2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Delay-agnostic Asynchronous Distributed Optimization

Xuyang Wu, Changxin Liu, Sindri Magnússon, and Mikael Johansson

Abstract— **Existing asynchronous distributed optimization algorithms often use diminishing step-sizes that cause slow practical convergence, or fixed step-sizes that depend on an assumed upper bound of delays. Not only is such a delay bound hard to obtain in advance, but it is also large and therefore results in unnecessarily slow convergence. This paper develops asynchronous versions of two distributed algorithms, DGD and DGD-ATC, for solving consensus optimization problems over undirected networks. In contrast to alternatives, our algorithms can converge to the fixed point set of their synchronous counterparts using step-sizes that are independent of the delays. We establish convergence guarantees under both partial and total asynchrony. The practical performance of our algorithms is demonstrated by numerical experiments.**

I. Introduction

Distributed optimization has attracted much attention in the last decade and has found applications in diverse areas such as cooperative control, machine learning, and power systems. The literature on distributed optimization has primarily focused on synchronous methods that iterate in a serialized manner, proceeding to the next iteration only after the current one is completed. Synchronous methods also require all nodes to maintain a consistent view of optimization variables without any information delay, which makes the algorithms easier to analyze. Nevertheless, synchronization through a network can be challenging. Additionally, synchronized update is inefficient and unreliable since the time taken per iteration is determined by the slowest node and the optimization process is vulnerable to single-node failure.

Asynchronous distributed methods that do not require synchronization between nodes are often better suited for practical implementation [1]. However, asynchronous methods are subject to information delays and nodes do not have a consistent view of the optimization variables, which makes them difficult to analyze. Despite the inherent challenges, there have been notable successes in studying the mathematical properties of asynchronous optimization algorithms. One area of focus has been on asynchronous consensus optimization algorithms [2]–[10], including asynchronous variants of well-established consensus optimization algorithms such as DGD, PG-EXTRA, and gradient-tracking-based methods. Asynchronous distributed algorithms on other optimization problems include ADGD [11], Asy-FLEXA [12], the asynchronous primal-dual algorithm [13], and the asynchronous coordinate descent method [14], [15].

The above work mainly focused on two types of step-size strategies: diminishing step-sizes [3]–[7] and fixed delay-dependent step-sizes [8]–[10], [12]–[14]. While diminishing step-sizes are effective in stochastic optimization or non-smooth optimization, they can result in slow convergence rates in deterministic smooth problems. For these types of problems, faster algorithms can often be obtained with non-diminishing step-sizes. Fixed step-sizes that depend on delay, in contrast, usually require an upper bound on the worst-case delay that is challenging to compute prior to executing the algorithm. Moreover, the use of worst-case delay can result in a conservative step-size condition and consequently, slow down the practical convergence speed. This is because the actual delays experienced in practice may be significantly smaller than the worst-case delay. For example, [16] implements an asynchronous SGD on a 40-core CPU, and reports a maximum and average delay of around 1200 and 40, respectively. Convergence of asynchronous distributed algorithms with fixed step-sizes that do not include any delay information have been considered in [2], [11], [15]. However, [2], [15] only consider quadratic programming and [11] studies only star networks.

In this paper, we study the asynchronous variants of two distributed algorithms, the decentralized gradient descent (DGD) [17] and the DGD using the adapt-then-combine technique (DGD-ATC) [18], for solving consensus optimization over undirected networks. Our contributions include:

1) We establish the optimality gap between the fixed point of DGD-ATC with fixed step-sizes and the optimum of the consensus optimization problem. This result is absent in the literature.
2) We show theoretically that, under the total asynchrony assumption, the two asynchronous methods can converge to the same fixed point sets of their synchronous counterparts with *fixed step-sizes that do not include delay information*.
3) We improve the above asymptotic convergence to linear convergence by assuming bounded information delays.

Compared to the delay-dependent fixed step-sizes, our proposed delay-free step-sizes are easy to tune and, in general, less restrictive. Although algorithms that use delay-dependent fixed step-sizes [8]–[10], [12]–[14] or diminishing step-sizes [3]–[7] can theoretically converge to the optimum while our algorithms suffer from unfavourable inexact convergence inherited from their synchronous counterparts, our

X. Wu, C. Liu, and M. Johansson are with the Division of Decision and Control Systems, School of EECS, KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden. Email: {xuyangw,changxin,mikaelj}@kth.se.

S. Magnússon is with the Department of Computer and System Science, Stockholm University, SE-164 07 Stockholm, Sweden. Email: sindri.magnusson@dsv.su.se.

This work was supported by WASP and the Swedish Research Council (Vetenskapsrådet) under grants 2019-05319 and 2020-03607.

algorithms may achieve faster practical convergence due to their less restrictive fixed step-sizes, which is demonstrated by numerical experiments.

The outline of this paper is as follows: Section II formulates the problem, revisits the synchronous algorithms DGD and DGD-ATC, and reviews/establishes their optimality error bounds. Section III introduces the asynchronous DGD and the asynchronous DGD-ATC, and Section IV provides convergence results. Finally, Section V tests the practical performance of the two asynchronous algorithms by numerical experiments and Section VI concludes the paper.

Notation and Preliminaries

We use $\mathbf{1}_d$, $\mathbf{0}_{d\times d}$, and I_d to denote the d-dimensional all-one vector, the $d \times d$ all-zero matrix, and the $d \times d$ identity matrix, respectively, where the subscript is omitted when it is clear from context. The notation $\otimes$ represents the Kronecker product and $\mathbb{N}_0$ is the set of natural numbers including 0. For any symmetric matrix $W \in \mathbb{R}^{n\times n}$, $\lambda_i(W)$, $1 \le i \le n$ denotes the ith largest eigenvalue of W, $\mathrm{Range}(W)$ is its range, and $W \succ \mathbf{0}$ means that W is positive definite. For any vector $x \in \mathbb{R}^n$, we use $\|x\|$ to represent the ℓ_2 norm and define $\|x\|_W = \sqrt{x^T W x}$ for any positive definite matrix $W \in \mathbb{R}^{n\times n}$. For any differentiable function $f : \mathbb{R}^d \to \mathbb{R}$, we say it is L-smooth for some $L > 0$ if

$$\|\nabla f(y) - \nabla f(x)\| \le L\|y - x\|,\ \forall x, y \in \mathbb{R}^d$$

and it is μ-strongly convex for some $\mu > 0$ if

$$\langle \nabla f(y) - \nabla f(x), y - x\rangle \ge \mu\|y - x\|^2,\ \forall x, y \in \mathbb{R}^d.$$

II. Problem Formulation and Synchronous distributed Algorithms

This section describes consensus optimization and revisits the synchronous distributed algorithms, DGD [17] and DGD-ATC [18], for solving it. The asynchronous version of the two methods will be introduced in Section III.

A. Consensus Optimization

Consider a network of n agents described by an undirected, connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \dots, n\}$ is the vertex set and $\mathcal{E} \subseteq \mathcal{V}\times\mathcal{V}$ is the edge set. In the network, each agent i observes a local cost function $f_i : \mathbb{R}^d \to \mathbb{R}$ and can only interact with its neighbors in $\mathcal{N}_i = \{j : \{i, j\} \in \mathcal{E}\}$. Consensus optimization aims to find a common decision vector that minimizes the total cost of all agents:

$$\underset{x\in\mathbb{R}^d}{\text{minimize}}\quad f(x) = \sum_{i\in\mathcal{V}} f_i(x). \tag{1}$$

Distributed algorithms for solving Problem (1) include the distributed subgradient method [19], DGD [17], distributed gradient-tracking-based algorithm [20], distributed dual averaging [21], and PG-EXTRA [22]. While these algorithms were originally designed to be executed synchronously, they have since been extended to allow for asynchronous implementations. However, existing asynchronous methods often suffer from slow convergence due to the use of either diminishing step-sizes or fixed step-sizes that depend on a (usually unknown and large) upper bound on all delays.

In this paper, we analyse the asynchronous version of two algorithms with delay-free fixed step-sizes: Decentralized Gradient Descent (DGD) and DGD using Adapt-Then-Combine Technique (DGD-ATC).

B. Decentralized Gradient Descent (DGD)

The first algorithm is DGD [17]. To present the algorithm compactly, define $\mathbf{x} = (x_1^T, \dots, x_n^T)^T \in \mathbb{R}^{nd}$, $F(\mathbf{x}) = \sum_{i\in\mathcal{V}} f_i(x_i)$, and let $\mathbf{W} = W \otimes I_d$ where W is an averaging matrix[1] associated with $\mathcal{G}$. We use $k \in \mathbb{N}_0$ as iteration index and $\mathbf{x}^k$ as the value of $\mathbf{x}$ at iteration k. Then the DGD algorithm progresses according to the following iterations:

$$\mathbf{x}^{k+1} = \mathbf{W}\mathbf{x}^k - \alpha\nabla F(\mathbf{x}^k), \tag{2}$$

where $\alpha > 0$ is the step-size.

As shown in [17], the DGD algorithm converges to a fixed point under reasonable assumptions. However, while the set of fixed points of DGD is not identical to the set of optimal solution of Problem (1), it is possible to bound the difference between the two sets under the following assumptions:

Assumption 1: Each f_i is proper closed convex, lower bounded, and L_i-smooth for some $L_i > 0$. Further, Problem (1) has a non-empty and bounded optimal solution set.

Assumption 2: Each f_i is μ_i-strongly convex.

We are now in a position to quantify the gap between the fixed point of DGD and the optimal solution. Define

$$L = \max_{i\in\mathcal{V}} L_i,\ \bar{L} = \frac{1}{n}\sum_{i\in\mathcal{V}} L_i, \tag{3}$$

$$\beta = \max\{|\lambda_2(W)|, |\lambda_n(W)|\}, \tag{4}$$

where $\beta \in (0, 1)$ since $\mathcal{G}$ is connected [20]. We first state the following lemma that follows similarly to Lemma 2 and Theorem 4 in [17].

Lemma 1: Suppose that Assumption 1 holds. If

$$\alpha \le \min\left(\frac{1 + \lambda_n(W)}{L}, \frac{1}{\bar{L}}\right),$$

then the fixed point set of DGD (2) is non-empty, and DGD converges to a point $\mathbf{x}^\star \in \mathbb{R}^{nd}$ satisfying

$$\|x_i^\star - \bar{x}^\star\| \le \frac{\alpha\sqrt{C}}{1 - \beta},\quad \forall i \in \mathcal{V}, \tag{5}$$

$$f(\bar{x}^\star) - f^\star \le \left(\frac{\alpha}{1 - \beta} + \frac{L\alpha^2}{2(1 - \beta)^2}\right) C, \tag{6}$$

where $\bar{x}^\star = \frac{1}{n}\sum_{i\in\mathcal{V}} x_i^\star$, $f^\star$ is the optimal value of (1), L, $\bar{L}$, and β are given in (3)–(4), and

$$C = 2L(f^\star - \sum_{i\in\mathcal{V}} \inf_{x_i\in\mathbb{R}^d} f_i(x_i)). \tag{7}$$

The fixed point is unique if, in addition, Assumption 2 holds.

[1]We say a matrix $W = (w_{ij}) \in \mathbb{R}^{n\times n}$ is an averaging matrix associated with $\mathcal{G}$ if it is non-negative, symmetric ($W = W^T$), stochastic ($W\mathbf{1} = \mathbf{1}$), and satisfies $w_{ij} = 0$ if and only if $\{i, j\} \notin \mathcal{E}$ and $i \ne j$. This matrix can be easily formed in a distributed manner, with many options listed in [23, Section 2.4].

C. DGD using Adapt-Then-Combine Technique (DGD-ATC)

DGD-ATC [18] is a variant of DGD that uses the adapt-then-combine technique and follows the update

$$\mathbf{x}^{k+1} = \mathbf{W}(\mathbf{x}^k - \alpha\nabla F(\mathbf{x}^k)), \tag{8}$$

where $\mathbf{W}$ is the same as in (2) and $\alpha > 0$ is the step-size.

We are unaware of any previous work that analyses the convergence of DGD-ATC with fixed step-sizes. In the lemma below, we show that DGD-ATC has a similar optimality gap as DGD. The convergence of DGD-ATC (8) follows as a special case of Theorem 2 in Section III.

Lemma 2: Suppose that Assumption 1 holds. If $W \succ \mathbf{0}$, then the fixed point set of DGD-ATC (8) is non-empty and for any fixed point $\mathbf{x}^\star \in \mathbb{R}^{nd}$, (5)–(6) hold. If, in addition, Assumption 2 holds, then the fixed point is unique.

III. Asynchronous distributed Algorithms

In this section, we introduce the asynchronous DGD and DGD-ATC algorithms. A key advantage of these algorithms is that they do not require global synchronization between nodes or a global clock. Both algorithms are analyzed in a setting where each node $i \in \mathcal{V}$ is activated at discrete time points, and can update and share its local variables once it is activated. In addition, every node $i \in \mathcal{V}$ has a buffer $\mathcal{B}_i$ in which it can receive and store messages from neighbors all the time (even when it is inactive).

A. Asynchronous DGD

In the asynchronous DGD, we let each node $i \in \mathcal{V}$ hold $x_i \in \mathbb{R}^d$ and $x_{ij} \in \mathbb{R}^d$ $\forall j \in \mathcal{N}_i$, where x_i is the current local iterate of node i and x_{ij} records the most recent x_j it received from node $j \in \mathcal{N}_i$. Once activated, node i reads all x_j in the buffer $\mathcal{B}_i$ and then sets $x_{ij} = x_j$ and, in case $\mathcal{B}_i$ contains multiple x_j's for a particular $j \in \mathcal{N}_i$, node i sets x_{ij} as the most recently received x_j. Next, it updates x_i by

$$x_i \leftarrow w_{ii}x_i + \sum_{j\in\mathcal{N}_i} w_{ij}x_{ij} - \alpha\nabla f_i(x_i) \tag{9}$$

and broadcasts the new x_i to all its neighbors. Once a node $j \in \mathcal{N}_i$ receives x_i, it stores x_i in its buffer $\mathcal{B}_j$. A detailed implementation is given in Algorithm 1.

To describe the asynchronous DGD mathematically, we index the iterates by $k \in \mathbb{N}_0$. The index k is increased by 1 whenever an update is performed on a local variable x_i of some nodes $i \in \mathcal{V}$. The index k does not need to be known by the nodes – it is only introduced to order events in our theoretical analysis. We can now see that each x_{ij} in (9) is a delayed x_j – each node $i \in \mathcal{V}$ updates using the most recently received x_j for higher efficiency but it is, in general, not the newest x_j computed by node j. Let $\mathcal{K}_i \subseteq \mathbb{N}_0$ denote the set of iterations where node i updates its iterate. For convenient notation, we define $\bar{\mathcal{N}}_i = \mathcal{N}_i \cup \{i\}$ for all $i \in \mathcal{V}$. Then, the asynchronous DGD can be described as follows. For each $i \in \mathcal{V}$ and $k \in \mathbb{N}_0$,

$$x_i^{k+1} = \begin{cases} \sum_{j\in\bar{\mathcal{N}}_i} w_{ij} x_j^{s_{ij}^k} - \alpha\nabla f_i(x_i^k), & k \in \mathcal{K}_i, \\ x_i^k, & \text{otherwise}, \end{cases} \tag{10}$$

Algorithm 1 Asynchronous DGD

1: **Initialization:** All the nodes agree on $\alpha > 0$, and cooperatively set w_{ij} $\forall\{i,j\} \in \mathcal{E}$.
2: Each node $i \in \mathcal{V}$ chooses $x_i \in \mathbb{R}^d$, creates a local buffer $\mathcal{B}_i$, and shares x_i with all neighbors in $\mathcal{N}_i$.
3: **for** each node $i \in \mathcal{V}$ **do**
4: keep *receiving* x_j *from neighbors and store* (x_j, j) *in* $\mathcal{B}_i$ until node i is activated[2].
5: set $x_{ij} = x_j$ $\forall(x_j, j) \in \mathcal{B}_i$. If multiple $(x_j, j) \in \mathcal{B}_i$ for some j, choose the most recently received one.
6: empty $\mathcal{B}_i$.
7: update x_i according to (9).
8: send x_i to all neighbors $j \in \mathcal{N}_i$.
9: **Until** a termination criterion is met.

where $s_{ij}^k \in [0, k]$ for $j \in \mathcal{N}_i$ is the iteration index of the most recent version of x_j available to node i at iteration k and $s_{ii}^k = k$. If $\mathcal{K}_i = \mathbb{N}_0$ $\forall i \in \mathcal{V}$ and $s_{ij}^k = k$ $\forall\{i,j\} \in \mathcal{E}, \forall k \in \mathbb{N}_0$, then (10) reduces to the synchronous DGD (2).

B. Asynchronous DGD-ATC

To implement the asynchronous DGD-ATC, each node $i \in \mathcal{V}$ holds $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}^d$, and $y_{ij} \in \mathbb{R}^d$ for $j \in \mathcal{N}_i$, where x_i is the current local iterate of node i, $y_i = x_i - \alpha\nabla f_i(x_i)$, and y_{ij}, $j \in \mathcal{N}_i$ records the most recent value of y_j it received from node j. Once activated, node $i \in \mathcal{V}$ first reads all y_j in its buffer $\mathcal{B}_i$ and then sets $y_{ij} = y_j$ and, in case $\mathcal{B}_i$ contains multiple values of y_j for a particular $j \in \mathcal{N}_i$, node i sets y_{ij} as the most recent y_j it has received. Next, it updates x_i by

$$x_i \leftarrow w_{ii}y_i + \sum_{j\in\mathcal{N}_i} w_{ij}y_{ij}, \tag{11}$$

computes $y_i = x_i - \alpha\nabla f_i(x_i)$, and broadcasts y_i to all $j \in \mathcal{N}_i$. Once a node $j \in \mathcal{N}_i$ receives y_i, it stores y_i in its buffer $\mathcal{B}_j$. A detailed implementation of the asynchronous DGD-ATC is described in Algorithm 2.

Note that each y_{ij} in (11) is a delayed $x_j - \alpha\nabla f_j(x_j)$. Then, similar to (10), the asynchronous DGD-ATC can be described as follows. For each $i \in \mathcal{V}$ and $k \in \mathbb{N}_0$,

$$x_i^{k+1} = \begin{cases} \sum_{j\in\bar{\mathcal{N}}_i} w_{ij}(x_j^{s_{ij}^k} - \alpha\nabla f_j(x_j^{s_{ij}^k})), & k \in \mathcal{K}_i, \\ x_i^k, & \text{otherwise}, \end{cases} \tag{12}$$

where all the notations follow their definitions in Section III-A. When $\mathcal{K}_i = \mathbb{N}_0$ $\forall i \in \mathcal{V}$ and $s_{ij}^k = k$ $\forall\{i,j\} \in \mathcal{E}, \forall k \in \mathbb{N}_0$, (12) reduces to the synchronous DGD-ATC.

IV. Convergence Analysis

In this section, we analyse the convergence of the asynchronous DGD and the asynchronous DGD-ATC under two different models of asynchrony. Our first results allow for total asynchrony in the sense of Bertsekas and Tsitsiklis [24],

[2]In the first iteration, each node $i \in \mathcal{V}$ can be activated only after it received (x_j, j) (Algorithm 1) or (y_j, j) (Algorithm 2) from all $j \in \mathcal{N}_i$.

Algorithm 2 Asynchronous DGD-ATC

Initialization: All the nodes agree on $\alpha > 0$, and cooperatively set $w_{ij}\ \forall\{i,j\} \in \mathcal{E}$.
Each node $i \in \mathcal{V}$ chooses $x_i \in \mathbb{R}^d$, creates a local buffer $\mathcal{B}_i$, sets $y_i = x_i - \alpha\nabla f_i(x_i)$ and shares it with all $j \in \mathcal{N}_i$.
for each node $i \in \mathcal{V}$ **do**
  keep *receiving* y_j *from neighbors and store* (y_j, j) *in* $\mathcal{B}_i$ until node i is activated[2].
  set $y_{ij} = y_j$ for all $(y_j, j) \in \mathcal{B}_i$. If multiple $(y_j, j) \in \mathcal{B}_i$ for some j, choose the most recently received one.
  empty $\mathcal{B}_i$.
  update x_i by (11).
  set $y_i = x_i - \alpha\nabla f_i(x_i)$.
  share y_i with all neighbors $j \in \mathcal{N}_i$.
Until a termination criterion is met.

i.e. the information delays $k - s_{ij}^k$ may grow arbitrarily large but no node can cease to update and old information must eventually be purged from the system. This assumption is well-suited for scenarios where communication and computation delays are "unstable", e.g., in massively parallel computing grids with heterogeneous computing nodes, delays can quickly add up if a node is saturated [25]. More formally, we make the following assumption.

Assumption 3 (total asynchrony): The following holds:

1) $\mathcal{K}_i$ is an infinite subset of $\mathbb{N}_0$ for each $i \in \mathcal{V}$.
2) $\lim_{k\to+\infty} s_{ij}^k = +\infty$ for any $i \in \mathcal{V}$ and $j \in \mathcal{N}_i$.

The following theorem provides delay-free step-size conditions that guarantee that the asynchronous DGD and DGD-ATC algorithms converge under total asynchrony.

Theorem 1 (total asynchrony): Suppose that Assumptions 1–3 hold. Also suppose that in the asynchronous DGD,

$$\alpha \in \left(0, 2\min_{i\in\mathcal{V}} \frac{w_{ii}}{L_i}\right), \tag{13}$$

and in the asynchronous DGD-ATC,

$$\alpha \in \left(0, \frac{2}{\max_{i\in\mathcal{V}} L_i}\right). \tag{14}$$

Then, $\{\mathbf{x}^k\}$ generated by either method converges to some element in the fixed point set of the synchronous counterpart.

Under total asynchrony, there is no lower bound on the update frequency of nodes and no upper bound on the information delays, and we are only able to give asymptotic convergence guarantees. To derive non-asymptotic convergence rate guarantees, we consider the more restrictive notion of partial asynchrony [24].

Assumption 4 (partial asynchrony): There exist positive integers B and D such that

1) For every $i \in \mathcal{V}$ and for every $k \geq 0$, at least one element in the set $\{k, \ldots, k+B\}$ belongs to $\mathcal{K}_i$.
2) There holds
$$k - D \leq s_{ij}^k \leq k$$
for all $i \in \mathcal{V}$, $j \in \mathcal{N}_i$, and $k \in \mathcal{K}_i$.

In Assumption 4, B and D characterize the minimum update frequency and the maximal information delay, respectively. If $B = D = 0$, then Assumption 4 reduces to the synchronous scheme where all local variables $x_i^k\ \forall i \in \mathcal{V}$ are instantaneously updated at every iteration $k \in \mathbb{N}_0$.

To state our convergence result, we define the block-wise maximum norm for any $\mathbf{x} = (x_1^T, \ldots, x_n^T)^T \in \mathbb{R}^{nd}$ as

$$\|\mathbf{x}\|_\infty^b = \max_{i\in\mathcal{V}} \|x_i\|.$$

The following theorem establishes linear convergence for the two algorithms under partial asynchrony.

Theorem 2 (partial asynchrony): Suppose that Assumptions 1, 2, 4 hold. Also suppose that (13) holds in the asynchronous DGD and (14) holds in the asynchronous DGD-ATC. Then, $\{\mathbf{x}^k\}$ generated by either method satisfies

$$\|\mathbf{x}^k - \mathbf{x}^\star\|_\infty^b \leq \rho^{\lfloor k/(B+D+1)\rfloor} \|\mathbf{x}^0 - \mathbf{x}^\star\|_\infty^b,$$

where $\mathbf{x}^\star$ is the fixed point of their synchronous counterpart and $\rho \in (0,1)$. Specifically, for

$$\text{async DGD}: \ \rho = \sqrt{1 - \alpha \min_{i\in\mathcal{V}} \left(\mu_i \left(2 - \frac{\alpha L_i}{w_{ii}}\right)\right)}, \tag{15}$$

$$\text{async DGD-ATC}: \ \rho = \sqrt{1 - \alpha \min_{i\in\mathcal{V}} \left(\mu_i (2 - \alpha L_i)\right)}. \tag{16}$$

By Lemmas 1–2 and Theorems 1-2, the two asynchronous methods can converge to an approximate optimum of Problem (1), where the optimality gap is given in Lemmas 1–2. Note that the range of step-sizes that guarantees convergence is independent of the degree of asynchrony in the system. The two algorithms converge even under total asynchrony, but the guarantees we can give improve as the amount of asynchrony decreases. Moreover, Theorem 2 indicates two advantages of the asynchronous DGD-ATC over the asynchronous DGD. First, it allows for a larger step-size range (14) than (13), which may lead to faster practical convergence. Second, even using the same α, the asynchronous DGD-ATC has a faster convergence rate: Let ρ, $\hat{\rho}$ denote the values in (15) and (16), respectively. Then,

$$\begin{aligned}
\rho^2 - \hat{\rho}^2 &= \alpha \left(\min_{i\in\mathcal{V}} \mu_i(2 - \alpha L_i) - \min_{i\in\mathcal{V}} \mu_i \left(2 - \alpha\frac{L_i}{w_{ii}}\right)\right) \\
&\geq \alpha \min_{i\in\mathcal{V}} \left(\mu_i(2 - \alpha L_i) - \mu_i \left(2 - \alpha\frac{L_i}{w_{ii}}\right)\right) \\
&= \alpha^2 \min_{i\in\mathcal{V}} \mu_i L_i \left(\frac{1}{w_{ii}} - 1\right) \geq 0.
\end{aligned} \tag{17}$$

The faster convergence of the asynchronous DGD-ATC is also demonstrated by experiments in Section V.

A. Comparison with Related Methods

To the best of our knowledge, Theorem 1 provides the first convergence result for solving (1) with non-quadratic f_i on general networks under total asynchrony. Other works considering total asynchrony include [15], [26]. In particular, the asynchronous coordinate descent method in [15] can solve Problem (1) with quadratic objective functions over undirected, connected networks, and the asynchronous

proximal gradient method in [26] can address (1) with non-quadratic f_i's, but only considers star networks.

In order to distinguish our results from the state-of-the-art on asynchronous consensus optimization algorithms [2]–[10], [15], [26], we categorize these works based on their step-sizes and compare them to our results.

delay-dependent step-size: [8]–[10] assume the existence of an upper bound on the information delay and use fixed parameters relying on and decreasing with the delay bound. Although the works [8]–[10] can achieve convergence to the exact optimum under partial asynchrony, which is more desirable than the inexact convergence of our algorithms, they suffer from difficult parameter determination and unnecessary slow convergence for two reasons. Firstly, the delay bound is often unknown and hard to obtain in advance. Secondly, the delay bound is typically large, which leads to small step-sizes and further slows down the convergence process. Our numerical experiments in Section V suggest that the asynchronous DGD and DGD-ATC can significantly outperform PG-EXTRA [9] for the simulated problem. In addition, our algorithms can converge under total asynchrony that is not allowed in [8]–[10].

delay-free and non-diminishing step-size: This category includes [2], [15], [26]. However, [2], [15] can only solve simple problems. The work [2] focuses on the consensus problem which is equivalent to Problem (1) with $f_i(x) \equiv 0$, and [15] can only deal with Problem (1) with quadratic objective functions. The work [26] can solve Problem (1) with non-quadratic objective functions, but requires star networks. In contrast, our results in Theorem 1–2 allow for non-quadratic objective functions and non-star communication networks, which is a substantial improvement.

diminishing step-size: [3]–[7] consider diminishing step-sizes that are also delay-free. However, the diminishing step-sizes decrease rapidly and can lead to slow practical convergence. Moreover, [3]–[7] all focus on partial asynchrony, while our algorithms can converge under total asynchrony.

V. Numerical Experiments

We evaluate the practical performance of the asynchronous DGD and the asynchronous DGD-ATC on decentralized learning using the ℓ_2- regularized logistic loss:

$$\underset{x\in\mathbb{R}^d}{\text{minimize}} \ \frac{1}{N}\sum_{i=1}^{N}\left(\log(1+e^{-b_i(a_i^T x)})+\frac{\lambda}{2}\|x\|^2\right), \quad (18)$$

where N is the number of samples, a_i is the feature of the ith sample, b_i is the corresponding label, and $\lambda = 10^{-3}$ is the regularization parameter. The experiments use the training set of Covertype [27] and MNIST [28] summarized below:

TABLE I: Information about training data sets.

Data set	sample number N	feature dimension d
Covertype	581012	54
MNIST	60000	784

We compare our algorithms with the asynchronous PG-EXTRA [9]. We do not compare with the algorithms in [3]–[7] with diminishing step-sizes because [3]–[6] require Lipschitz continuous objective functions which does not hold for Problem (18) and the maximum allowable step-size in [7] is excessively small ($\leq 10^{-10}$ in our experiment setting). We set $n = 8$, evenly partition and allocate all data samples to each node, and implement all the methods on a multi-core computer using the message-passing framework MPI4py [29], where each core serves as a node and the communication graph $\mathcal{G}$ is displayed in Figure 1. In the experiments, each node $i \in \mathcal{V}$ is activated once its buffer $\mathcal{B}_i$ is non-empty, and the delays are generated by real interactions between the nodes and not by any theoretical delay model. We set $\alpha = \min_{i\in\mathcal{V}} w_{ii}/\max_{i\in\mathcal{V}} L_i$ in the asynchronous DGD and $\alpha = 1/\max_{i\in\mathcal{V}} L_i$ in the asynchronous DGD-ATC, which meet their conditions in Theorems 1–2. We fine-tune the parameters of the asynchronous PG-EXTRA within their theoretical ranges for guaranteeing convergence. The theoretical ranges involve the maximum delay, which is determined by recording the maximum observed delay during a 20-second run of the method.

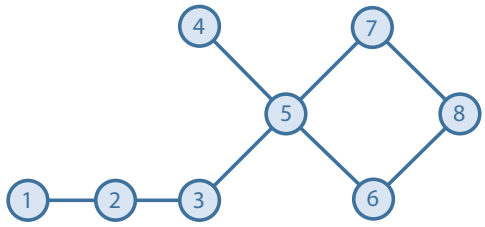
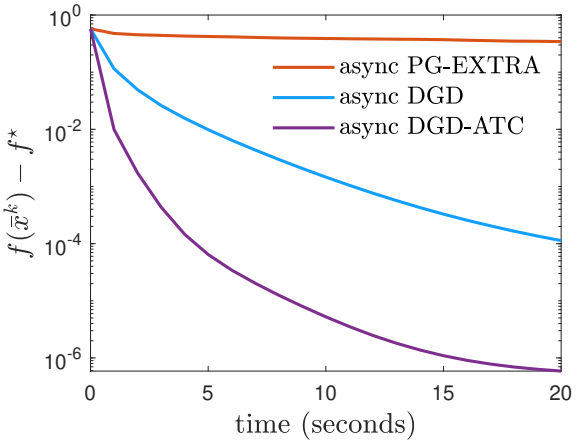


Fig. 1: Communication graph in simulation.

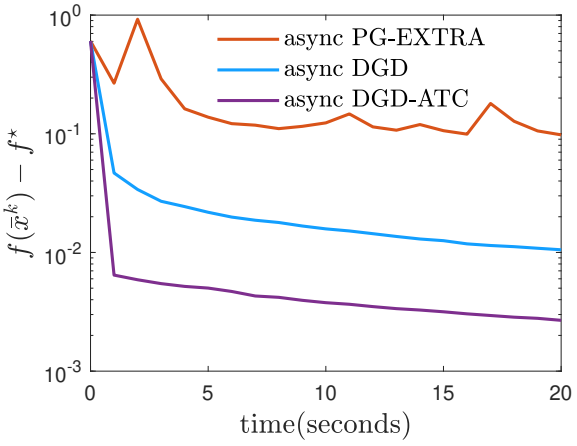
We run all methods for 20 seconds and plot the training error $f(\bar{x}^k) - f^\star$ at the average iterate $\bar{x}^k = \frac{1}{n}\sum_{i=1}^n x_i^k$ in Figure 2, where $f^\star$ is the optimal value of (18). We can see that for both datasets, the asynchronous DGD-ATC outperforms the asynchronous DGD as indicated by (17), and they both converge faster than the asynchronous PG-EXTRA. The slow convergence of the asynchronous PG-EXTRA may be because of its conservative parameters caused by the large delay, while our algorithms can converge under much more relaxed delay-free parameter conditions.

VI. Conclusion

We have investigated the asynchronous version of two distributed algorithms, DGD and DGD-ATC, for solving consensus optimization problems. We first reviewed existing results on the optimality gap of DGD and developed a corresponding results for the optimality gap of DGD-ATC. Then, we developed *delay-free* parameter conditions under which both asynchronous methods converge to the fixed point set of their synchronous counterparts under total and partial asynchrony. Finally, we demonstrated superior practical convergence of the two asynchronous algorithms via numerical experiments. Future work includes developing asynchronous algorithms with delay-free parameter conditions for other distributed optimization problems.



(a) Covertype



(b) MNIST

Fig. 2: Convergence on logistic regression

REFERENCES

[1] M. Assran, A. Aytekin, H. R. Feyzmahdavian, M. Johansson, and M. G. Rabbat, "Advances in asynchronous parallel and distributed optimization," *Proceedings of the IEEE*, vol. 108, no. 11, pp. 2013–2031, 2020.

[2] A. Nedić and A. Ozdaglar, "Convergence rate for consensus with delays," *Journal of Global Optimization*, vol. 47, pp. 437–456, 2010.

[3] J. Zhang and K. You, "AsySPA: An exact asynchronous algorithm for convex optimization over digraphs," *IEEE Transactions on Automatic Control*, vol. 65, no. 6, pp. 2494–2509, 2019.

[4] B. Sirb and X. Ye, "Consensus optimization with delayed and stochastic gradients on decentralized networks," in *IEEE International Conference on Big Data (Big Data)*, 2016, pp. 76–85.

[5] T. T. Doan, C. L. Beck, and R. Srikant, "On the convergence rate of distributed gradient methods for finite-sum optimization under communication delays," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, pp. 1–27, 2017.

[6] V. Kungurtsev, M. Morafah, T. Javidi, and G. Scutari, "Decentralized asynchronous non-convex stochastic optimization on directed graphs," accepted to *IEEE Transactions on Control of Network Systems*, 2023.

[7] M. S. Assran and M. G. Rabbat, "Asynchronous gradient push," *IEEE Transactions on Automatic Control*, vol. 66, no. 1, pp. 168–183, 2020.

[8] J. Zhang and K. You, "Fully asynchronous distributed optimization with linear convergence in directed networks," *arXiv preprint arXiv:1901.08215*, 2019.

[9] T. Wu, K. Yuan, Q. Ling, W. Yin, and A. H. Sayed, "Decentralized consensus optimization with asynchrony and delays," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 4, no. 2, pp. 293–307, 2017.

[10] Y. Tian, Y. Sun, and G. Scutari, "Achieving linear convergence in distributed asynchronous multiagent optimization," *IEEE Transactions on Automatic Control*, vol. 65, no. 12, pp. 5264–5279, 2020.

[11] Y. Wang, Q. Zhao, and X. Wang, "An asynchronous gradient descent based method for distributed resource allocation with bounded variables," *IEEE Transactions on Automatic Control*, vol. 67, no. 11, pp. 6106–6111, 2021.

[12] L. Cannelli, F. Facchinei, G. Scutari, and V. Kungurtsev, "Asynchronous optimization over graphs: Linear convergence under error bound conditions," *IEEE Transactions on Automatic Control*, vol. 66, no. 10, pp. 4604–4619, 2020.

[13] P. Latafat and P. Patrinos, "Primal-dual algorithms for multi-agent structured optimization over message-passing architectures with bounded communication delays," *Optimization Methods and Software*, vol. 37, no. 6, pp. 2052–2079, 2022.

[14] M. Ubl and M. T. Hale, "Faster asynchronous nonconvex block coordinate descent with locally chosen stepsizes," in *IEEE Conference on Decision and Control (CDC)*, 2022, pp. 4559–4564.

[15] M. Ubl and M. Hale, "Totally asynchronous large-scale quadratic programming: Regularization, convergence rates, and parameter selection," *IEEE Transactions on Control of Network Systems*, vol. 8, no. 3, pp. 1465–1476, 2021.

[16] K. Mishchenko, F. Bach, M. Even, and B. Woodworth, "Asynchronous SGD beats minibatch SGD under arbitrary delays," in *Advances in Neural Information Processing Systems*, 2022.

[17] K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," *SIAM Journal on Optimization*, vol. 26, no. 3, pp. 1835–1854, 2016.

[18] S. Pu, A. Olshevsky, and I. C. Paschalidis, "Asymptotic network independence in distributed stochastic optimization for machine learning: Examining distributed and centralized stochastic gradient descent," *IEEE signal processing magazine*, vol. 37, no. 3, pp. 114–122, 2020.

[19] 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.

[20] A. Nedić, A. Olshevsky, and W. Shi, "Achieving geometric convergence for distributed optimization over time-varying graphs," *SIAM Journal on Optimization*, vol. 27, no. 4, pp. 2597–2633, 2017.

[21] C. Liu, Z. Zhou, J. Pei, Y. Zhang, and Y. Shi, "Decentralized composite optimization in stochastic networks: A dual averaging approach with linear convergence," accepted to *IEEE Transactions on Automatic Control*, 2022.

[22] W. Shi, Q. Ling, G. Wu, and W. Yin, "A proximal gradient algorithm for decentralized composite optimization," *IEEE Transactions on Signal Processing*, vol. 63, no. 22, pp. 6013–6023, 2015.

[23] ——, "EXTRA: An exact first-order algorithm for decentralized consensus optimization," *SIAM Journal on Optimization*, vol. 25, no. 2, pp. 944–966, 2015.

[24] D. Bertsekas and J. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Athena Scientific, 2015.

[25] Z. Zhou, P. Mertikopoulos, N. Bambos, P. Glynn, Y. Ye, L.-J. Li, and L. Fei-Fei, "Distributed asynchronous optimization with unbounded delays: How slow can you go?" in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80. PMLR, 10–15 Jul 2018, pp. 5970–5979.

[26] K. Mishchenko, F. Iutzeler, J. Malick, and M.-R. Amini, "A delay-tolerant proximal-gradient algorithm for distributed learning," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3587–3595.

[27] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[28] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[29] L. Dalcín, R. Paz, M. Storti, and J. D'Elía, "MPI for python: Performance improvements and MPI-2 extensions," *Journal of Parallel and Distributed Computing*, vol. 68, no. 5, pp. 655–662, 2008.