Distributed Optimization with Gradient Descent and Quantized Communication [⋆]

Apostolos I. Rikos [*] Wei Jiang [**]
Themistoklis Charalambous [***] Karl H. Johansson [*]

[*] *Division of Decision and Control Systems, KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden, and Digital Futures, SE-100 44 Stockholm, Sweden. E-mails:* {rikos,kallej}@kth.se
[**] *Department of Electrical Engineering and Automation, School of Electrical Engineering, Aalto University, Espoo, Finland. E-mail:* wei.jiang@aalto.fi
[***] *Department of Electrical and Computer Engineering, School of Engineering, University of Cyprus, Nicosia, Cyprus. E-mail:* charalambous.themistoklis@ucy.ac.cy

Abstract: In this paper, we consider the unconstrained distributed optimization problem, in which the exchange of information in the network is captured by a directed graph topology, thus, nodes can only communicate with their neighbors. Additionally, in our problem, the communication channels among the nodes have limited bandwidth. In order to alleviate this limitation, quantized messages should be exchanged among the nodes. For solving this distributed optimization problem, we combine a gradient descent method with a distributed quantized consensus algorithm (which requires the nodes to exchange quantized messages and converges in a finite number of steps). Specifically, at every optimization step, each node (i) performs a gradient descent step (i.e., subtracts the scaled gradient from its current estimate), and (ii) performs a finite-time calculation of the quantized average of every node's estimate in the network. As a consequence, this algorithm approximately mimics the centralized gradient descent algorithm. We show that our algorithm asymptotically converges to a neighborhood of the optimal solution with linear convergence rate. The performance of the proposed algorithm is demonstrated via simple illustrative examples.

Keywords: Distributed optimization, quantized communication, directed graphs, finite-time consensus.

1. INTRODUCTION

The problem of distributed optimization has received extensive attention over the recent years from the control and machine learning communities, due to the wide area of applications in research areas such as resource allocation (Rikos et al., 2021; Doostmohammadian et al., 2022), sensor networks (Zhu et al., 2013), smart grids (Cady et al., 2015), and federated learning (Reisizadeh et al., 2020). The main idea is to optimize a global objective function by utilizing multiple nodes over a distributed network. Specifically, each node has access to a local function which is part of the global objective function. Each node aims to optimize the global objective function by optimizing its own local objective function and then coordinating with other nodes in the network. The distributed optimization problem can be defined formally as following:

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

where n is the number of nodes in the network, $x \in \mathbb{R}^p$ is the decision variable that all nodes are trying to optimize, $f_i : \mathbb{R}^p \mapsto \mathbb{R}$, is the local cost function for every node $v_i = \{1, 2, ..., n\}$, and $F : \mathbb{R}^p \mapsto \mathbb{R}$, is the global cost function. Note that due to the distributed nature of the problem, each node communicates only with its neighbors in the underlying network. This means that each node is required to perform only local operations such as sensing, communication, and computation in order to cooperatively solve the problem in (1).

For solving the distributed problem in (1), there are two main optimization methods: (i) primal-based, and (ii) dual-based. Examples of algorithms which rely on primal-based optimization methods are gradient/subgradient descent (see, e.g., Nedic and Ozdaglar (2009)). Examples of algorithms that rely on dual-based optimization methods are alternating direction method of multipliers (ADMM) algorithms (see, e.g., Makhdoumi and Ozdaglar (2017); Khatana and Salapaka (2020); Jiang and Charalambous (2021) and references therein). In this paper, we focus on primal-based optimization algorithms.

[⋆] This work was supported by the Knut and Alice Wallenberg Foundation and the Swedish Research Council. It was also partly supported by the project MINERVA, funded by the European Research Council (ERC) under the European Union's Horizon 2022 research and innovation programme (Grant agreement No. 101044629).

There have been various primal-based approaches for distributed optimization in the literature. In Nedic and Ozdaglar (2009), the authors present a distributed optimization algorithm known as distributed gradient / subgradient descent. This work converges to the optimal solution with sub-linear convergence rate due to the diminishing step size. Improving convergence rate of Nedic and Ozdaglar (2009) is possible by utilizing a constant step size. However, utilizing a constant step size leads to convergence to a neighborhood of the optimal solution. In Chen and Ozdaglar (2012); Jakovetic et al. (2014), the authors presented distributed optimization algorithms which execute multiple average consensus steps between each gradient descent update. In this way, they achieved faster convergence rate. A major drawback of this technique is that the average consensus steps achieve only asymptotic convergence and impose heavy communication requirements over the nodes and the network. Qu and Li (2018) presented a distributed optimization algorithm which achieves linear convergence rate. In order to achieve linear convergence rate, each node relies on historic gradient information and executes one dynamic average consensus step after each gradient descent update. In Nedic et al. (2017), the authors present the DIGing algorithm which achieves geometric convergence rate as long as some constraints over the fixed step size are fulfilled. In Shi et al. (2015), the authors present the EXTRA algorithm which uses a gradient difference strategy and achieves a geometric convergence rate. In Xin and Khan (2018), the authors present a distributed algorithm that geometrically converges to the global minimizer with a sufficiently small step-size. The proposed algorithm is based on an inexact gradient method and a gradient estimation technique. In Pu et al. (2021), the authors present the Push–Pull algorithm which converges to the optimal solution in a linear fashion. The Push–Pull algorithm pushes the information about the gradients to its neighbors, and pulls information about the decision variable from them.

It is important to note that most algorithms in current literature (and all the aforementioned works) assume the processing and exchange of real-valued messages between the nodes in the network. For large-scale networks with possibly limited bandwidth capacity, the communication overhead during each iteration becomes a major bottleneck. Quantization of information is one of the major approaches to overcome this issue. The main idea of quantization is that nodes transmit a compressed value (i.e., quantized) of their stored information as they require a few bits for representation compared to the non-compressed ones (i.e., real values) which in theory require infinite number of bits. For this reason, communication efficient distributed optimization has received significant attention recently in the control and machine learning communities. Specifically, in Pu et al. (2015), the authors present a distributed optimization algorithm with progressive quantization. They improve the convergence speed via a warm-starting strategy, and show that there exists a trade-off between accuracy and required number of iterations for convergence. In Koloskova et al. (2019) the authors present a gossip-based stochastic gradient descent algorithm, which utilizes arbitrary compressed messages and exhibits linear convergence. In Basu et al. (2019), the authors present a distributed optimization algorithm which combines aggressive sparsification with quantization. The algorithm keeps track of the difference between the true and compressed gradients, and converges with equal convergence rate as its non-quantized version. In Ivkin et al. (2019), the authors introduce SKETCHED-SGD. This algorithm performs distributed sub-gradient decent (SGD) by communicating sketches instead of full gradients. In Reisizadeh et al. (2020), the authors present a communication-efficient federated learning algorithm, which relies on periodic averaging and quantized message-passing, and achieves near-optimal theoretical guarantees. In Li et al. (2020), the authors present a distributed optimization algorithm which employs stochastic variance reduction and achieves linear convergence rate. In Khatana et al. (2020), the authors present a distributed optimization algorithm, called gradient-consensus. In this algorithm an *approximate* finite-time consensus protocol is combined with gradient descent. In Jiang and Charalambous (2022) the authors propose a distributed algorithm which combines gradient descent and finite-time *exact* ratio consensus. Both Khatana et al. (2020); Jiang and Charalambous (2022) achieve linear convergence rate.

We emphasize that most current approaches mainly focused on methods which are mainly quantizing values of an asymptotic coordination algorithm and are only able to exhibit asymptotic convergence to the consensus value, rendering them inappropriate for use in consensus-based distributed optimization methods. Notwithstanding this, the problem of how to design communication-efficient algorithms suitable for consensus-based distributed optimization still remains largely unexplored.

Main Contributions. It is often assumed that the exchange of information among agents is seamless and the exact value is communicated. However, in most cases, the exact value is an irrational number, whose transmission would require an infinite number of bits. Hence, if the channel has limited capacity, most of the current distributed approaches are impractical. Additionally, most quantized consensus methods are mainly quantizing values of an asymptotic consensus algorithm and, as a result, they do not converge in a finite number of steps. In this paper, a quantized consensus approach is combined with a gradient decent algorithm yielding the following appealing characteristics:

- exchange of quantized information, which complies with channels of limited bandwidth;
- the operation of the proposed algorithm relies on a novel finite-time quantized averaging strategy. The averaging strategy adjusts to the required quantization level and exhibits distributed stopping capabilites. Our algorithm's performance is equivalent to approximate centralized gradient descent iteration to solve the distributed optimization problem.

The main idea behind our proposed algorithm is the following. Initially, each node stores its quantized estimation regarding the optimal solution. During the algorithm's operation, each node performs a gradient descent step (i.e., subtracts the scaled gradient from its current estimate). Then, each node performs a finite-time calculation of the quantized average of every node's estimate in the network. This operation allows our algorithm to asymptotically converge to a neighborhood of the optimal solution with linear convergence rate.

This work was inspired by Khatana et al. (2020); Jiang and Charalambous (2022). However, our algorithm is substantially different than Jiang and Charalambous (2022); Khatana et al. (2020) in terms of operation and operational advantages. Specifically, the operation of our proposed algorithm relies on a novel quantized averaging algorithm adjusted to the desired quantization level. Furthermore, due to its quantized operation, in practical scenarios our proposed algorithm may exhibit (i) reduced complexity (i.e., quantized values can be represented using fewer bits than real values), (ii) faster computation (i.e., operating on quantized values can be faster than operating on real values because the former requires fewer computational resources), and (iii) improved accuracy in certain cases such as processing signals that have low signal-to-noise ratios (i.e., in some cases, using quantized values can actually improve the accuracy of the algorithm, e.g., when processing signals that have low signal-to-noise ratios, quantization can help to reduce the effects of noise and improve the signal quality).

2. NOTATION AND BACKGROUND

Notation. We denote the following sets of numbers: real $\mathbb{R}$, rational $\mathbb{Q}$, integer $\mathbb{Z}$, and natural $\mathbb{N}$. The set of nonnegative integers is denoted as $\mathbb{Z}_{\geq 0}$. The set of positive rationals is denoted as $\mathbb{Q}_{>0}$. For any $a \in \mathbb{R}$, the greatest integer less than or equal to a is denoted $\lfloor a \rfloor$, and the least integer greater than or equal to a is denoted as $\lceil a \rceil$. 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. The Euclidean norm of a vector is denoted as $\|x\|$. By $\mathbf{1}$ we denote the all-ones vector and by I we denote the identity matrix (of appropriate dimensions). By ∇ we denote the standard derivative of a function.

Graph Theory. The communication topology of the network consists of n ($n \geq 2$) nodes communicating only with their immediate neighbors. This can be captured by a directed graph (digraph) defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. In $\mathcal{G}$, the set of nodes is denoted as $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$, and the set of edges as $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} \cup \{(v_j, v_j) \mid v_j \in \mathcal{V}\}$ (note that each node has also a virtual self-edge). The cardinality of the set of nodes is denoted as $|\mathcal{V}| = n$, and the cardinality of the set of edges 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). 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 $\mathcal{D}_j^- = |\mathcal{N}_j^-|$. The subset of nodes that can directly receive information from node v_j is called the set of out-neighbors of v_j and is represented by $\mathcal{N}_j^+ = \{v_l \in \mathcal{V} \mid (v_l, v_j) \in \mathcal{E}\}$. The cardinality of $\mathcal{N}_j^+$ is called the *out-degree* of v_j and is denoted by $\mathcal{D}_j^+ = |\mathcal{N}_j^+|$. A directed *path* of length t 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 $\tau = 0, 1, \ldots, t-1$. 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.

Node Operation. At each time step $k \in \mathbb{Z}_{\geq 0}$ each node v_j maintains: (i) its local estimate variable $x_j^{[k]} \in \mathbb{Q}$ which is used to calculate the optimal solution, (ii) the stopping variables M_j, $m_j \in \mathbb{Q}$, which are used to determine whether convergence has been achieved, (iii) the mass variables $y_j^{[k]} \in \mathbb{Q}$ and $z_j^{[k]} \in \mathbb{Q}$, which are used to communicate with other nodes by either transmitting or receiving messages, and (iv) the state variables $y_{j,(s)}^{[k]} \in \mathbb{Q}$, $z_{j,(s)}^{[k]} \in \mathbb{Q}$ and $q_{j,(s)}^{[k]} = y_{j,(s)}^{[k]} / z_{j,(s)}^{[k]}$, which are used to store the received messages and calculate the result of the optimization operation.

Synchronous max/min **- Consensus.** The distributed max-consensus algorithm computes the maximum value of the network in a finite number of time steps (see Cortés (2008)). Every node $v_j \in \mathcal{V}$ updates its state in a synchronous fashion with the following update rule:

$$q_j^{[k+1]} = \max_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} \{q_i^{[k]}\}. \tag{2}$$

The max-consensus algorithm converges to the maximum value among all nodes in a finite number of steps s_m, where $s_m \leq D$ (see, (Giannini et al., 2013, Theorem 5.4)). Similar results hold for the min-consensus algorithm.

Asymmetric Quantizer. In distributed networks, quantization is a common procedure to reduce the required communication bandwidth and to increase power and computation efficiency. Quantization lessens the number of bits needed to represent information. It is mainly used to describe communication constraints and imperfect information exchanges between nodes (Wei et al., 2019). The three main types of quantizers are (i) uniform, (ii) asymmetric, and (iii) logarithmic. In this paper we rely on asymmetric quantizers to lessen the number of bits needed to represent information (but the results can also be extended to logarithmic and uniform quantizers). Assymetric quantizers are defined as

$$q_\Delta^a(\xi) = \left\lfloor \frac{\xi}{\Delta} \right\rfloor \Delta, \tag{3}$$

where $\xi \in \mathbb{R}$ is the value to be quantized, $q_\Delta^a(\xi) \in \mathbb{Q}$ is the quantized version of ξ, and $\Delta \in \mathbb{Q}$ is the quantization level.

3. PROBLEM FORMULATION

Let us consider a digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ nodes. Each node v_j is endowed with a local cost function $f_j(x) : \mathbb{R}^P \mapsto \mathbb{R}$ only known to node v_j. We aim to develop a distributed algorithm which allows nodes to cooperatively solve the following optimization problem $P1$:

$$\min_{x \in \mathcal{X}} \; F(x_1, x_2, ..., x_n) \equiv \sum_{i=1}^{n} f_i(x_i), \tag{4a}$$

$$\text{s.t. } x_i = x_j, \forall v_i, v_j, \in \mathcal{V}, \tag{4b}$$

$$x_i^{[0]} \in \mathcal{X} \subset \mathbb{Q}_{\geq 0}, \forall v_i \in \mathcal{V}, \tag{4c}$$

$$\text{nodes communicate with quantized values.} \tag{4d}$$

We denote $\mathcal{X}$ the set of feasible values of parameter x, and x^* the optimal solution of the optimization problem. Eq. (4a) means that we aim to minimize the global cost function which is defined as the sum of the local cost functions in the network. Eq. (4b) means that nodes need to calculate equal optimal solutions. Eq. (4c) means that the initial estimations of nodes belong in a common set. Note that it is not necessary for the initial values of nodes to be rational numbers, i.e., $x_i^{[0]} \in \mathcal{X} \subset \mathbb{Q}_{\geq 0}$. However, nodes can generate a quantized version of their initial states by utilizing the Asymetric Quantizer presented in Section 2. Eq. (4d) means that nodes are transmitting and receiving quantized values with their neighbors.

4. FINITE TIME DISTRIBUTED OPTIMIZATION WITH AVERAGED QUANTIZED GRADIENTS

In this section we present a distributed algorithm which solves the problem described in Section 3. Our distributed algorithm is detailed below as Algorithm 1 (with name QuAGD). Before presenting QuAGD, we consider the following assumptions for the development of the results in this paper.

Assumption 1. We assume $\mathcal{G}$ is *strongly connected*. This means that there exists a directed *path* from v_i to v_j, for every $v_j, v_i \in \{\mathcal{V} \mid v_j \neq v_i\}$.

Assumption 2. For every node v_j, the local cost function $f_j(x)$ is smooth and strongly convex. This means that for every node v_j, for every $x_1, x_2, \in \mathcal{X}$,

- there exists positive constant L_j such that
$$\|\nabla f_j(x_1) - \nabla f_j(x_2)\|_2 \leq L_j \|x_1 - x_2\|_2, \quad (5)$$
- there exists positive constant μ_j such that
$$f_j(x_2) \geq f_j(x_1) + \nabla f_j(x_1)^\top (x_2 - x_1) + \frac{\mu_j}{2}\|x_2 - x_1\|_2^2. \quad (6)$$

This means that the Lipschitz-continuity and strong-convexity constants of the global cost function F (see (4a)) are L, μ, respectively (L, μ are defined later in Theorem 1).

Assumption 3. Every node $v_j \in \mathcal{V}$ knows the diameter of the network D or an upper bound D' (i.e., $D' \geq D$), and a common quantization level Δ.

Assumption 1 is a necessary condition so that each node is able to calculate the optimal solution x^* of $P1$.

In assumption 2, Lipschitz-continuity (see (5)) is a necessary condition that guarantees the existence of the solution. Lipschitz-continuity is a standard assumption in distributed first-order optimization problems (see Xu et al. (2018); Qu and Li (2018)) and guarantees that nodes are able to calculate the global optimal minimizer x^* for (4a). Also, strong-convexity (see (6)) is useful for guaranteeing a linear convergence rate and that the global function F has no more than one minimum.

Assumption 3 allows each node to determine whether it has calculated the quantized average of every node's estimate of the optimal solution in finite time (and then proceed to perform gradient descent).

4.1 Quantized Averaged Gradient Descent Algorithm

The details of the distributed optimization algorithm can be seen in Algorithm 1.

Algorithm 1 Quantized Averaged Gradient Descent (QuAGD)

```
Input: A strongly connected digraph $\mathcal{G}$ with $n = |\mathcal{V}|$ nodes and $m = |\mathcal{E}|$ edges. Static step-size $\alpha \in \mathbb{R}$, digraph diameter $D$, initial value $x_j^{[0]}$, local cost function $f_j$, quantization level $\Delta \in \mathbb{Q}$, for every node $v_j \in \mathcal{V}$.
Iteration: For $k = 0, 1, 2, \ldots$, each node $v_j \in \mathcal{V}$ does:
1) $x_j^{[k+\frac{1}{2}]} = x_j^{[k]} - \alpha \nabla f_j(x_j^{[k]})$;
2) $x_j^{[k+1]} =$ Algorithm 1a$(x_j^{[k+\frac{1}{2}]}, D, \Delta)$;
Output: Each node $v_j \in \mathcal{V}$ calculates $x^*$ which solves problem $P1$ in Section 3.
```

The intuition of Algorithm 1 (QuAGD) is the following. Initially, each node maintains an estimate of the optimal solution, and the desired quantization level. Quantization level (i) is the same for every node, (ii) allows quantized communication between nodes, and (iii) determines the desired precision of the solution. At each time step k, each node updates the estimate of the optimal solution by performing a gradient descent step. This step is performed towards the negative direction the node's gradient. Then, each node utilizes a fast asymmetrically quantized averaging algorithm[1] Algorithm 1a (FAQuA). FAQuA allows each node to update its estimate of the optimal solution. Specifically, FAQuA allows each node to calculate the quantized average of each node's estimate in finite time by processing and transmitting quantized messages, with precision determined by the quantization level. The intuition of FAQuA is explained below.

The intuition of Algorithm 1a (FAQuA) is the following. FAQuA algorithm utilizes (i) asymmetric quantization, (ii) quantized averaging, and (iii) a stopping strategy. Specifically, each node v_j uses an asymmetric quantizer to its state and doubles its mass variables (this change has no effect on the average calculation). Then, at each time step λ, each node v_j checks if $z_j[\lambda] > 1$ (i) it updates its state variables to be equal to the mass variables and (ii) it *splits* $y_j[\lambda]$ into $z_j[\lambda]$ equal integer pieces (the value of some pieces might be greater than others by one). It chooses one piece with minimum y-value and transmits it to itself, and it transmits each of the remaining $z_j[\lambda] - 1$ pieces to randomly selected out-neighbors or to itself. It receives the values $y_i[\lambda]$ and $z_i[\lambda]$ from its in-neighbors, sums them with its stored $y_j[\lambda]$ and $z_j[\lambda]$ values and repeats the operation. Every D time steps performs a max and min consensus operation. If the stopping condition holds, it scales the solution according to the quantization level.

4.2 Convergence of Algorithm 1

We now analyze the convergence of Algorithm 1. Specifically, we show that during the operation of Algorithm 1,

[1] Algorithm 1a (FAQuA) runs between every two consecutive optimization steps k and $k+1$ of Algorithm 1 (QuAGD). For this reason Algorithm 1a uses a different time index λ (and not k as Algorithm 1).

Algorithm 1a FAQuA

```
Input: $x_j^{[k+\frac{1}{2}]}, D, \Delta$. Output: $x_j^{[k+1]}$.
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) $\text{flag}_j = 0$, $z_j^{[1]} = 2$, $y_j^{[1]} = 2\left\lfloor \frac{x_j^{[k+\frac{1}{2}]}}{\Delta} \right\rfloor$,
3) $y_{j,(s)}^{[1]} := y_j^{[1]}$, $z_{j,(s)}^{[1]} = z_j^{[1]}$, $q_{j,(s)}^{[1]} := y_{j,(s)}^{[1]}/z_{j,(s)}^{[1]}$,
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 $v_l$.
Iteration: For $\lambda = 1, 2, \dots$, each node $v_j \in \mathcal{V}$, does:
1) if $\lambda \mod D = 1$ then sets $M_j = \lceil y_{j,(s)}^{[\lambda]}/z_{j,(s)}^{[\lambda]} \rceil$, $m_j = \lfloor y_{j,(s)}^{[\lambda]}/z_{j,(s)}^{[\lambda]} \rfloor$;
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) if $z_j^{[\lambda]} > 1$, then
   5.1) sets $z_{j,(s)}^{[\lambda]} = z_j^{[\lambda]}$, $y_{j,(s)}^{[\lambda]} = y_j^{[\lambda]}$, $q_{j,(s)}^{[\lambda]} = \left\lfloor \frac{y_{j,(s)}^{[\lambda]}}{z_{j,(s)}^{[\lambda]}} \right\rfloor$;
   5.2) sets (i) $mas^{y,[\lambda]} = y_j^{[\lambda]}$, $mas^{z,[\lambda]} = z_j^{[\lambda]}$; (ii) $c_{lj}^{y,[\lambda]} = 0$, $c_{lj}^{z,[\lambda]} = 0$, for every $v_l \in \mathcal{N}_j^+ \cup \{v_j\}$; (iii) $\delta = \lfloor mas^{y[\lambda]}/mas^{z,[\lambda]} \rfloor$, $mas^{rem,[\lambda]} = y_j^{[\lambda]} - \delta\ mas^{z,[\lambda]}$;
   5.3) while $mas^{z,[\lambda]} > 1$, then
      5.3a) chooses $v_l \in \mathcal{N}_j^+ \cup \{v_j\}$ randomly according to $b_{lj}$;
      5.3b) sets (i) $c_{lj}^{z,[\lambda]} := c_{lj}^{z,[\lambda]} + 1$, $c_{lj}^{y,[\lambda]} := c_{lj}^{y,[\lambda]} + \delta$; (ii) $mas^{z,[\lambda]} := mas^{z,[\lambda]} - 1$, $mas^{y,[\lambda]} := mas^{y,[\lambda]} - \delta$.
      5.3c) If $mas^{rem,[\lambda]} > 1$, sets $c_{lj}^{y,[\lambda]} := c_{lj}^{y,[\lambda]} + 1$, $mas^{rem,[\lambda]} := mas^{rem,[\lambda]} - 1$;
   5.4) sets $c_{jj}^{y,[\lambda]} := c_{jj}^{y,[\lambda]} + mas^{y,[\lambda]}$, $c_{jj}^{z,[\lambda]} := c_{jj}^{z,[\lambda]} + mas^{z,[\lambda]}$;
   5.5) for every $v_l \in \mathcal{N}_j^+$, if $c_{lj}^{z,[\lambda]} > 0$ transmits $c_{lj}^{y,[\lambda]}$, $c_{lj}^{z,[\lambda]}$ to out-neighbor $v_l$;
• else if $z_j^{[\lambda]} \leq 1$, sets $c_{jj}^{y,[\lambda]} = y_j^{[\lambda]}$, $c_{jj}^{z,[\lambda]} = z_j^{[\lambda]}$;
6) receives $c_{ji}^{y,[\lambda]}$, $c_{ji}^{z,[\lambda]}$ from $v_i \in \mathcal{N}_j^-$ and sets
   $$y_j^{[\lambda+1]} = c_{jj}^{y,[\lambda]} + \sum_{i=1}^{n} w_{ji}^{[\lambda]}\ c_{ji}^{y,[\lambda]}, \quad (7)$$
   $$z_j^{[\lambda+1]} = c_{jj}^{z,[\lambda]} + \sum_{i=1}^{n} w_{ji}^{[\lambda]}\ c_{ji}^{z,[\lambda]}, \quad (8)$$
   where $w_{ji}^{[\lambda]} = 1$ if node $v_j$ receives $c_{jj}^{y,[\lambda]}$, $c_{jj}^{z,[\lambda]}$ from $v_i \in \mathcal{N}_j^-$ at iteration $k$ (otherwise $w_{ji}^{[\lambda]} = 0$);
7) if $\lambda \mod D = 0$ then, if $M_j - m_j \leq 1$ then sets $x_j^{[k+1]} = m_j\Delta$ and stops operation.
```

the variable $x_i^{[k]}$ of each node $v_i \in \mathcal{V}$ converges to a neighborhood of the optimal solution x^* with linear convergence rate. In Step 1) of Algorithm 1, for the convenience of presentation of mathematics in this section, we denote $z_i^{[k+1]} := x_i^{[k+\frac{1}{2}]}$. Thus, the Steps 1) and 2) in Algorithm 1 are changed to:

$$z_i^{[k+1]} = x_i^{[k]} - \alpha\nabla f_i(x_i^{[k]}), \tag{9}$$

$$x_i^{[k+1]} = \text{Algorithm } 1a(z_i^{[k+1]}, D, \Delta). \tag{10}$$

From (10) and the property of Algorithm 1a, we have

$$x_i^{[k+1]} = \frac{1}{n}\sum_{i=1}^{n} \Delta\left\lfloor \frac{z_i^{[k+1]}}{\Delta} \right\rfloor + \varrho_i^{[k+1]}, \text{ where } \|\varrho_i^{[k+1]}\| \leq \Delta, \tag{11}$$

for every $k \geq 0$, where $\varrho_i^{[k+1]}$ is the total error due to Algorithm 1a calculating the quantized average of the node's initial quantized states at optimization step k.

In addition, from the quantization definition, we have

$$x_i^{[k]} = \Delta\left\lfloor \frac{x_i^{[k]}}{\Delta} \right\rfloor + \epsilon_i^{[k]}, \text{ where } 0 \leq \epsilon_i^{[k]} \leq \Delta, \tag{12}$$

for every $v_i \in \mathcal{V}$, where $\epsilon_i^{[k]}$ is the error due to applying asymmetric quantization to the value $x_i^{[k]}$ (see Section 2) at time step k. Denote

$$\hat{z}^{[k+1]} := \frac{1}{n}\sum_{i=1}^{n} z_i^{[k+1]}, \hat{x}^{[k+1]} := \frac{1}{n}\sum_{i=1}^{n} x_i^{[k+1]}, k \geq 0.$$

Based on the quantizer property (12), it is easy to have

$$\begin{aligned} \hat{z}^{[k+1]} - x_i^{[k+1]} &= \frac{1}{n}\sum_{i=1}^{n} z_i^{[k+1]} - \frac{1}{n}\sum_{i=1}^{n} \Delta\left\lfloor \frac{z_i^{[k+1]}}{\Delta} \right\rfloor - \varrho_i^{[k+1]} \\ &= \frac{1}{n}\sum_{i=1}^{n} \epsilon_i^{[k+1]} - \varrho_i^{[k+1]} \leq 2\Delta. \end{aligned} \tag{13}$$

Lemma 1. For $k \geq 1$, the following inequalities hold:

$$\|\hat{x}^{[k]} - \hat{z}^{[k]}\| \leq 2\Delta, \tag{14}$$

$$\|x_i^{[k]} - \hat{x}^{[k]}\| \leq 4\Delta. \tag{15}$$

Proof. From (11), we have

$$\begin{aligned} \|\hat{x}^{[k]} - \hat{z}^{[k]}\| &= \|\frac{1}{n}\sum_{i=1}^{n}(x_i^{[k]} - z_i^{[k]})\| \\ &= \|\frac{1}{n}\sum_{i=1}^{n}(\varrho_i^{[k]} - z_i^{[k]} + \frac{1}{n}\sum_{i=1}^{n}\Delta\left\lfloor \frac{z_i^{[k]}}{\Delta} \right\rfloor)\| \\ &\leq \|\frac{1}{n}\sum_{i=1}^{n}\varrho_i^{[k]}\| + \|\frac{1}{n}\sum_{i=1}^{n}(\Delta\left\lfloor \frac{z_i^{[k]}}{\Delta} \right\rfloor - z_i^{[k]})\| \\ &\leq 2\Delta, \end{aligned} \tag{16}$$

which proves (14).

Based on (13) and (14), we have

$$\|x_i^{[k]} - \hat{x}^{[k]}\| \leq \|x_i^{[k]} - \hat{z}^{[k]}\| + \|\hat{z}^{[k]} - \hat{x}^{[k]}\| \leq 4\Delta,$$

which can prove (15). The proof is finished.

Denote

$$u^{[k]} := \sum_{i=1}^{n} \nabla f_i(x_i^{[k]}), \hat{u}^{[k]} := \sum_{i=1}^{n} \nabla f_i(\hat{x}^{[k]}).$$

From Assumption 2 with (15), we have

$$\|\nabla f_i(x_i^{[k]}) - \nabla f_i(\hat{x}^{[k]})\| \leq L_i \|x_i^{[k]} - \hat{x}^{[k]}\| \leq 4L_i\Delta \quad (17)$$

$$\|u^{[k]} - \hat{u}^{[k]}\| \leq \sum_{i=1}^{n} \|\nabla f_i(x_i^{[k]}) - \nabla f_i(\hat{x}^{[k]})\|$$

$$\leq \sum_{i=1}^{n} L_i \|x_i^{[k]} - \hat{x}^{[k]}\| \leq 4nL\Delta, L = \max\{L_i\}. \quad (18)$$

Denote $e^{[k]} = \hat{x}^{[k]} - \hat{z}^{[k]}, \hat{\alpha} := \frac{\alpha}{n}$. From (14) we have $\|e^{[k]}\| \leq 2\Delta$. From (9) we get $\hat{x}^{[k+1]} = \hat{z}^{[k+1]} + e^{[k+1]} = \frac{1}{n}\sum_{i=1}^{n}(x_i^{[k]} - \alpha\nabla f_i(x_i^{[k]})) + e^{[k+1]} = \hat{x}^{[k]} - \hat{\alpha}u^{[k]} + e^{[k+1]}$, $k \geq 1$.

At present, we are ready to provide the result of the step-size upper bound and algorithm convergence rate in the following theorem. However, due to space limitations we omit the proof of Theorem 1. It will be available at an extended version of our paper.

Theorem 1. Under Assumptions 1–3, when the step-size α satisfies $\alpha \in (\frac{n(\mu+L)}{4\mu L}, \frac{2n}{\mu+L})$ and $\delta \in (0, \frac{n[4\alpha\mu L - n(\mu+L)]}{2\alpha[n(\mu+L)-2\alpha\mu L]})$ where $L = \max\{L_i\}, \mu = \min\{\mu_i\}$, Algorithm 1 generates a sequence of points $\{x^{[k]}\}$ (i.e., the variable $x_i^{[k]}$ of each node $v_i \in \mathcal{V}$) which satisfy

$$\|\hat{x}^{[k+1]} - x^*\|^2 < \vartheta\|\hat{x}^{[k]} - x^*\|^2 + \mathcal{O}(\Delta^2), \quad (19)$$

where Δ is the quantizer and

$$\vartheta := 2(1 + \frac{\alpha\delta}{n})(1 - \frac{2\alpha\mu L}{n(\mu + L)}) \in (0, 1), \quad (20a)$$

$$\mathcal{O}(\Delta^2) = (8 + 32n^2\hat{\alpha}^2L^2 + \frac{32n^2\hat{\alpha}L^2}{\delta})\Delta^2. \quad (20b)$$

Theorem 1 shows that Algorithm 1 converges linearly to a neighborhood of the optimal solution. This neighborhood is determined by the quantization level Δ.

Remark 1. Based on the Theorem 1, if we have that $\alpha \in (\frac{n(\mu+L)}{4\mu L}, \frac{2n}{\mu+L})$, then we always have $\frac{n[4\alpha\mu L - n(\mu+L)]}{2\alpha[n(\mu+L)-2\alpha\mu L]} > 0$, thus, δ exists. Also, we always have $\vartheta \in (0, 1)$. For the step-size interval $\alpha \in (\frac{n(\mu+L)}{4\mu L}, \frac{2n}{\mu+L})$ to be always not empty, we need $\frac{n(\mu+L)}{4\mu L} < \frac{2n}{\mu+L}$. This means that we need $(L - \mu)^2 < 4\mu L$. Herein, we could provide a sufficient condition to have this interval be always available. Due to $\mu \leq L$, if we get $(L - \mu)^2 < 4\mu^2$, then, we always have $(L-\mu)^2 < 4\mu L$. This means that when $L < 3\mu$, the interval for the step-size α is always not empty.

5. SIMULATION RESULTS

In this section, we present simulation results in order to demonstrate the operation of Algorithm 1. We focus on a random digraph of 20 nodes and show how the nodes' states converge to the optimal solution for various quantization levels. Furthermore, we present comparisons against existing algorithms, and we emphasize on the improvements introduced by Algorithm 1.

In Fig. 1, we compare the operation of Algorithm 1 for different quantization levels with Jiang and Charalambous (2022); Khatana et al. (2020). We plot the error $e^{[k]}$ in a logarithmic scale against the number of iterations. The error $e^{[k]}$ is defined as

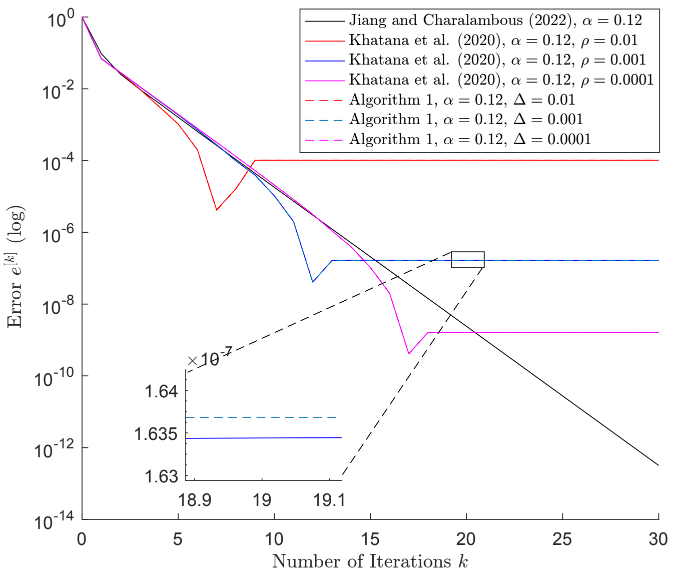

Fig. 1. Comparison of Algorithm 1 for different quantization levels with Jiang and Charalambous (2022); Khatana et al. (2020).

$$e^{[k]} = \sqrt{\sum_{j=1}^{n} \frac{(x_j^{[k]} - x^*)^2}{(x_j^{[0]} - x^*)^2}}, \quad (21)$$

where x^* is the optimal solution of the optimization problem $P1$.

We can see that Algorithm 1 exhibits almost equal performance compared with Khatana et al. (2020), for the case where the quantization level is equal to the pre-specified tolerance value ρ (see Khatana et al. (2020)). However, our proposed algorithm uses quantized values and hence can be used in channels with limited/finite capacity. Furthermore, Algorithm 1 exhibits comparable performance compared with Jiang and Charalambous (2022), even though the results of Jiang and Charalambous (2022) do not have an error floor. Note, however, that Algorithm 1 is able to approximate the optimal solution with precision that depends on the quantization level. This means that if we reduce the value of the quantization level, nodes are able to approximate the optimal solution with higher precision. Specifically, for the method in Jiang and Charalambous (2022), forming the Hankel matrix and performing additional computations when the matrix loses rank, requires the exact values from each node. This translates to nodes exchanging messages of infinite capacity. Thus, the main advantage of Algorithm 1 compared to Jiang and Charalambous (2022) is that nodes operate with quantized values (while in Jiang and Charalambous (2022) nodes exchange values of infinite prevision).

6. CONCLUSIONS AND FUTURE DIRECTIONS

In this paper, we focused on designing a communication-efficient algorithm for the unconstrained distributed optimization problem. Specifically, each node performs a gradient descent step, and then performs a finite-time calculation of the quantized average of every node's estimate in the network. This algorithm, to the best of the authors' knowledge, is the first distributed optimization algorithm which relies on a finite time quantized coordination mechanism which operates over directed graphs, and the only one with quantization used in this context of approximating the centralized gradient decent algorithm.

REFERENCES

Basu, D., Data, D., Karakus, C., and Diggavi, S. (2019). Qsparse-local-SGD: Distributed SGD with quantization, sparsification and local computations. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Cady, S.T., Domínguez-García, A.D., and Hadjicostis, C.N. (2015). Finite-time approximate consensus and its application to distributed frequency regulation in islanded AC microgrids. In *Proceedings of Hawaii International Conference on System Sciences*, 2664–2670.

Chen, A.I. and Ozdaglar, A. (2012). A fast distributed proximal-gradient method. In *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 601–608.

Cortés, J. (2008). Distributed algorithms for reaching consensus on general functions. *Automatica*, 44, 726–737.

Doostmohammadian, M., Aghasi, A., Rikos, A.I., Grammenos, A., Kalyvianaki, E., Hadjicostis, C.N., Johansson, K.H., and Charalambous, T. (2022). Distributed anytime-feasible resource allocation subject to heterogeneous time-varying delays. *IEEE Open Journal of Control Systems*, 1, 255–267.

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

Ivkin, N., Rothchild, D., Ullah, E., braverman, V., Stoica, I., and Arora, R. (2019). Communication-efficient distributed SGD with sketching. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Jakovetic, D., Xavier, J., and Moura, J.M.F. (2014). Fast distributed gradient methods. *IEEE Transactions on Automatic Control*, 59(5), 1131–1146.

Jiang, W. and Charalambous, T. (2022). A fast finite-time consensus based gradient method for distributed optimization over digraphs. In *Proceedings of IEEE Conference on Decision and Control (CDC)*, 6848–6854.

Jiang, W. and Charalambous, T. (2021). Distributed alternating direction method of multipliers using finite-time exact ratio consensus in digraphs. In *European Control Conference*, 2205–2212.

Khatana, V., Saraswat, G., Patel, S., and Salapaka, M.V. (2020). Gradient-consensus method for distributed optimization in directed multi-agent networks. In *American Control Conference (ACC)*, 4689–4694.

Khatana, V. and Salapaka, M.V. (2020). D-DistADMM: A O(1/k) distributed admm for distributed optimization in directed graph topologies. In *2020 59th IEEE Conference on Decision and Control (CDC)*, 2992–2997.

Koloskova, A., Stich, S., and Jaggi, M. (2019). Decentralized stochastic optimization and gossip algorithms with compressed communication. In *Proceedings of the 36th International Conference on Machine Learning*, 3478–3487.

Li, B., Cen, S., Chen, Y., and Chi, Y. (2020). Communication-efficient distributed optimization in networks with gradient tracking and variance reduction. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, 1662–1672.

Makhdoumi, A. and Ozdaglar, A. (2017). Convergence rate of distributed ADMM over networks. *IEEE Transactions on Automatic Control*, 62(10), 5082–5095.

Nedic, A. and Ozdaglar, A. (2009). Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1), 48–61.

Nedic, A., Olshevsky, A., and Shi, W. (2017). Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM Journal on Optimization*, 27(4), 2597–2633.

Pu, S., Shi, W., Xu, J., and Nedic, A. (2021). Push–pull gradient methods for distributed optimization in networks. *IEEE Transactions on Automatic Control*, 66(1), 1–16.

Pu, Y., Zeilinger, M.N., and Jones, C.N. (2015). Quantization design for distributed optimization with time-varying parameters. In *54th IEEE Conference on Decision and Control (CDC)*, 2037–2042.

Qu, G. and Li, N. (2018). Harnessing smoothness to accelerate distributed optimization. *IEEE Transactions on Control of Network Systems*, 5(3), 1245–1260.

Reisizadeh, A., Mokhtari, A., Hassani, H., Jadbabaie, A., and Pedarsani, R. (2020). FedPAQ: A communication-efficient federated learning method with periodic averaging and quantization. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, 2021–2031.

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

Shi, W., Ling, Q., Wu, G., and Yin, W. (2015). EXTRA: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2), 944–966.

Wei, J., Yi, X., Sandberg, H., and Johansson, K.H. (2019). Nonlinear consensus protocols with applications to quantized communication and actuation. *IEEE Transactions on Control of Network Systems*, 6(2), 598–608.

Xin, R. and Khan, U.A. (2018). A linear algorithm for optimization over directed graphs with geometric convergence. *IEEE Control Systems Letters*, 2(3), 315–320.

Xu, J., Zhu, S., Soh, Y.C., and Xie, L. (2018). Convergence of asynchronous distributed gradient methods over stochastic networks. *IEEE Transactions on Automatic Control*, 63(2), 434–448.

Zhu, S., Chen, C., Li, W., Yang, B., and Guan, X. (2013). Distributed optimal consensus filter for target tracking in heterogeneous sensor networks. *IEEE Transactions on Cybernetics*, 43(6), 1963–1976.