2022 IEEE 61st Conference on Decision and Control (CDC) | 978-1-6654-6761-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/CDC51059.2022.9993016

Distributed Optimal Data Allocation in Finite Time with Efficient Communication and Transmission Stopping over Dynamic Networks

Apostolos I. Rikos, Christoforos N. Hadjicostis, and Karl H. Johansson

Abstract— **In this paper, we focus on the problem of data sharing over a wireless computer network (i.e., a wireless grid). Given a set of available data, we present a distributed algorithm, which operates over a dynamically changing network and allows each node to calculate the optimal allocation of data in a finite number of time steps. We show that our proposed algorithm (i) converges to the optimal solution in finite time with very high probability, and (ii) once the optimal solution is reached, each node is able to cease transmissions without needing knowledge of a global parameter such as the network diameter. Furthermore, our algorithm (i) operates exclusively with quantized values (i.e., each node processes and transmits quantized information), (ii) relies on event-driven updates, and (iii) calculates the optimal solution in the form of a quantized fraction which avoids errors due to quantization. Finally, we demonstrate the operation, performance, and potential advantages of our algorithm over random dynamic networks.**

I. Introduction

Wireless computer networks (or wireless grids) comprise of different electronic devices (or nodes), which share their resources with other devices in a distributed manner. Various users or devices may request access to stored data in other devices. In order to reduce the average waiting time for users or devices who request access, data allocation is the procedure of allocating the available data to different nodes according to their available memory capacity such that specific performance objectives are achieved.

In wireless computer networks, data comprise an important resource that needs to be managed efficiently. Optimal allocation of data can heavily influence the operational performance of the network [1]. In general, resource allocation can be formulated as an optimization problem but solving the optimal allocation problem over dynamic networks with quantized communication is challenging due to the heterogeneity of the network and the nonlinear nature of the communication constraints. Centralized solutions consider gathering the available data to a central scheduler; however, these solutions are not ideal as they lack scalability and they impose heavy computational and storage requirements on the central scheduler. For this reason, there has been interest towards distributed algorithms that solve the optimal allocation problem [2]–[5].

Apostolos I. Rikos and K. H. Johansson are with the Division of Decision and Control Systems, KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden. They are also affiliated with Digital Futures, SE-100 44 Stockholm, Sweden. E-mails: {rikos,kallej}@kth.se.

C. N. Hadjicostis is with the Department of Electrical and Computer Engineering, University of Cyprus, 1678 Nicosia, Cyprus. E-mail: chadjic@ucy.ac.cy.

This work was supported by the Knut and Alice Wallenberg Foundation and the Swedish Research Council.

Distributed optimization has received significant attention recently due to its wide variety of applications [6]–[14]. Most works in the current literature assume that network devices process and exchange real values and are able to reach asymptotic convergence within some error [2], [5]. In practical applications of wireless computer networks, devices need to exchange information messages with finite length (i.e., quantized messages) which allows for more efficient usage of the available network resources (e.g., energy, processing power, etc.). Also, they need to operate over networks which may be dynamic due to changes over the sensing radius of the various devices [3], [4]. Additionally, in order to preserve available energy resources, it is desirable for devices to converge in finite time and to stop transmitting once convergence has been achieved [2], [4]. In this paper we propose an algorithm that combines all previously mentioned characteristics: it solves the optimal allocation problem over dynamic networks in finite time, while exhibiting transmission stopping capabilities.

Main Contributions. We focus on the problem of data sharing over a wireless computer network (i.e., a wireless grid). We aim to balance the data storage between nodes by distributively allocating the available data per available memory in the network. We consider the realistic scenario where nodes process and exchange quantized information. We also consider that various changes of each node's sensing radius result in dynamically changing connections in the communication network. Our algorithm is analyzed for optimal data allocation over a wireless computer network. In this scenario, we want to reduce the average waiting time for users or devices who request access to the stored data. However, please note that the proposed algorithm could be adopted in a wide variety of other related applications. The main contributions of the paper are the following.

A. We present a distributed algorithm which solves the optimal data allocation problem over a dynamic network; see Algorithm 1.

B. We show that our algorithm converges in a finite number of time steps, and each node is able to calculate the exact solution without introducing errors due to quantization; see Section IV-B.

C. Once our algorithm converges to the optimal solution, each node ceases transmissions without needing knowledge of a global network parameter (e.g., the diameter of the network). Note that this means that our algorithm does not require reinitialization when there is a change over the network (e.g., when a node enters or leaves the network or

Authorized licensed use limited to: KTH Royal Institute of Technology. Downloaded on April 15,2023 at 12:53:36 UTC from IEEE Xplore. Restrictions apply.

when the diameter changes); see Section IV-A.

D. We analyze the convergence time of the algorithm and show that it relies on the time-varying connectivity (which is determined by the time needed to communicate among pairs of nodes), rather than the size of the network; see Theorem 1.

E. We present simulations of our algorithm where we show its finite time convergence to the exact solution and its transmission stopping capabilities; see Section V.

Unlike our work in this paper, most of the current literature considers algorithms which operate with real values and converge asymptotically within some error [2], [5]. This paper, along with [3], [4], aims to pave the way for finite time algorithms, that operate solely with quantized values and address resource allocation problems. To the authors' knowledge, the proposed algorithm is the first algorithm in the current literature which guarantees finite time convergence and transmission stopping for the case where the underlying network is dynamic, without needing knowledge of a global parameter such as the diameter of the network (e.g., see [4]).

II. NOTATION AND BACKGROUND

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. 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.

Graph-Theoretic Notions. Consider a *dynamic* network of n ($n \geq 2$) nodes communicating only with their immediate neighbors. The communication topology can be captured by a dynamic undirected graph, called *dynamic communication graph*. A dynamic graph is defined as a sequence of undirected graphs $\mathcal{G}[k] = (\mathcal{V}, \mathcal{E}[k])$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes and $\mathcal{E}[k] \subseteq \mathcal{V} \times \mathcal{V} - \{(v_j, v_j) \mid v_j \in \mathcal{V}\}$ is the set of edges (self-edges excluded). An edge between node v_i and node v_j is denoted by $m_{ji}[k] \triangleq (v_j, v_i) \in \mathcal{E}[k]$, and captures the fact that node v_j and node v_i can exchange information (v_j can transmit to v_i and v_i can transmit to v_j) at time step k. Note here that if $(v_j, v_i) \in \mathcal{E}[k]$ then $(v_i, v_j) \in \mathcal{E}[k]$. At time step k, the subset of nodes that can directly transmit information to node v_j is called the set of neighbors of v_j and is represented by $\mathcal{N}_j[k] = \{v_i \in \mathcal{V} \mid (v_j, v_i) \in \mathcal{E}[k]\}$. The cardinality of $\mathcal{N}_j[k]$ at time step k, is called the *degree* of v_j and is denoted by $\mathcal{D}_j[k] = |\mathcal{N}_j[k]|$. Given a collection of graphs $\mathcal{G}[k] = (\mathcal{V}, \mathcal{E}[k])$ for $k = 1, 2, ..., m$, where $m \in \mathbb{N}$, the *union graph* is defined as $\mathcal{G}_d^{1,2,\ldots,m} = (\mathcal{V}, \cup_{k=1}^{m} \mathcal{E}[k])$. A collection of graphs is said to be *jointly connected*, if its corresponding union graph $\mathcal{G}^{1,2,\ldots,m}$ forms a connected graph (i.e., for each pair $v_j, v_i \in \mathcal{V}$, $v_j \neq v_i$, there exists a *path*[1] from v_i to v_j).

Node Operation. The operation of each node $v_j \in \mathcal{V}$ respects the quantization of information flow. At time step $k \in \mathbb{Z}_+$ (where $\mathbb{Z}_+$ is the set of nonnegative integers), each node v_j: maintains the mass variables $y_j[k] \in \mathbb{Z}$ and $z_j[k] \in \mathbb{Z}_+$, which are used to communicate with other nodes by either transmitting or receiving messages; the state variables $y_j^s[k] \in \mathbb{Z}$, $z_j^s[k] \in \mathbb{Z}_+$ and $q_j^s[k] = \frac{y_j^s[k]}{z_j^s[k]}$, which are used to store the received messages and calculate the result of the optimization operation; the transmission variables $S\_br_j \in \mathbb{N}$ and $M\_tr_j \in \mathbb{N}$, which are used to decide whether v_j will broadcast its state variables or transmit its mass variables via a direct transmission.

For the case where each node v_j is required to perform a direct transmission, we assume that v_j is aware of its out-neighbors and can directly transmit messages to each out-neighbor separately. In the proposed distributed algorithm, in order to randomly choose an out-neighbor to transmit to, each node v_j assigns a nonzero probability $b_{lj}[k]$ to each of its edges m_{lj} where $v_l \in \mathcal{N}_j[k]$ (note that there is always a virtual self-edge which means that the probability assigned to the self-edge is nonzero). This probability assignment can be captured for all nodes by an $n \times n$ column stochastic matrix $\mathcal{B}[k] = [b_{lj}[k]]$. A simple choice is to set these probabilities to be equal, i.e.,

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

Each nonzero entry $b_{lj}[k]$ of matrix $\mathcal{B}[k]$ represents the probability of node v_j transmitting towards neighbor $v_l \in \mathcal{N}_j[k]$ through the edge m_{lj}.

Modelling of Wireless Grid and Database. A wireless grid is modeled as a set of $\mathcal{V}$ nodes (or wireless sensors) and each node is denoted as $v_j \in \mathcal{V}$. In most data grids, all participating nodes are interconnected with undirected communication links. Furthermore, various changes over the sensing range of each node impose a dynamic nature to the network topology. This means that the network topology forms a dynamic undirected graph.

For modelling databases, we borrow notation from [4]. Specifically, the data to be allocated in the network is D_{dat}. The data sets which comprise the database are $d_j \in D_{dat}$ (where $j \in \{1, \ldots, |D_{dat}|\}$). The required memory for each data set d_j to be stored is μ_j and is assumed to be known before the optimization operation. Thus, the total required memory for database D_{dat} is $\mu := \sum_{v_j \in \mathcal{V}} \mu_j$. The total load of data at each node v_j, due to incoming data in the network is l_j. The time period for which the optimization operation is executed (before the next optimization operation) is T_o. The total memory of node v_j is $\nu_j^{\max}$, so that the total memory in the network is $\nu^{\max} := \sum_{v_j \in \mathcal{V}} \nu_j^{\max}$. The amount of unavailable memory at node v_j due to previously stored data is $\delta_j[m]$, and the total amount of unavailable memory in the network is $\delta_{\text{tot}}[m] = \sum_{v_j \in \mathcal{V}} \delta_j[m]$. The amount of available memory of node v_j at optimization step m (i.e., at time step mT_o) is $\nu_j^{\text{avail}}[m] := \nu_j^{\max} - \delta_j[m]$, so that the total amount of available memory in the network is $\nu^{\text{avail}}[m] := \sum_{v_j \in \mathcal{V}} \nu_j^{\text{avail}}[m]$.

[1]A *path* from v_i to v_j in the union graph $\mathcal{G}_d^{1,2,\ldots,m}$ exists if we can find a sequence of vertices $v_i \equiv v_{l_0}, v_{l_1}, \ldots, v_{l_t} \equiv v_j$ such that $(v_{l_{\tau+1}}, v_{l_\tau}) \in \bigcup_{k=1}^{m} \mathcal{E}[k+\tau]$ for $\tau = 0, 1, \ldots, t-1$.

Authorized licensed use limited to: KTH Royal Institute of Technology. Downloaded on April 15,2023 at 12:53:36 UTC from IEEE Xplore. Restrictions apply.

III. Problem Formulation

Let us consider a wireless computer network modeled as a dynamic graph $\mathcal{G}[k] = (\mathcal{V}, \mathcal{E}[k])$ with $n = |\mathcal{V}|$ nodes. Each node v_i has a scalar quadratic local cost function $f_i : \mathbb{R}^n \mapsto \mathbb{R}$ (see [8] and references therein) defined as:

$$f_i(z) = \frac{1}{2}\alpha_i(z - \mu_i)^2, \tag{1}$$

where $\alpha_i > 0$, $\mu_i \in \mathbb{R}$ is the demand of node v_i and z is a global optimization variable which determines the data to be stored at each node. The global cost function is the sum of every local cost function f_i (see (1)) in the network, i.e.,

$$F(z) = \sum_{v_i \in \mathcal{V}} f_i(z). \tag{2}$$

The main goal of the nodes is to distributively allocate the available data in order to calculate z^*, which minimizes the global cost function in (2) and is defined as

$$z^* = \arg\min_{z \in \mathcal{Z}} \sum_{v_i \in \mathcal{V}} f_i(z), \tag{3}$$

where $\mathcal{Z}$ is the set of feasible values of parameter z. Note that the solution z^* in (3) can be given in closed form as

$$z^* = \frac{\sum_{v_i \in \mathcal{V}} \alpha_i \mu_i}{\sum_{v_i \in \mathcal{V}} \alpha_i}. \tag{4}$$

Also, note here that if $\alpha_i = 1$ for all $v_i \in \mathcal{V}$, the solution is the average.

The problem we present in this paper is borrowed from [2], [4], but is adjusted in the context of data allocation over dynamic wireless computer networks. Specifically, each node v_i aims to calculate the optimal amount of data to receive $w_i^*[m]$ at each optimization step m, which fulfills

$$\begin{aligned} \frac{w_i^*[m] + \delta_i[m]}{\nu_i^{\max}} &= \frac{w_j^*[m] + \delta_j[m]}{\nu_j^{\max}} \\ &= \frac{\mu[m] + \delta_{\text{tot}}[m]}{\nu^{\max}}, \ \forall v_i, v_j \in \mathcal{V}. \end{aligned} \tag{5}$$

This means that every node aims to balance its data storage (i.e., maintain the same percentage of stored data per available memory) during the algorithm's execution. In the remainder of this paper, we consider a single optimization step (i.e., without loss of generality, we drop index m). From [2], in order to fulfill (5), we need

$$z^* = \frac{\sum_{v_i \in \mathcal{V}} \nu_i^{\max} \frac{\mu_i + \delta_i}{\nu_i^{\max}}}{\sum_{v_i \in \mathcal{V}} \nu_i^{\max}} = \frac{\mu + \delta_{\text{tot}}}{\nu^{\max}}. \tag{6}$$

Thus, the cost function $f_i(z)$ in (1) is given by

$$f_i(z) = \frac{1}{2}\nu_i^{\max}\left(z - \frac{\mu_i + \delta_i}{\nu_i^{\max}}\right)^2. \tag{7}$$

This means that each node computes the optimal amount of data to store and then it is able to find the amount of data w_i^* to receive, i.e.,

$$w_i^* = \frac{\mu + \delta_{\text{tot}}}{\nu^{\max}}\nu_i^{\max} - \delta_i. \tag{8}$$

In this paper we develop a distributed algorithm which operates over dynamic networks and the following features:

- It allows each node v_i to calculate the optimal solution w_i^* in (5) (at every optimization step m).
- Nodes converge to the optimal solution after a finite number of time steps while processing and transmitting quantized values.
- Nodes cease transmissions once convergence has been achieved in order to preserve the available resources at each node without having knowledge of global parameters.

IV. Quantized Data Allocation Algorithm with Finite Transmission Capabilities

In this section we present a distributed algorithm which solves the problem described in Section III. The distributed algorithm is detailed below as Algorithm 1. In order to solve the finite time data allocation problem, we make the following assumptions.

Assumption 1. *Let us consider an infinite sequence of undirected graphs $\mathcal{G}[0], \mathcal{G}[1]$, $\mathcal{G}[2]$, ..., $\mathcal{G}[k]$, ..., describing a dynamic graph. 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 $\mathcal{G}^{t_m, \ldots, t_{m+1}-1}$, is equal to the nominal undirected graph $\mathcal{G}$ which is assumed to be connected. Furthermore, the diameter of the connected union graph $\mathcal{G}^{t_m, \ldots, t_{m+1}-1}$ is denoted as D^{un} and is the longest shortest path between any two nodes $v_j, v_i \in \mathcal{V}$ (note that D^{un} is also the diameter of the nominal graph $\mathcal{G}$).*

Assumption 2. *Each node v_j has a unique ID. This ID is used to distinguish node v_j from other nodes in the network.*

Assumption 3. *The time horizon T_o at step m is chosen such that $\mu[m] \leq \nu^{\text{avail}}[m]$. This means that the total amount of data to be allocated at a specific optimization step m is smaller or equal to the total available memory of the network.*

Assumption 1 allows each node to compute the quantized average of the nodes' quantized states, in a finite number of time steps. Assumption 2 is a necessary condition in order for each node to cease transmissions once the optimal allocation is calculated after a finite number of time steps. Assumption 3 is a necessary condition so that the total demand of data storage does not exceed the total available memory in the network. Note that T_o can be chosen appropriately to fulfill this requirement. Furthermore, note that in case Assumption 3 does not hold, some data will not be allocated due to the lack of available storage in the system.

We now describe the main operations of Algorithm 1. The initialization involves the following steps:

Initialization: Each node $v_j \in \mathcal{V}$ does the following: (i) it initializes its mass variables, (ii) it initializes its transmission variables, and (iii) it sets its state variables to be equal to the mass variables. Then, it broadcasts the values of its state variables to every neighbor. Finally, it initializes set S_j, to

Authorized licensed use limited to: KTH Royal Institute of Technology. Downloaded on April 15,2023 at 12:53:36 UTC from IEEE Xplore. Restrictions apply.

contain the out-neighbors to which it transmitted its state variables at time step $k = 0$.

The iteration involves the following steps:

Iteration - Step 1. Probability Assignment and Receiving: Each node $v_j \in \mathcal{V}$ assigns a nonzero probability to its self-edge and to each of its outgoing edges available at time step k. The assigned probabilities have equal values, and their sum is equal to one, at each step k. Then, it receives from every neighbor (i) the transmitted set of state variables, and (ii) the transmitted set of mass variables (if no set of mass variables is received from a specific neighbor, v_j assumes it received mass variables equal to zero from this neighbor).

Iteration - Step 2. Transmission Conditions According to Mass and State Variables: If node v_j received at least one set of mass variables or state variables during Iteration - Step 1, each node v_j checks the following conditions:

- If the received set of state variables is "greater" (in the way clarified later in this section) than the current set of state variables, it sets its state variables to be equal to the received (greater) set of state variables and decides to broadcast its updated state variables.
- If the stored set of mass variables is "greater" than the state variables, it sets its state variables equal to the mass variables and decides to broadcast its updated state variables.
- If the set of state variables is "greater" than the set of mass variables and the fraction of its mass variables is not equal to the fraction of its state variables, then it decides to directly transmit its mass variables to a randomly chosen neighbor.

Then, if node v_j decided to broadcast its state variables, it updates the stored set S_j to be equal to the current set of neighbors (in order to remember which neighbors received the current state variables).

Iteration - Step 3. Transmission Conditions According to Dynamic Network: At each time step k, each node v_j checks whether the current set of neighbors is not included in the stored set S_j. Note here that the stored set S_j denotes the neighbors which have received (or will receive) the current set of state variables. In case there is one (or multiple) neighbor(s) who is (are) not included in the set S_j, node v_j decides to broadcast its state variables so that this one neighbor (or multiple neighbors) receives the updated set of state variables.

Iteration - Step 4. Transmitting: Each node v_j checks its transmission variables. It transmits its mass variables via a direct transmission, or broadcasts its state variables. Then, it sets its transmission variables equal to zero and repeats the operation.

The details of the dynamic algorithm with transmission stopping capabilities can be seen in Algorithm 1.

Remark 1. *Assumption 2 requires that each node v_j has a unique ID. This is used by Algorithm 1 in order to cease transmissions once the optimal allocation is calculated after a finite number of time steps. In particular, node IDs are used by node v_j in order to update the set S_j and determine whether to broadcast its state variables. To the authors' knowledge, this is the first algorithm which operates over dynamic networks in which nodes are able to cease transmissions without any global parameter, such as the network diameter. However, note that if nodes have knowledge of the parameter l in Definition 1, then they do not need to have unique IDs (Assumption 2 is not needed). Specifically, every time their state variables are updated (see Algorithm 1.A) they can broadcast their state variables for l time steps. From Definition 1, every l time steps, there is a link from node v_j to every neighboring node. As a result, if nodes broadcast their state variables for l time steps, every neighbor will receive the updated states at least once and the algorithm will converge to the optimal solution in finite time.*

Algorithm 1 Optimal Data Allocation Algorithm with Efficient Communication and Transmission Stopping

```
Input: A set of graphs $\mathcal{G}[k] = (\mathcal{V}, \mathcal{E}[k])$ with $n = |\mathcal{V}|$ nodes and $m[k] = |\mathcal{E}[k]|$ edges for which Assumption 1 holds.
Initialization: Each node $v_j \in \mathcal{V}$ does the following:
1) Sets $z_j[0] := l_j + \delta_j$, $y_j[0] = \nu_j^{\max}$, $z_j^s[0] = z_j[0]$, $y_j^s[0] = y_j[0]$, $q_j^s[0] = y_j^s[0]/z_j^s[0]$ and $S\_br_j = 0$, $M\_tr_j = 0$.
2) Broadcasts $z_j^s[0]$, $y_j^s[0]$ to every $v_l \in \mathcal{N}_j[0]$.
3) Sets $S = \mathcal{N}_j[0]$.
Iteration: For $k = 0, 1, 2, \ldots$, each node $v_j \in \mathcal{V}$ does the following:
1) Assigns a nonzero probability $b_{lj}[k]$ to each of its edges $m_{lj}$, where $v_l \in \mathcal{N}_j[k]$, as follows
   $$b_{lj}[k] = \begin{cases} \frac{1}{\mathcal{D}_j[k]+1}, & \text{if } l = j \text{ or } v_l \in \mathcal{N}_j[k], \\ 0, & \text{if } l \neq j \text{ and } v_l \notin \mathcal{N}_j[k]. \end{cases}$$
2) Receives $y_i^s[k]$, $z_i^s[k]$ from every $v_i \in \mathcal{N}_j[k]$ (if no message is received, it sets $y_i^s[k] = 0$, $z_i^s[k] = 0$).
3) Receives $y_i[k]$, $z_i[k]$ from each $v_i \in \mathcal{N}_j[k]$ and sets
   $$y_j[k+1] = y_j[k] + \sum_{v_i \in \mathcal{N}_j[k]} w_{ji}[k] y_i[k],$$
   $$z_j[k+1] = z_j[k] + \sum_{v_i \in \mathcal{N}_j[k]} w_{ji}[k] z_i[k],$$
   where $w_{ji}[k] = 1$ if a message with $y_i[k]$, $z_i[k]$ is received from neighbor $v_i$, otherwise $w_{ji}[k] = 0$.
4) If $w_{ji}[k] \neq 0$ or $z_i^s[k] \neq 0$ for some $v_i \in \mathcal{N}_j[k]$ then
   4a) Calls Algorithm 1.A.
   4b) Event Trigger Conditions 1: If $S_j \cap \mathcal{N}_j[k] \neq \emptyset$, then node $v_j$ sets $S\_br_j = 1$, and $S_j = S_j \cup \mathcal{N}_j[k]$.
   4c) If $M\_tr_j = 1$ then node $v_j$ chooses $v_l \in \mathcal{N}_j[k]$ randomly according to $b_{lj}[k]$ and transmits $y_j[k]$, $z_j[k]$. Then, node $v_j$ sets $y_j[k] = 0$, $z_j[k] = 0$, $M\_tr_j = 0$.
   4d) If $S\_br_j = 1$ then, node $v_j$ broadcasts $z_j^s[k+1]$, $y_j^s[k+1]$ to every $v_l \in \mathcal{N}_j$. Then, it sets $S\_br_j = 0$.
5) Repeats (increases $k$ to $k+1$ and goes back to Step 1).
Output: Sets $w_j^* + \delta_j = (\nu_j^{\max}/q_j^s[k])$ and (5) holds for every $v_j \in \mathcal{V}$.
```

Authorized licensed use limited to: KTH Royal Institute of Technology. Downloaded on April 15,2023 at 12:53:36 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1.A Event-Triggered Conditions for Algorithm 1 (for each node v_j)

```
Input
$y_j^s[k]$, $z_j^s[k]$, $q_j^s[k]$, $y_j[k+1]$, $z_j[k+1]$, $S\_br_j$, $M\_tr_j$, $S_j$, $\mathcal{N}_j[k]$, and the received $y_i^s[k]$, $z_i^s[k]$ from every $v_i \in \mathcal{N}_j[k]$.
Execution
• Event Trigger Conditions 1: If
    Condition $(i)$: $z_i^s[k] > z_j^s[k]$, or
    Condition $(ii)$: $z_i^s[k] = z_j^s[k]$ and $y_i^s[k] > y_j^s[k]$,
    then node $v_j$ sets
        $z_j^s[k+1] = \max_{v_i \in \mathcal{N}_j[k]} z_i^s[k]$,   and
        $y_j^s[k+1] = \max_{v_i \in \{v_{i'} \in \mathcal{N}_j[k] | z_{i'}^s[k] = z_j^s[k+1]\}} y_i^s[k]$,
    and also sets $q_j^s[k+1] = \frac{y_j^s[k+1]}{z_j^s[k+1]}$, and $S\_br_j = 1$.
• Event Trigger Conditions 2: If
    Condition $(i)$: $z_j[k+1] > z_j^s[k+1]$, or
    Condition $(ii)$: $z_j[k+1] = z_j^s[k+1]$ and $y_j[k+1] > y_j^s[k+1]$,
    then node $v_j$ sets $z_j^s[k+1] = z_j[k+1]$, $y_j^s[k+1] = y_j[k+1]$, and $q_j^s[k+1] = \frac{y_j^s[k+1]}{z_j^s[k+1]}$ and $S\_br_j = 1$.
• Event Trigger Conditions 3: If
    Condition $(i)$: $0 < z_j[k+1] < z_j^s[k+1]$ or
    Condition $(ii)$: $z_j[k+1] = z_j^s[k+1]$ and $y_j[k+1] < y_j^s[k+1]$,
    then node $v_j$ sets $M\_tr_j = 1$.
• Event Trigger Conditions 4: If $0 < z_j[k+1]$ and
        $\frac{y_j[k+1]}{z_j[k+1]} = \frac{y_j^s[k+1]}{z_j^s[k+1]}$,
    then node $v_j$ sets $M\_tr_j = 0$.
• Event Trigger Conditions 5: If $S\_br_j = 1$
    then node $v_j$ sets $S_j = \mathcal{N}_j[k]$.
Output
$y_j^s[k]$, $z_j^s[k]$, $q_j^s[k]$, $S\_br_j$, $M\_tr_j$, $S_j$.
```

A. Operation over Dynamic Graphs

We now analyze the functionality of Algorithm 1 over dynamic networks. We consider the following two definitions which are important for our subsequent development.

Definition 1. *Consider a set of graphs $\mathcal{G}[k] = (\mathcal{V}, \mathcal{E}[k])$, $k = 0, 1, 2, \ldots$, with $n = |\mathcal{V}|$ nodes and $m[k] = |\mathcal{E}[k]|$ edges for which Assumption 1 holds. During the execution of Algorithm 1, at time step k_0, there is at least one node $v_{j'} \in \mathcal{V}$, for which*

$$z_{j'}[k_0] \geq z_i[k_0], \; \forall v_i \in \mathcal{V}. \tag{9}$$

Then, among the nodes $v_{j'}$ for which (9) holds, there is at least one node v_j for which

$$y_j[k_0] \geq y_l[k_0], \text{ where } v_j, v_l \in \{v_{j'} \in \mathcal{V} \mid (9) \text{ holds}\}. \tag{10}$$

For notational convenience we will call the pair of mass variables of node v_j for which (9) and (10) hold as the "leading mass" (or "leading masses" if multiple nodes hold such a pair of values) and the pairs of mass variables of a node v_l for which $z_l[k_0] > 0$ but (9) and (10) do not hold as the "follower mass" (or "follower masses").

Definition 2. *Consider a set of graphs $\mathcal{G}[k] = (\mathcal{V}, \mathcal{E}[k])$, $k = 0, 1, 2, \ldots$, with $n = |\mathcal{V}|$ nodes and $m[k] = |\mathcal{E}[k]|$ edges for which Assumption 1 holds. During the execution of Algorithm 1, at time step k_0, if two (or more) masses (for which $z \neq 0$) reach a node simultaneously then we say that they "merge". This means that the receiving node "merges" the mass variables it receives by summing their numerators and their denominators (according to Step 3 of the Iteration of Algorithm 1). This way a set of mass variables with a greater denominator is created.*

The intuition behind Algorithm 1 can be described through the following three stages.
Stage 1: Initially every node assumes that its mass variables are the leading mass and broadcasts its state variables. After a finite number of time steps, the state variables of every node in the network are equal to the leading mass. Note that we consider the simple scenario where no mass variables merged until the state variables of every node in the network become equal to the leading mass, (thus the leading mass does not change until the state variables of every node in the network become equal to the leading mass). The scenario where two or more mass variables merge and the leading mass changes until the state variables of every node in the network become equal to the leading mass, can be proved identically.
Stage 2: Once the state variables of every node become equal to the leading mass, every node transmits its mass variables towards a randomly chosen neighbor. This means that the mass variables of every node (except the node whose mass variables are the leading mass) perform a random walk. During their random walk, the mass variables either merge with the leading mass (which is not transmitted), or merge between them (if they visit a common node). In the first case, the leading mass is updated and the corresponding node broadcasts its updated state variables. In the second case, if two mass variables visit a common node, the node checks if the merged mass variables are now the leading mass (note that the node's state variables are equal to the leading mass from Stage 1). If the merged mass variables are the leading mass, then the corresponding node broadcasts its state variables and does not transmit its mass variables. However, if the merged mass mass variables are not the leading mass, then they are transmitted to a randomly chosen neighbor.
Stage 3: Once the leading mass is updated (i.e., the mass variables either merge with the leading mass, or merge between them), the corresponding node broadcasts its state variables. Thus, after a finite number of time steps, the state variables of every node in the network become equal to the updated leading mass, and then Stage 2 is repeated.

Note here that during the operation of Algorithm 1, there is always a set of mass variables which is the leading mass. The follower masses perform random walks until they merge

Authorized licensed use limited to: KTH Royal Institute of Technology. Downloaded on April 15,2023 at 12:53:36 UTC from IEEE Xplore. Restrictions apply.

with the leading mass or they merge between them and become the leading mass. Once the leading mass is updated (i.e., either a follower mass merges with the leading mass or two follower masses merge between them and become the leading mass), every node in the network receives the updated state variables of the node whose mass variables are the leading mass (see Stage 1). As a result, after a finite number of time steps (i) the leading mass becomes equal to the quantized average of the initial states, (ii) each node sets its state variables equal to the leading mass (i.e., the average of the initial states), and (iii) once each node's state becomes equal to the average of the initial states, transmissions are ceased (since there are no more updates of the leading mass).

B. Convergence of Algorithm 1

We now analyze the convergence time of Algorithm 1. We first consider Lemma 1, *mutatis mutandis*, which is necessary for our subsequent development.

Lemma 1 ([15])**.** *Consider a sequence of graphs $\mathcal{G}[k] = (\mathcal{V}, \mathcal{E}[k])$, $k = 0, 1, 2, ...$, with $n = |\mathcal{V}|$ nodes and $m[k] = |\mathcal{E}[k]|$ edges for which Assumption 1 holds. At each time step k, suppose that each node v_j assigns a nonzero probability $b_{lj}[k]$ to each of its edges $m_{lj}[k]$, where $v_l \in \mathcal{N}_j[k] \cup \{v_j\}$, as*

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

At time step $k = 0$, node v_j holds a "token" while the other nodes $v_l \in \mathcal{V} - \{v_j\}$ do not. At each time step k, each node v_j transmits the "token" (if it has the token, otherwise it performs no transmission) according to the nonzero probability $b_{lj}[k]$ it assigned to its edges $m_{lj}[k]$. The probability $P_{DT_i}^{D^{un}}$ that the token is at node v_i after lD^{un} time steps satisfies

$$P_{DT_i}^{lD^{un}} \geq (1 + \mathcal{D}_{max})^{-(lD^{un})} > 0,$$

where l is the time window defined in Assumption 1 (for which the union graph $\mathcal{G}_d^{t_m,...,t_{m+1}-1}$ is equal to the nominal graph $\mathcal{G}$ which is connected), and $\mathcal{D}_{max}$ is the maximum degree of every node in the nominal graph $\mathcal{G}$.

We now consider Lemma 2, which analyzes the probability according to which a token performing a random walk visits a specific node. Then, we present Theorem 1 which analyzes the finite time convergence of Algorithm 1. Due to space limitations we omit the proof of Lemma 2, and Theorem 1; they will be available in an extended version of our paper.

Lemma 2. *Consider a sequence of graphs $\mathcal{G}[k] = (\mathcal{V}, \mathcal{E}[k])$, $k = 0, 1, 2, ...$, with $n = |\mathcal{V}|$ nodes and $m[k] = |\mathcal{E}[k]|$ edges for which Assumption 1 holds. At each time step k, suppose that each node v_j assigns a nonzero probability $b_{lj}[k]$ to each of its edges $m_{lj}[k]$, where $v_l \in \mathcal{N}_j[k] \cup \{v_j\}$, as follows*

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

At time step $k = 0$, node v_j holds a "token" while the other nodes $v_l \in \mathcal{V} - \{v_j\}$ do not. At each time step k, each node v_j transmits the "token" (if it has the token, otherwise it performs no transmission) according to the nonzero probability $b_{lj}[k]$ it assigned to its edges $m_{lj}[k]$. For any probability p_0, where $0 < p_0 < 1$, there exists $k_0 \in \mathbb{Z}_+$, so that with probability at least p_0, the token has visited a specific node v_i, (where l is the time window defined in Assumption 1 for which the union graph $\mathcal{G}^{t_m,...,t_{m+1}-1}$ is equal to the nominal graph $\mathcal{G}$ which is connected).

Theorem 1. *Consider a sequence of graphs $\mathcal{G}[k] = (\mathcal{V}, \mathcal{E}[k])$, $k = 0, 1, 2, ...$, with $n = |\mathcal{V}|$ nodes and $m[k] = |\mathcal{E}[k]|$ edges for which Assumption 1, Assumption 2, and Assumption 3 hold. Suppose that each node $v_j \in \mathcal{V}$ follows the Initialization and Iteration steps as described in Algorithm 1, where $l_j, \delta_j, \nu_j^{\max} \in \mathbb{N}$ for every node $v_j \in \mathcal{V}$ at time step $k = 0$. During the operation of Algorithm 1, for any probability p_0' (where $0 < p_0' < 1$) there exists $k_0' \in \mathbb{Z}_+$, so that with probability at least p_0' each node v_j is able to (i) calculate the optimal amount of data w_j^* (shown in (8)) after a finite number of time steps k_0, and (ii) cease transmissions after calculating w_j^*.*

V. Simulation Results

In this section, we present simulation results in order to demonstrate the operation of Algorithm 1 and its potential advantages. We focus on a random graph of 20 nodes and show how the nodes' states converge to the optimal solution. Furthermore, we show the total accumulated number of transmissions and the number transmissions at every time step.

Evaluation over a Dynamic Network of 20 Nodes. The dynamic network comprises 20 nodes and the union of the dynamic networks is equal to the nominal graph after $l = 5$ time steps. The nominal graph is assumed to be connected and has a diameter equal to 3. At each node v_j, the total load of data l_j was generated via a random distribution uniformly picked within the range $[1, 50]$. The total load of data in the network is equal to 504 (i.e., $\sum_{v_j \in \mathcal{V}} l_j = 504$). For a randomly chosen set of seven nodes the total memory was set to be 31752, for a randomly chosen set of seven nodes the total memory was set to be 63504, and for a randomly chosen set of six nodes the total memory was set to be 95256. Our simulation results are shown in Fig. 1 and Fig. 2.

In Fig. 1 (A), we can see that each node v_j is able to calculate the exact ratio of memory per data after 29 time steps. The ratio is equal to 504/20 and and is calculated exactly in the form of a quantized fraction without any errors due to quantized communication and processing. In Fig. 1 (B), we can see that each node v_j is able to calculate the optimal amount of data to receive after 29 time steps. The amount of data is proportional to the node's memory capacity. Specifically, from the result of Fig. 1, each node calculates the ratio of data per memory and then scales with its available memory capacity. Specifically, the 7 nodes with total memory equal to 31752, receive 1260 amount of data.

Authorized licensed use limited to: KTH Royal Institute of Technology. Downloaded on April 15,2023 at 12:53:36 UTC from IEEE Xplore. Restrictions apply.

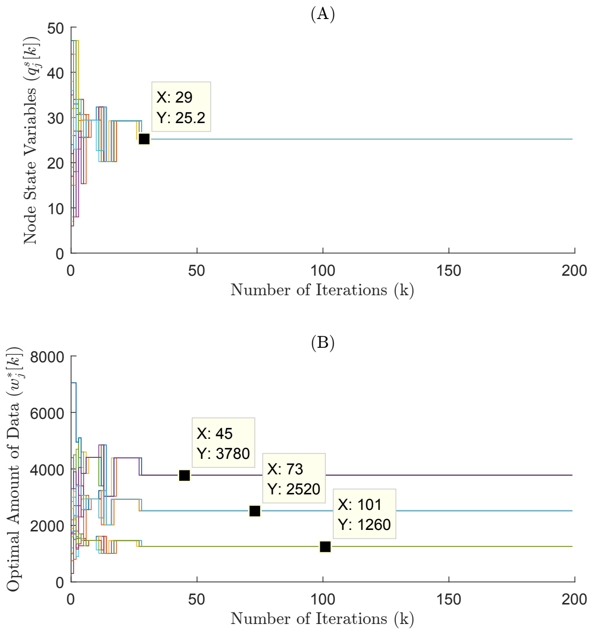

Fig. 1. Execution of Algorithm 1 over a random dynamic graph comprised of 20 nodes where the union of the dynamic graphs is equal to the nominal graph after $l = 5$ time steps. *(A):* Following Algorithm 1, each node calculates the amount of data per memory after 29 time steps. *(B):* Algorithm 1 converges to the optimal solution after 29 time steps.

The 7 nodes with total memory equal to 63504, receive 2520 amount of data (double the amount received by the nodes with 31752). Finally, the 6 nodes with total memory equal to 95256, receive 3780 amount of data.

In Fig. 2 (A), we can see the accumulated total number of transmissions performed from nodes in the network during the operation of Algorithm 1. The total number of transmissions performed is equal to 291 during 29 time steps. In Fig. 2 (B), we can see the number of transmissions during the operation of Algorithm 1 at every time step k. We can see, that in the beginning, the number of transmissions is high. However, after 10 time steps, it decreases and only increases at specific instances due to the dynamic nature of the communication network. The number of transmissions performed becomes equal to 0 after 29 time steps.

VI. Conclusions and Future Directions

In this paper, we focused on the problem of optimal data scheduling over a wireless computing network. We proposed a distributed algorithm which operates over dynamic networks and converges in finite time. We showed that our algorithm converges to the exact optimal solution in finite time. Our algorithm operates with quantized values (i.e., each node processes and transmits quantized values) and, once it converges to the optimal solution, each node ceases transmissions. Finally, we have demonstrated the operation of our algorithm over random dynamic networks and exhorted its finite time convergence.

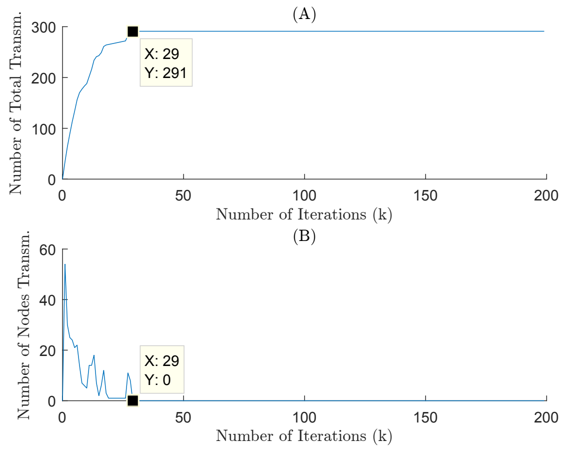

Fig. 2. Execution of Algorithm 1 over a random dynamic graph comprised of 20 nodes where the union of the dynamic graphs is equal to the nominal graph after $l = 5$ time steps. *(A):* Following Algorithm 1, the total number of transmissions performed is equal to 291. *(B):* Following Algorithm 1, every node ceases transmissions after 29 time steps.

References

[1] J. Y. Wang and K. F. Jea, "A near-optimal database allocation for reducing the average waiting time in the grid computing environment," *Information Sciences*, vol. 179, no. 21, pp. 3772–3790, 2009.

[2] A. Grammenos, T. Charalambous, and E. Kalyvianaki, "CPU scheduling in data centers using asynchronous finite-time distributed coordination mechanisms," *arXiv preprint arXiv:2101.06139*, 2020.

[3] M. Doostmohammadian, A. Aghasi, M. Pirani, E. Nekouei, U. A. Khan, and T. Charalambous, "Fast-convergent dynamics for distributed allocation of resources over switching sparse networks with quantized communication links," *arXiv preprint arXiv:2012.08181*, 2020.

[4] A. I. Rikos, A. Grammenos, E. Kalyvianaki, C. N. Hadjicostis, T. Charalambous, and K. H. Johansson, "Optimal CPU scheduling in data centers via a finite-time distributed quantized coordination mechanism," *in Proceedings of* 60^{th} *IEEE Conference on Decision and Control (CDC)*, pp. 6276–6281, 2021.

[5] A. D. Dominguez-Garcia and C. N. Hadjicostis, "Distributed resource coordination in networked systems described by digraphs," *Systems & Control Letters*, vol. 82, pp. 33–39, 2015.

[6] M. Rabbat and R. D. Nowak, "Distributed optimization in sensor networks," *Proceedings of* 3^{rd} *International Symposium on Information Processing in Sensor Networks (IPSN)*, pp. 20–27, 2004.

[7] K. Tsianos, S. Lawlor, and M. Rabbat, "Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning," *Proceedings of* 50^{th} *Annual Allerton Conference on Communication, Control, and Computing*, pp. 1543–1550, 2012.

[8] T. Yang, X. Yi, J. Wu, Y. Yuan, D. Wu, Z. Meng, Y. Hong, H. Wang, Z. Lin, and K. H. Johansson, "A survey of distributed optimization," *Annual Reviews in Control*, vol. 47, pp. 278–305, 2019.

[9] L. Xiao and S. Boyd, "Optimal scaling of a gradient method for distributed resource allocation," *Journal of Optimization Theory and Applications*, vol. 129, no. 3, pp. 469–488, 2006.

[10] A. Nedić, A. Olshevsky, and W. Shi, "Improved convergence rates for distributed resource allocation," in *Proceedings of* 57^{th} *IEEE Conference on Decision and Control*, 2018, pp. 172–177.

[11] B. Johansson, M. Rabi, and M. Johansson, "A randomized incremental subgradient method for distributed optimization in networked systems," *SIAM J. on Optimization*, vol. 20, no. 3, pp. 1157–1170, August 2010.

[12] A. Nedic, A. Ozdaglar, and P. A. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, 2010.

[13] A. Nedić, A. Olshevsky, and M. G. Rabbat, "Network topology and communication-computation tradeoffs in decentralized optimization," *Proceedings of IEEE*, vol. 106, no. 5, pp. 953–976, 2018.

[14] A. Cherukuri and J. Cortes, "Distributed generator coordination for initialization and anytime optimization in economic dispatch," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 3, pp. 226–237, December 2015.

[15] A. I. Rikos and C. N. Hadjicostis, "Distributed average consensus under quantized communication via event-triggered mass splitting," in *Proceedings of* 20^{th} *IFAC World Congress*, 2020, pp. 3019–3024.

Authorized licensed use limited to: KTH Royal Institute of Technology. Downloaded on April 15,2023 at 12:53:36 UTC from IEEE Xplore. Restrictions apply.