Asynchronous Distributed Optimization via ADMM
with Efficient Communication

Apostolos I. Rikos, Wei Jiang, Themistoklis Charalambous, and Karl H. Johansson

Abstract— In this paper, we focus on an asynchronous distributed optimization problem. In our problem, each node is endowed with a convex local cost function, and is able to communicate with its neighbors over a directed communication network. Furthermore, we assume that the communication channels between nodes have limited bandwidth, and each node suffers from processing delays. We present a distributed algorithm which combines the Alternating Direction Method of Multipliers (ADMM) strategy with a finite time quantized averaging algorithm. In our proposed algorithm, nodes exchange quantized valued messages and operate in an asynchronous fashion. More specifically, during every iteration of our algorithm each node (i) solves a local convex optimization problem (for the one of its primal variables), and (ii) utilizes a finite-time quantized averaging algorithm to obtain the value of the second primal variable (since the cost function for the second primal variable is not decomposable). We show that our algorithm converges to the optimal solution at a rate of $O(1/k)$ (where k is the number of time steps) for the case where the local cost function of every node is convex and not-necessarily differentiable. Finally, we demonstrate the operational advantages of our algorithm against other algorithms from the literature.

I. Introduction

The problem of distributed optimization has received extensive attention in recent years. Due to the rise of large-scale machine learning [1], control [2], and other data-driven applications [3], there is a growing need to solve optimization problems that involve massive amounts of data. Solving these problems in a centralized way is proven to be infeasible since it is difficult or impossible to store and process large amounts of data on a single node.

Distributed optimization is a method that distributes data across multiple nodes. Each node performs computations on its stored data and collaborates with others to solve the optimization problem collectively. This approach optimizes a global objective function by combining each node's local objective function and coordinating with the network. The advantage is reducing computational and storage requirements for individual nodes. However, frequent communication with neighboring nodes is necessary to update optimization variables. This can become a bottleneck with increasing nodes or data. To address this issue, recent attention from the scientific community focuses on developing optimization algorithms with efficient communication. This leads to enhancements on scalability and operational efficiency, while mitigating issues like network congestion, latency, and bandwidth limitations.

Existing Literature. Most works in the literature assume that nodes can process and exchange real values. This may result in communication overhead, especially for algorithms requiring frequent and complex communication (see, e.g., [4]–[10]). In practical applications, nodes must exchange quantized messages to efficiently utilize network resources like energy and processing power. For this reason, recent research focuses on communication-efficient algorithms (e.g., [6], [7], [11]–[15]), but they often assume perfectly synchronized nodes or bidirectional communication, limiting their applicability. Addressing communication overhead remains a key challenge, necessitating the development of communication-efficient algorithms that can operate over directed networks asynchronously. Therefore, continued research in this area is crucial to overcoming this bottleneck and enhancing the performance of distributed optimization methods.

Main Contributions. Existing algorithms in the literature often assume that nodes can exchange precise values of their optimization variables and operate synchronously. However, transmitting exact values (often irrational numbers) necessitates an infinite number of bits and becomes infeasible. Moreover, synchronizing nodes within a distributed network involves costly protocols, time-consuming to execute. In this paper, we present a distributed optimization algorithm, which aims to address these challenges. More specifically, we make the following contributions.

A. We present a distributed optimization algorithm that leverages the advantages of the ADMM optimization strategy and operates over a directed communication graph. Our algorithm allows nodes to operate in an asynchronous fashion, and enables efficient communication as nodes communicate with quantized messages; see Algorithm 1.

B. We prove that our algorithm converges to the optimal solution at a rate of $O(1/k)$ even for non-differentiable and convex local cost functions (as it is the case for similar algorithms with real-valued states). This rate is justified in our simulations in which our algorithm exhibits com-

Apostolos I. Rikos is with the Department of Electrical and Computer Engineering, Division of Systems Engineering, Boston University, Boston, MA 02215, US. E-mail: `arikos@bu.edu`.

Wei Jiang resides in Hong Kong, China. Email: `wjiang.lab@gmail.com`.

T. Charalambous is with the Department of Electrical and Computer Engineering, School of Engineering, University of Cyprus, 1678 Nicosia, Cyprus. He is also with the Department of Electrical Engineering and Automation, School of Electrical Engineering, Aalto University, Espoo, Finland. Email: `charalambous.themistoklis@ucy.ac.cy`.

K. H. Johansson is with the Division of Decision and Control Systems, KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden. He is also affiliated with Digital Futures. E-mail: `kallej@kth.se`.

parable performance with real-valued communication algorithms while guaranteeing efficient (quantized) communication among nodes; see Section VI. Furthermore, we show that the optimal solution is calculated within an error bound that depends on the quantization level; see Theorem 1.

II. NOTATION AND PRELIMINARIES

Notation. The sets of real, rational, integer and natural numbers are denoted by $\mathbb{R}, \mathbb{Q}, \mathbb{Z}$ and $\mathbb{N}$, respectively. The symbol $\mathbb{Z}_{\geq 0}$ ($\mathbb{Z}_{>0}$) denotes the set of nonnegative (positive) integer numbers. The symbol $\mathbb{R}_{\geq 0}$ ($\mathbb{R}_{>0}$) denotes the set of nonnegative (positive) real numbers. The symbol $\mathbb{R}_{\geq 0}^n$ denotes the nonnegative orthant of the n-dimensional real space $\mathbb{R}^n$. Matrices are denoted with capital letters (e.g., A), and vectors with small letters (e.g., x). The transpose of matrix A and vector x are denoted as $A^\top$, $x^\top$, respectively. 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. For any matrix $A \in \mathbb{R}^{n \times n}$, the a_{ij} denotes the entry in row i and column j. By $\mathbb{1}$ and $\mathbb{I}$ we denote the all-ones vector and the identity matrix of appropriate dimensions, respectively. By $\| \cdot \|$, we denote the Euclidean norm of a vector.

Graph Theory. The communication network is captured by a directed graph (digraph) defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. This digraph consists of n ($n \geq 2$) agents communicating only with their immediate neighbors, and is static (i.e., it does not change over time). In $\mathcal{G}$, the set of nodes is denoted as $\mathcal{V} = \{v_1, v_2, ..., v_n\}$, and the set of edges as $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} \cup \{(v_i, v_i) \mid v_i \in \mathcal{V}\}$ (note that each agent has also a virtual self-edge). The cardinality of the sets of nodes, edges are denoted as $|\mathcal{V}| = N$, $|\mathcal{E}| = m$, respectively. A directed edge from node v_i to node v_l is denoted by $(v_l, v_i) \in \mathcal{E}$, and captures the fact that node v_l can receive information from node v_i at time step k (but not the other way around). The subset of nodes that can directly transmit information to node v_i is called the set of in-neighbors of v_i and is represented by $\mathcal{N}_i^- = \{v_j \in \mathcal{V} \mid (v_i, v_j) \in \mathcal{E}\}$. The subset of nodes that can directly receive information from node v_i is called the set of out-neighbors of v_i and is represented by $\mathcal{N}_i^+ = \{v_l \in \mathcal{V} \mid (v_l, v_i) \in \mathcal{E}\}$. The *in-degree*, and *out-degree* of v_j are denoted by $\mathcal{D}_i^- = |\mathcal{N}_i^-|$, $\mathcal{D}_i^+ = |\mathcal{N}_i^+|$, respectively. The diameter D of a digraph is the longest shortest path between any two nodes $v_l, v_i \in \mathcal{V}$. A directed *path* from v_i to v_l of length t exists if we can find a sequence of agents $i \equiv l_0, l_1, \ldots, l_t \equiv l$ such that $(l_{\tau+1}, l_\tau) \in \mathcal{E}$ for $\tau = 0, 1, \ldots, t-1$. A digraph is *strongly connected* if there exists a directed path from every node v_i to every node v_l, for every $v_i, v_l \in \mathcal{V}$.

ADMM Algorithm. The standard ADMM algorithm [16] is designed to solve the following problem:

$$\min_{x \in \mathbb{R}^p, z \in \mathbb{R}^m} \quad f(x) + g(x), \tag{1}$$
$$\text{s.t. } Ax + Bz = c,$$

where $A \in \mathbb{R}^{q \times p}$, $B \in \mathbb{R}^{q \times m}$ and $c \in \mathbb{R}^q$ (for $q, p, m \in \mathbb{N}$).

In order to solve (1), the augmented Lagrangian is:

$$\begin{aligned} L_\rho(x, z, \lambda) = f(x) + g(x) + \lambda(Ax + Bz - c) \\ + \frac{\rho}{2}\|Ax + Bz - c\|^2, \end{aligned} \tag{2}$$

where $\lambda \in \mathbb{R}$ is the Lagrange multiplier, and $\rho \in \mathbb{R}$ is the positive penalty parameter. The primary variables x, z and the Lagrangian multiplier λ are initialized as $[x, z, \lambda]^\top = [x^{[0]}, z^{[0]}, \lambda^{[0]}]^\top$. Then, during every ADMM time step, the x, z and λ are updated as:

$$x^{[k+1]} = \arg\min_{x \in \mathbb{R}^p} L_\rho(x, z^{[k]}, \lambda^{[k]}), \tag{3}$$

$$z^{[k+1]} = \arg\min_{z \in \mathbb{R}^m} L_\rho(x^{[k+1]}, z, \lambda^{[k]}), \tag{4}$$

$$\lambda^{[k+1]} = \lambda^{[k]} + \rho(Ax^{[k+1]} + Bz^{[k+1]} - c), \tag{5}$$

where ρ in (5) is the penalty parameter from (2).

Asymmetric Quantizers. Quantization is a strategy that lessens the number of bits needed to represent information. This reduces the required communication bandwidth and increases power and computation efficiency. Quantization is mainly used to describe communication constraints and imperfect information exchanges between nodes [17]. The three main types of quantizers are (i) asymmetric, (ii) uniform, and (iii) logarithmic. In this paper, we rely on asymmetric quantizers to reduce the required communication bandwidth. Note that the results of this paper are transferable to other quantizer types (e.g., logarithmic or uniform). Asymmetric quantizers are defined as

$$q_\Delta^a(\xi) = \left\lfloor \frac{\xi}{\Delta} \right\rfloor, \tag{6}$$

where $\Delta \in \mathbb{Q}$ is the quantization level, $\xi \in \mathbb{R}$ is the value to be quantized, and $q_\Delta^a(\xi) \in \mathbb{Q}$ is the quantized version of ξ with quantization level Δ (note that the superscript "a" indicates that the quantizer is asymmetric).

III. PROBLEM FORMULATION

Problem Statement. Let us consider a distributed network modeled as a digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ nodes. In our network $\mathcal{G}$, we assume that the communication channels among nodes have limited bandwidth. Each node v_i is endowed with a scalar local cost function $f_i(x) : \mathbb{R}^p \mapsto \mathbb{R}$ only known to node v_i. In this paper we aim to develop a distributed algorithm which allows nodes to cooperatively solve the following optimization problem

$$\min_{x \in \mathbb{R}^p} \sum_{i=1}^n f_i(x), \tag{7}$$

where $x \in \mathbb{R}^p$ is the global optimization variable (or common decision variable). We will solve (7) via the distributed ADMM strategy. Furthermore, in our solution we guarantee efficient communication between nodes (due to communication channels of limited bandwidth in the network).

Modification of the Optimization Problem. In order to solve (7) via the ADMM and guarantee efficient communication between nodes, we introduce (i) the variable x_i for every

node v_i, (ii) the constraint $|x_i - x_j| \leq \epsilon$ for every $v_i, v_j \in \mathcal{V}$ (where $\epsilon \in \mathbb{R}$ is an error tolerance which is predefined), and (iii) the constraint that nodes communicate with quantized values. The second constraint is introduced to allow an asynchronous implementation of the distributed ADMM strategy, and the third constraint to guarantee efficient communication between nodes. Considering the aforementioned constraints (i), (ii) and (iii), (7) becomes:

$$\min_{x_i} \ \sum_{i=1}^{n} f_i(x_i), i = 1, ..., n \tag{8}$$

$$\text{s.t. } |x_i - x_j| \leq \epsilon, \forall v_i, v_j \in \mathcal{V}, \tag{9}$$

$$\text{nodes communicate with quantized values.} \tag{10}$$

Let us now define a closed nonempty convex set $\mathcal{C}$ as

$$\mathcal{C} = \left\{ \begin{bmatrix} x_1^\mathsf{T} & x_2^\mathsf{T} & \cdots & x_n^\mathsf{T} \end{bmatrix}^\mathsf{T} \in \mathbb{R}^{np} : \|x_i - x_j\| \leq \epsilon \right\}. \tag{11}$$

Furthermore, denote $X \coloneqq \begin{bmatrix} x_1^\mathsf{T} & x_2^\mathsf{T} & \cdots & x_n^\mathsf{T} \end{bmatrix}^\mathsf{T}$ and its copy variable $z \in \mathbb{R}^{np}$. This means that (9) and (11) become

$$X = z, \forall z \in \mathcal{C}. \tag{12}$$

Now let us define the indicator function $g(z)$ of set $\mathcal{C}$ as

$$g(z) = \begin{cases} 0, & \text{if } z \in \mathcal{C}, \\ \infty, & \text{otherwise}. \end{cases} \tag{13}$$

Incorporating (12) and (13) into (8), we have that (8) becomes

$$\min_{z, x_i} \left\{ \sum_{i=1}^{n} f_i(x_i) + g(z) \right\}, i = 1, \ldots, n$$
$$\text{s.t. } X - z = 0, \forall z \in \mathcal{C}, \tag{14}$$
$$\text{nodes communicate with quantized values.}$$

As a result, in this paper, we aim to design a distributed algorithm that solves (14) via the distributed ADMM strategy.

IV. PRELIMINARIES ON DISTRIBUTED COORDINATION

We now present a definition of asynchrony (borrowed from [8]) that defines the operation of nodes in the network. Furthermore, we present a distributed coordination algorithm that operates with quantized values and is necessary for our subsequent development.

A. Definition of Asynchronous Operation

During their optimization operation, nodes aim to coordinate in an asynchronous fashion. Specifically, let us assume that the iterations for the optimization operation start at time step $t(0) \in \mathbb{R}_+$. Furthermore, we assume that one (or more) nodes transmit values to their out-neighbors at a set of time instances $\mathcal{T} = \{t(1), t(2), t(3), \ldots\}$. During the nodes' asynchronous operation, a message that is received at time step $t(\eta_1)$ from node v_i, is processed at time step $t(\eta_2)$ where $\eta_2 > \eta_1$. This means that the message received at time step $t(\eta_1)$ suffers from a processing delay of $t(\eta_2) - t(\eta_1)$ time steps. An example of how processing delays affecting transmissions is shown in Fig. 1 (that is borrowed from [8]).
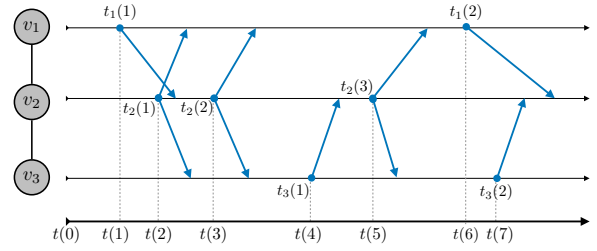

Fig. 1. Example of how processing and transmission delays affect the operation of nodes v_1, v_2, v_3. Blue dots indicate the iterations and blue arrows indicate the transmissions. Transmissions occur at time steps $t_i(\eta)$, and $t_i(\eta+1) - t_i(\eta)$ is the processing delay, where $i \in \{1, 2, 3\}, \eta \in \mathbb{Z}_{\geq 0}$. The time difference from the blue dot to the blue arrow is the transmission delay [8].

Note here that the nodes states at time step $t(\eta)$ are indexed by η. This means that the state of node v_i at time step $t(\eta)$ is denoted as $x_i^\eta \in \mathbb{R}^p$.

We now present the following assumption which is necessary for the asynchronous operation of every node.

Assumption 1: The number of time steps required for a node v_i to process the information received from its in-neighbors is upper bounded by $\mathcal{B} \in \mathbb{N}$. Furthermore, the actual time (in seconds) required for a node v_i to process the information received from its in-neighbors is upper bounded by $T \in \mathbb{R}_{\geq 0}$.

Assumption 1 states that there exists a finite number of steps $\mathcal{B}$ before which all nodes have updated their states and proceed to perform transmissions to their neighboring nodes. The upper bound $\mathcal{B}$ is translated to an upper bound of T in actual time (in seconds). This is mainly because it is not possible for nodes to count the number of time steps elapsed in the network (and understand when $\mathcal{B}$ time steps have passed. The value T can be counted by each node individually.

B. Asynchronous max/min - Consensus

In asynchronous max/min consensus (see [18]), the update rule for every node $v_i \in \mathcal{V}$ is:

$$x_i^{[k+\theta_i^{[k]}]} = \max_{v_j \in \mathcal{N}_i^- \cup \{v_i\}} \{ x_j^{[k+\theta_{ij}^{[k]}]} \}, \tag{15}$$

where $\theta_i^{[k]}$ is the update instance of node v_i, $x_j^{[k+\theta_{ij}^{[k]}]}$ are the states of the in-neighbors $v_j \in \mathcal{N}_i^- \cup \{v_i\}$ during the time instant of v_i's update, $\theta_{ij}^{[k]}$ are the asynchronous state updates of the in-neighbors of node v_i that occur between two consecutive updates of node v_i's state. The asynchronous max/min consensus in (15) converges to the maximum value among all nodes in a finite number of steps $s' \leq D\mathcal{B}$ (see [18]), where D is the diameter of the network, and $\mathcal{B}$ is the upper bound on the number of time steps required for a node v_j to process the information received from its in-neighbors.

V. DISTRIBUTED ASYNCHRONOUS OPTIMIZATION VIA ADMM WITH EFFICIENT COMMUNICATION

In this section we present a distributed algorithm which solves problem (14). Before presenting the operation of the

proposed algorithm, we analyze the ADMM operation over the problem (14).

In (14), let us denote $F(X) \coloneqq \sum_{i=1}^{n} f_i(x_i)$. This means that the Lagrangian function is equal to

$$L(X, z, \lambda) = F(X) + g(z) + \lambda^{\mathrm{T}}(X - z), \qquad (16)$$

where $\lambda \in \mathbb{R}^{np}$ is the Lagrange multiplier. We now make the following assumptions to solve the problem (14).

Assumption 2: Every cost function $f_i : \mathbb{R}^p \to \mathbb{R}$ is closed, proper and convex.

Assumption 3: The Lagrangian $L(X, z, \lambda)$ has a saddle point. This means that there exists (X^*, z^*, λ^*), for which

$$L(X^*, z^*, \lambda) \leq L(X^*, z^*, \lambda^*) \leq L(X, z, \lambda^*), \qquad (17)$$

for all $X \in \mathbb{R}^{np}$, $z \in \mathbb{R}^{np}$, and $\lambda \in \mathbb{R}^{np}$.

Assumption 2 means that the local cost function f_i of every node v_i can be non-differentiable (see [19]). Furthermore, Assumptions 2 and 3 mean that $L(X, z, \lambda^*)$ is convex in (X, z) and (X^*, z^*) is a solution to problem (14) (see [19], [20]). Note that this is also based on the definition of $g(z)$ in (13). Note here that our results extend naturally to strongly convex cost functions, since strong convexity implies convexity.

Let us now focus on the Lagrangian of the problem in (14). At time step k, the augmented Lagrangian of (14) is

$$
\begin{aligned}
&L_\rho(X^{[k]}, z^{[k]}, \lambda^{[k]}) \qquad (18) \\
&= \sum_{i=1}^{n} f_i(x_i^{[k]}) + g(z^{[k]}) + \lambda^{[k]\mathrm{T}}(X^{[k]} - z^{[k]}) \\
&+ \frac{\rho}{2}\|X^{[k]} - z^{[k]}\|^2 \\
&= \sum_{i=1}^{n} \left(f_i(x_i^{[k]}) + \lambda_i^{[k]\mathrm{T}}(x_i^{[k]} - z_i^{[k]}) + \frac{\rho}{2}\|x_i^{[k]} - z_i^{[k]}\|^2 \right) \\
&+ g(z^{[k]}),
\end{aligned}
$$

where $z_i \in \mathbb{R}^p$ is the ith element of vector z. In (3)–(5) we ignore terms that are independent of the optimization variables such as x_i, z for node v_i. This means that (3)–(5) become:

$$x_i^{[k+1]} = \underset{x_i}{\operatorname{argmin}} f_i(x_i) + \lambda_i^{[k]\mathrm{T}} x_i + \frac{\rho}{2}\|x_i - z_i^{[k]}\|^2, \qquad (19)$$

$$z^{[k+1]} = \underset{z}{\operatorname{argmin}} g(z) + \lambda^{[k]\mathrm{T}}(X^{[k+1]} - z) + \frac{\rho}{2}\|X^{[k+1]} - z\|^2$$

$$= \underset{z}{\operatorname{argmin}} g(z) + \frac{\rho}{2}\|X^{[k+1]} - z + \frac{1}{\rho}\lambda^{[k]}\|^2, \qquad (20)$$

$$\lambda_i^{[k+1]} = \lambda_i^{[k]} + \rho(x_i^{[k+1]} - z_i^{[k+1]}). \qquad (21)$$

Note that for (20) we use the identity $2a^T b + b^2 = (a+b)^2 - a^2$ for $a = \lambda^{[k]}/\rho$ and $b = X^{[k+1]} - z$.

Equations (19), (21) can be executed independently by node v_i in a parallel fashion. Specifically, node v_i can solve (19) for $x_i^{[k+1]}$ by a classical method (e.g., a proximity operator [19, Section 4]), and implement trivially (21) for $\lambda_i^{[k+1]}$.

In (13), $g(z)$ is the indicator function of the closed nonempty convex set $\mathcal{C}$. This means that (20) becomes

$$z^{[k+1]} = \Pi_{\mathcal{C}}(X^{[k+1]} + \lambda^{[k]}/\rho), \qquad (22)$$

where $\Pi_{\mathcal{C}}$ is the projection (in the Euclidean norm) onto $\mathcal{C}$. It is important to note here that the elements of z (i.e., $z_1, z_2, \ldots, z_n$) should belong into the set $\mathcal{C}$ in finite time. This is due to the definition of $g(z)$ in (13). Specifically, if the elements of z do not belong in $\mathcal{C}$ then $g(z) = \infty$ (thus (20) cannot be executed). Therefore, we need to adjust the values of the elements of z so that they belong in the set $\mathcal{C}$ in finite time (i.e., we need to set the elements of z such that $\|z_i - z_j\| \leq \epsilon, \forall v_i, v_j \in \mathcal{V}$). Note that if $z_i - z_j = 0, \forall v_i, v_j \in \mathcal{V}$, then every node $v_i \in \mathcal{V}$ has reached consensus. Specifically, we can have that in finite time the state z_i becomes

$$z_i = \frac{1}{n}\sum_{l=1}^{n} z_l^{[0]}, \forall v_i \in \mathcal{V}, \qquad (23)$$

where $z_l^{[0]} = x_l^{[k+1]} + \lambda_l^{[k]}/\rho$. Furthermore, $\|z_i - z_j\| \leq \epsilon, \forall v_i, v_j \in \mathcal{V}$ means that

$$z_i \in \left[\frac{1}{n}\sum_{l=1}^{n} z_l^{[0]} - \frac{\epsilon}{2}, \frac{1}{n}\sum_{l=1}^{n} z_l^{[0]} + \frac{\epsilon}{2}\right], \forall v_i \in \mathcal{V}, \qquad (24)$$

where $z_l^{[0]} = x_l^{[k+1]} + \lambda_l^{[k]}/\rho$. This means that for every node v_i, z_i enters a circle with its center at $\frac{1}{n}\sum_{l=1}^{n}(x_l^{[k+1]} + \lambda_l^{[k]}/\rho)$ and its radius as $\epsilon/2$. Finally, from (14), we have that each node in the network needs to communicate with its neighbors in an efficient manner. For this reason, we aim to allow each node v_i coordinate with its neighboring nodes by exchanging quantized values in order to fulfil (24).

A. Distributed Optimization Algorithm

We now present our distributed optimization algorithm. The algorithm is detailed below as Algorithm 1 and allows each node in the network to solve the problem presented in (14). The operation of the proposed algorithm is based on two parts. During these parts, each node v_i (i) calculates $x_i^{[k+1]}$, $z_i^{[k+1]}$, $\lambda_i^{[k+1]}$ according to (19)–(21) (see Algorithm 1), and (ii) coordinates with other nodes in a communication efficient manner in order to calculate $z_i^{[k+1]}$ that belongs in $\mathcal{C}$ in (11) (see Algorithm 2). Note that Algorithm 2 is a finite time coordination algorithm with quantized communication and is executed as a step of Algorithm 1.

Note that during Algorithm 1, nodes operate in an asynchronous fashion. Synchronous operation requires synchronization among nodes or the existence of a global clock so that all nodes to agree on their update time. In our setting, asynchronous operation arises when each node (i) starts calculating $x_i^{[k+1]}$, $z_i^{[k+1]}$, $\lambda_i^{[k+1]}$ according to (19)–(21) in Algorithm 1, and (ii) calculates $z_i^{[k+1]}$ that belongs in $\mathcal{C}$ in (11) in Algorithm 2. This can be achieved by making the internal clocks of all nodes have similar pacing, which will allow them to execute the optimization step at roughly the same time [21]. Furthermore, making the internal clocks

of all nodes have similar pacing does not mean that we have to synchronize the clocks of the nodes (or their time-zones). Note that this is a common procedure in most modern computers as the clock pacing specification is defined within the Advanced Configuration and Power Interface (ACPI) specification [22].

We now make the following assumption which is important for the operation of our algorithm.

Assumption 4: The diameter D (or an upper bound) is known to every node v_i in the network.

Assumption 4 is necessary so that each node v_i is able to determine whether calculation of z_i that belongs in $\mathcal{C}$ in (11) has been achieved in a distributed manner. We now present the details of Algorithm 1.

Algorithm 1 QuAsyADMM - Quantized Asynchronous ADMM

Input: Strongly connected $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, parameter ρ, diameter D, error tolerance $\epsilon \in \mathbb{Q}$, upper bound on processing delays $\mathcal{B}$. Assumptions 1, 2, 3, 4 hold. $k_{\max}$ (ADMM maximum number of iterations).
Initialization: Each node $v_i \in \mathcal{V}$ sets randomly $x^{[0]}, z^{[0]}, \lambda^{[0]}$, and sets $\Delta = \epsilon/3$.
Iteration: For $k = 0, 1, 2, \ldots, k_{\max}$, each node $v_i \in \mathcal{V}$ does the following:
1) Calculate $x_i^{[k+1]}$ via (19);
2) Calculate $z_i^{[k+1]} = $ Algorithm $2(x_i^{[k+1]} + \lambda_i^{[k]}/\rho, D, \Delta, \mathcal{B})$;
3) Calculate $\lambda_i^{[k+1]}$ via (21).
Output: Each node $v_i \in \mathcal{V}$ calculates x_i^* which solves problem (14) in Section III.

Note here that Algorithm 1 shares similarities with [8]. However, Algorithm 1 is adjusted to guarantee efficient communication between nodes by allowing them to exchange quatized valued messages (this characteristic is not present in [8]). More specifically, the intuition of Algorithm 1 is the following. Each node v_i aims to solve the problem presented in (14) via the ADMM strategy. Initially, each node chooses a suitable quantization level Δ so that the constraints of (14) are fulfilled (as explained later in Remark 1). During each time step k each node v_i calculates $x_i^{[k+1]}$ via (19). Then, each node v_i executes Algorithm 2 in order to calculate $z_i^{[k+1]}$ which belongs in the set $\mathcal{C}$ (i.e., $z_i^{[k+1]} \in \mathcal{C}, \forall v_i \in \mathcal{V}$). Finally, each node v_i uses $x_i^{[k+1]}$ and the result of Algorithm 2 in order to calculate $\lambda_i^{[k+1]}$ via (21). Algorithm 2 is a distributed coordination algorithm which allows each node to calculate the quantized average of each node's initial state. The main characteristic of Algorithm 2 is that it combines (i) (asymmetric) quantization, (ii) quantized averaging, and (iii) a stopping strategy. In Algorithm 2, initially each node v_i uses an asymmetric quantizer to quantize its state; see Initialization-step 2. Then, at each time step η each node v_i:
- splits the y_i into ξ_i equal pieces (the value of some pieces might be greater than others by one); see Iteration-steps 4.1, 4.2,

---

Algorithm 2 QuAsAvCo - Quantized Asynchronous Average Consensus

---

Input: $x_i^{[k+1]} + \lambda_i^{[k]}/\rho, D, \Delta, \mathcal{B}$.
Initialization: Each node $v_i \in \mathcal{V}$ does the following:
1) Assigns probability b_{li} to each out-neigbor $v_l \in \mathcal{N}_i^+ \cup \{v_i\}$, as follows

$$b_{li} = \begin{cases} \frac{1}{1 + \mathcal{D}_i^+}, & \text{if } l = i \text{ or } v_l \in \mathcal{N}_i^+, \\ 0, & \text{if } l \neq i \text{ and } v_l \notin \mathcal{N}_i^+; \end{cases}$$

2) $\text{flag}_i = 0$, $\xi_i = 2$, $y_i = 2\, q_\Delta^a(x_i^{[k+1]} + \lambda_i^{[k]}/\rho)$ (see (6));
Iteration: For $\eta = 1, 2, \ldots$, each node $v_i \in \mathcal{V}$, does:
1) **if** $\eta \mod (D\mathcal{B}) = 1$ **then** sets $M_i = \lceil y_i/\xi_i \rceil$, $m_i = \lfloor y_i/\xi_i \rfloor$;
2) broadcasts M_i, m_i to every $v_l \in \mathcal{N}_i^+$; receives M_j, m_j from every $v_j \in \mathcal{N}_i^-$; sets $M_i = \max_{v_j \in \mathcal{N}_i^- \cup \{v_i\}} M_j$, $m_i = \min_{v_j \in \mathcal{N}_i^- \cup \{v_i\}} m_j$;
3) sets $d_i = \xi_i$;
4) **while** $d_i > 1$ **do**
   4.1) $c_i^{[\eta]} = \lfloor y_i / \xi_i \rfloor$;
   4.2) sets $y_i = y_i - c_i^{[\eta]}$, $\xi_i = \xi_i - 1$, and $d_i = d_i - 1$;
   4.3) transmits $c_i^{[\eta]}$ to randomly chosen out-neighbor $v_l \in \mathcal{N}_i^+ \cup \{v_i\}$ according to b_{li};
   4.4) receives $c_j^{[\eta]}$ from $v_j \in \mathcal{N}_i^-$ and sets

$$y_i = y_i + \sum_{j=1}^{n} \sum_{r=0}^{\mathcal{B}} w_{\eta-r,ij}^{[r]}\, c_j^{[\eta-r]}, \qquad (25)$$

$$\xi_i = \xi_i + \sum_{j=1}^{n} \sum_{r=0}^{\mathcal{B}} w_{\eta-r,ij}^{[r]}, \qquad (26)$$

where $w_{\eta-r,ij}^{[r]} = 1$ when the processing time of node v_i is equal to r at time step $\eta-r$, so that node v_i receives $c_i^{[\eta]}$, 1 from v_j at time step η (otherwise $w_{\eta-r,ij}^{[r]} = 0$ and v_i receives no message at time step η from v_j);
5) **if** $\eta \mod (D\mathcal{B}) = 0$ **and** $M_i - m_i \leq 1$ **then** sets $z_i^{[k+1]} = m_i \Delta$ and stops operation.
Output: $z_i^{[k+1]}$.

- transmits each piece to a randomly selected out-neighbor or to itself; see Iteration-step 4.3,
- receives the pieces transmitted from its in-neighbors, sums them with y_i and ξ_i, and repeats the operation; see Iteration-step 4.4.

Finally, every $D\mathcal{B}$ time steps, each node v_i performs in parallel a max-consensus and a min-consensus operation; see Iteration-steps 1 and 2. If the results of the max-consensus and min-consensus have a difference less or equal to one (see Iteration-steps 5), each node v_i (i) scales the solution according to the quantization level, (ii) stops the operation of Algorithm 2, (iii) uses the value $z_i^{[k+1]}$ to continue the operation of Algorithm 1. Note that Algorithm 2 converges in finite number of steps according to [23, Theorem 1], since QuAsAvCo has similar structure to that in [23].

Remark 1: It is important to note here that during the initialization of Algorithm 1, the error tolerance ϵ is chosen to be a rational number (i.e., $\epsilon \in \mathbb{Q}$). This is not a limitation for the ADMM optimization process in Algorithm 1. The real-valued ϵ can be chosen such that it can be represented as a rational value. Furthermore, this choice facilitates the operation of Algorithm 2. Specifically, a rational value for ϵ facilitates the choice of a suitable quantization level Δ (since $\Delta = \epsilon/3$). During the execution of Algorithm 2 nodes quantize their states, thus an error $e_{q_1} \leq \Delta$ is imposed to every state. Then, Algorithm 2 converges to the quantized average thus, the final states of the nodes have an error $e_{q_2} \leq \Delta$. This means that after executing Algorithm 2, we have $|z_i - z_j| \leq 2\Delta < \epsilon$, and thus we have $z_i^{[k+1]} \in \mathcal{C}$ in (11), $\forall v_i \in \mathcal{V}$. For this reason, any choice of Δ for which $\Delta < \epsilon/2$ is suitable for the operation of our algorithm for a given error tolerance ϵ.

Remark 2: In practical applications, nodes do not know the value of $\mathcal{B}$. However, B time-steps (which is its upper bound) is guaranteed to be executed within T seconds (see Assumption 1). As noted previously, consistent pacing of each node's clock ensures that the check for convergence at each node will happen at roughly the same time (see [21]). Therefore, at every DT seconds, each node checks whether Algorithm 2 can be terminated.

B. Convergence of Algorithm 1

We now analyze the convergence time of Algorithm 1 via the following theorem. Our theorem is inspired from [8] but is adjusted to the quantized nature of Algorithm 1. However, due to space limitations we omit the proof (we will include it at an extended version of our paper).

Theorem 1: Let us consider a strongly connected digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each node $v_i \in \mathcal{V}$, is endowed with a scalar local cost function $f_i(x) : \mathbb{R}^p \mapsto \mathbb{R}$, and Assumptions 1-4 hold. Furthermore, every node v_i has knowledge of a parameter ρ, the network diameter D, an error tolerance $\epsilon \in \mathbb{Q}$, and an upper bound on processing delays $\mathcal{B}$. During the operation of Algorithm 1, let us consider the variables $\{X^{[k]}, z^{[k]}, \lambda^{[k]}\}$, where $X^{[k]} = [x_1^{[k]^\top}, x_2^{[k]^\top}, \ldots, x_n^{[k]^\top}]^\top$ and $\lambda^{[k]} = [\lambda_1^{[k]^\top}, \lambda_2^{[k]^\top}, \ldots, \lambda_n^{[k]^\top}]^\top$; then, define $\bar{X}^{[k]} = \frac{1}{k}\sum_{s=0}^{k-1} X^{[s+1]}, \bar{z}^{[k]} = \frac{1}{k}\sum_{s=0}^{k-1} z^{[s+1]}$. During the operation of Algorithm 1 we have

$$0 \leq L(\bar{X}^{[k]}, \bar{z}^{[k]}, \lambda^*) - L(X^*, z^*, \lambda^*) \tag{27}$$
$$\leq \frac{1}{k}\left(\frac{1}{2\rho}\|\lambda^* - \lambda^{[0]}\|^2 + \frac{\rho}{2}\|X^* - z^{[0]}\|^2\right) + \mathcal{O}(2\Delta\sqrt{n}),$$

for every time step k, where Δ is the quantization level for calculating $z_i \in \mathcal{C}$ in (11) during the operation of Algorithm 2.

It is important to note that in Theorem 1 we focus on the convergence of the optimization steps, i.e., the steps executed during the operation of Algorithm 1. Due to the operation of Algorithm 2 we have that in (27) an additional term $\mathcal{O}(2\Delta\sqrt{n})$ appears. This term (as will be seen later in Section VI) affects the precision according to which the

optimal solution is calculated. However, we can adjust Algorithm 2 to operate with a dynamically refined quantization level Δ. For example, we can initially set $\Delta = \epsilon/3$ (where $\epsilon \in \mathbb{Q}$). Then, execute Algorithm 2 during every time step k with quantization level $\Delta' = \frac{\Delta}{10(k+1)}$. Since we have $\frac{\Delta}{10(k+1)} < \frac{\Delta}{10(k)}$ for every k, then, Algorithm 2 will lead to a reduction of the error on the optimal solution that depends on the quantization level (i.e., the term $\mathcal{O}(2\Delta\sqrt{n})$ in (27) will be reduced after every execution of Algorithm 2). However, please note that this analysis is outside of the scope of this paper and will be considered in an extended version.

VI. SIMULATION RESULTS

In this section, we present simulation results in order to demonstrate the operation of Algorithm 1 and its advantages. Furthermore, we compare Algorithm 1 against existing algorithms and emphasize on the introduced improvements.

In Fig. 2, we focus on a network comprised of 100 nodes modelled as a directed graph. Each node v_i is endowed with a scalar local cost function $f_i(x) = 0.5x^\top P_i x + q_i^\top x + r_i$. This cost function is quadratic and convex. Furthermore, for $f_i(x)$ we have that (i) P_i was initialized as the square of a randomly generated symmetric matrix A_i (ensuring it is is positive definite), (ii) q_i is initialized as the negation of the product of the transpose of A_i and a randomly generated vector b_i (i.e., it is a linear term), (iii) and r_i is initialized as half of the squared norm of the randomly generated vector b_i (i.e., it is a scalar constant). We execute Algorithm 1 and we show how the nodes' states converge to the optimal solution for $\epsilon = 0.03, 0.003, 0.0003$, and $\Delta = 0.01, 0.001, 0.0001$, respectively. We plot the error $e^{[k]}$ defined as

$$e^{[k]} = \frac{\sqrt{\sum_{j=1}^n (x_j^{[k]} - x^*)^2}}{\sqrt{\sum_{j=1}^n (x_j^{[0]} - x^*)^2}}, \tag{28}$$

where x^* is the optimal solution of the optimization problem in (14). Note that from Remark 1, we have that any $\Delta < \epsilon/2$ is suitable for the operation of Algorithm 1 for a given ϵ. In Fig. 2, we execute Algorithm 1 for $\Delta = \epsilon/3$. We can see that Algorithm 1 converges to the optimal solution for the three different values of ϵ. However, Algorithm 1 is able to approximate the optimal solution with precision that depends on the quantization level (i.e., during Algorithm 1, nodes are able to calculate a neighborhood of the optimal solution). Reducing the quantization level Δ allows calculation of the optimal solution with higher precision. Furthermore, we can see that after calculating the optimal solution our algorithm exhibits an oscillatory behavior due to quantized communication. This means quantized communication introduces nonlinearities to the consensus calculation which in turn affect the values of other parameters such as x and z, and λ (see iteration steps 1, 2, 3), and for this reason we have this oscillatory behavior. Finally, we can see that Algorithm 1 exhibits comparable performance with [24] (which is plotted until optimization step 14) until the neighborhood of the optimal solution is calculated. However, in [24] nodes are able to exchange real-valued messages. Specifically, in [24]
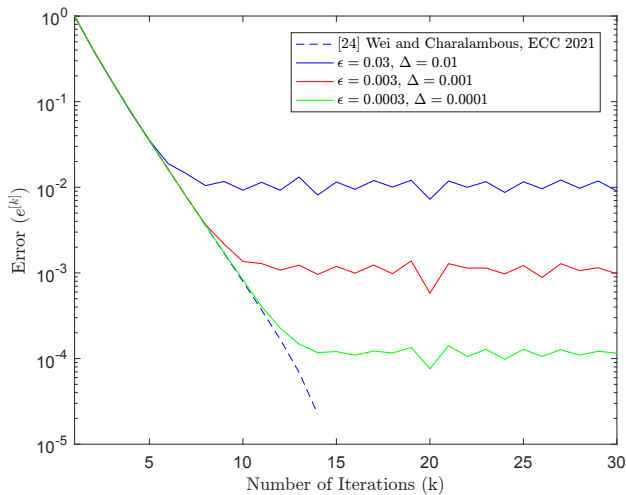
Fig. 2. Comparison of Algorithm 1 with [24] over a directed graph comprised of 100 nodes for $\epsilon = 0.03, 0.003, 0.0003$, and $\Delta = 0.01, 0.001, 0.0001$, respectively.

nodes are required to form the Hankel matrix and perform additional computations when the matrix loses rank. This requires nodes to exchange the exact values of their states. Therefore, the main advantage of Algorithm 1 compared to [24], is that it exhibits comparable performance while guaranteeing efficient (quantized) communication among nodes.

VII. CONCLUSIONS AND FUTURE DIRECTIONS

In this paper, we presented an asynchronous distributed optimization algorithm which combines the Alternating Direction Method of Multipliers (ADMM) strategy with a finite time quantized averaging algorithm. We showed that our proposed algorithm is able to calculate the optimal solution while operating over directed communication networks in an asynchronous fashion, and guaranteeing efficient (quantized) communication between nodes. We analyzed the operation of our algorithm and showed that it converges to a neighborhood of the optimal solution (that depends on the quantization level) at a rate of $O(1/k)$. Finally, we demonstrated the operation of our algorithm and compared it against other algorithms from the literature.

In the future, we aim to enhance the operation of our algorithm to avoid the oscillatory behavior after calculating the optimal solution. Furthermore, we plan to develop strategies that allow calculation of the *exact* optimal solution while guaranteeing efficient communication among nodes. Finally, we will focus on designing efficient communication strategies for non-convex distributed optimization problems.

REFERENCES

[1] A. Nedich, "Distributed gradient methods for convex machine learning problems in networks: Distributed optimization," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 92–101, 2020.

[2] G. S. Seyboth, D. V. Dimarogonas, and K. H. Johansson, "Event-based broadcasting for multi-agent average consensus," *Automatica*, vol. 49, no. 1, pp. 245–252, 2013.

[3] S. U. Stich, J. B. Cordonnier, and M. Jaggi, "Sparsified SGD with memory," in *Advances in Neural Information Processing Systems*, vol. 31. Curran Associates, Inc., 2018, pp. 4447–4458.

[4] A. Makhdoumi and A. Ozdaglar, "Convergence rate of distributed ADMM over networks," *IEEE Transactions on Automatic Control*, vol. 62, no. 10, pp. 5082–5095, 2017.

[5] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[6] T. Qin, S. R. Etesami, and C. A. Uribe, "Communication-efficient decentralized local SGD over undirected networks," in *2021 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 3361–3366.

[7] L. Fang and Y. Lei, "An asynchronous distributed ADMM algorithm and efficient communication model," in *2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, 2016, pp. 136–140.

[8] W. Jiang, A. Grammenos, E. Kalyvianaki, and T. Charalambous, "An asynchronous approximate distributed alternating direction method of multipliers in digraphs," in *2021 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 3406–3413.

[9] N. Bastianello, R. Carli, L. Schenato, and M. Todescato, "Asynchronous distributed optimization over lossy networks via relaxed ADMM: Stability and linear convergence," *IEEE Transactions on Automatic Control*, vol. 66, no. 6, pp. 2620–2635, 2021.

[10] V. Khatana and M. V. Salapaka, "DC-DistADMM: ADMM algorithm for constrained optimization over directed graphs," *IEEE Transactions on Automatic Control*, pp. 1–16, 2022.

[11] J. F. C. Mota, J. M. F. Xavier, P. M. Q. Aguiar, and M. Püschel, "D-ADMM: A communication-efficient distributed algorithm for separable optimization," *IEEE Transactions on Signal Processing*, vol. 61, no. 10, pp. 2718–2723, 2013.

[12] S. Zhu, M. Hong, and B. Chen, "Quantized consensus ADMM for multi-agent distributed optimization," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 4134–4138.

[13] S.-C. Tsai, Y.-H. Tseng, and T.-H. Chang, "Communication-efficient distributed demand response: A randomized ADMM approach," *IEEE Transactions on Smart Grid*, vol. 8, no. 3, pp. 1085–1095, 2017.

[14] Y. Liu, G. Wu, Z. Tian, and Q. Ling, "DQC-ADMM: Decentralized dynamic ADMM with quantized and censored communications," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 8, pp. 3290–3304, 2022.

[15] A. I. Rikos, W. Jiang, T. Charalambous, and K. H. Johansson, "Distributed optimization with gradient descent and quantized communication," in *Proceedings of 22nd IFAC World Congress*, 2023, pp. 6433–6439.

[16] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the ADMM in decentralized consensus optimization," *IEEE Transactions on Signal Processing*, vol. 62, no. 7, pp. 1750–1761, 2014.

[17] J. Wei, X. Yi, H. Sandberg, and K. H. Johansson, "Nonlinear consensus protocols with applications to quantized communication and actuation," *IEEE Transactions on Control of Network Systems*, vol. 6, no. 2, pp. 598–608, 2019.

[18] S. Giannini, D. Di Paola, A. Petitti, and A. Rizzo, "On the convergence of the max-consensus protocol with asynchronous updates," in *Proceedings of IEEE Conference on Decision and Control (CDC)*, 2013, pp. 2605–2610.

[19] S. Boyd, N. Parikh, and E. Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[20] E. Wei and A. Ozdaglar, "Distributed alternating direction method of multipliers," in *Proceedings of the IEEE Conference on Decision and Control*, 2012, pp. 5445–5450.

[21] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," in *Concurrency: the Works of Leslie Lamport*, 2019, pp. 179–196.

[22] Advanced configuration and power interface (ACPI) specification - ACPI specification 6.4 documentation. [Online]. Available: https://uefi.org/htmlspecs/ACPI_Spec_6_4_html/

[23] A. I. Rikos, C. N. Hadjicostis, and K. H. Johansson, "Non-oscillating quantized average consensus over dynamic directed topologies," *Automatica*, vol. 146, p. 110621, 2022.

[24] W. Jiang and T. Charalambous, "Distributed alternating direction method of multipliers using finite-time exact ratio consensus in digraphs," in *European Control Conference*, 2021, pp. 2205–2212.