and/or electing a leader node in a finite number of steps with quantized communication (as done in this paper) remains largely unexplored. In this work, we demonstrate that it is possible to distributively compute rational numbers (such as the average degree and number of nodes) and elect a leader node in a network without relying on algorithms designed to operate with real numbers, thus reducing the need for high-bandwidth communication.

Main Contributions. Our paper is a major departure from the current literature and aims to bridge the gap between theoretical approaches and practical needs. Compared to existing approaches, the operation of our algorithms allows nodes to process and transmit quantized values. This is significant because quantized operation allows more efficient use of network resources compared to real-valued operation. The use of quantized values can result in significant reductions in communication and storage requirements, which can make distributed algorithms more efficient and scalable. Specifically, nodes require less bits to store and transmit information, which reduces network congestion and energy consumption. Furthermore, compared to algorithms that exhibit asymptotic convergence, our algorithms converge in a finite number of steps without introducing any final error, as nodes compute the final result in the form of a quantized fraction. This is an important property for practical applications, where it is essential to obtain accurate results in a timely manner. More specifically, in applications where the accuracy of the result is critical, waiting for an algorithm to converge asymptotically may not be practical. The finite-time convergence property enables our algorithms to be used in time-sensitive applications, where obtaining an accurate result quickly is of utmost importance. The main contributions are the following.

- We present a novel distributed algorithm for computing the average degree of a given network. Our algorithm operates with quantized values and is able to calculate the exact result without any error; see Algorithm 1.
- We present a novel distributed algorithm for computing the size of a given network. Our algorithm operates with quantized values and calculates the exact result without introducing any final error; see Algorithm 2. Furthermore, the algorithm's operation relies on the election of a leader node (if a leader is not already assigned/decided). For this reason, we present a novel strategy for leader election with quantized processing and communication. Note that this is the first leader election strategy which relies on quantized operation; see Algorithm 3. We show that the leader election strategy achieves its objective (i.e., the election of one leader node) after a small number of time steps, with high probability; see Theorem 3. Note that when a leader is not already assigned, the algorithm converges to the exact result with high probability, since the leader election process is successful with high probability.
- We show that both our algorithms complete in finite time, and we provide upper bounds on the number of time steps needed for completion. Our provided bounds rely on a known upper bound on the diameter rather than the size of the network.
- Both algorithms utilize a distributed stopping strategy which allows nodes to determine whether completion has been reached, and thus terminate their operation. For implementing this strategy, we assume that each node has knowledge of an upper bound on the network diameter.

The main advantage of our proposed algorithms is that each node's state is represented as a quantized fraction. This offers improved communication efficiency and reduced memory requirements. The numerator of the initial fraction is the nodes initial state value (which in our case is an integer value) and the denominator is equal to one. Then, each node transmits the initial fraction to a randomly chosen neighbor node. If two or more fractions are transmitted to the same node, then the receiving node sums separately the received numerators and denominators, and forms a new fraction. In our case, throughout this process each fraction is represented by an integer numerator and an integer denominator; however, more generally, the numerator and the denominator could be any two quantized values. In this way, after a finite number of time steps each node will receive a fraction whose numerator is the sum of each node's initial state value and the denominator is equal to the number of nodes in the network. This finite-time convergence characteristic is essential for practical applications where timely and accurate results are crucial.

Compared to our prior works in [27,28], our proposed algorithms exhibit significant differences. First, both algorithms presented in this paper incorporate distributed stopping strategies to ensure that nodes terminate their operations upon completion, thereby conserving network resources (this feature is abscent in [28]). Second, our algorithm for computing the network size deploys a new strategy for leader election that involves quantized processing and communication (this feature is abscent in [27,28]). This strategy enables our algorithm to converge to the correct result. In summary, our algorithms are better suited for networks with limited resources (since they ensure operation termination), and applications where timely and accurate results are critical (due to the leader election strategy and ability to terminate).

2. Notation and Preliminaries

The sets of real, rational, and integer numbers are denoted by $\mathbb{R}, \mathbb{Q}$, and $\mathbb{Z}$, respectively. The symbol $\mathbb{Z}_{\geq 0}$ ($\mathbb{Z}_{>0}$) denotes the set of nonnegative (positive) integer numbers (similarly, $\mathbb{Z}_{\leq 0}$ and $\mathbb{Z}_{<0}$). Vectors are denoted by small letters, matrices are denoted by capital letters and the transpose of a matrix A is denoted by A^T. For a matrix $A \in \mathbb{R}^{n \times n}$, the entry at row i and column j is denoted by A_{ij}. By $\mathbf{1}$ we denote the all-ones vector and by I we denote the identity matrix (of appropriate dimensions).

Graph-Theoretic Notions. Consider a network of n ($n \geq 2$) nodes communicating only with their immediate neighbors. The communication topology is captured by a directed graph (digraph) defined as $\mathcal{G}_d = (\mathcal{V}, \mathcal{E})$. In digraph $\mathcal{G}_d$, $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes, whose cardinality is denoted as $n = |\mathcal{V}| \geq 2$, 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) whose cardinality is denoted as $m = |\mathcal{E}|$. 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 assume that the given digraph $\mathcal{G}_d = (\mathcal{V}, \mathcal{E})$ is *strongly connected*. This means that for each pair of nodes $v_j, v_i \in \mathcal{V}$, $v_j \neq v_i$, there exists a directed *path*[1] from v_i to v_j. Furthermore, the diameter D of a digraph is the longest shortest path between any two nodes $v_j, v_i \in \mathcal{V}$ in the network. 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 $\mathcal{N}_j^- = \{v_i \in \mathcal{V} \mid (v_j, v_i) \in \mathcal{E}\}$. The cardinality of $\mathcal{N}_j^-$ is called the *in-degree* of v_j and is denoted by $\mathcal{D}_j^-$. The subset of nodes that can

[1] A directed *path* from v_i to v_j exists if we can find a sequence of nodes $v_i \equiv v_{l_0}, v_{l_1}, \ldots, v_{l_t} \equiv v_j$ such that $(v_{l_{\tau+1}}, v_{l_\tau}) \in \mathcal{E}$ for $\tau = 0, 1, \ldots, t-1$.

directly receive information from node v_j is called the set of out-neighbors of v_j and is represented by $\mathcal{N}_j^+ = \{v_l \in \mathcal{V} \mid (v_l, v_j) \in \mathcal{E}\}$. The cardinality of $\mathcal{N}_j^+$ is called the *out-degree* of v_j and is denoted by $\mathcal{D}_j^+$.

Node Operation. The operation of each node $v_j \in \mathcal{V}$ respects the quantization of information flow. At time step $k \in \mathbb{Z}_{>0}$, each node v_j maintains the mass variables $y_j[k] \in \mathbb{Z}$ and $z_j[k] \in \mathbb{Z}_{\geq 0}$, which are used to communicate with other nodes. The state variables $y_j^s \in \mathbb{Z}$, $z_j^s \in \mathbb{Z}_{>0}$ and $q_j^s \in \mathbb{Q}$, (where $q_j^s = \frac{y_j^s}{z_j^s}$) are used to store the received messages. The voting variables m_j and M_j are used to determine whether convergence has been achieved (thus, the nodes can stop their operation). Furthermore, we assume that each node v_j is aware of its out-neighbors and can directly transmit messages to each out-neighbor separately. In order to randomly determine which out-neighbor to transmit to, each node v_j assigns a nonzero probability b_{lj} to each of its outgoing edges $v_l \in \mathcal{N}_j^+$. For every node, this probability assignment can be captured by an $n \times n$ column stochastic matrix $\mathcal{B} = [b_{lj}]$. A simple choice is to set these probabilities to be equal, i.e.,

$$b_{lj} = \begin{cases} \frac{1}{1+\mathcal{D}_j^+}, & \text{if } l = j \text{ or } v_l \in \mathcal{N}_j^+, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Each nonzero entry b_{lj} of matrix $\mathcal{B}$ represents the probability of node v_j transmitting towards out-neighbor $v_l \in \mathcal{N}_j^+$. Note that in (1) each node assigns a nonzero probability to a virtual self-edge. As long as the underlying digraph $\mathcal{G}_d$ is strongly connected, this means that the stochastic matrix $\mathcal{B}$ is primitive and has a unique eigenvector associated with the eigenvalue 1. This characteristic is essential for the development of the results in this paper. If every node does not have a virtual self-edge then $\mathcal{B}$ is not necessarily primitive, which could cause problems for our proposed algorithms.

2.1. *Synchronous* max/min - *Consensus*

The max-consensus algorithm computes the maximum value of the network in a finite number of time steps in a distributed fashion [5]. For every node $v_j \in \mathcal{V}$, if the updates of each node's state are synchronous, then the update rule is: $x_j[k+1] = \max_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} \{x_i[k]\}$, for $k = 0, 1, \ldots$. It has been shown (see, e.g., [10, Theorem 5.4]) that the max-consensus algorithm converges to the maximum value among all nodes in a finite number of steps s, where $s \leq D$ (i.e., $x_j[s] = \max_{v_l \in \mathcal{V}} \{x_l[0]\}$, for all nodes $v_j \in \mathcal{V}$). Similar results hold for the min-consensus algorithm.

2.2. *Quantized average consensus*

The objective of quantized average consensus problems is the development of distributed average consensus algorithms which allow nodes to process and transmit quantized information. During their operation, each node utilizes short communication packages and eventually obtains after a finite number of time steps a state q^s in the form of a quantized fraction, which is equal to the *exact* real average q of the initial states. Note that in this paper we consider the case where quantized values are represented by integer[2] numbers. As we will see, this does not impose a restriction since all initial and computed values in our algorithms are integers.

Since each node processes and transmits quantized information, we adopt the algorithm in [28]. This algorithm, which is preliminary for the results in this paper, allows nodes to achieve quantized average consensus after a finite number of time steps. The operation of the algorithm presented in [28], assumes that each node v_j in the network has an integer initial state $y_j[1] \in \mathbb{Z}$. At initialization, each node v_j assigns a nonzero probability to each outgoing edge and to a virtual self-edge as in (1). At each time step k, each node $v_j \in \mathcal{V}$ maintains its mass variables $y_j[k], z_j[k]$, and its state variables $y_j^s[k], z_j^s[k], q_j^s[k]$. It updates the mass variables as

$$y_j[k+1] = y_j[k] + \sum_{v_i \in \mathcal{N}_j^-} \mathbb{1}_{ji}[k] y_i[k], \tag{2a}$$

$$z_j[k+1] = z_j[k] + \sum_{v_i \in \mathcal{N}_j^-} \mathbb{1}_{ji}[k] z_i[k], \tag{2b}$$

where $\mathbb{1}_{ji}[k] = 1$ if a message is received at v_j from v_i at k (0 otherwise). If the following condition holds:

(C1): $z_j[k+1] \geq 1$,

then, node v_j updates its state variables as

$$z_j^s[k+1] = z_j[k+1], \tag{3a}$$

$$y_j^s[k+1] = y_j[k+1], \tag{3b}$$

$$q_j^s[k+1] = \frac{y_j^s[k+1]}{z_j^s[k+1]}. \tag{3c}$$

Then, it transmits its mass variables $z_j[k+1]$, $y_j[k+1]$ to one randomly selected out-neighbor or to itself according to (1). If it transmits its mass variables, it sets them equal to zero (i.e., $z_j[k+1] = 0$, $y_j[k+1] = 0$). Finally, it receives the values $y_i[k]$ and $z_i[k]$ from its in-neighbors $v_i \in \mathcal{N}_j^+$, it performs the calculations in (2a), (2b), and repeats the operation.

Definition 1. The system achieves *exact* quantized average consensus if there exists $k_0 \in \mathbb{Z}_{>0}$ so that for every $v_j \in \mathcal{V}$, we have $y_j^s[k] = \sum_{l=1}^n y_l[1]$ and $z_j^s[k] = n$, which means that $q_j^s[k] = \frac{\sum_{l=1}^n y_l[1]}{n} = q$, for $k \geq k_0$. Notice that q is the desirable (real) average of the initial states.

The following result from [28] analyzes the convergence of the quantized average consensus algorithm.

Theorem 1 ([28]). *Consider a strongly connected digraph $\mathcal{G}_d = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ nodes and $m = |\mathcal{E}|$ edges and $z_j[1] = 1$ and $y_j[1] \in \mathbb{Z}$ for every node $v_j \in \mathcal{V}$ at time step $k = 1$. Suppose that each node $v_j \in \mathcal{V}$ follows the Initialization and Iteration steps as described in the algorithm in [28]. We can find $k_0 \in \mathbb{N}$, so that for every $k \geq k_0$ we have $y_j^s[k] = \sum_{l=1}^n y_l[1]$ and $z_j^s[k] = n$, with probability arbitrarily close to 1. This means that $q_j^s[k] = \frac{\sum_{l=1}^n y_l[1]}{n}$, for every $v_j \in \mathcal{V}$.*

Remark 1. The operation of [28] can be be interpreted as the "random walk" of n "tokens" in a Markov chain. Each token has a pair of values y, z. If a token visits a node, then the state variables of the node become equal to the y, z variables of the token. Also, if two (or more) tokens visit the same node at the same time step k, they "merge" to a new single token (i.e., their y values sum to the new y value, and their z values sum to the new z value). The new token continues to perform a random walk in a Markov chain. This means that, after a finite number of time steps, all n tokens merge to a final single token, and this token has values y, z whose ratio y/z is equal to the desired result. Thus, this final single token will visit every node in the network (because it performs a random walk), and the state variables of each node will become equal to the desired result.

[2] Following [13] we assume that the state of each node is integer valued. This abstraction subsumes a class of quantization effects (e.g., uniform quantization).

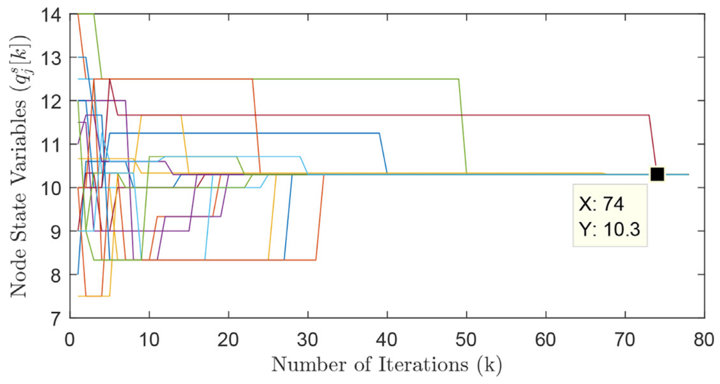

Fig. 1. Execution of Algorithm 1 over a random digraph of 20 nodes with diameter $D = 3$.

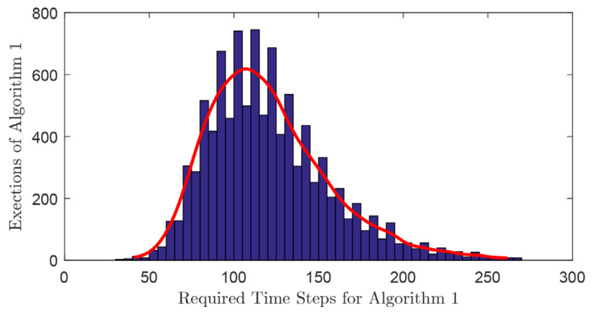

Fig. 2. Required time steps for completion of 10000 executions of Algorithm 1 over a random digraph of 20 nodes with diameter $D = 3$.

3. Problem formulation

Consider a network modelled as a directed graph $\mathcal{G}_d = (\mathcal{V}, \mathcal{E})$. In this paper, we develop a distributed algorithm that allows nodes to address problems **P1** and **P2** presented below, while processing and transmitting *quantized* information via available communication links.

P1. Average degree estimation: After a finite number of time steps, each node v_j obtains a fraction q^s_j that is equal to the average degree of the network. Specifically, q^s_j is equal to:

$$q^s_j = \frac{\sum_{l=1}^{n} \mathcal{D}^+_l}{n} \tag{4}$$

where $\mathcal{D}^+_l$ is the out-degree of node v_l, and n is the total number of nodes in the network. Each node v_j processes and transmits quantized values, and stops transmitting once (4) holds for every node.

P2. Network size estimation: After a finite number of time steps, each node v_j obtains a value z^s_j that is equal to the number of nodes in the network. Specifically, z^s_j is equal to:

$$z^s_j = n \tag{5}$$

where n is the total number of nodes in the network. Each node v_j processes and transmits quantized values, and stops transmitting once (5) holds for every node.

4. Distributed average degree computation

In this section we present a distributed algorithm which solves problem **P1**. Our algorithm is detailed below as Algorithm 1. For solving the problem in a distributed manner we make the following assumption.

Assumption 1. An upper bound D' of the diameter of the network D is known to all nodes $v_j \in \mathcal{V}$.

Assumption 1 is necessary for coordinating min- and max-consensus algorithms, as described later. It is important to note here that the feasibility of knowing the network diameter in real-world applications depends on the specific application and the available network information. In some cases, the network diameter may be explicitly known or can be estimated via distributed algorithms such as [24]. More specifically, [24] can be executed as an initialization step of our algorithm.

We now describe the main operations of Algorithm 1.

Initialization. Each node $v_j \in \mathcal{V}$ assigns a nonzero probability to each outgoing edge and a virtual self-edge, so that the sum of the nonzero probabilities is equal to one (an example is shown in (1)). It sets its mass variable $y_j[1]$ to be equal to the node's out-degree and its mass variable $z_j[1]$ to be equal to one. Also it sets its initial state variables $y^s_j[1]$, $z^s_j[1]$ to be equal to the initial mass variables $y_j[1]$, $z_j[1]$, respectively, and the state variable $q^s_j[1]$ to be equal to the fraction $y^s_j[1]/z^s_j[1]$. Then, it transmits its mass variables to a randomly chosen out-neighbor (or itself) and sets them equal to zero (unless the transmission was towards itself).

Iteration-Step 1. Calculating the Average Network Degree: This step can be executed in an asynchronous fashion. Every node v_j receives the transmitted mass variables from its in-neighbors, and sums them with its stored mass variables to obtain new mass variables $y_j[k+1]$, $z_j[k+1]$. Then, if its mass variable $z_j[k+1]$ is nonzero, (i) it updates its state variables to be equal to the mass variables, and (ii) it chooses randomly an out-neighbor (or itself) and transmits the mass variables $y_j[k+1]$ and $z_j[k+1]$. Eventually, after a finite number of time steps, the ratio of state variables $y^s_j[k+1]/z^s_j[k+1]$ of each node v_j is equal to the average degree in the network.

Iteration-Step 2. Distributed Stopping: This step is executed in a synchronous fashion. Every $k = tD' + 1$ time steps, where $t \in \mathbb{N}$, each node v_j sets its voting variables m_j and M_j to be equal to the fraction $y^s_j[k]/z^s_j[k]$ of the state variables. It broadcasts its voting variables to its out-neighbors and receives the corresponding m_i and M_i from its in-neighbors $v_i \in \mathcal{N}^-_j$. It stores the min and max among all received and its own voting values to the variables m_j and M_j, respectively. The min- and max-consensus algorithms are performed for $D' - 1$ steps. When $k = (t+1)D'$ time steps, each node v_j checks whether m_j, M_j have equal values; if this holds, then every node terminates the operation of the algorithm. If not, the process continues, with step $k = (t+1)D' + 1$, in which each node v_j sets its voting variables m_j and M_j to be equal to the fraction $y^s_j[k]/z^s_j[k]$ of the state variables, and the min- and max-consensus algorithms are restarted. Note that m_j and M_j are fractions of integers. Therefore, for both the min- and max-consensus, each node v_j at time step k sends a pair of integer values $(y^s_j[k], z^s_j[k])$ whose ratio $y^s_j[k]/z^s_j[k]$ is used to perform com-

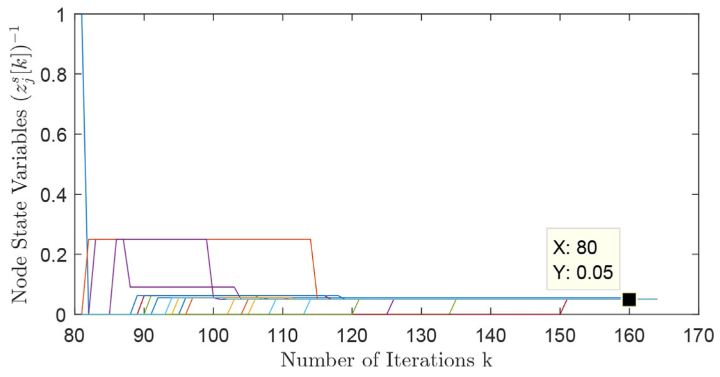

Fig. 3. Execution of Algorithm 2 over a random digraph of 20 nodes with diameter $D = 3$.

parisons during the consensus operation. Note that each node v_j, can store and compare M_j, m_j with M_i, m_i without having to convert them to real values. Specifically, let us suppose that $M_j = \frac{A_j}{B_j}$, $m_j = \frac{a_j}{b_j}$, and $M_i = \frac{A_i}{B_i}$, $m_i = \frac{a_i}{b_i}$. Each node v_j compares the fractions M_j with M_i (comparison of m_j with m_i is done similarly) by following the steps: (i) if $A_jB_i \geq B_jA_i$, then $M_j \geq M_i$ and node v_j stores M_j, (ii) else if $A_jB_i < B_jA_i$ then $M_j < M_i$ and node v_j stores M_i.

We now analyze the convergence of Algorithm 1.

Theorem 2. *Consider a strongly connected digraph $\mathcal{G}_d = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ nodes and $m = |\mathcal{E}|$ edges. At time step $k = 1$, each node v_j follows the Initialization and Iteration steps as described in Algorithm 1. For any probability p_0 (where $0 < p_0 < 1$), after $k_0 \geq (n-1)\tau' D + (n-1)\tau'' D + D'$ time steps, where*

$$\tau' \geq \left\lceil \frac{\log \varepsilon'}{\log\left(1 - \sum_{l=1}^{n} (1 + \mathcal{D}_{\max}^{+})^{-(2D)}\right)} \right\rceil,$$

for $\varepsilon' \leq 1 - 2^{\frac{\log_2 \sqrt{p_0}}{n-1}}$, and

$$\tau'' \geq \left\lceil \frac{\log \varepsilon''}{\log\left(1 - (1 + \mathcal{D}_{\max}^{+})^{-D}\right)} \right\rceil,$$

for $\varepsilon'' \leq 1 - 2^{\frac{\log_2 \sqrt{p_0}}{n-1}}$, we have that each node addresses problem ***P1*** *in Section 3 with probability at least p_0.*

Proof. See Appendix Appendix A. □

5. Distributed Network Size Computation

In this section we present a distributed algorithm which solves problem **P2**. Our algorithm is detailed below as Algorithm 2. Note here that Assumption 1 also holds during the operation of the proposed algorithm and we additionally make the following assumption.

Assumption 2. All nodes $v_j \in \mathcal{V}$ have knowledge of a constant $U_v \in \mathbb{N}$.

Assumption 2 is necessary for executing the max-consensus algorithm in order to elect a single leader node with high probability. This can be a preset value for executing the protocol, irrespective of the network (i.e., the constant U_v is simply a parameter that is used by the algorithm, and it is assumed that all nodes have access to this parameter.). Alternatively, U_v can be obtained during initialization by having each node choose a constant and executing a max-consensus algorithm for D' time steps. The resulting value from the consensus algorithm can then be used as the common U_v value across all nodes.

We now describe the main operations of Algorithm 2.

Initialization-Step 1. Probability Assignment: Each node $v_j \in \mathcal{V}$ assigns a nonzero probability to each outgoing edge and a virtual self edge, so that the sum of nonzero probabilities is equal to one (an example is shown in (1)).

Initialization-Step 2. Leader Election: This step is executed in a synchronous fashion. Each node executes Algorithm 3. During its operation, each node in the network executes a max-consensus for $U_v D'$ time steps (i.e., it executes a max-consensus algorithm U_v times). More specifically, each node randomly picks a nonnegative integer from the set $\{0, 1, \ldots, Up_j\}$, and executes the first max-consensus for D' time steps. Once the first max-consensus completes, the node (or nodes) that picked the maximum value pick again randomly a nonnegative integer, whereas the node (or nodes) that did not pick the maximum value choose a value equal to -1. Then, the max-consensus is executed again for D' time steps. This process is repeated U_v times (i.e., for a total $U_v D'$ time steps). After the execution of Algorithm 3, we have that one node $v_j \in \mathcal{V}$ is the leader (i.e., $\text{flag}_j^{\text{ld}} = 1$) and every node $v_i \in \mathcal{V} \setminus \{v_j\}$ is a follower (i.e., $\text{flag}_i^{\text{ld}} = 0$), with high probability.

Initialization-Step 3. Initialization of Mass and State Variables: Each node initializes its mass variables and its state variables according to the result of Algorithm 3 (i.e., the leader node initializes its mass variables different than the follower nodes). Specifically, the leader node initializes both y and z to be equal to 1. Every follower node initializes y to be equal to 0 and z to be equal to 1. Then, every node sets its state variables to be equal to the mass variables.

Iteration-Step 1. Calculating the Network Size: This step can be executed in an asynchronous fashion. Every node v_j receives the transmitted mass variables of its in-neighbors, and sums them with the stored mass variables. Then, if its mass variable $z_j[k+1]$ is nonzero, (i) it updates its state variables to be equal to the mass variables, and (ii) it chooses randomly an out-neighbor (or itself) and transmits the mass variables $y_j[k+1]$ and $z_j[k+1]$ (if it transmits its mass variables towards an out-neighbor, it sets them to zero). Eventually, after a finite number of time steps, the state variable $z_j^s[k+1]$ of each node v_j is equal to the number of nodes in the network.

Iteration-Step 2. Distributed Stopping: The operation of this step is identical to "Iteration-Step 2. Distributed Stopping" of Algorithm 1. It is omitted due to space considerations.

Remark 2. Note that the execution of Algorithm 3 as an initialization step of Algorithm 2 is essential for Algorithm 2 to converge to the correct solution. For example, let us consider the scenario where we do not execute Algorithm 3. Instead we execute Algorithm 2 with Initialization steps 1, 2.2, 3, 4. Then, it is possible that Algorithm 2 terminates its operation before calculating the correct result. More specifically, let us consider a network with n nodes, where n is an even number. At time step $1 + (a-1)D'$ (where a is a natural number, and

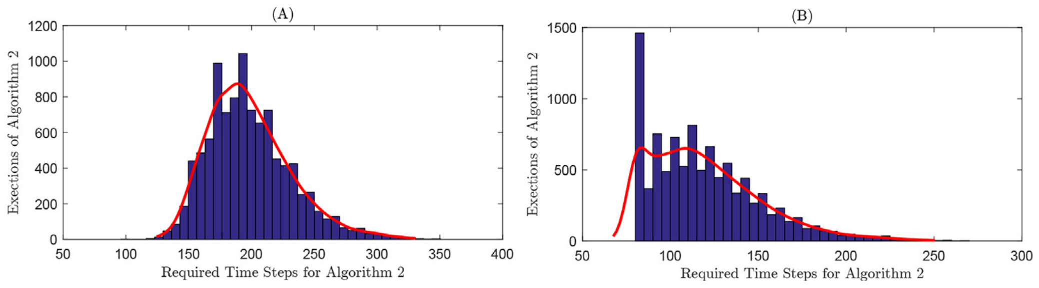

Fig. 4. Required time steps for convergence of 10000 executions of Algorithm 2 over a random digraph of 20 nodes with diameter $D = 3$. (A) Execution of Algorithm 3 before Algorithm 2. (B) Execution of Algorithm 3 in parallel with Algorithm 2.

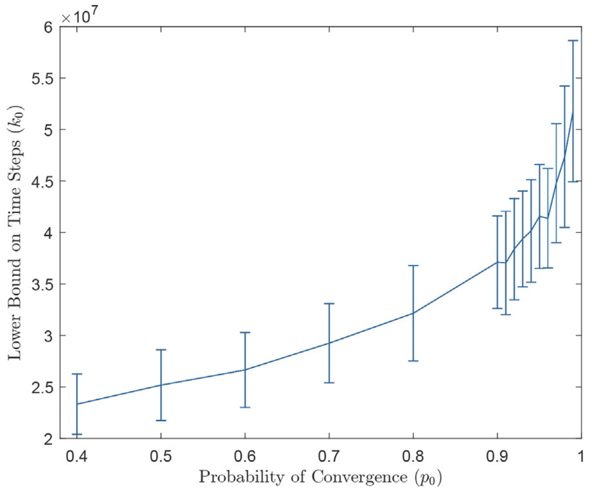

Fig. 5. Lower bounds on the required number of time steps for convergence of Algorithm 1, and Algorithm 2 for different probability values, according to Theorem 2, and Theorem 3.

$a > 1$) we have that $y_j[1 + (a-1)D'] = 0$, $z_j[1 + (a-1)D'] = n/2$, and $y_l[1 + (a-1)D'] = 0$, $z_l[1 + (a-1)D'] = n/2$, for nodes v_l, v_j, respectively. Also, $y_i[1 + (a-1)D'] = 0$, $z_i[1 + (a-1)D'] = 0$, for every $v_i \in \mathcal{V} \setminus \{v_j, v_l\}$. Furthermore, let us assume that $z_i^s[1 + (a-1)D'] = n/2$, $y_i^s[1 + (a-1)D'] = 0$ for every $v_i \in \mathcal{V} \setminus \{v_j, v_l\}$. Let us suppose now that at time step aD', we have $y_{j'}[aD'] = 0$, $z_{j'}[aD'] = n/2$, and $y_{l'}[aD'] = 0$, $z_{l'}[aD'] = n/2$, for nodes $v_{j'}, v_{l'}$, respectively. This means that Algorithm 2, will terminate its operation at time step aD' (since $M_j = m_j$ for every node). However, the state variable z^s of every node is not equal to the size of the network. On the contrary, the execution of Algorithm 3 guarantees that Algorithm 2 will terminate its operation only when the state variable z^s for every node is equal to the network size.

We now analyze the convergence of Algorithm 2.

Theorem 3. *Consider a strongly connected digraph $\mathcal{G}_d = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ nodes and $m = |\mathcal{E}|$ edges. At time step $k = 1$, each node v_j follows the Initialization and Iteration steps as described in Algorithm 2. For any probability p_0 (where $0 < p_0 < 1$), after $k_0 \geq (n-1)\tau' D + (n-1)\tau'' D + D'$ time steps (where τ', τ'' are defined in Theorem 2), every node v_j addresses problem **P2** in Section 3 with probability at least p_0.*

Proof. See Appendix Appendix B. □

Remark 3. Apart from guaranteeing that the exact number of nodes is computed, Algorithm 2 establishes also that the process is terminated (so that other algorithms can be initiated after the completion of Algorithm 2). If operation termination was not required, then a variation of Algorithm 2 could be used to calculate the network size in a finite number of steps without electing a leader node. Specifically, in this variation each node v_j initializes $z_j[1] = 1$ and executes iteration steps 2–6 of Algorithm 2. During this execution, there exists k_0, for which $z_j^s[k] = n$ for every $v_j \in \mathcal{V}$, for $k \geq k_0$.

Remark 4. Algorithm 3 is executed as an initialization step of Algorithm 2. However, both can be executed in parallel. The strategy of parallel execution significantly decreases the required number of time steps for convergence of Algorithm 2 as demonstrated in Section 6 (see Fig. 4).

6. Simulation results

In this section, we present simulation results in order to demonstrate the operation of our proposed algorithms and their potential advantages. For both algorithms we focus on a random digraph of 20 nodes and show how the nodes' states converge to the desired value in finite time. We also demonstrate the distributed stopping capabilities of our algorithms. Finally, we analyze numerically the completion times of our algorithms, by presenting histograms of the required number of time steps for completion over several runs of the algorithm. Our simulations emphasize the novelty of our algorithms which, to the best of our knowledge, are the first that use quantized values to calculate the exact values of the average degree and size of a network while also providing strong theoretical guarantees. Note here that comparing against other algorithms from current literature is not feasible due to the fact that all other methods in the literature operate with real-values and exhibit asymptotic convergence.

Average Degree Computation of a Random Network of 20 Nodes. In this section we demonstrate the operation of Algorithm 1 over a random digraph of 20 nodes. Each edge of the digraph was generated with probability 0.5, and the diameter is equal to $D = 3$. The average degree of the digraph is equal to $\frac{206}{20} = 10.3$. During the execution of Algorithm 1, each node has knowledge of an upper bound on the network's diameter equal to $D' = 4$.

In Fig. 1, we plot the evolution of the state variable $q_j^s[k]$ of every node v_j. We can see that Algorithm 1 converges after 74 time steps to the exact solution. Specifically, each node calculates the quantized fraction 206/20 which is equal to the average degree in the network. Furthermore, we can see that after 78 iterations each node terminates its operation since it has knowledge of $D' = 4$, which is an upper bound on the network diameter.

In Fig. 2, we present 10000 executions of Algorithm 1 over a random digraph of 20 nodes. Each edge of the digraph was generated with probability 0.5, and the diameter is equal to $D = 3$. The average number of time steps for completion of Algorithm 1 is equal to 120.96. In Fig. 2 we can see that in most cases Algorithm 1 requires 70–160 iterations for completion.

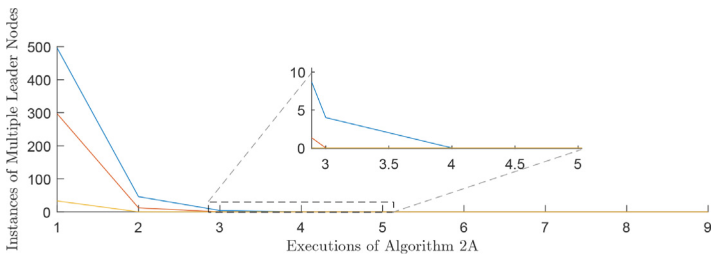

Fig. 6. Instances where multiple nodes are leader nodes during 10000 executions of Algorithm 3.

Algorithm 1 Average Degree Computation Algorithm.

```
Input: A strongly connected digraph $\mathcal{G}_d = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ nodes and $m = |\mathcal{E}|$ edges.Each node $v_j \in \mathcal{V}$ has knowledge of an upper bound $D'$ of the network diameter.
Initialization: Each node $v_j \in \mathcal{V}$:
  1) assigns a nonzero probability $b_{lj}$ to each of its outgoing edges $m_{lj}$, where $v_l \in \mathcal{N}_j^+ \cup \{v_j\}$, as follows
     $b_{lj} = \begin{cases} \frac{1}{1+\mathcal{D}_j^+}, & \text{if } l = j \text{ or } v_l \in \mathcal{N}_j^+, \\ 0, & \text{if } l \neq j \text{ and } v_l \notin \mathcal{N}_j^+, \end{cases}$
  2) sets $y_j[1] := \mathcal{D}_j^+$, $z_j[1] = 1$.
  3) sets $y_j^s[1] := y_j[1]$, $z_j^s[1] = 1$, $q_j^s[1] := y_j^s[1]/z_j^s[1]$.
  4) chooses $v_l \in \mathcal{N}_j^+ \cup \{v_j\}$ randomly according to $b_{lj}$, and transmits $y_j[1]$ and $z_j[1]$ towards $v_l$.
  5) Sets $\text{flag}_j^{\text{st}} = 0$.
Iteration: For $k = 1, 2, \dots$, each node $v_j \in \mathcal{V}$, does the following:
  • while $\text{flag}_j^{\text{st}} = 0$ then
    1) if $k \mod D' = 1$ then sets $M_j = m_j = y_j^s[k]/z_j^s[k]$;
    2) broadcasts $M_j$, $m_j$ to every $v_l \in \mathcal{N}_j^+$;
    3) receives $M_i$, $m_i$ from every $v_i \in \mathcal{N}_j^-$;
    4) sets $M_j = \max_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} M_i$, $m_j = \min_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} m_i$;
    5) receives $y_i[k]$ and $z_i[k]$ from $v_i \in \mathcal{N}_j^-$ and updates according to (2a)-(2b);
    6) if $z_j[k+1] > 1$, then
       6.1) sets $z_j^s[k+1] = z_j[k+1]$, $y_j^s[k+1] = y_j[k+1]$,$q_j^s[k+1] = \frac{y_j^s[k+1]}{z_j^s[k+1]}$ ;
       6.2) chooses $v_l \in \mathcal{N}_j^+ \cup \{v_j\}$ randomly according to $b_{lj}$, and transmits $y_j[k+1]$ and $z_j[k+1]$ towards $v_l$.
    7) if $k \mod D' = 0$ then, if $M_j = m_j$ then sets $\text{flag}_j^{\text{st}} = 1$.
Output: (4) holds for every $v_j \in \mathcal{V}$.
```

Algorithm 2 Distributed Network Size Computation Algorithm.

```
Input: A strongly connected digraph $\mathcal{G}_d = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ nodes and $m = |\mathcal{E}|$ edges.Each node $v_j \in \mathcal{V}$ has knowledge of an upper bound $D'$ of the network diameter.
Initialization: Each node $v_j \in \mathcal{V}$ does the following:
  1) Assigns a nonzero probability $b_{lj}$ to each of its outgoing edges $m_{lj}$, where $v_l \in \mathcal{N}_j^+ \cup \{v_j\}$, as follows
     $b_{lj} = \begin{cases} \frac{1}{1+\mathcal{D}_j^+}, & \text{if } l = j \text{ or } v_l \in \mathcal{N}_j^+, \\ 0, & \text{if } l \neq j \text{ and } v_l \notin \mathcal{N}_j^+. \end{cases}$
  2) Calls Algorithm 2A;
     2.1) if $\text{flag}_j^{\text{ld}} = 1$, sets $y_j[1] := 1$, $z_j[1] = 1$;
     2.2) if $\text{flag}_j^{\text{ld}} = 0$, sets $y_j[1] := 0$, $z_j[1] = 1$;
  3) Sets $y_j^s[1] := y_j[1]$, $z_j^s[1] = 1$;
  4) sets $\text{flag}_j^{\text{st}} = 0$.
Iteration: For $k = 1, 2, \dots$, each node $v_j \in \mathcal{V}$, does the following:
  • while $\text{flag}_j^{\text{st}} = 0$ then
    1) if $k \mod D' = 1$ then sets $M_j = m_j = y_j^s[k]$;
    2) broadcasts $M_j$, $m_j$ to every $v_l \in \mathcal{N}_j^+$;
    3) receives $M_i$, $m_i$ from every $v_i \in \mathcal{N}_j^-$;
    4) sets $M_j = \max_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} M_i$, $m_j = \min_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} m_i$;
    5) receives $y_i[k]$ and $z_i[k]$ from $v_i \in \mathcal{N}_j^-$ and updates according to (2a)-(2b);
    6) if $z_j[k+1] > 1$, then
       6.1) sets $z_j^s[k+1] = z_j[k+1]$, $y_j^s[k+1] = y_j[k+1]$,$q_j^s[k+1] = \frac{y_j^s[k+1]}{z_j^s[k+1]}$ ;
       6.2) chooses $v_l \in \mathcal{N}_j^+ \cup \{v_j\}$ randomly according to $b_{lj}$, and transmits $y_j[k+1]$ and $z_j[k+1]$ towards $v_l$.
    7) if $k \mod D' = 0$ then, if $M_j = m_j$ then sets $\text{flag}_j^{\text{st}} = 1$.
Output: (5) holds for every $v_j \in \mathcal{V}$.
```

Size Computation of a Random Network of 20 Nodes. In this section we demonstrate the operation of Algorithm 2 over a random digraph of 20 nodes. The parameters of the digraph are the same as in the previous example where we demonstrated Algorithm 1, and each node also has knowledge of $D' = 4$. Furthermore, we set $U_v = 20$. This means that Algorithm 3 is executed for 80 time steps.

In Fig. 3, we plot the evolution of the state variable $(z_j^s[k])^{-1}$ of every node v_j. We can see that Algorithm 2 allows nodes to calculate after 160 time steps the quantized fraction 1/20, where the denominator is equal to the number of nodes in the network. Finally, after 164 iterations each node terminates its operation since it has knowledge of $D' = 4$. Note that in Fig. 3, we plot the evolution of $z_j^s[k]$ for time steps $k \geq 80$, since Algorithm 3 is executed for the first 80 time steps.

In Fig. 4, we present 10000 executions of Algorithm 2 over a random digraph of 20 nodes. Each edge of the digraph was generated with probability 0.5, the diameter is equal to $D = 3$, and each node also has knowledge of $D' = 4$. In Fig. 4(A) we execute Algorithm 3 before Algorithm 2 with $U_v = 20$ and $D' = 4$ (i.e., we execute Algorithm 3 for 80 time steps). In Fig. 4(B) we execute Algorithm 3 in parallel with Algorithm 2. The average number of time steps for convergence of Algorithm 2 is equal to 198.64 for (A), and 119.60 for (B). In Fig. 4(A) we can see that in most cases

Algorithm 3 Leader Election with Quantized Information.

```
Input: A strongly connected digraph $\mathcal{G}_d = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ nodes and $m = |\mathcal{E}|$ edges.Each node $v_j \in \mathcal{V}$ has knowledge of an upper bound $D'$ of the network diameter.
Initialization: Each node $v_j \in \mathcal{V}$ sets $\text{flag}_j^{\text{ld}} = 1$.
Iteration: For $k = 1, 2, \ldots, U_v D'$, each node $v_j \in \mathcal{V}$, does the following:
  1) if $k \mod D' = 1$ then
      1.1) if $\text{flag}_j^{\text{ld}} = 1$ then chooses randomly $\eta'_j \in \{0, 1, \ldots, Up_j\}$, and sets $M'_j = \eta'_j$;
      1.2) if $\text{flag}_j^{\text{ld}} = 0$ then sets $\eta'_j = -1$ and $M'_j = \eta'_j$;
  2) broadcasts $M'_j$ to every $v_l \in \mathcal{N}_j^+$;
  3) receives $M'_i$ from every $v_i \in \mathcal{N}_j^-$;
  4) sets $M'_j = \max_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} M'_i$;
  5) if $k \mod D' = 0$ then if $M'_j \neq \eta'_j$ then sets $\text{flag}_j^{\text{ld}} = 0$;
Output: $\text{flag}_j^{\text{ld}}$ for every $v_j \in \mathcal{V}$.
```

Algorithm 2 requires 150–240 iterations for convergence. However, in Fig. 4(B) we can see that in most cases it requires 80–140. More specifically, in Fig. 4(B) we can see that 1400 executions of Algorithm 2 require 80 – 90 time steps to converge. This is mainly due to the fact that the Iteration Steps of Algorithm 2 have reached convergence, but nodes need to implement Algorithm 3 for 80 time steps. As a result, parallel execution of Algorithm 3 with Algorithm 2 is advantageous, since a significantly smaller number of time steps is required for convergence.

Lower bounds on required time steps for convergence according to Theorem 2, and Theorem 3. We now present lower bounds on the required time steps k_0 for convergence of Algorithms 1 and 2, for a given probability p_0 according to the results in Theorem 2, and Theorem 3. Specifically, from these theorems we have that $k_0 \geq (n-1)\tau' D + (n-1)\tau'' D + D'$, where τ' fulfills (8) and τ'' fulfills (12). For simplicity, we assume that $D' = D$. We analyze the different values of k_0 for probability values of $p_0 \in \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99\}$ over random digraphs of 20 nodes. Each value of k_0 for a specific probability p_0 was averaged over 100 random digraphs of 20 nodes. In Fig. 5, the errorbars (i.e., upper and lower limit of each point) are due to the different values of D for the different generated networks of $n = 20$ nodes. In Fig. 5 we can see that the lower bound of required time steps increases in a linear fashion for $0.4 \leq p_0 \leq 0.8$. For $0.8 \leq p_0 \leq 0.99$, the lower bound of required time steps increases exponentially. In general, the bounds in Fig. 5 appear to be conservative, and improving them is one of our main future research directions. However, note that in practice Algorithms 1 and 2, require much less time steps for convergence (see Fig. 2 and Fig. 4) and they also exhibit finite time convergence guarantees (which is the main objective of this work).

Leader Election via Algorithm 3. We now analyze the probability of having one leader after the execution of Algorithm 3, over a random digraph of 20 nodes. In Fig. 6 we present 10000 executions of Algorithm 3 with $U_v = 10$. We plot the instances for which we have more than one leader nodes during the execution of Algorithm 3, for the cases where (i) $\eta'_j \in \{0, 1, \ldots, 15\}$, (ii) $\eta'_j \in \{0, 1, \ldots, 31\}$, and (iii) $\eta'_j \in \{0, 1, \ldots, 255\}$ for every node v_j. We can see that if we increase the range of η'_j for every node v_j, the convergence rate of Algorithm 3 improves greatly. Also, after 5 executions of Algorithm 3 (i.e., $U_v = 5$), the number of instances where we have multiple leader nodes is equal to zero. This means that during Algorithm 3 for $U_v \geq 5$, only one node is a leader node with high probability.

7. Conclusions and future directions

We proposed and analyzed two algorithms that able to compute the exact values of the average degree and the size of a network after a finite number of time steps. Our algorithms allow each node to determine in a distributed fashion whether convergence has been achieved, and thus terminate its operation. In order to implement our algorithms, we also devised the first leader election strategy which relies on quantized operation (i.e., nodes process and transmit quantized information). Finally, we demonstrated the operation of our algorithms over random directed networks and illustrated their finite time convergence.

Extending our algorithms to achieve fully asynchronous operation (which is desirable in large-scale networks), and to operate over unreliable networks (e.g., packet dropping links) are two of our main future directions. In addition, we aim to investigate how different choices of the upper bound of the network diameter D', may affect the performance of our proposed algorithms. We also aim to improve the calculated bounds regarding the required time steps for convergence presented in Theorems 2 and 3. Finally, we also intend to analyze the complexity or stopping time of the algorithms by using bounds on the graph diameter D (instead of Assumption 1).

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The work of Prof Themistoklis Charalambous was partly funded by MINERVA, European Research Council (ERC) Grant under the European Union's Horizon 2022 research and innovation programme (Grant agreement No. 101044629).

Appendix A. Proof of Theorem 2

We first consider Lemma 1, which is necessary for our subsequent development.

Lemma 1 [29]**.** *Consider a strongly connected digraph $\mathcal{G}_d = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ nodes and $m = |\mathcal{E}|$ edges. Suppose that each node v_j assigns a nonzero probability b_{lj} to each of its outgoing edges m_{lj}, where $v_l \in \mathcal{N}_j^+ \cup \{v_j\}$, as follows*

$$b_{lj} = \begin{cases} \frac{1}{1+\mathcal{D}_j^+}, & \text{if } l = j \text{ or } v_l \in \mathcal{N}_j^+, \\ 0, & \text{if } l \neq j \text{ and } v_l \notin \mathcal{N}_j^+. \end{cases}$$

At time step $k = 0$, node v_j holds a "token" while the other nodes $v_l \in \mathcal{V} - \{v_j\}$ do not. Each node v_j transmits the "token" (if it has it, otherwise it performs no transmission) according to the nonzero probability b_{lj} it assigned to its outgoing edges m_{lj}. The probability $P_{T_i}^D$ that the token is at node v_i after D time steps satisfies $P_{T_i}^D \geq (1 + \mathcal{D}_{\max}^+)^{-D} > 0$, where $\mathcal{D}_{\max}^+ = \max_{v_j \in \mathcal{V}} \mathcal{D}_j^+$.

We now consider Lemma 2, which analyzes the probability according to which two tokens performing a random walk visit a specific node at the same time step. The proof is similar to Lemma 1, *mutatis mutandis*, and is omitted due to space limitations.

Lemma 2. *Consider a strongly connected digraph $\mathcal{G}_d = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ nodes and $m = |\mathcal{E}|$ edges. Suppose that each node v_j assigns* *a nonzero probability b_{lj} to each of its outgoing edges m_{lj}, where $v_l \in \mathcal{N}_j^+ \cup \{v_j\}$, as follows*

$$b_{lj} = \begin{cases} \frac{1}{1+\mathcal{D}_j^+}, & \text{if } l = j \text{ or } v_l \in \mathcal{N}_j^+, \\ 0, & \text{if } l \neq j \text{ and } v_l \notin \mathcal{N}_j^+. \end{cases}$$

At time step $k = 0$, nodes v_i, v_j hold a "token" while the other nodes $v_l \in \mathcal{V} - \{v_j, v_i\}$ do not (i.e., there are two tokens in the network). Each node v_j transmits the "token" (if it has it, otherwise it performs no transmission) according to the nonzero probability b_{lj} it assigned to its outgoing edges m_{lj}. After D time steps, the probability $P^D_{TT_l}$ that the two tokens visit a specific node at the same time step is $P^D_{TT_l} \geq \sum_{l=1}^n (1 + \mathcal{D}^+_{\max})^{-(2D)} > 0$, where $\mathcal{D}^+_{\max} = \max_{v_j \in \mathcal{V}} \mathcal{D}_j^+$.

Now we present the proof of Theorem 2. The operation of Algorithm 1 can be interpreted as the "random walk" of n "tokens" in a Markov chain. Each token has a pair of values y, z. If two (or more) tokens visit the same node at the same time step k, they "merge" to a new single token (i.e., if two (or more) tokens merge, their y values sum to the new y value, and their z values sum to the new z value). Then, the new token performs a random walk in a Markov chain. Once all n tokens merge to a final single token, this final single token has a pair of values y, z whose ratio y/z is equal to the average degree in the network. Thus, executing Algorithm 1 for an additional finite number of time steps, this final single token will visit every node in the network.

The structure of the proof comprises of three parts. In the first part (**Part I**), we calculate the number of time steps k_0' after which all tokens have merged to one single final token with probability at least $\sqrt{p_0}$. In the second part (**Part II**), we calculate the number of time steps k_0'' after which the final single token has visited every node in the network with probability at least $\sqrt{p_0}$. In the third part (**Part III**), we calculate the number of time steps k_0 after which (4) holds for every node, and each node ceases transmissions.

Part I. During the operation of Algorithm 1, from Lemma 2 we have that, after D time steps, the probability $P^D_{TT_l}$ that two (or more) tokens visit a specific node at the same time step is $P^D_{TT_l} \geq \sum_{l=1}^n (1 + \mathcal{D}^+_{\max})^{-(2D)} > 0$, where $\mathcal{D}^+_{\max} = \max_{v_j \in \mathcal{V}} \mathcal{D}_j^+$. This means that, after D time steps, the probability $P^D_{NTT_l}$ that two (or more) tokens *do not* visit a specific node at the same time step is

$$P^D_{NTT_l} \leq 1 - \sum_{l=1}^{n} (1 + \mathcal{D}^+_{\max})^{-(2D)}. \tag{6}$$

By extending this analysis, we choose ε' (where $0 < \varepsilon' < 1$) for which it holds that

$$\varepsilon' \leq 1 - 2^{\frac{\log_2 \sqrt{p_0}}{n-1}}. \tag{7}$$

After $\tau' D$ time steps where

$$\tau' \geq \left\lceil \frac{\log \varepsilon'}{\log\left(1 - \sum_{l=1}^n (1 + \mathcal{D}^+_{\max})^{-(2D)}\right)} \right\rceil, \tag{8}$$

and ε' fulfills (7), we have that the probability $P^{\tau D}_{NTT_l}$ that two (or more) tokens *do not* visit a specific node at the same time step is $P^{\tau D}_{NTT_l} \leq [P^D_{NTT_l}]^{\tau'} \leq \varepsilon'$. This means that after $\tau' D$ time steps, the probability $P^{\tau D}_{TT_l}$ that two (or more) tokens visit a specific node at the same time step is $P^{\tau D}_{TT_l} \geq 1 - \varepsilon'$. Therefore, after $k_0' \geq (n-1)\tau' D$ time steps, the probability $P^{(n-1)\tau D}_{TT_l}$ that two (or more) tokens visit a specific node for $n-1$ instances at the same time step is

$$P^{(n-1)\tau D}_{TT_l} \geq (1 - \varepsilon')^{(n-1)} \geq \sqrt{p_0}. \tag{9}$$

Part II. During the operation of Algorithm 1, from Lemma 1 we have that the probability $P^D_{T_l}$ that a token visits a specific node after D time steps is $P^D_{T_l} \geq (1 + \mathcal{D}^+_{\max})^{-D} > 0$, where $\mathcal{D}^+_{\max} = \max_{v_j \in \mathcal{V}} \mathcal{D}_j^+$. This means that the probability $P^D_{NT_l}$ that a token *does not* visit a specific node after D steps is

$$P^D_{NT_l} \leq 1 - (1 + \mathcal{D}^+_{\max})^{-D}. \tag{10}$$

We choose ε'' (where $0 < \varepsilon'' < 1$) for which it holds that

$$\varepsilon'' \leq 1 - 2^{\frac{\log_2 \sqrt{p_0}}{n-1}}. \tag{11}$$

After $\tau'' D$ time steps where

$$\tau'' \geq \left\lceil \frac{\log \varepsilon''}{\log\left(1 - (1 + \mathcal{D}^+_{\max})^{-D}\right)} \right\rceil, \tag{12}$$

and ε'' fulfills (11), we have that the probability $P^{\tau D}_{NT_l}$ that one token *does not* visit a specific node is $P^{\tau D}_{NT_l} \leq [P^D_{NTT_l}]^{\tau''} \leq \varepsilon''$. This means that after $\tau'' D$ time steps, the probability $P^{\tau D}_{T_l}$ that one token visits a specific node is $P^{\tau D}_{T_l} \geq 1 - \varepsilon''$. Therefore, after $k_0'' \geq (n-1)\tau'' D$ time steps, the probability $P^{(n-1)\tau D}_{T_l}$ that one token visits a specific node for $n-1$ instances (i.e., it visits every node in the network) is $P^{(n-1)\tau D}_{T_l} \geq (1 - \varepsilon'')^{(n-1)} \geq \sqrt{p_0}$.

Part III. During the operation of Algorithm 1, from Lemmas 1 and 2, we can state that after $(n-1)\tau' D + (n-1)\tau'' D$ time steps, where τ' fulfills (8) and τ'' fulfills (12), we have that (4) holds for every $v_j \in \mathcal{V}$ with probability at least p_0. Then, after an additional number of D' time steps, each node will determine whether convergence has been achieved, and thus it will cease transmissions. As a result, during the operation of Algorithm 1, after $k_0 \geq (n-1)\tau' D + (n-1)\tau'' D + D'$ time steps, we have that each node addresses problem **P1** in Section 3 with probability at least p_0.

Appendix B. Proof of Theorem 3

We first consider Lemma 3, which is necessary for our subsequent development.

Lemma 3. *Consider a strongly connected digraph $\mathcal{G}_d = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ nodes and $m = |\mathcal{E}|$ edges. Each node executes Algorithm 3. After $U_v D$ time steps, a single node v_j is the leader and every other node $v_i \in \mathcal{V} \setminus \{v_j\}$ is a follower, with probability that goes to one, as U_v goes to infinity.*

Proof. When node v_j selects a value randomly depending on the number of bits allocated for communication, it basically samples from a random variable X_j with probability mass function a discrete uniform distribution on the integers $0, 1, 2, \ldots, M-1$, where $M := 2^{\text{bits}}$. Note here that M is equal to the value η_j' (see Algorithm 3). Let $X_1, X_2, \ldots, X_n$ be independent identically distributed (i.i.d.) random variables (representing the random variables of the n nodes in the network).

Let $Y = \max\{X_1, \ldots, X_n\}$. The probability of event A_ℓ being that $\ell < n$ nodes have the maximum value is given by

$$P[A_\ell] = \binom{n}{\ell} \left(\frac{1}{M}\right)^{\ell} \left(\frac{Y-1}{M}\right)^{n-\ell}.$$

Let $n(j)$ denote the number of nodes participating in the max-consensus algorithm in round j, $j \in \{1, 2, \ldots, U_v\}$. In the worst case (when $n(j) = 2$), a node is eliminated at round j with probability at least $1 - 1/M$. Now, we proceed with a very conservative analysis to prove Lemma 3 Consider a sequence of rounds $\{1, 2, \ldots, U_v\}$: at each round a node is eliminated with probability (at least) $p = 1 - 1/M$. Otherwise, with probability (less than) $1 - p = 1/M$, no node is eliminated. Thus, the probability that we have a leader after U_v rounds is the probability that we have $n-1$ eliminations: this is bounded from below by

$$\sum_{k=n-1}^{U_v} \binom{U_v}{k} p^k (1-p)^{U_v-k} = 1 - \sum_{k=0}^{n-2} \binom{U_v}{k} p^k (1-p)^{U_v-k}$$
$$> 1 - (n-1)\binom{U_v}{n-1} p^{n-1} (1-p)^{U_v-(n-1)} ,$$

where U_v is assumed to be larger than $2(n-1)$ for the last inequality to hold (but U_v is not required to be even relevant with the network size, as shown in Fig. 6).

The major advantage of this method is that the number of nodes participating in the next max-consensus is limited to the number of nodes that had the same maximum value in the preceding max-consensus round. Therefore, especially for a considerably large value of M, the number of nodes participating in the next max-consensus is much smaller. The procedure continues until there is a round with no two nodes that select the same (maximum) integer value (cf. Fig. 6). □

Once Algorithm 2 finishes the Initialization steps and elects a leader node (see Algorithm 3) with high probability (see Lemma 3), it executes its Iteration steps. However, the Iteration steps of Algorithm 2 are identical to Algorithm 1. Thus, the proof of Theorem 3 is similar to Theorem 2 and is omitted.

References

[1] J. Augustine, G. Pandurangan, P. Robinson, Fast byzantine leader election in dynamic networks, in: Distributed Computing, 2015, pp. 276–291.
[2] P. Berenbrink, D. Kaaser, P. Kling, L. Otterbach, Simple and efficient leader election, in: Symposium on Simplicity in Algorithms, volume 61, 2018, pp. 1–11.
[3] A. Biswas, A.K. Maurya, A.K. Tripathi, S. Aknine, FRLLE: a failure rate and load-based leader election algorithm for a bidirectional ring in distributed systems, J. Supercomput. 77 (2021) 751–779.
[4] C. Borgs, J. Chayes, A. Ganesh, A. Saberi, How to distribute antidote to control epidemics, Random Structures & Algorithms 37 (2) (2010) 204–222.
[5] J. Cortés, Distributed algorithms for reaching consensus on general functions, Automatica 44 (3) (2008) 726–737.
[6] A. Dasgupta, R. Kumar, T. Sarlos, On estimating the average degree, in: Proceedings of the 23rd International Conference on World Wide Web, 2014, pp. 795–806.
[7] A.D. Domínguez-García, C.N. Hadjicostis, Coordination and control of distributed energy resources for provision of ancillary services, in: Proceedings of the First IEEE International Conference on Smart Grid Communications, 2010, pp. 537–542, doi:10.1109/SMARTGRID.2010.5621991.
[8] L. Faramondi, G. Oliva, R. Setola, Multi-criteria node criticality assessment framework for critical infrastructure networks, Int. J. Crit. Infrastruct. Protect. 28 (2020).
[9] F. Garin, D. Varagnolo, K.H. Johansson, Distributed estimation of diameter, radius and eccentricities in anonymous networks, IFAC Proceedings Volumes 45 (26) (2012) 13–18.
[10] S. Giannini, D. Di Paola, A. Petitti, A. Rizzo, On the convergence of the max-consensus protocol with asynchronous updates, in: Proceedings of IEEE Conference on Decision and Control, 2013, pp. 2605–2610.
[11] M. Hay, C. Li, G. Miklau, D. Jensen, Accurate estimation of the degree distribution of private networks, in: Proceedings of IEEE International Conference on Data Mining, 2009, pp. 169–178.
[12] M.A. Javed, M.S. Younis, S. Latif, J. Qadir, A. Baig, Community detection in networks: A multidisciplinary review, J. Netw. Comput. Appl. 108 (2018) 87–111.
[13] A. Kashyap, T. Basar, R. Srikant, Quantized consensus, Automatica 43 (7) (2007) 1192–1203.
[14] M. Kenyeres, J. Kenyeres, Distributed network size estimation executed by average consensus bounded by stopping criterion for wireless sensor networks, in: Proceedings of International Conference on Applied Electronics (AE), 2019, pp. 1–6.
[15] M. Kenyeres, J. Kenyeres, I. Budinska, On performance evaluation of distributed system size estimation executed by average consensus weights, in: Proceedings of Recent Advances in Soft Computing and Cybernetics, 2021, pp. 15–24.
[16] S. Kutten, G. Pandurangan, D. Peleg, P. Robinson, A. Trehan, On the complexity of universal leader election, J. ACM 62 (1) (2015).
[17] J. Leskovec, J. Kleinberg, C. Faloutsos, Graph evolution: Densification and shrinking diameters, ACM Transactions on Knowledge Discovery from Data 1 (1) (2007) 1–41.
[18] R. Lucchese, D. Varagnolo, J.C. Delvenne, J. Hendrickx, Network cardinality estimation using max consensus: The case of Bernoulli trials, in: Proceedings of IEEE Conference on Decision and Control, 2015, pp. 895–901.
[19] R. Lucchese, D. Varagnolo, K.H. Johansson, Distributed detection of topological changes in communication networks, IFAC Proceedings Volumes 47 (3) (2014) 1928–1934.
[20] N. Malpani, J.L. Welch, N. Vaidya, Leader election algorithms for mobile ad hoc networks, in: Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, 2000, pp. 96–103.
[21] L. Massoulie, E.L. Merrer, A.M. Kermarrec, A. Ganesh, Peer counting and sampling in overlay networks: random walk methods, in: Proceedings of the 25th Annual ACM Symposium on Principles of Distributed Computing, 2006, pp. 123–132.
[22] A. Nedić, A. Olshevsky, M.G. Rabbat, Network topology and communication–computation tradeoffs in decentralized optimization, Proceedings of the IEEE 106 (5) (2018) 953–976.
[23] G. Oliva, R. Setola, C.N. Hadjicostis, Distributed k-means algorithm, arXiv preprint arXiv:1312.4176 (2015).
[24] G. Oliva, R. Setola, C.N. Hadjicostis, Distributed finite-time calculation of node eccentricities, graph radius and graph diameter, System Control Letters 92 (2016) 20–27.
[25] S.L. Peng, S.S. Li, X.K. Liao, Y.X. Peng, N. Xiao, Estimation of a population size in large-scale wireless sensor networks, Journal of Computer Science and Technology 24 (5) (2009) 987–997.
[26] B. Ribeiro, D. Towsley, On the estimation accuracy of degree distributions from graph sampling, in: Proceedings of IEEE Conference on Decision and Control, 2012, pp. 5240–5247.
[27] A.I. Rikos, A. Grammenos, E. Kalyvianaki, C.N. Hadjicostis, T. Charalambous, K.H. Johansson, Optimal CPU scheduling in data centers via a finite-time distributed quantized coordination mechanism, in: IEEE Conference on Decision and Control, 2021, pp. 6276–6281.
[28] A.I. Rikos, C.N. Hadjicostis, Distributed average consensus under quantized communication via event-triggered mass summation, in: Proceedings of IEEE Conference on Decision and Control, 2018, pp. 894–899.
[29] A.I. Rikos, C.N. Hadjicostis, Distributed average consensus under quantized communication via event-triggered mass splitting, in: Proceedings of 20th IFAC World Congress, 2020, pp. 3019–3024.
[30] I. Shames, T. Charalambous, C.N. Hadjicostis, M. Johansson, Distributed network size estimation and average degree estimation and control in networks isomorphic to directed graphs, in: Proceedings of Allerton Conference on Communication, Control, and Computing, 2012, pp. 1885–1892.
[31] Y. Sun, S. Wang, X. Tang, T.-Y. Hsieh, V. Honavar, Adversarial attacks on graph neural networks via node injections: a hierarchical reinforcement learning approach, in: Proceedings of The Web Conference, 2020, pp. 673–683.
[32] C. Tang, W. Wang, X. Wu, B. Wang, Effects of average degree on cooperation in networked evolutionary game, The European Physical Journal B - Condensed Matter and Complex Systems 53 (3) (2006) 411–415.
[33] D. Varagnolo, G. Pillonetto, L. Schenato, Distributed statistical estimation of the number of nodes in sensor seworks, in: Proceedings of IEEE Conference on Decision and Control, 2010, pp. 1498–1503.
[34] S. Zhang, C. Tepedelenlioglu, M.K. Banavar, A. Spanias, Maxconsensus using the soft maximum, in: Proceedings of Asilomar Conference on Signals, Systems and Computers, 2013, pp. 433–437.