On the Convergence of Inexact Gradient Descent with Controlled Synchronization Steps

Sandushan Ranaweera, Chathuranga Weeraddana, Prathapasinghe Dharmawansa, and Carlo Fischione

Abstract—We develop a gradient-like algorithm to minimize a sum of peer objective functions based on coordination through a peer interconnection network. The coordination admits two stages: the first is to constitute a gradient, possibly with errors, for updating locally replicated decision variables at each peer and the second is used for error-free averaging for synchronizing local replicas. Unlike many related algorithms, the errors permitted in our algorithm can cover a wide range of inexactnesses, as long as they are bounded. Moreover, we do not impose any gradient boundedness conditions for the objective functions. Furthermore, the second stage is not conducted in a periodic manner, like many related algorithms. Instead, a locally verifiable criterion is devised to dynamically trigger the peer-to-peer coordination at the second stage, so that expensive communication overhead for error-free averaging can significantly be reduced. Finally, the convergence of the algorithm is established under mild conditions.

Index Terms—Distributed optimization, inexact algorithms

I. INTRODUCTION

GRadient descent and its variants often lend themselves fully amenable to parallel and distributed algorithms, which are highly desirable in large-scale optimization problems [1]. As a result, solution methods for many problems of recent interest are predominantly based on such gradient-like algorithms [2]–[14]. Broadly speaking, those algorithms developments are twofold [15]: a) a federated setting where a central controller (CC) intervenes for decision variable update [2]–[8]; b) a peer-to-peer (PP) setting where subsystems (SSs), each with its replicated decision variable, perform locally the update through some peer interconnection network, often modeled by a connected graph [9]–[14]. In this setting, the algorithm relies on neighbors specified by the graph and does not rely on a CC like in the federated setting. As such, it appears that PP setting is more appealing than the CC setting due to many reasons, such as higher scalability and inherently decentralized collection of big data sets, among others [1], [15]. In the context of a PP setting, a more fundamental concern is that the distributed algorithms usually undergo inevitable inexact conditions, e.g., unreliable and often limited communication capabilities [1], [15], [16]. Thus, unlike the inexactnesses under CC settings [17]–[22], those under PP settings influence the optimality, convergence, and effective implementation of algorithms. Consequently, there is an appeal to design effective algorithms under PP setting [23]–[33].

S. Ranaweera (e-mail: sandushan@ieee.org) and P. Dharmawansa (e-mail: prathapa@uom.lk) are with the Dept. of Electronic and Telecom. Eng., University of Moratuwa, Sri Lanka. C. Weeraddana (e-mail: Chathuranga.Weeraddana@oulu.fi) is with the Faculty of Information Technology and Electrical Eng., University of Oulu, Finland. C. Fischione (e-mail: carlofi@kth.se) is with the School of Electrical Eng. and Computer Science, KTH Royal Institute of Technology, Sweden.

Algorithms in [23]–[28] are based on distributed subgradient methods due to [13]. Some of these methods consider quantization models [23]–[25] and others consider event-triggered models [26]–[28], so as to reduce the communication burden between SSs. The gradient boundedness of underlying objective functions, although a restriction, has been considered in [23]–[28], a technical assumption that enables convergences. The errors introduced in [23]–[28] can be viewed as controllable, in the sense that they are at the disposal of the algorithm. For example, quantization models in [23], [25] are chosen to be unbiased, a favorable condition for convergence. However, a peer interconnection network can often admit errors that are not at the disposal of the algorithm, e.g., wireless links [34, § 9], limiting the applicability of developments in [23]–[28]

Works in [29]–[33] rely on PP coordination to constitute a gradient, in contrast to common federated settings where primal variables are coordinated instead. Then the resulting gradients are for updating their locally replicated decision variables. They are persuaded again under quantization settings (e.g., [29], [30]) and event-triggered settings (e.g., [31], [32]). Hybrid variants have also been considered by some authors, e.g., [33]. Similar to the developments noted in the preceding discussion, errors introduced in [29]–[33] are also controlled by the algorithms. For example, the quantization models in [29] and [30] are chosen so that the errors are diminishing and unbiased, respectively. Moreover, the authors in [31], [33] have specific impositions on the gradient boundedness.

It is worth noting that many algorithms in either of the setting federated or PP (e.g., [6]–[8], [31]–[33]) have considered an averaging step performed at periodic or predefined epochs to enable the consistency of the locally replicates decision variables. Depending on the context, this entails periodic communication through the CC or through the PP interconnection network. From a communication overhead point of view, however, such an overhead for periodic communication seems like a restriction. This may be avoided by dynamically choosing the averaging epochs for synchronization.

In this paper, we develop an algorithm that relies on PP coordination to constitute a gradient for updating locally replicated decision variables associated with a problem of minimizing the sum of peer objective functions. The algorithm iterates two stages. The first is used to exchange gradients possibly with errors. We have no restrictions on the errors of local gradient estimates, except that they are bounded. As a result, our modeling can handle errors beyond those of classic quantization models with restrictions, such as diminishing and unbiasedness. For instance, a cheap low-bit quantization can be used throughout the algorithm iterates under the first

stage. The second stage is used to error-free averaging for synchronizing local replicas. In this respect, unlike other related algorithms, we do not rely on periodic communication over the PP network. Instead, a locally verifiable criterion is devised to dynamically trigger the averaging step, only when necessary. This has the advantage of minimizing expensive communication overhead. Throughout this paper, we consider the PP network to be fully connected. [1] Subsequently, the convergence of the algorithm is established and is shown to be linear.

II. PROBLEM FORMULATION

Consider N peers or subsystems which solve the problem

$$\text{minimize} \quad f(\boldsymbol{x}) = \sum_{i=1}^{N} f_i(\boldsymbol{x}) \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ and $f_i : \mathbb{R}^n \to \mathbb{R}$, $i \in \mathcal{N} \triangleq \{1, \ldots, N\}$, be a function satisfying the following standard assumption:

AS 1. *The objective function f_i, $i \in \mathcal{N}$, is strongly convex with constant $\ell_i > 0$ and is L_i-smooth, i.e., ∇f_i is Lipschitz continuous with the constant $L_i > 0$.*

A commonly used iterative algorithms for solving problem (1) is the gradient descent (GD) algorithm $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \gamma \sum_{j=1}^{N} \nabla f_j(\boldsymbol{x}^{(k)})$, where $k \in \mathbb{Z}^+ \triangleq \{0, 1, 2, \ldots\}$ is the iteration index and γ is the step size. In contrast, here we assume a setting where each subsystem (SS) i performs locally the variable GD update of its own copy $\boldsymbol{x}_i^{(k+1)}$ of $\boldsymbol{x}^{(k+1)}$. This setting facilitates a distributed implementation of GD and thus each SS i relies on a communication with SS j to get a rough measurement of $\nabla f_j(\boldsymbol{x}_j^{(k)})$ as specified below:

AS 2. $\forall i, j \in \mathcal{N}$, s.t. $i \neq j$, *gradient measurement $\boldsymbol{h}_{ij}^{(k)} \in \mathbb{R}^n$ received by i-th SS from j-th SS at k-th iteration is given by*

$$\boldsymbol{h}_{ij}^{(k)} = \nabla f_j(\boldsymbol{x}_j^{(k)}) + \boldsymbol{\epsilon}_{ij}^{(k)} \tag{2}$$

where $\boldsymbol{\epsilon}_{ij}^{(k)} \in \mathbb{R}^n$ is a error such that $\|\boldsymbol{\epsilon}_{ij}^{(k)}\| \leq \epsilon$ with $\|\cdot\|$ denoting the Euclidean norm.

The parameters $\boldsymbol{\epsilon}_{ij}^{(k)}$ model measurement errors, noises, quantization errors[2] due to compression, among others. However, note the upper bound condition on $\boldsymbol{\epsilon}_{ij}^{(k)}$ in **AS 2**, where ϵ can be thought of as the worst-case characteristic of errors throughout the algorithm. Under **AS 2**, the gradient $\nabla f(\boldsymbol{x}_i^{(k)})$ is distorted, which in turn admits the following iterate:

$$\boldsymbol{x}_i^{(k+1)} = \boldsymbol{x}_i^{(k)} - \gamma \sum_{j=1}^{N} \boldsymbol{h}_{ij}^{(k)}, \quad i \in \mathcal{N}. \tag{3}$$

Strictly speaking, the local variables updates should be consistent in the sense that $\forall k \in \mathbb{Z}^+$, $\forall i, j \in \mathcal{N}$, $\boldsymbol{x}_j^{(k)} = \boldsymbol{x}_i^{(k)}$. However, (3) with distinct SSs do not admit at least a weaker form of the consistency, called *synchrony* given by

$$\forall i, j \in \mathcal{N}, i \neq j, \boldsymbol{x}_j^{(m)} = \boldsymbol{x}_i^{(m)} \tag{4}$$

where m is an iteration index of practical interest, e.g., the iteration index at the termination. Thus, the main challenge in

this research is to establish the convergence properties of (3), while maintaining the synchrony. [3] This challenge is taken up next, where the iterate (3) is integrated with potential SS coordination to yield an algorithm with guaranteed convergence.

III. ALGORITHM DEVELOPMENT

Let us first focus on establishing the evolutionary characteristics of (3) to set the stage for our subsequent developments.

A. Evolutionary Characteristics of (3)

From (3), (2), together with some standard algebraic manipulations as shown in the *Appendix*, it can be shown that, under **AS 1**, **AS 2** and for $\gamma \in (0, 1/\sum_{j=i}^{N} L_j]$,

$$\|\nabla f(\boldsymbol{x}_i^{(k)}) - \boldsymbol{h}_i^{(k)}\| \leq 2\epsilon N (k + 1/2), \quad i \in \mathcal{N} \tag{5}$$

where $\boldsymbol{h}_i^{(k)} \triangleq \sum_{j=1}^{N} \boldsymbol{h}_{ij}^{(k)}$. The inequality (5) indicates that, in the worst case, the norm of the difference between $\nabla f(\boldsymbol{x}_i^{(k)})$ and its local representation $\boldsymbol{h}_i^{(k)}$ diverges as $k \to \infty$. Thus, it is of paramount importance to control such growth for establishing convergence of iterates of the form (3). To this end, it is customary to rely on SS coordination possibly through an error-free communication medium. However, error-free communications are usually more expensive. Therefore, unlike the commonly considered periodic SS coordination [33], we seek to reduce the communication overhead by dynamically choosing the coordination epochs, so as to make it still possible to ensure convergences of the underlying sequences. As such, we consider a relative deviation of the gradient of the objective function f and its measurement from the standpoint of ith SS, i.e., $e_i^{(k)} \triangleq \|\nabla f(\boldsymbol{x}_i^{(k)}) - \boldsymbol{h}_i^{(k)}\| / \|\nabla f(\boldsymbol{x}_i^{(k)})\|$, $i \in \mathcal{N}$.

Intuitively, when $e_i^{(k)}$ is sufficiently small, the influence of errors $\boldsymbol{\epsilon}_{ij}^{(k)}$ on (3) becomes relatively insignificant. On the other hand, when $e_i^{(k)}$ is sufficiently large, the consequences become more detrimental, and (3) may evolve anomalously. Thus, to circumvent such anomalies, the objective is to start with synchrony [*cf.* (4)] at $k = 0$ and to perform iterate (3) as long as $e_i^{(k)}$ is sufficiently small, for otherwise to trigger SS coordination. As a result, the iterates (3) at each SSs might tend to evolve in a meaningful direction.

Let us next discuss how the preceding concept can be integrated into devise our algorithm. In this respect, the most crucial step is to identify an epoch at which the SS coordination is to be triggered. In other words, each SS needs a locally verifiable characterization of the iterates k for which $e_i^{(k)}$ is sufficiently small, despite the dependence of $e_i^{(k)}$ on global information $\nabla f(\cdot)$. As such, we rely on the condition

$$k \leq r\|\boldsymbol{h}_i^{(k)}\| / (2\epsilon N) - 1/2 \implies e_i^{(k)} \leq r/(1-r) \tag{6}$$

where $r \in (0, 1)$ is a design parameter, suitably chosen based on the strong convexity constants and the Lipschitz constants of the objective functions. The condition (6) follows from (5), together with that $\|\boldsymbol{h}_i^{(k)}\| - \|\nabla f(\boldsymbol{x}_i^{(k)})\| \leq \|\nabla f(\boldsymbol{x}_i^{(k)}) - \boldsymbol{h}_i^{(k)}\|$.

---

[1] An extension to an arbitrary graph is possible with an additional assumption on the gradient boundedness. The details are provided in the Appendix.

[2] *cf.* [19, Definition 2] for such a quantization that yield an error as in **AS 2**.

[3] Under imperfect conditions, iterates of the form (3) are commonplace in many distributed algorithms such as primal or dual-decomposition methods, among others, see e.g., [12] and references therein.

Thus, the SSs perform the iterate (3) independent of each other, as long as, for all $i \in \mathcal{N}$, $k \leq r\|\boldsymbol{h}_i^{(k)}\|/(2\epsilon N) - 1/2$, and is referred to as IndComp. If $k > r\|\boldsymbol{h}_i^{(k)}\|/(2\epsilon N) - 1/2$ for at least one SS, SSs communicate with others to average their local copies $\boldsymbol{x}_i^{(k)}$, which is referred to as the intermittent synchronization (IntSync). IntSync is performed through an error-free communication system. Having presented the evolutionary characteristics of (3), we are now ready to propose our new algorithm.

B. Algorithm and Its Convergence Analysis

The two stages IndComp and IntSync are implemented in an iterative manner to yield the following algorithm.

Algorithm 1 Inexact GD with IndComp−IntSync

Input: $\boldsymbol{x}_j^{(0)} = \boldsymbol{x}_i^{(0)} \ \forall \ i,j \in \mathcal{N}$, $\epsilon \geq 0$, $r \in (0, \sqrt{\ell}/(\sqrt{L} + \sqrt{\ell}))$, $s = 0, k = 0$

1: **repeat**
2:     **repeat**
3:         $\forall \ i \in \mathcal{N}$, compute $\boldsymbol{x}_i^{(s+k+1)}$ from (3), $k \leftarrow k+1$
4:     **until** $\exists \ i \in \mathcal{N}$, $k-1 > r\|\boldsymbol{h}_i^{(s+k-1)}\|/(2\epsilon N) - 1/2$
5:     **if** $k \neq 1$ **then**
6:         $\forall \ i \in \mathcal{N}$, $\boldsymbol{x}_i^{(s+k-1)} \leftarrow \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{x}_j^{(s+k-1)}$
7:         $s \leftarrow s+k-1$, $k \leftarrow 0$
8:     **else**
9:         $\forall \ i \in \mathcal{N}$, $\boldsymbol{x}_i^{(s+k)} \leftarrow \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{x}_j^{(s+k)}$
10:     $s \leftarrow s+k$, $k \leftarrow 0$
11:     **end if**
12: **until** a stopping criterion true

It is worth emphasizing that the indices s and k of the algorithm have an important interpretation. The index s always represents an iteration at which the synchrony [see (4)] of the local copies of the decision variables is imposed, *cf.* step 7, step 10. The inner loop [*cf.* steps 2-4] always starts with synchrony. Thus, k represents the local iteration index within the inner loop, which is reset every time the synchrony is imposed, *cf.* step 7, step 10. Consequently, $s+k$ is simply the global iteration index of Algorithm 1. The following Proposition establishes the convergence of Algorithm 1.

Proposition 1. *Suppose AS 1, AS 2 hold. Let $\{\boldsymbol{x}_i^{(k)}\}_{k \in \mathbb{Z}^+}$, $i \in \mathcal{N}$, be the sequence of local copies of the decision variable generated by Algorithm 1. Then for $\gamma \in (0, 1/L]$*

   1) $\limsup_{k \to \infty} (f(\boldsymbol{x}_i^{(k)}) - f(\boldsymbol{x}^\star)) \leq \epsilon^2 N^2/(2(\ell - L\bar{r}^2))$

   2) $\limsup_{k \to \infty} \|\nabla f(\boldsymbol{x}_i^{(k)})\| \leq \sqrt{L\epsilon^2 N^2/(\ell - L\bar{r}^2)}$

   3) $\limsup_{k \to \infty} \|\boldsymbol{x}_i^{(k)} - \boldsymbol{x}^\star\| \leq \sqrt{L\epsilon^2 N^2/(\ell^2 - L\bar{r}^2\ell)}$

where $L = \sum_{j=1}^{N} L_j$, $\ell = \min_{j \in \mathcal{N}} \ell_j$, $\bar{r} = r/(1-r)$, and $\boldsymbol{x}^\star = \arg\min_{\boldsymbol{x}} f(\boldsymbol{x})$.

It is not difficult to see that the Proposition holds even if $\boldsymbol{x}_i^{(k)}$ is set as $\boldsymbol{x}^{(k)} = \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{x}_j^{(k)}$ for all $k \in \mathbb{Z}^+$. Note that until the termination of the algorithm [*cf.* step 12], the inner loop is in either of the following states: 1) it repeats more than once 2) it repeats only once. Thus, the proof of the Proposition is simply based on the characterization of the evolution of the

sequence $\{f(\boldsymbol{x}_i^{(s+k+1)}) - f(\boldsymbol{x}^\star)\}$ when the algorithm is in either of the states. To this end, we shall require the following results, the proofs of which are given in the *Appendix*.

Lemma 1. *Let AS 1, AS 2 hold, $s \in \mathbb{Z}^+$ be any iteration index at which synchrony is imposed, $i \in \mathcal{N}$, and $r \in (0,1)$. Moreover, suppose the inner loop of Algorithm 1 repeats for iteration indices $\bar{k} \in \{s, s+1, \ldots, s+\kappa\}$, for some $\kappa \geq 2$. Then for $k \in \{0, 1, \ldots, \kappa - 1\}$*

$$f(\boldsymbol{x}_i^{(s+k+1)}) - f(\boldsymbol{x}^\star) \leq q\left(f(\boldsymbol{x}_i^{(s+k)}) - f(\boldsymbol{x}^\star)\right) \quad (7)$$

where $q = (1 + \gamma L\bar{r}^2 - \gamma\ell)$ is a positive constant.

Lemma 1 characterizes the evolution of the sequence $\{f(\boldsymbol{x}_i^{(s+k+1)}) - f(\boldsymbol{x}^\star)\}$ when the algorithm is in states 1. Consequently, the recursive application of (7), together with the Jensen's inequality yields

$$f(\boldsymbol{x}_i^{(s+\kappa)}) - f(\boldsymbol{x}^\star) \leq q^\kappa \left(f(\boldsymbol{x}_i^{(s)}) - f(\boldsymbol{x}^\star)\right). \quad (8)$$

The evolution of the sequence $\{f(\boldsymbol{x}_i^{(s+k+1)}) - f(\boldsymbol{x}^\star)\}$ when the algorithm is in state 2 is established by the following result.

Lemma 2. *Let AS 1, AS 2 hold, $s \in \mathbb{Z}^+$ be any iteration index at which synchrony is imposed, $r \in (0,1)$, and $i \in \mathcal{N}$. Moreover, suppose the inner loop of Algorithm 1 repeats only once, where the iteration index is s. Then*

$$f(\boldsymbol{x}_i^{(s+1)}) - f(\boldsymbol{x}^\star) \leq (1 - \gamma\ell)\left(f(\boldsymbol{x}_i^{(s)}) - f(\boldsymbol{x}^\star)\right) + \frac{\gamma\epsilon^2 N^2}{2}. \quad (9)$$

The inequality (9) holds even if $\boldsymbol{x}_i^{(s+1)}$ from the inner loop of Algorithm 1 is set as $\boldsymbol{x}_i^{(s+1)} = \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{x}_j^{(s+1)}$.

Finally, the following Lemma asserts that the algorithm necessarily switches to state 2 from state 1.

Lemma 3. *Let AS 1, AS 2 hold. Moreover, suppose $\forall \ i \in \mathcal{N}$, $r\|\boldsymbol{h}_i^{(0)}\| \geq \epsilon N$, and thus, the algorithm starts at state 1, where $r \in (0, \sqrt{\ell}/(\sqrt{L}+\sqrt{\ell}))$. Then $\exists \ \bar{s}, \bar{k} \in \mathbb{Z}^+$ such that Algorithm 1 switches to state 2 from state 1, where $\bar{s}$ is an iteration index at which the synchrony is imposed and $\bar{k}$ is a local iteration index within the inner loop.*
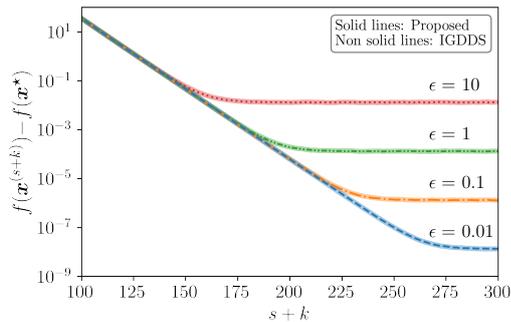
Having armed with the above results, we are now ready to give the proof of Proposition 1.

Proof of Proposition 1. From Lemma 1 and (8), for any consecutive sequence of state 1, starting at some global iteration index $n \in \mathbb{Z}^+$ and ending at $n + k \in \mathbb{Z}^+$, we have
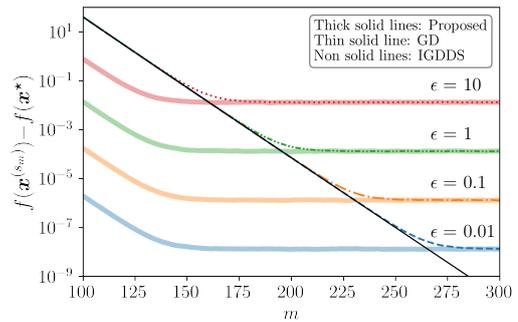
$$f(\boldsymbol{x}_i^{(n+k)}) - f(\boldsymbol{x}^\star) \leq q^k \left(f(\boldsymbol{x}_i^{(n)}) - f(\boldsymbol{x}^\star)\right) \quad (10)$$

$$\leq q^k \left(f(\boldsymbol{x}_i^{(n)}) - f(\boldsymbol{x}^\star)\right) + \frac{\gamma\epsilon^2 N^2}{2}\sum_{j=0}^{k-1} q^j. \quad (11)$$

Similarly, recursively applying (9) in Lemma 2 for any consecutive sequence of state 2, starting at some global iteration index $n \in \mathbb{Z}^+$ and ending at $n + k \in \mathbb{Z}^+$, together with that $1 - \gamma\ell \leq q$, we again have an equivalent form of (11).

(a) Error Vs `IntSync` + `IndComp`

(b) Error Vs `IntSync`

Fig. 1: Comparison of error for different distortion levels (i.e, ϵ). Results are shown for $\epsilon = 0.01, 0.1, 1,$ and 10.

Moreover, the algorithm necessarily switches to state 2 from state 1, *cf.* Lemma 3. Thus, from (11), $\forall\ k \in \mathbb{Z}^+$, we have

$$f\big(\boldsymbol{x}_i^{(k)}\big) - f\big(\boldsymbol{x}^\star\big) \leq q^k \left(f\big(\boldsymbol{x}_i^{(0)}\big) - f\big(\boldsymbol{x}^\star\big) \right) + \frac{\gamma \epsilon^2 N^2}{2} \sum_{j=0}^{k-1} q^j.$$

Noting that $q < 1$, we take the limit as $k \to \infty$ to yield Part 1. Part 2 follows from Part 1 and [35, eq. 10, § 1.4]. Finally, Part 3 follows from Part 1 and [35, eq. 35, § 1.1]. $\square$

IV. NUMERICAL RESULTS

Let us first verify the convergence results of Proposition 1. To this end, we consider problem (1) with quadratic f_is, i.e., $f_i(\boldsymbol{x}) = \boldsymbol{x}^\mathsf{T} \boldsymbol{B}_i^\mathsf{T} \boldsymbol{B}_i \boldsymbol{x} + \boldsymbol{c}_i^\mathsf{T} \boldsymbol{x}$, where $\boldsymbol{B}_i^\mathsf{T} \boldsymbol{B}_i \in \mathbb{S}_{++}^n$, $\boldsymbol{c}_i \in \mathbb{R}^n$, and $\mathbb{S}_{++}^n$ is the positive definite cone. The entries of $\boldsymbol{B}_i$ and $\boldsymbol{c}_i$ are generated from a normal distribution. Note that ℓ_i and L_i are determined by $\boldsymbol{B}_i$, *cf.* **AS** 1. We let $N = 4$, $n = 10$, $\gamma = 1/(2L)$, and $r = 0.03$. Only the results related to Proposition 1-(1) is presented, since those related to Proposition 1-(2) and (3) behave similarly.

For comparison, we consider two algorithms. The first one is the classic GD, i.e., Algorithm 1 with $\epsilon = 0$ and $r = 0$. We also consider another algorithm which we refer to as inexact-GD with distributed synchrony (IGDDS), i.e., Algorithm 1 with $r = 0$ and $\forall\ i \in \mathcal{N}$, $\boldsymbol{\epsilon}_{ij}^{(k)} = \boldsymbol{\epsilon}_j^{(k)}$ [*cf.* (2)]. In this respect, the synchrony (4) holds for all $k \in \mathbb{Z}^+$ and we have $\limsup_{k \to \infty} \big(f\big(\boldsymbol{x}_i^{(k)}\big) - f(\boldsymbol{x}^\star) \big) \leq \epsilon^2 N^2 / (2\ell)$ [35], [36, § 4].

Figure 1(a) shows the error $f\big(\boldsymbol{x}^{(s+k)}\big) - f(\boldsymbol{x}^\star)$ vs global iteration index $s + k$ for different ϵ, *cf.* solid lines. Results are averaged over 1000 initializations $\boldsymbol{x}^{(0)}$, whose entries are normally distributed. Plots agree with Proposition 1-(1), i.e., the smaller the ϵ, the smaller the error of the optimality. Results with IGDDS are given in non-solid lines. Convergence rates and the suboptimality obtained by Algorithm 1 and IGDDS seem almost identical. This is expected because the convergence rate of Algorithm 1, i.e., $(1 - \gamma L \bar{r}^2 - \gamma \ell)$ and that of IGDDS, i.e., $(1 - \gamma \ell)$ are almost identical when $\gamma L \bar{r}^2 \ll 1 - \gamma \ell$. This condition is always realizable in practice, e.g., we have $\gamma L \bar{r}^2 = 0.0005$ and $1 - \gamma \ell = 0.9986$ in our simulation. A similar comparison holds for the suboptimality as well. Thus, results suggest that Algorithm 1 yields almost identical results to that of more constrained IGDDS.

Since IGDDS is technically equivalent to Algorithm 1 with $r = 0$, error-free communication is needed in every iteration to yield synchrony (4). However, Algorithm 1 does not require synchrony in every iteration. Therefore, for a fair comparison of Algorithm 1 and IGDDS in terms of communication overhead, it is instructive to plot the error versus the number of `IntSync` steps m, where s_m, $m \in \mathbb{Z}^+$ is the iteration index of s within Algorithm 1 at which the mth-synchrony is imposed.

Figure 1(b) shows the error $f\big(\boldsymbol{x}^{(s_m)}\big) - f(\boldsymbol{x}^\star)$ vs m with Algorithm 1, see thick solid lines. Results related to IGDDS are also plotted, see the non-solid lines. Clearly, there is a shift of the plots with IGDDS towards the right relative to the plots with Algorithm 1. Therefore, for all considered ϵ values, the number of `IntSync` steps m required to obtain a specified error with Algorithm 1 is smaller than with IGDDS. Moreover, if the number of `IntSync` steps m is fixed, the error with Algorithm 1 can be on the order of magnitude smaller than with IGDDS. This is useful in practice, because the cost of the error-free communication required for `IntSync` can be reduced with Algorithm 1 than with IGDDS. The benefits become greater as ϵ decreases. Finally, we plot results due to GD, see the thin solid line in Fig. 1(b). Results show that still the Algorithm 1 can benefit from less expensive `IndComp` steps. For example, in 150 `IntSync` steps, Algorithm 1 manages to yield an error significantly less than that from GD despite the value of ϵ. Clearly, GD outperforms Algorithm 1 if m is sufficiently large, since there are no inexactnesses. Thus, the results suggest if there is a choice for less expensive communication for `IndComp`, or a choice for allowing some inexactnesses, one can operate Algorithm 1 in a way there is a trade-off between the error and `IntSync` steps (m).

V. CONCLUSION

A gradient-like algorithm with guaranteed convergence has been developed to minimize a sum of peer objective functions through an interconnection network with multi-peer broadcast and multi-peer accumulation capabilities. Peer coordination can usually admit communications with bounded errors, however with some infrequent error-free synchronization epochs, which are dynamically triggered. Our algorithm can be attractive in many distributed applications, under inexact communication settings, such as decomposition with dual-subgradient methods and distributed learning systems with in-network computing capabilities, among others.

APPENDIX

A. Derivation of (5)

Suppose **AS** 1 and **AS** 2 hold. Moreover, let $\gamma \in (0, 1/\sum_{j=i}^{N} L_j]$. Then by recursively applying (3) followed by the use of the triangular inequality gives

$$\|\boldsymbol{x}_i^{(k)} - \boldsymbol{x}_j^{(k)}\| \le 2\epsilon N\gamma k, \quad \forall i,j \in \mathcal{N}. \tag{12}$$

From (2) and the gradient Lipshitz continuity of f_is, it follows that

$$\|\nabla f_j(\boldsymbol{x}_i^{(k)}) - \boldsymbol{h}_{ij}^{(k)}\| \le L_j\|\boldsymbol{x}_i^{(k)} - \boldsymbol{x}_j^{(k)}\| + \epsilon. \tag{13}$$

Finally, (5) follows from (13) by noting that $\nabla f(\boldsymbol{x}_i^{(k)}) = \sum_{j=1}^{N} \nabla f_j(\boldsymbol{x}_j^{(k)})$, the definition of $\boldsymbol{h}_i^{(k)}$, (12), and $\gamma \le \left(1/\sum_{j=i}^{N} L_j\right)$.

B. Proof of Lemma 1

Without loss of generality we may assume $s = 0$. Now, one can bound $f(\boldsymbol{x}_i^{(k+1)})$ as follows:

$$f(\boldsymbol{x}_i^{(k+1)}) \le f(\boldsymbol{x}_i^{(k)}) + \nabla f(\boldsymbol{x}_i^{(k)})^{\mathrm{T}}(\boldsymbol{x}_i^{(k+1)} - \boldsymbol{x}_i^{(k)})$$
$$(L/2)\|\boldsymbol{x}_i^{(k+1)} - \boldsymbol{x}_i^{(k)}\|^2 \tag{14}$$
$$\le f(\boldsymbol{x}_i^{(k)}) - \gamma\nabla f(\boldsymbol{x}_i^{(k)})^{\mathrm{T}}\boldsymbol{h}_i^{(k)} + (\gamma/2)\|\boldsymbol{h}_i^{(k)}\|^2 \tag{15}$$
$$= f(\boldsymbol{x}_i^{(k)}) - (\gamma/2)\|\nabla f(\boldsymbol{x}_i^{(k)})\|^2$$
$$(\gamma/2)\|\nabla f(\boldsymbol{x}_i^{(k)}) - \boldsymbol{h}_i^{(k)}\|^2 \tag{16}$$
$$\le f(\boldsymbol{x}_i^{(k)}) + \left(\frac{\gamma\bar{r}^2}{2} - \frac{\gamma}{2}\right)\|\nabla f(\boldsymbol{x}_i^{(k)})\|^2 \tag{17}$$
$$\le f(\boldsymbol{x}_i^{(k)}) + (\gamma L\bar{r}^2 - \gamma\ell)\left(f(\boldsymbol{x}_i^{(k)}) - f(\boldsymbol{x}^\star)\right) \tag{18}$$

where $\bar{r} = r/(1-r)$. Here (14) follows from the *descent lemma* [37, Lemma 5.7], (15) follows from (3) and noting that $\gamma L \le 1$, (16) follows from simple algebraic identities, (17) follows from (6), (18) follows from [35, Lemma 3, § 1.4] and [35, eq. 10, § 1.4] for bounding $-\|\nabla f(\boldsymbol{x}_i^{(k)})\|^2$ and $\|\nabla f(\boldsymbol{x}_i^{(k)})\|^2$, respectively. Now, subtracting $f(\boldsymbol{x}^\star)$ from the both sides of (18) yields the final result.

C. Proof of Lemma 2

To begin with, let us bound $f(\boldsymbol{x}_i^{(s+1)})$ as follows:

$$f(\boldsymbol{x}_i^{(s+1)}) \le f(\boldsymbol{x}_i^{(s)}) - (\gamma/2)\|\nabla f(\boldsymbol{x}_i^{(s)})\|^2$$
$$+ (\gamma/2)\|\nabla f(\boldsymbol{x}_i^{(s)}) - \boldsymbol{h}_i^{(s)}\|^2 \tag{19}$$
$$\le f(\boldsymbol{x}_i^{(s)}) - \gamma\ell\left(f(\boldsymbol{x}_i^{(s)}) - f(\boldsymbol{x}^\star)\right) + (\gamma\epsilon^2 N^2)/2 \tag{20}$$

where (19) is similar to (16) of the preceding lemma. (20) follows from [35, Lemma 3, § 1.4] for bounding $-\|\nabla f(\boldsymbol{x}_i^{(s)})\|^2$ and from that $\|\nabla f(\boldsymbol{x}_i^{(s)}) - \boldsymbol{h}_i^{(s)}\|^2 \le \epsilon N$, since the inner loop always starts from synchrony, *cf.* (5). Subtracting $f(\boldsymbol{x}^\star)$ from both sides yields (9). The latter part of the lemma is immediate from the Jensen's inequality.

D. Proof of Lemma 3

Suppose the algorithm remains in state 1 [4] without switching to state 2. For clarity, let $s \in \mathbb{Z}^+$ and $k \in \mathbb{Z}^+$ denote arbitrary iteration indices at which the *synchrony* is imposed and corresponding local iteration index within the inner loop, respectively. Thus, from Lemma 1 and (8), we have

$$f(\boldsymbol{x}_i^{(s+k)}) - f(\boldsymbol{x}^\star) \le q^{s+k}\left(f(\boldsymbol{x}_i^{(0)}) - f(\boldsymbol{x}^\star)\right). \tag{21}$$

Consequently, bounding $f(\boldsymbol{x}_i^{(s+k)}) - f(\boldsymbol{x}^\star)$ and $f(\boldsymbol{x}_i^{(0)}) - f(\boldsymbol{x}^\star)$ using [35, Lemma 3, § 1.4] and [35, eq. 10 § 1.4] respectively, we have

$$\|\nabla f(\boldsymbol{x}_i^{(s+k)})\|^2 \le (L/\ell)q^{s+k}\|\nabla f(\boldsymbol{x}_i^{(0)})\|^2. \tag{22}$$

Moreover, for $r \in (0, \sqrt{\ell}/(\sqrt{L}+\sqrt{\ell}))$, we have $q \in (0,1)$. Thus, $\exists s+k \in \mathbb{Z}^+$ such that guarantees [5]

$$\|\nabla f(\boldsymbol{x}_i^{(s+k)})\| < \epsilon N/\bar{r}. \tag{23}$$

It holds that

$$\|\boldsymbol{h}_i^{(s+k)}\| \le \|\nabla f(\boldsymbol{x}_i^{(s+k)})\| + \|\nabla f(\boldsymbol{x}_i^{(s+k)}) - \boldsymbol{h}_i^{(s+k)}\| \tag{24}$$
$$< \epsilon N/\bar{r} + 2\epsilon N(k+1/2) \tag{25}$$
$$< 2\epsilon N(k+1/2)(1/\bar{r}+1) = 2\epsilon N(k+1/2)/r \tag{26}$$

where (24) follows from triangular inequality, (25) follows from (23) and (5).

The inequality (26) is the inner loop exit criterion [*cf.* step 4] which transfers the control of the algorithm to `IntSync` at steps 6-7 of the algorithm. From (8) it follows that the inequality (21) holds even after the synchrony at `IntSync`. Thus, by following arguments identical to that of (22) - (26), we conclude that the control of the algorithm is next transferred to `IntSync` at steps 9-10. That is, the previous inner loop has been repeated only once, which is a contradiction. Therefore, the algorithm must switch to state 2.

E. Analysis with a General Peer-to-Peer Setting

In § II and § III, we focused on a network that can be modeled using a fully connected graph. However, the mathematical derivations can be extended to a more generalized peer-to-peer network that is modeled using a connected graph. Therefore, communication need not be coordinated by a central controller like in a federated setting. In the sequel, the main points of the derivations and related results are discussed.

Let us consider an arbitrary graph $\mathcal{G}(\mathcal{N}, \mathcal{L})$, where $\mathcal{N} = \{1, 2, \ldots, N\}$ represents the set of subsystems (SSs) of problem (1). Moreover, $\mathcal{L}$ represents a set of edges between SSs, where an edge is given by a pair (i,j), $i,j \in \mathcal{N}$. The graph is considered to be undirected. In other words, $(i,j) \in \mathcal{L} \iff (j,i) \in \mathcal{L}$. Communication from SS j to i is allowed if and only if there is a link (i,j) between the two nodes. We denote by $\mathcal{N}_i = \{j \mid (i,j) \in \mathcal{L}\}$, the set of

---

[4]More generally, the algorithm can be in a consecutive sequence of inner loops that are of state 1.

[5]To be precise $s + k \ge \left\lceil \frac{\ln(\epsilon^2 N^2 L\|\nabla f(\boldsymbol{x}_i^{(0)})\|^2) - \ln(\bar{r}^2\ell)}{\ln q}\right\rceil$, where $\lceil\cdot\rceil$ is the ceiling function.

neighbours of the SS i. Furthermore, we denote by $\mathcal{T}(\mathcal{N}, \bar{\mathcal{L}})$, a spanning tree of the graph $\mathcal{G}$ where $\bar{\mathcal{L}}$ denotes the set of edges in $\mathcal{T}$. We also define $\mathcal{T}_i = \{j \mid (i,j) \in \bar{\mathcal{L}}\}$, the neighbors of the SS i in the spanning tree.

Now, we note that the gradient measurement model in (2) is going to be modified as follows in the general setting:

$$h_{ij}^{(k)} = \begin{cases} \nabla f_j(x_j^{(k)}) + \epsilon_{ij}^{(k)} & \text{if } (i,j) \in \mathcal{L} \\ \mathbf{0} & \text{otherwise.} \end{cases} \quad (27)$$

Consequently, it is immediate that

$$\left\| h_{ij}^{(k)} - \nabla f_j(x_j^{(k)}) \right\| = \begin{cases} \left\| \epsilon_{ij}^{(k)} \right\| & \text{if } (i,j) \in \mathcal{L} \\ \left\| \nabla f_j(x_j^{(k)}) \right\| & \text{otherwise.} \end{cases} \quad (28)$$

It is worth highlighting that the generalized setting requires an additional assumption unlike the fully connected setting considered in § II and § III, which we will outline next.

AS 3. *The gradients ∇f_is of the objective functions f_i, $i \in \mathcal{N}$ are bounded, i.e., $\|\nabla f_i(x)\| \leq \zeta$ for some $\zeta > 0$ for all x, i.*

Hence, from **AS** 2 and **AS** 3, together with (28), it is easily verified that

$$\left\| h_{ij}^{(k)} - \nabla f_j(x_j^{(k)}) \right\| \leq \tau \quad (29)$$

where $\tau = \max(\epsilon, \zeta)$. Thus, the gradient measurement $h_{ij}^{(k)}$ obeys the following remark:

Remark 1. $\forall\, i, j \in \mathcal{N}$, s.t. $i \neq j$, *gradient measurement $h_{ij}^{(k)} \in \mathbb{R}^n$ received by i-th SS from j-th SS at k-th iteration is given by*

$$h_{ij}^{(k)} = \nabla f_j(x_j^{(k)}) + \tau_{ij}^{(k)} \quad (30)$$

where $\tau_{ij}^{(k)} \in \mathbb{R}^n$ is a error such that $\|\tau_{ij}^{(k)}\| \leq \tau$ with $\|\cdot\|$ denoting the Euclidean norm.

Note that Remark 1 play the role of **AS** 2.

Let us next outline the modified version of Algorithm 1.

Algorithm 2 Inexact GD with `IndComp−IntSync` over a General Graph

Input: $x_j^{(0)} = x_i^{(0)} \ \forall\, i, j \in \mathcal{N}$, $\tau \geq 0$, $r \in (0, \sqrt{\ell}/(\sqrt{L} + \sqrt{\ell}))$, $s = 0, k = 0$

1: **repeat**
2:    **repeat**
3:       $\forall\, i \in \mathcal{N}$, compute $x_i^{(s+k+1)}$ from (3), $k \leftarrow k + 1$
4:    **until** $\exists\, i \in \mathcal{N}$, $k - 1 > r\|h_i^{(s+k-1)}\|/(2\tau N) - 1/2$
5:    **if** $k \neq 1$ **then**
6:       $\forall\, i \in \mathcal{N}$, $x_i^{(s+k-1)} \leftarrow \frac{1}{N}\sum_{j=1}^N x_j^{(s+k-1)}$ ▷ performed through $\mathcal{T}(\mathcal{G}, \bar{\mathcal{L}})$.
7:       $s \leftarrow s + k - 1, \ k \leftarrow 0$
8:    **else**
9:       $\forall\, i \in \mathcal{N}$, $x_i^{(s+k)} \leftarrow \frac{1}{N}\sum_{j=1}^N x_j^{(s+k)}$ ▷ performed through $\mathcal{T}(\mathcal{G}, \bar{\mathcal{L}})$.
10:       $s \leftarrow s + k, \ k \leftarrow 0$
11:    **end if**
12: **until** a stopping criterion true

Now, one can easily see that an identical result to Proposition 1 holds even in the general setting if **AS** 2 is replaced by

Remark 1 above. More specifically, we have the following result:

Proposition 2. *Suppose AS 1, Remark 1 hold. Let $\{x_i^{(k)}\}_{k \in \mathbb{Z}^+}$, $i \in \mathcal{N}$, be the sequence of local copies of the decision variable generated by Algorithm 2. Then for $\gamma \in (0, 1/L]$*

1) $\limsup\limits_{k \to \infty} \left(f(x_i^{(k)}) - f(x^\star) \right) \leq \tau^2 N^2/(2(\ell - L\bar{r}^2))$

2) $\limsup\limits_{k \to \infty} \|\nabla f(x_i^{(k)})\| \leq \sqrt{L\tau^2 N^2/(\ell - L\bar{r}^2)}$

3) $\limsup\limits_{k \to \infty} \|x_i^{(k)} - x^\star\| \leq \sqrt{L\tau^2 N^2/(\ell^2 - L\bar{r}^2 \ell)}$

where $L = \sum_{j=1}^N L_j$, $\ell = \min_{j \in \mathcal{N}} \ell_j$, $\bar{r} = r/(1-r)$, and $x^\star = \arg\min_x f(x)$.

REFERENCES

[1] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods.* MA: Athena Scientific, 1997.

[2] J. Dean, G. Corrado, R. Monga, *et al.*, "Large scale distributed deep networks," *Adv. Neural Inf. Process. Syst.*, vol. 25, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., 2012.

[3] T. Chilimbi, Y. Suzue, J. Apacible, and K. Kalyanaraman, "Project ADAM: Building an efficient and scalable deep learning training system," *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation*, pp. 571–582, Oct. 2014.

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-efficient learning of deep networks from decentralized data," *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, A. Singh and J. Zhu, Eds., ser. Proceedings of Machine Learning Research, vol. 54, PMLR, 20–22 Apr 2017, pp. 1273–1282.

[5] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," 2016. arXiv: 1610.02527.

[6] A. Khaled, K. Mishchenko, and P. Richtárik, "First analysis of local GD on heterogeneous data," 2019. arXiv: 1909.04715.

[7] S. U. Stich, "Local SGD converges fast and communicates little," *7th International Conference on Learning Representations*, May 2019.

[8] J. Wang and G. Joshi, "Adaptive communication strategies to achieve the best error-runtime trade-off in local-update SGD," *Proceedings of Machine Learning and Systems 1*, 2019.

[9] D. P. Palomar and Y. C. Eldar, *Convex Optimization in Signal Processing and Communications.* NY: Cambridge Univ. Press, 2010.

[10] A. Nedić, "Distributed gradient methods for convex machine learning problems in networks: Distributed optimization," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 92–101, May 2020.

[11] L. Xiao, M. Johansson, and S. P. Boyd, "Simultaneous routing and resource allocation via dual decomposition," *IEEE. Trans. Commun.*, vol. 52, no. 7, pp. 1136–1144, Jul. 2004.

[12] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, "Layering as optimization decomposition: A mathematical theory of network architectures," *Proc. IEEE*, vol. 95, no. 1, pp. 255–312, Jan. 2007.

[13] A. Nedić and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Automat. Contr.*, vol. 54, no. 1, pp. 48–61, Jan. 2009.

[14] A. S. Berahas, R. Bollapragada, and E. Wei, "On the convergence of nested decentralized gradient methods with multiple consensus and gradient steps," *IEEE Trans. Signal Process.*, vol. 69, pp. 4192–4203, Jul. 2021.

[15] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A Survey on distributed machine learning," *ACM Computing Surveys*, vol. 53, no. 2, Mar. 2020.

[16] P. Kairouz, H. B. McMahan, B. Avent, *et al.*, "Advances and open problems in federated learning," *Foundations and Trends in Machine Learning*, vol. 14, no. 1-2, Jun. 2021.

[17] Y. Chen, R. S. Blum, M. Takac, and B. M. Sadler, "Distributed learning with sparsified gradient differences," *IEEE J. Sel. Top. Signal Process.*, vol. 16, no. 3, pp. 585–600, Apr. 2022.

[18] S. Magnússon, C. Enyioha, N. Li, C. Fischione, and V. Tarokh, "Convergence of limited communication gradient methods," *IEEE Trans. Automat. Contr.*, vol. 63, no. 5, pp. 1356–1371, May 2018.

[19] S. Magnússon, H. Shokri-Ghadikolaei, and N. Li, "On maintaining linear convergence of distributed learning and optimization under limited communication," *IEEE Trans. Signal Process.*, vol. 68, pp. 6101–6116, Nov. 2020.

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

[21] S. Magnusson, C. Enyioha, N. Li, C. Fischione, and V. Tarokh, "Communication complexity of dual decomposition methods for distributed resource allocation optimization," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 4, pp. 717–732, Aug. 2018.

[22] S. Khirirat, S. Magnússon, and M. Johansson, "Compressed gradient methods with Hessian-aided error compensation," *IEEE Trans. Signal Process.*, vol. 69, pp. 998–1011, 2021.

[23] C. S. Lee, N. Michelusi, and G. Scutari, "Finite rate quantized distributed optimization with geometric convergence," *52nd Asilomar Conf. Signals, Syst., Comput.*, Oct. 2018.

[24] A. Nedić, A. Olshevsky, A. Ozdaglar, and J. N. Tsitsiklis, "Distributed subgradient methods and quantization effects," *Proc. IEEE Conf. Decis. Control*, pp. 4177–4184, Dec. 2008.

[25] A. Koloskova, S. U. Stich, and M. Jaggi, "Decentralized stochastic optimization and gossip algorithms with compressed communication," *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, pp. 3478–3487, Jun. 2019.

[26] J. George and P. Gurram, "Distributed stochastic gradient descent with event-triggered communication," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, pp. 7169–7178, Apr. 2020.

[27] X. Cao and T. Başar, "Decentralized online convex optimization with event-triggered communications," *IEEE Trans. Signal Process.*, vol. 69, pp. 284–299, 2021.

[28] D. Jakovetić, D. Bajović, N. Krejić, and N. K. Jerinkić, "Distributed gradient methods with variable number of working nodes," *IEEE Trans. Signal Process.*, vol. 64, no. 15, pp. 4080–4095, Aug. 2016.

[29] D. Alistarh, T. Hoefler, M. Johansson, S. Khirirat, N. Konstantinov, and C. Renggli, "The convergence of sparsified gradient methods," *Adv. Neural Inf. Process. Syst.*, vol. 31, pp. 5973–5983, 2018.

[30] D. Alistarh, D. Grubic, J. Z. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," *Adv. Neural Inf. Process. Syst.*, vol. 30, no. 1, pp. 1710–1721, Dec. 2017.

[31] H. Yu, S. Yang, and S. Zhu, "Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 5693–5700, Jul. 2019.

[32] F. Zhou and G. Cong, "On the convergence properties of a K-step averaging stochastic gradient descent algorithm for nonconvex optimization," *IJCAI International Joint Conference on Artificial Intelligence*, pp. 3219–3227, Jul. 2018.

[33] C. Xie, S. Zheng, O. Koyejo, I. Gupta, M. Li, and H. Lin, "CSER: Communication-efficient SGD with error reset," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 12 593–12 603, 2020.

[34] A. Goldsmith, *Wireless Communications*. UK: Cambridge University Press, 2005.

[35] B. T. Polyak, *Introduction to Optimization* (Translations Series in Mathematics and Engineering). NY: Optimization Software, Publications Division, 1987.

[36] A. Ajalloeian and S. U. Stich, "On the convergence of SGD with biased gradients," 2020. arXiv: 2008.00051.

[37] A. Beck, *First-Order Methods in Optimization* (MOS-SIAM Series on Optimization). Philadelphia, PA: Society for Industrial and Applied Mathematics, 2017.