Low Sample and Communication Complexities in Decentralized Learning: A Triple Hybrid Approach

Xin Zhang[1] Jia Liu[2] Zhengyuan Zhu[1] Elizabeth Serena Bentley[3]
[1]Department of Statistics, Iowa State University
[2]Department of Electrical and Computer Engineering, The Ohio State University
[3]Air Force Research Laboratory, Information Directorate

***Abstract*—Network-consensus-based decentralized learning optimization algorithms have attracted a significant amount of attention in recent years due to their rapidly growing applications. However, most of the existing decentralized learning algorithms could not achieve low sample and communication complexities simultaneously – two important metrics in evaluating the trade-off between computation and communication costs of decentralized learning. To overcome these limitations, in this paper, we propose a triple hybrid decentralized stochastic gradient descent (TH-DSGD) algorithm for efficiently solving non-convex network-consensus optimization problems for decentralized learning. We show that to reach an ϵ^2-stationary solution, the total sample complexity of TH-DSGD is $O(\epsilon^{-3})$ and the communication complexity is $O(\epsilon^{-3})$, both of which are *independent* of dataset sizes and significantly improve the sample and communication complexities of the existing works. We conduct extensive experiments with a variety of learning models to verify our theoretical findings. We also show that our TH-DSGD algorithm is stable as the network topology gets sparse and enjoys better convergence in the large-system regime.**

I. Introduction

In recent years, advances in machine learning have achieved enormous success in a many areas, including medical image analysis, financial prediction, natural language processing, just to name a few. Mathematically, the training phase of machine learning can be formulated as solving an optimization problem. However, with the rapidly growing size of training datasets and high dimensionality of modern (deep) learning models, efficient and scalable learning is increasingly challenging. To address this difficulty, a natural idea is to utilize *decentralized* computational resources in a networked system, which could be in either a server-worker architecture [1], [2], [3] or a distributed peer-to-peer network structure [4], [5]. Moreover, thanks to the advantages in system robustness, data privacy and implementation simplicity, decentralized learning over networked systems has received a lot of attention recently, and has found many new and emerging applications in various science and engineering fields (e.g., dictionary learning [6], multi-agent systems [7], [8], [9], multi-task learning [10], [11], information retrieval [12], energy allocation [13], etc.).

Among the large body of literature of decentralized learning over networks, one of the most popular and effective approaches is the so-called network consensus optimization, which traces its roots to the seminal work by Tsitsiklis in 1984 [14]. In addition, there have been ever-increasing applications of network consensus optimization in the areas of robotics control [8], [9], network resource allocation[13], [15], etc. In network consensus optimization, there are a set of geographically dispersed computing nodes collaborating to solve an optimization problem. Each node is associated with a local dataset that could be either too large to be aggregated at a central location due to high communication costs, or need to stay locally because of privacy/security concerns. A defining feature of network consensus optimization is that there is *no* shared memory due to the lack of a dedicated parameter server. As a result, every node needs to exchange information with its neighbors to arrive at a consensus of a global optimal solution, hence the name "network consensus optimization."

However, the design of high-performance network consensus optimization algorithms faces two fundamentally *conflicting* challenges: i) On one hand, due to the high dimensionality of most learning problems, it is only feasible to use first-order (stochastic) gradient information to determine the update direction in each iteration. The variance of a stochastic gradient highly depends on the number of samples in each mini-batch. However, the more samples an algorithm uses, the higher the computational cost. The high computational cost is further exacerbated by the fact that in edge-computing environments, the computing devices could be limited by hardware resource and/or battery capacity. ii) On the other hand, if one intends to use fewer samples per iteration to lower the computational cost, the stochastic gradient information is noisier, which necessitates more communication rounds to reach certain training accuracy (i.e., slow convergence). The low communication efficiency is particularly problematic in wireless edge networks, where the communication links could be low-speed and highly dynamic.

Due to the above fundamental trade-off between computation and communication costs, the notions of sample complexity and communication complexity (to be formally defined in Section II) become two of the most important performance metrics in evaluating the performances of a given decentralized learning algorithm. However, in the literature, most existing

This work has been supported in part by NSF grants CAREER-CNS-1943226, ECCS-1818791, CCF-2110252, ONR grant N00014-17-1-2417, AFRL grant FA8750-18-1-0107, USDA grant 68-7482-17-009. Distribution A. Approved for public release: Distribution unlimited 88ABW-2020-3447 on 04 Nov 2020. ACKNOWLEDGMENT OF SUPPORT AND DISCLAIMER: (a) Contractor acknowledges Government's support in the publication of this paper. This material is based upon work funded by AFRL, under AFRL Contract No. #FA87501810107. (b) Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of AFRL.

decentralized learning algorithms have either low sample complexity or low communication complexity, but not both (see Section II for more in-depth discussions). The limitations of these existing approaches naturally prompt the following question: *Could we design a decentralized learning algorithm that strikes a good balance between sample complexity and communication complexity?*

In this paper, we show that the answer to the above question is affirmative by proposing a new *triple hybrid approach* that achieves low sample and communication complexities. Our main results and contributions are summarized as follows:

- We propose a new decentralized stochastic gradient descent algorithm for solving non-convex network consensus optimization problems. Specifically, inspired by a hybrid SARAH-SGD estimator developed for centralized machine learning [16], we proposed a *decentralized* learning algorithm that first updates the local gradient estimator with a modified SARAH-SGD hybrid estimator and then estimates the global gradient with the gradient tracking method. In this sense, our proposed algorithm can be viewed as a *triple hybrid* of the decentralized stochastic gradient descent, variance reduction and gradient tracking techniques. For this reason, we name our proposed algorithm "triple hybrid decentralized stochastic gradient descent" (TH-DSGD) method.
- We show that under some mild assumptions and parameter conditions, our TH-DSGD algorithm enjoys an $O(T^{-2/3})$ convergence rate, where T is the maximum number of iterations the algorithm is executed. Note that this rate is much faster than the rate of $O(T^{-1/2})$ for the classic decentralized stochastic algorithms, e.g., DSGD [17], PSGD [5] and GNSD [18]. Also, we show that to reach an ϵ^2-stationary solution (to be defined in Section II) in non-convex learning, the sample complexity of TH-DSGD is $O(m\epsilon^{-3})$ and the communication complexity is $O(\epsilon^{-3})$, which are *independent* of dataset sizes and strike a good balance between these two performance metrics.
- To examine the performance of our TH-DSGD algorithm and verify our theoretical results, we conduct extensive numerical experiments by using non-convex logistic regression with the LibSVM datasets [19]. Our results show that the proposed TH-DSGD algorithm outperforms the other existing state-of-the-art algorithms for problem instances with large datasets and networked systems with low computational speeds. Also, we numerically show that our TH-DSGD is stable as the network topology gets sparse and enjoys better convergence performance in the large-system regime.

The rest of the paper is organized as follows. In Section II, we first provide the preliminaries of network consensus optimization and discuss related works with a focus on sample and communication complexities. In Section III, we present our proposed TH-DSGD algorithm. The convergence rate and sample complexity analysis of TH-DSGD are also provided in Section III. We provide numerical results in Section IV to verify the theoretical results of our TH-DSGD algorithm. In Section V, we provide concluding remarks.

II. Preliminaries and Related Work

To facilitate subsequent technical discussions, in Section II-A, we first provide a primer on the basics of network consensus optimization and formally define the notions of sample and communication complexities of decentralized optimization algorithms for solving network consensus optimization problems. Next, in Section II-B, we give an overview on the historical development of (centralized) stochastic first-order optimization algorithms for solving non-convex learning problems in terms of their sample and communication complexities. Here, we pay special attention to several algorithmic acceleration techniques that motivate our TH-DSGD algorithm. Lastly, in Section II-C, we focus on the recent advances of optimization algorithms for solving decentralized learning problems, putting our work into comparative perspectives.

A. Network Consensus Optimization: A Primer

As mentioned in Section I, in decentralized learning, there are a set of geographically dispersed computing nodes that form a network. We represent a networked distributed computing system as an undirected connected network $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ and $\mathcal{L}$ are the sets of nodes and edges, respectively, with $|\mathcal{N}| = m$. The nodes have local computation capabilities and are able to communicate with their neighbors via the edges in $\mathcal{L}$. The goal of decentralized learning is to have the nodes *distributively* and *collaboratively* solving a network-wide optimization problem as follows:

$$\min_{\mathbf{x}\in\mathbb{R}^p} f(\mathbf{x}) = \min_{\mathbf{x}\in\mathbb{R}^p} \frac{1}{m}\sum_{i=1}^{m} f_i(\mathbf{x}), \tag{1}$$

where each local objective function $f_i(\mathbf{x}) \triangleq \mathbb{E}_{\zeta\sim\mathcal{D}_i} f_i(\mathbf{x};\zeta)$ is only observable to node i and not necessarily convex. Here, $\mathcal{D}_i$ represents the distribution of the dataset at node i, which could be a finite dataset (offline finite-sum setting) or an infinite dataset (online setting), and $f_i(\mathbf{x},\zeta)$ represents a loss function that evaluates the discrepancy between the learning model's output and the ground truth of a training sample ζ. To solve Problem (1) in a decentralized fashion, a common approach is to reformulate Problem (1) in the following equivalent form:

$$\begin{aligned} \text{Minimize} \quad & \frac{1}{m}\sum_{i=1}^{m} f_i(\mathbf{x}_i) \\ \text{subject to} \quad & \mathbf{x}_i = \mathbf{x}_j, \qquad \forall (i,j)\in\mathcal{L}. \end{aligned} \tag{2}$$

where $\mathbf{x} \triangleq [\mathbf{x}_1^\top, \cdots, \mathbf{x}_m^\top]^\top$, and $\mathbf{x}_i$ is an introduced local copy at node i. In Problem (2), the constraints enforce that the local copy at each node is equal to those of its neighbors, hence the name "consensus." Clearly, the solution of Problems (1) solves Problem (2) and vice versa.

In the literature of network consensus optimization, a main goal in algorithm design for solving Problem (2) is to attain an ϵ^2-stationary point $\mathbf{x}$ defined as follows:

$$\underbrace{\left\|\frac{1}{m}\sum_{i=1}^{m}\nabla f_i(\bar{\mathbf{x}})\right\|^2}_{\text{Global gradient magnitude}} + \underbrace{\frac{1}{m}\sum_{i=1}^{m}\|\mathbf{x}_i - \bar{\mathbf{x}}\|^2}_{\text{Consensus error}} \le \epsilon^2, \tag{3}$$

where $\bar{\mathbf{x}} \triangleq \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i$ represents the spatial average across all nodes. Note that different from the ϵ^2-stationary point for traditional centralized optimization problems, the above criterion in Eq. (3) contains two components: the first term is the global gradient magnitude for the non-convex objectives and the second term is the average consensus error across all local copies in the network. So far, various decentralized algorithms have been proposed to find the above ϵ^2-stationary point (see more in-depth discussion in Section II-C). However, most of these algorithms suffer either a high sample complexity due to full gradient evaluation or a high communication complexity. To facilitate subsequent discussions, we first state the formal definitions of sample complexity and communication complexity from the literature (see, e.g., [20]):

Definition 1 (Sample Complexity)**.** *The sample complexity is defined as the total number of the incremental first-order oracle (IFO) calls required across all the nodes to find an ϵ^2-stationary point defined in Eq. (3), where one IFO call evaluates a pair of $(f_i(\mathbf{x};\zeta), \nabla f_i(\mathbf{x};\zeta))$ on a sample $\zeta \sim \mathcal{D}_i$ and parameter $\mathbf{x} \in \mathbb{R}^p$ at node i.*

Definition 2 (Communication Complexity)**.** *The communication complexity is defined as the total rounds of communications required to find an ϵ^2-stationary point defined in Eq. (3), where each node can send and receive a p-dimensional vector with its neighboring nodes in one communication round.*

To put these two performance metrics into perspective, consider the classic (centralized) gradient descent (GD) method in optimization for solving Problem (1) as an example. It is well-known that the GD method has an $O(1/T)$ convergence rate for non-convex objective functions, which implies $O(\epsilon^{-2})$ communication complexity. Also, it requires a full gradient evaluation in each iteration, i.e., $O(n)$ sample complexity per iteration, where n is the total number of training samples. This implies $O(n\epsilon^{-2})$ sample complexity to reach an ϵ^2-stationary point. Clearly, if the dataset size n is large, the sample complexity of the GD method is high.

In comparison, consider yet another classic algorithm – the stochastic gradient descent (SGD), which is the most widely used algorithm in machine learning. A key motivation of using the SGD algorithm is to reduce the computational cost by using only a mini-batch of training samples for gradient estimation in each iteration. It is well-known that (see, e.g.,[21], [22], [23]), due to the noise from the random training samples in the mini-batch, the convergence rate of the SGD algorithm for non-convex optimization is reduced to $O(1/\sqrt{T})$. Thus, to attain an ϵ^2-stationary point $\mathbf{x}$ with $\|\nabla f(\mathbf{x})\|^2 \leq \epsilon^2$, the SGD method has $O(\epsilon^{-4})$ sample complexity, which could be either higher or lower than the $O(n\epsilon^{-2})$ sample complexity of the GD method, depending on the relationship between n and ϵ.

B. Related Work: Centralized Stochastic First-Order Methods with Improved Sample and Communication Complexities

Next, we provide an overview on several centralized stochastic first-order optimization methods, which inspire our TH-DSGD algorithm design. In the context of centralized learning, to reduce the overall sample and communication complexities of the classic GD and SGD algorithms, various variance reduction methods have been proposed, most notably SVRG [24], [25], SAGA [26] and SCSG [27]. It have been shown that these algorithms yield an overall sample complexity of $O(n + n^{2/3}\epsilon^{-2})$. Recently, Fang *et al.* [28] proposed another variance reduction method named stochastic path-integrated differential estimator (SPIDER) based on the SARAH gradient estimator developed by Nguyen *et al.* [29]. SPIDER further improves the overall sample complexity to $O(n + \sqrt{n}\epsilon^{-2})$, which matches the theoretical lower bound $\Omega(\sqrt{n}\epsilon^{-2})$ for finding an ϵ^2-stationary point when $n = O(\epsilon^{-4})$. However, in practice, SPIDER does not perform well due to the constraint on the small step-size $O(\epsilon L^{-1})$. In [30], Wang *et al.* proposed an algorithm named SpiderBoost that allows a larger constant step-size $O(L^{-1})$, while the sample complexity remains being $O(n+\sqrt{n}\epsilon^{-2})$. However, it should be noted that the significant complexity improvement is due to a more restrictive assumption that the same universal Lipschitz smoothness parameter exists for all individual components $f(\cdot;\zeta_i)\ \forall i$, which implies that the objectives are "similar" across all training samples in terms of smoothness. Meanwhile, to achieve the optimal complexity, SPIDER and SpiderBoost require a (nearly) full gradient evaluation every $\sqrt{n}$ iterations and a mini-batch stochastic gradient evaluation with batch size $\sqrt{n}$ at each iteration.

To address the above limitations, very recently, Tran-Dinh *et al.* [16] introduced a hybrid stochastic gradient descent (Hybrid-SGD) algorithm, where the main idea is to use a convex combination of the SARAH estimator [29] and an unbiased stochastic gradient as the gradient estimator. In Hybrid-SGD, the individual Lipschitz smoothness assumption in SpiderBoost is relaxed to an average Lipschitz smoothness assumption. Also, it only needs to evaluate gradient with *two* samples per iteration. It is shown that Hybrid-SGD yields a sample complexity of $O(\epsilon^{-3})$, which is *independent* of dataset size. These salient features motivate us to propose a "triple hybrid" approach for *decentralized learning* following a similar token. Interestingly, we show that in decentralized settings, our TH-DSGD method can further improve the gradient evaluation to only *one sample* per iteration, while not degrading the communication complexity order.

C. Related Work: Decentralized Optimization Algorithms

In the literature, various decentralized learning optimization algorithms have been developed to solve Problem (1), e.g., first-order methods [4], [31], [32], [33], prime-dual methods [34], [35], the Newton's method [36], [37], etc. We refer the readers to recent surveys [38], [39] for a comprehensive review. In this work, we focus on decentralized first-order methods for non-convex network consensus optimization in Problem (2). Towards this end, Zeng and Yin [40] studied the convergence rate of the well-known decentralized gradient descent (DGD) algorithm [4] for non-convex optimization and showed that the algorithm converges with an $O(1/T)$ rate to a step-size-dependent error neighborhood of a stationary point when a constant step-size is used. Later, Lorenzo and Scutari [33]

proposed a gradient tracking method to find an ϵ^2-stationary point with $O(1/T)$ by adopting a constant step-size. However, these methods utilize a full gradient evaluation at each iteration, which leads to $O(n\epsilon^{-2})$ sample complexity.

To reduce the per-iteration sample complexity, stochastic algorithms have been adopted in the decentralized optimization framework, e.g., DSGD [17], PSGD [5], GNSD [18], etc. In these algorithms, a stochastic gradient is used to approximate the true gradient. Due to the random noise in stochastic gradients, the convergence rate is decayed to $O(1/\sqrt{T})$. Thus, the total sample and communication complexities for these stochastic algorithms are $O(\epsilon^{-4})$ and $O(m^{-1}\epsilon^{-4})$, much higher than $O(mn\epsilon^{-2})$ total sample and $O(\epsilon^{-2})$ communication complexities for the deterministic counterparts.

To address the limitations in stochastic decentralized algorithms, in this work, we propose to adopt the variance reduction techniques from centralized optimization in Section II-B to reduce the sample and communication complexities for the non-convex network consensus optimization. In the literature, there are several works on the decentralized stochastic variance reduction methods, such as DSA [41], diffusion-AVRG [42] and GT-SAGA [43] etc. However, most of these works focus on convex objectives. To our knowledge, the only algorithm for non-convex problems is the decentralized gradient estimation and tracking (D-GET) algorithm proposed by Sun *et al.* [20], which is the most related work to ours. D-GET combines the decentralized gradient tracking method [18] and the SpiderBoost gradient estimator [30], and has $O(mn + m\sqrt{n}\epsilon^{-2})$ *dataset-size-dependent* sample complexity and $O(\epsilon^{-2})$ communication complexity. Recall that the sample and communication complexities of TH-DSGD are $O(m\epsilon^{-3})$ and $O(\epsilon^{-3})$, respectively. Thus, if dataset size $n = \Omega(\epsilon^{-2})$, D-GET has a higher sample complexity than TH-DSGD. For example, for $\epsilon = 0.01$, if n is on the order of ten thousands (not uncommon in modern machine learning datasets), D-GET has higher sample complexity. Moreover, D-GET still suffers the same Lipschitz smoothness weakness of SpiderBoost as we discussed in Section II-B.

III. Triple Hybrid Decentralized Stochastic Gradient Descent Method (TH-DSGD)

In this Section, we will first introduce our TH-DSGD algorithm in Section III-A. Then, we will present the main theoretical results and their key insights in Section III-B. The proofs for the main results will be provided in Section III-C.

A. The TH-DSGD Algorithm

To solve Problem (2), in the literature, a well-known starting point is to reformulate the problem as [4]:

$$\begin{aligned} \text{Minimize} \quad & \frac{1}{m}\sum_{i=1}^{m} f_i(\mathbf{x}_i) \\ \text{subject to} \quad & (\mathbf{W}\otimes \mathbf{I}_p)\mathbf{x} = \mathbf{x}, \end{aligned} \tag{4}$$

where $\mathbf{I}_p$ denotes the p-dimensional identity matrix, the operator $\otimes$ denotes the Kronecker product, and $\mathbf{W} \in \mathbb{R}^{m\times m}$ is often referred to as the consensus matrix. We use $[\mathbf{W}]_{ij}$ to represent the entry in the i-th row and the j-th column in $\mathbf{W}$. In order for Problem (4) to be equivalent to Probelm (2), $\mathbf{W}$ should satisfy the following properties:

(a) *Doubly Stochastic:* $\sum_{i=1}^{m}[\mathbf{W}]_{ij} = \sum_{j=1}^{m}[\mathbf{W}]_{ij} = 1$, $\forall i, j \in \mathcal{N}$.
(b) *Symmetric:* $[\mathbf{W}]_{ij} = [\mathbf{W}]_{ji}$, $\forall i, j \in \mathcal{N}$.
(c) *Network-Defined Sparsity Pattern:* $[\mathbf{W}]_{ij} > 0$ if $(i, j) \in \mathcal{L}$; otherwise $[\mathbf{W}]_{ij} = 0$, $\forall i, j \in \mathcal{N}$.

These properties imply that all eigenvalues of $\mathbf{W}$ are real and lie in the interval $(-1, 1]$, and thus can be sorted. Without loss of generality, we denote these eigenvalues as $-1 < \lambda_m \leq \cdots \leq \lambda_1 = 1$. We let $\lambda \triangleq \max\{|\lambda_2|, |\lambda_m|\}$ denote the second-largest eigenvalue in magnitude. We will see later that λ plays an important role in the step-size selection and characterizing the convergence rate of our proposed algorithm.

As mentioned in Section II-B, our TH-DSGD algorithm is inspired by the Hybrid-SGD estimator [16] proposed for centralized learning. Therefore, we restate the search direction of Hybrid-SGD (using our notation) as follows:

$$\begin{aligned} \mathbf{v}_t = \beta \underbrace{\left(\mathbf{v}_{t-1} + \nabla f(\mathbf{x}_t;\zeta_t) - \nabla f(\mathbf{x}_{t-1};\zeta_t)\right)}_{\text{SARAH estimator}} \\ + (1-\beta)\underbrace{\nabla f(\mathbf{x}_t;\xi_t)}_{\text{Stochastic gradient}}, \end{aligned} \tag{5}$$

where ζ_t and ξ_t are two independent random training samples and $\beta \in (0, 1)$ is a parameter of the algorithm. It can be seen that the gradient estimator in (5) is a convex combination of two independent components: the first one is a *biased* SARAH estimator [29] and the second one is an *unbiased* stochastic gradient estimator.

Clearly, the hybrid estimator in (5) requires two independent training samples (or two independent mini-batches of training samples), which complicates the estimation procedure, (e.g., waiting for enough data and/or splitting the dataset, etc.). Moreover, from the theoretical result in [16], the optimal $(1-\beta)$ should be on the order of $O(T^{-2/3})$, which is close to zero for typical T-values in practice. As a result, the information in the unbiased part could be overwhelmed by the biased SARAH estimator. Moreover, due to the need for minimizing the consensus error (i.e., $\sum_{i=1}^{m}\|\mathbf{x}_i - \bar{\mathbf{x}}\|^2$), it is infeasible to directly apply the hybrid estimator in (5) in decentralized consensus optimization.

To achieve the benefits of (5) in decentralized learning while avoiding its limitations, in this paper, we propose a new hybrid estimator. First, each node estimates its local gradient as:

$$\mathbf{v}_{i,t} = \beta\mathbf{v}_{i,t-1} + \nabla f_i(\mathbf{x}_{i,t};\zeta_{i,t}) - \beta\nabla f_i(\mathbf{x}_{i,t-1};\zeta_{i,t}). \tag{6}$$

It is easy to see that the structure of Eq. (6) is also a hybrid of the biased SARAH estimator and an unbiased stochastic gradient estimator similar to Eq. (5), with the exception that Eq. (6) is only based on *a single sample*.

Then, we track the global gradient by performing the following local weighted aggregation:

$$\mathbf{y}_{i,t} = \sum\nolimits_{j\in\mathcal{N}_i}[\mathbf{W}]_{ij}\mathbf{y}_{i,t-1} + \mathbf{v}_{i,t} - \mathbf{v}_{i,t-1} \tag{7}$$

where $\mathbf{x}_{i,t}$, $\mathbf{v}_{i,t}$ and $\zeta_{i,t}$ are the parameter, hybrid gradient estimator and a random sample at node i in the tth iteration, respectively, and $\mathcal{N}_i \triangleq \{j \in \mathcal{N} : (i,j) \in \mathcal{L}\}$. Note that Eq. (7) shares a similar spirit with the gradient tracking technique [33].

Lastly, we update local parameters following the conventional decentralized stochastic gradient descent (DSGD) step [4]:

$$\mathbf{x}_{i,t+1} = \sum\nolimits_{j\in\mathcal{N}_i} [\mathbf{W}]_{ij}\mathbf{x}_{j,t} - \eta \mathbf{y}_{i,t}, \tag{8}$$

where the constant η is the step-size. We can see from the above three steps that our proposed algorithm is a *triple hybrid* scheme that integrates the SARAH estimator, the gradient tracking, and the decentralized stochastic gradient descent techniques. We formally state our algorithm in Algorithm 1.

Algorithm 1: Triple Hybrid Decentralized Stochastic Gradient Descent Algorithm (TH-DSGD).

Initialization:

1. Choose $T > 0$ and let $t = 1$. Set initial value $\mathbf{x}_{i,0} = \mathbf{x}^0$, $\forall i$, and compute $\mathbf{v}_{i,0} = \frac{1}{|\mathcal{S}_{i,0}|}\sum_{j\in\mathcal{S}_{i,0}} \nabla f_i(\mathbf{x}_{i,0};\zeta_{i,j})$ based on a mini-batch of samples $\mathcal{S}_{i,0}$ and set $\mathbf{y}_{i,0} = \mathbf{v}_{i,0}$, $\forall i$.

Main Loop:

2. In the t-th iteration, each node sends $\mathbf{x}_{i,t-1}$ and $\mathbf{y}_{i,t-1}$ to its neighbors. Meanwhile, upon the reception of all neighbors' information, each node performs the following:
   a). Update the paramter with the neighboring copies and global gradient estimator:
   $$\mathbf{x}_{i,t} = \sum\nolimits_{j\in\mathcal{N}_i} [\mathbf{W}]_{ij}\mathbf{x}_{j,t-1} - \eta \mathbf{y}_{i,t-1}; \tag{9}$$
   b). Sample data $\zeta_{i,t}$ and update local gradient estimator:
   $$\mathbf{v}_{i,t} = \beta\mathbf{v}_{i,t-1} + \nabla f_i(\mathbf{x}_{i,t};\zeta_{i,t}) - \beta\nabla f_i(\mathbf{x}_{i,t-1};\zeta_{i,t}); \tag{10}$$
   c). Track the global gradient estimator:
   $$\mathbf{y}_{i,t} = \sum\nolimits_{j\in\mathcal{N}_i} [\mathbf{W}]_{ij}\mathbf{y}_{i,t-1} + \mathbf{v}_{i,t} - \mathbf{v}_{i,t-1}; \tag{11}$$
3. Stop if $t > T$; otherwise, let $t \leftarrow t+1$ and go to Step 2.

B. Main Theoretical Results

In this section, we will establish the convergence properties of the proposed TH-DSGD algorithm. We first state several needed assumptions as follows:

Assumption 1. *The objective function $f(x) = \frac{1}{m}\sum_{i=1}^m f_i(\mathbf{x})$ with $f_i(\mathbf{x}) = \mathbb{E}_{\zeta\sim\mathcal{D}_i} f_i(\mathbf{x};\zeta)$ satisfies the following conditions:*

(a) (Boundedness from Below) There exists a finite lower bound $f^* = \inf_{\mathbf{x}} f(x) > -\infty$;

(b) (L-Average Smoothness) The function $f_i(\cdot;\zeta_i)$ is L-average smooth on $\mathbb{R}^p$, i.e. there exists a constant $L > 0$, such that $\mathbb{E}_{\zeta\sim\mathcal{D}_i}[\|\nabla f_i(\mathbf{x};\zeta) - \nabla f_i(\mathbf{y};\zeta)\|^2] \le L^2\|\mathbf{x}-\mathbf{y}\|^2$, $\forall \mathbf{x},\mathbf{y}\in\mathbb{R}^p, \forall i \in [m]$;[1]

(c) (Bounded Variance) There exists a constant $\sigma > 0$ such that $\mathbb{E}_{\zeta\sim\mathcal{D}_i}[\|\nabla f_i(\mathbf{x};\zeta) - \nabla f_i(\mathbf{x})\|^2] \le \sigma^2$, $\forall \mathbf{x}\in\mathbb{R}^p, i \in [m]$.

[1]For any positive integer m, we define $[m] := \{1,\dots,m\}$.

In Assumption 1, (a) and (c) are standard in convergence analysis of stochastic gradient-type algorithms in non-convex optimization; (b) is an *expected* Lipschitz smoothness condition over the data distribution, which implies the conventional global Lipschitz smoothness [21] by the Jensen's inequality. Note that (b) is weaker than the individual Lipschitz smoothness in [28], [30], [20]: $\|\nabla f_i(\mathbf{x};\zeta) - \nabla f_i(\mathbf{y};\zeta)\| \le L\|\mathbf{x}-\mathbf{y}\|$, $\forall \mathbf{x},\mathbf{y}\in\mathbb{R}^p$, $\forall i \in [m]$, $\forall \zeta \sim \mathcal{D}_i$. To see this, note that if there exists an outlier sample ζ, then the individual objective could have a large Lipschitz smoothness parameter, while the average Lipschitz smoothness constant could still be small.

For notational convenience in our subsequent convergence analysis, we can compactly rewrite the updates (9)–(11) in Algorithm 1 in a vector form. Towards this end, we define $\widetilde{\mathbf{W}} = \mathbf{W}\otimes\mathbf{I}_m$, $\mathbf{g}_{i,t} = \nabla f_i(\mathbf{x}_{i,t})$, $\mathbf{u}_{i,t} = \nabla f_i(\mathbf{x}_{i,t};\zeta_{i,t})$, $\mathbf{w}_{i,t} = \nabla f_i(\mathbf{x}_{i,t};\zeta_{i,t}) - \nabla f_i(\mathbf{x}_{i,t-1};\zeta_{i,t})$ and $\mathbf{a}_t = [\mathbf{a}_{1,t}^\top,\cdots,\mathbf{a}_{m,t}^\top]^\top$ and $\bar{\mathbf{a}}_t = \frac{1}{m}\sum_{i=1}^m \mathbf{a}_{i,t}$, for $\mathbf{a} \in \{\mathbf{x},\mathbf{u},\mathbf{w},\mathbf{v},\mathbf{g}\}$. Then, the updates in (9)–(11) can be re-written in the following matrix and vector form for $t \in [T]$:

$$\mathbf{x}_t = \widetilde{\mathbf{W}}\mathbf{x}_{t-1} - \eta\mathbf{y}_{t-1}, \tag{12}$$
$$\mathbf{v}_t = \beta\mathbf{v}_{t-1} + \beta\mathbf{w}_t + (1-\beta)\mathbf{u}_t, \tag{13}$$
$$\mathbf{y}_t = \widetilde{\mathbf{W}}\mathbf{y}_{t-1} + \mathbf{v}_t - \mathbf{v}_{t-1},. \tag{14}$$

Further, since $\mathbf{1}^\top\mathbf{W} = \mathbf{1}^\top$, we can simplify the above as:

$$\bar{\mathbf{x}}_t = \bar{\mathbf{x}}_{t-1} - \eta\bar{\mathbf{y}}_{t-1}, \tag{15}$$
$$\bar{\mathbf{v}}_t = \beta\bar{\mathbf{v}}_{t-1} + \beta\bar{\mathbf{w}}_t + (1-\beta)\bar{\mathbf{u}}_t, \tag{16}$$
$$\bar{\mathbf{y}}_t = \bar{\mathbf{y}}_{t-1} + \bar{\mathbf{v}}_t - \bar{\mathbf{v}}_{t-1}. \tag{17}$$

It then follows from (17) that $\bar{\mathbf{y}}_t = \bar{\mathbf{v}}_t$ since $\bar{\mathbf{y}}_0 = \bar{\mathbf{v}}_0$.

With the above compact form, we are now in a position to present the main convergence result for Algorithm 1 as follows:

Theorem 1 (Convergence)**.** *Under Assumption 1, for any constant c_0 satisfying $c_0 > 1/(1-\lambda^2)$, if the parameters η and β in Algorithm 1 and c_1 satisfy:*

$$\eta \le \min\left\{\frac{2L}{2L^2+1}, \frac{1}{9L}, k_1\right\},$$
$$\beta = 1 - 80L^2\eta^2, c_1 = \frac{c_0-1}{c_0\lambda^2} - 1,$$

where k_1 is a constant defined as:

$$k_1 := \big(1-(1+c_1)\lambda^2\big)/\Big(\frac{1}{2c_0} + \frac{(c_0-1)}{c_0 - 1 - c_0\lambda^2}\Big), \tag{18}$$

then we have the following convergence result for Algorithm 1:

$$\min_{t\in[T]}\left\{\mathbb{E}[\|\nabla f(\bar{\mathbf{x}}_t)\|^2] + \frac{1}{m}\|\mathbf{x}_t - \mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2\right\}$$
$$\le \frac{2(f(\bar{\mathbf{x}}_0) - \mathbb{E}[f(\bar{\mathbf{x}}_{T+1})])}{\eta(T+1)} + \frac{2c_0\mathbb{E}[\|\mathbf{v}_0 - \mathbf{1}\otimes\bar{\mathbf{v}}_0\|^2]}{m\eta(T+1)}$$
$$+ \frac{\epsilon_0^2}{20L^2m\eta^2(T+1)} + 800L^2\eta^2\sigma^2, \tag{19}$$

where $\epsilon_0^2 := \mathbb{E}[\|\bar{\mathbf{v}}_0 - \bar{\mathbf{g}}_0\|^2]$ is the average square error of the initial gradient estimator $\bar{\mathbf{v}}_0$.

Several important remarks for Theorem 1 are in order. First, in Theorem 1, c_0 and c_1 are two constants depending on the *network topology*, which in turn will affect the step-size η and convergence: In a sparse network (i.e., λ is close to 1), then c_0 is large and c_1 is near zero, which leads to a small k_1 and in turn implies a smaller step-size η and slower convergence. Second, for the convergence error in the RHS of (19), the first term is dependent on the initialization; the second term is the consensus error of $\mathbf{v}_0$, which is from our potential function defined in (27); the third term is the error of the initial gradient estimation, which will decrease as more samples are adopted in the initial step; and the last term is from the noise of the stochastic gradient after the initial step.

Based on the convergence result in Theorem 1, we immediately have the following key results on the sample and communication complexities for Algorithm 1.

Corollary 2 (Sample & Communication Complexities)**.** *Under the conditions in Theorem 1, if $\eta = O(1/\sqrt[3]{T+1})$, then*

$$\min_{t\in[T]} \Big\{\mathbb{E}[\|\nabla f(\bar{\mathbf{x}}_t)\|^2] + \frac{1}{m}\|\mathbf{x}_t - \mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2\Big\} = O\Big(\frac{1}{(T+1)^{2/3}} + \frac{\epsilon_0^2}{m(T+1)^{1/3}}\Big). \quad (20)$$

Therefore, if the initial batch size $|\mathcal{S}_{i,0}| = O(\epsilon^{-1})$, $\forall i$, and the maximum number of iterations $T = O(\epsilon^{-3})$, then the TH-DSGD algorithm achieves an ϵ^2-stationary solution. Hence, the TH-DSGD algorithm achieves $O(\epsilon^{-3})$ communication complexity and $O(m\epsilon^{-1} + m\epsilon^{-3}) = O(m\epsilon^{-3})$ sample complexity.

It is worth pointing out that, in Corollary 2, the step-size η is on the order of $O(T^{-1/3})$, which is larger than the $O(T^{-1/2})$ step-size for the classical SGD algorithms. With this larger step-size and the stated initial batch size, the convergence rate is $O(T^{-2/3})$, which is faster than the $O(T^{-1/2})$ for SGD.

C. Proof of the Main Theorem

For better readability, in this section, we organize the proof of Theorem 1 into several key lemmas. Due to the space limitation, we could only provide proof sketches for these lemmas and omit the detailed proofs in this paper.

Our first step to prove Theorem 1 is to bound the error of the gradient estimator $\sum_{t=0}^{T}\mathbb{E}[\|\mathbf{v}_t - \mathbf{g}_t\|^2]$, which is stated in the following lemma:

Lemma 1 (Error of Gradient Estimator)**.** *Under Assumption 1 and with $\mathbf{v}_t$ being defined as in (13), we have the following upper bound on $\mathbb{E}[\|\mathbf{v}_t - \mathbf{g}_t\|^2]$:*

$$\sum_{t=0}^{T}\mathbb{E}[\|\mathbf{v}_t - \mathbf{g}_t\|^2] \le \frac{\epsilon_0^2}{1-\beta^2} + \frac{2L^2}{1-\beta^2}\times \sum_{t=0}^{T}\mathbb{E}[\|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2] + \frac{2m\sigma^2(1-\beta)T}{1+\beta}, \quad (21)$$

where $\epsilon_0^2 = \mathbb{E}[\|\mathbf{v}_0 - \mathbf{g}_0\|^2]$ is the average square error of the initial gradient estimator $\mathbf{v}_0$.

Proof Sketch. Since $\mathbf{v}_t = \beta\mathbf{v}_{t-1} + \beta\mathbf{w}_t + (1-\beta)\mathbf{u}_t$, we have:

$$\|\mathbf{v}_t - \mathbf{g}_t\|^2 = \beta^2\|\mathbf{v}_{t-1} - \mathbf{g}_{t-1}\|^2 + \|\beta(\mathbf{w}_t - \mathbf{g}_t + \mathbf{g}_{t-1}) + (1-\beta)(\mathbf{u}_t - \mathbf{g}_t)\|^2 + 2\langle \mathbf{v}_{t-1} - \mathbf{g}_{t-1}, \beta(\mathbf{w}_t - \mathbf{g}_t + \mathbf{g}_{t-1}) + (1-\beta)(\mathbf{u}_t - \mathbf{g}_t)\rangle.$$

Note that $\mathbb{E}_{\zeta_t}[\mathbf{w}_t] = \mathbf{g}_t - \mathbf{g}_{t-1}$ and $\mathbb{E}_{\zeta_t}[\mathbf{u}_t] = \mathbf{g}_t$. Upon taking expectations, we can obtain:

$$\mathbb{E}[\|\mathbf{v}_t - \mathbf{g}_t\|^2] \le \beta^2\mathbb{E}[\|\mathbf{v}_{t-1} - \mathbf{g}_{t-1}\|^2] + 2\beta^2L^2\mathbb{E}[\|\mathbf{x}_t - \mathbf{x}_{t-1}\|^2] + 2m(1-\beta)^2\sigma^2.$$

Similar to [16, Lemma 2.2], by induction, we have:

$$\mathbb{E}[\|\mathbf{v}_t - \mathbf{g}_t\|^2] \le \beta^{2t}\mathbb{E}[\|\mathbf{v}_0 - \mathbf{g}_0\|^2] + 2L^2\sum_{i=0}^{t-1}\beta^{2(t-i)}\mathbb{E}[\|\mathbf{x}_{i+1} - \mathbf{x}_i\|^2] + \frac{2m\sigma^2(1-\beta)}{1+\beta}.$$

Telescoping this inequality from $t = 0$ to T and after some rearrangements, we arrive at the result stated in Lemma 1. □

We can see from (21) that the gradient estimator upper bound depends on the initial gradient estimation. Next, according to the algorithm updates in (12)–(14), we show the descent property of our TH-DSGD algorithm in the following lemma:

Lemma 2 (Descent Property)**.** *Under Assumption 1, we have the following relation for any given $T > 0$:*

$$\mathbb{E}[f(\bar{\mathbf{x}}_{T+1})] - f(\bar{\mathbf{x}}_0) \le -\frac{\eta}{2}\sum_{t=0}^{T}\mathbb{E}[\|\nabla f(\bar{\mathbf{x}}_t)\|^2] - (\frac{\eta}{2} - \frac{L\eta^2}{2})\sum_{t=0}^{T}\mathbb{E}[\|\bar{\mathbf{v}}_t\|^2] + \frac{\eta}{m}\sum_{t=0}^{T}\mathbb{E}[\|\mathbf{v}_t - \mathbf{g}_t\|^2] + \frac{L^2\eta}{m}\sum_{t=0}^{T}\mathbb{E}[\|\mathbf{x}_t - \mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2]. \quad (22)$$

Proof Sketech. From the L-smoothness of f and $\bar{\mathbf{x}}_t = \bar{\mathbf{x}}_{t-1} - \eta\bar{\mathbf{y}}_{t-1} = \bar{\mathbf{x}}_{t-1} - \eta\bar{\mathbf{v}}_{t-1}$, we can show that:

$$f(\bar{\mathbf{x}}_{t+1}) \le f(\bar{\mathbf{x}}_t) - \frac{\eta}{2}\|\nabla f(\bar{\mathbf{x}}_t)\|^2 - (\frac{\eta}{2} - \frac{L\eta^2}{2})\|\bar{\mathbf{v}}_t\|^2 + \frac{\eta}{m}\|\mathbf{v}_t - \mathbf{g}_t\|^2 + \frac{L^2\eta}{m}\|\mathbf{x}_t - \mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2.$$

Taking the full expectation on the above inequality and telescoping from $t = 0$ to T yields the result stated in Lemma 2 and the proof is complete. □

Note that the right-hand-size (RHS) of the inequality in (22) contains the consensus error of local parameters $\sum_{t=0}^{T}\mathbb{E}[\|\mathbf{x}_t - \mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2]$. This additional consensus error makes the algorithm harder to converge compared to centralized algorithms, and is the additional price one needs to pay for performing decentralized optimization over a network.

Next, we establish the contractions of the iterates in the following lemma, which is important in analyzing the performance of the decentralized gradient tracking component in our proposed TH-DSGD algorithm.

Lemma 3 (Iterates Contraction)**.** *The following contraction properties of the iterates hold for Algorithm 1:*

$$\|\mathbf{x}_t - \mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2 \le (1+c_1)\lambda^2\|\mathbf{x}_{t-1}-\mathbf{1}\otimes\bar{\mathbf{x}}_{t-1}\|^2 + \left(1+\frac{1}{c_1}\right)\eta^2\|\mathbf{y}_{t-1}-\mathbf{1}\otimes\bar{\mathbf{y}}_{t-1}\|^2, \quad (23)$$

$$\|\mathbf{y}_t - \mathbf{1}\otimes\bar{\mathbf{y}}_t\|^2 \le (1+c_1)\lambda^2\|\mathbf{y}_{t-1}-\mathbf{1}\otimes\bar{\mathbf{y}}_{t-1}\|^2 + \left(1+\frac{1}{c_1}\right)\|\mathbf{v}_t-\mathbf{v}_{t-1}\|^2, \quad (24)$$

where c_1 is a positive constant. Additionally, we have

$$\|\mathbf{x}_t-\mathbf{x}_{t-1}\|^2 \le 8\|(\mathbf{x}_{t-1}-\mathbf{1}\otimes\bar{\mathbf{x}}_{t-1})\|^2 + 4\eta^2\|\mathbf{y}_{t-1}-\mathbf{1}\otimes\bar{\mathbf{y}}_{t-1}\|^2 + 4\eta^2 m\|\bar{\mathbf{v}}_{t-1}\|^2, \quad (25)$$

$$\|\mathbf{v}_t-\mathbf{v}_{t-1}\|^2 \le 2\|\mathbf{w}_t\|^2 + 2(1-\beta)^2\|\nabla\mathbf{f}(\mathbf{x}_{t-1};\zeta_t)-\mathbf{v}_{t-1}\|^2, \quad (26)$$

where the vector $\nabla\mathbf{f}(\mathbf{x}_{t-1};\zeta_t)$ is defined as: $\nabla\mathbf{f}(\mathbf{x}_{t-1};\zeta_t) = [\nabla f_1(\mathbf{x}_{1,t-1};\zeta_{1,t})^\top,\cdots,\nabla f_m(\mathbf{x}_{m,t-1};\zeta_{m,t})^\top]^\top$.

Proof Sketch. First, for variable $\mathbf{x}_t$, we note the following contraction $\|\widetilde{\mathbf{W}}\mathbf{x}_t - \mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2 = \|\widetilde{\mathbf{W}}(\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t)\|^2 \le \lambda^2\|\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2$, Starting from $\bar{\mathbf{x}}_t = \bar{\mathbf{x}}_{t-1} - \eta\bar{\mathbf{y}}_{t-1}$, we can show Eq. (23) (Eq. (24) can also be proved similarly). For $\|\mathbf{v}_t-\mathbf{v}_{t-1}\|^2$, after derivations, we can show that $\|\mathbf{v}_t-\mathbf{v}_{t-1}\|^2 \le 2\|\mathbf{w}_t\|^2 + 2(1-\beta)^2\|\widetilde{\mathbf{u}}_t-\mathbf{v}_{t-1}\|^2$, where $\widetilde{\mathbf{u}}_t = [\widetilde{\mathbf{u}}_{1,t}^\top,\cdots,\widetilde{\mathbf{u}}_{m,t}^\top]^\top$ and $\widetilde{\mathbf{u}}_{i,t} = \nabla f_i(\mathbf{x}_{i,t-1};\zeta_{i,t})$. Lastly, according to the updating equation (12), we have that

$$\|\mathbf{x}_t-\mathbf{x}_{t-1}\|^2 = \|\widetilde{\mathbf{W}}\mathbf{x}_{t-1}-\eta\mathbf{y}_{t-1}-\mathbf{x}_{t-1}\|^2 = 2\|(\widetilde{\mathbf{W}}-\mathbf{I})(\mathbf{x}_{t-1}-\mathbf{1}\otimes\bar{\mathbf{x}}_{t-1})\|^2 + 2\eta^2\|\mathbf{y}_{t-1}\|^2.$$

Following similar procedure, we have a similar expression for $\|\mathbf{v}_t-\mathbf{v}_{t-1}\|^2$. Then, Eqs. (25)–(26) in Lemma 3 follow from noting that $\|\widetilde{\mathbf{W}}-\mathbf{I}\|\le 2$ and the proof is complete. □

Lastly, we define a potential function in Eq. (27) and prove the convergence bound based on the potential function.

Lemma 4. *(Convergence on Potential Function) Define the following potential function:*

$$H_t = \mathbb{E}\left[f(\bar{\mathbf{x}}_t) + \frac{c_0}{m\eta}\|\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2 + \frac{c_0}{m}\|\mathbf{v}_t-\mathbf{1}\otimes\bar{\mathbf{v}}_t\|^2\right], \quad (27)$$

where $c_0 \ge 1$ is a constant. If we set $4(1-\beta)(1+\frac{1}{c_1})c_0 \le \eta$ and $\beta\in[0,1)$, then under Assumption 1, it holds that:

$$\frac{\eta}{2}\sum_{t=0}^{T}\mathbb{E}[\|\nabla f(\bar{\mathbf{x}}_t)\|^2] \le H_0 - H_{T+1} + \frac{2\eta\epsilon_0^2}{m(1-\beta^2)} + 5\eta\sigma^2(1-\beta)T - \frac{c_0C_1}{m\eta}\sum_{t=0}^{T}\mathbb{E}[\|\mathbf{x}_t-\mathbf{1}\bar{\mathbf{x}}_t\|^2] - \frac{c_0C_2}{m}\sum_{t=0}^{T}\mathbb{E}[\|\mathbf{y}_t-\mathbf{1}\otimes\bar{\mathbf{y}}_t\|^2] - \frac{C_3\eta}{2}\sum_{t=0}^{T}\mathbb{E}[\|\bar{\mathbf{v}}_t\|^2], \quad (28)$$

where $\epsilon_0^2 = \mathbb{E}[\|\bar{\mathbf{v}}_0-\bar{\mathbf{g}}_0\|^2]$ is the error bound on the initial estimator $\bar{\mathbf{v}}_0$, and constants C_1, C_2, and C_3 are defined as:

$$C_1 := 1-(1+c_1)\lambda^2 - \frac{L^2\eta^2}{c_0} - \frac{40L^2\eta^2}{c_0(1-\beta)},$$
$$C_2 := 1-(1+c_1)\lambda^2 - (1+\frac{1}{c_1})\eta - \frac{20\eta^3L^2}{c_0(1-\beta)},$$
$$C_3 := 1-L\eta - \frac{40\eta^2L^2}{(1-\beta)}.$$

Proof Sketch. From Lemma 3, we have that $\|\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2 - \|\mathbf{x}_{t-1}-\mathbf{1}\otimes\bar{\mathbf{x}}_{t-1}\|^2 \le -(1-(1+c_1)\lambda^2)\|\mathbf{x}_{t-1}-\mathbf{1}\otimes\bar{\mathbf{x}}_{t-1}\|^2 + (1+\frac{1}{c_1})\eta^2\|\mathbf{y}_{t-1}-\mathbf{1}\otimes\bar{\mathbf{y}}_{t-1}\|^2$ and $\|\mathbf{y}_t-\mathbf{1}\otimes\bar{\mathbf{y}}_t\|^2 - \|\mathbf{y}_{t-1}-\mathbf{1}\otimes\bar{\mathbf{y}}_{t-1}\|^2 \le -(1-(1+c_1)\lambda^2)\|\mathbf{y}_{t-1}-\mathbf{1}\otimes\bar{\mathbf{y}}_{t-1}\|^2 + (1+\frac{1}{c_1})\|\mathbf{v}_t-\mathbf{v}_{t-1}\|^2$. Letting $P_i = \mathbb{E}[\|\mathbf{x}_i-\mathbf{1}\otimes\bar{\mathbf{x}}_i\|^2] + \eta\mathbb{E}[\|\mathbf{v}_i-\mathbf{1}\otimes\bar{\mathbf{v}}_i\|^2]$, combining the above two inequalities, and telescoping from $t=1$ to $T+1$ yield:

$$P_{T+1} - P_0 \le -\big(1-(1+c_1)\lambda^2\big)\sum_{t=0}^{T}\mathbb{E}[\|\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2] - \big(1-(1+c_1)\lambda^2-(1+\frac{1}{c_1})\eta\big)\eta\sum_{t=0}^{T}\mathbb{E}[\|\mathbf{y}_t-\mathbf{1}\otimes\bar{\mathbf{y}}_t\|^2] + (1+\frac{1}{c_1})\eta\sum_{t=1}^{T}\mathbb{E}[\|\mathbf{v}_t-\mathbf{v}_{t-1}\|^2].$$

Combining this inequality with Lemma 2 and after algebraic manipulations, we arrive at the stated result in Lemma 4. □

Note that by properly selecting the parameters as stated in Theorem 1, the constants C_1, C_2 and C_3 can be made non-negative and their associated terms in the RHS of (28) can be dropped. Finally, by combining Lemmas 3 and 4 and after some algebraic simplifications, we arrive at the desired result as stated in Theorem 1.

IV. Numerical Results

In this section, we present experimental results to evaluate the performance of our proposed TH-DSGD algorithm using several non-convex machine learning problems. In particular, we compare TH-DSGD with three state-of-the-art algorithms:

- *Decentralized Stochastic Gradient Descent (DSGD)*[4], [31], [17]: Each node performs the following update: $\mathbf{x}_{i,t+1} = \sum_{j\in\mathcal{N}_i}[\mathbf{W}]_{ij}\mathbf{x}_{j,t} - \eta\nabla f_i(\mathbf{x}_{i,t};\zeta_{i,t})$, where the stochastic gradient $\nabla f_i(\mathbf{x}_{i,t};\zeta_{i,t})$ depends on random sample $\zeta_{i,t}$. Upon finishing the update, each node exchanges the local variable $\mathbf{x}_{i,t}$ with its neighbors.
- *Gradient-Tracking-Based Non-Convex Stochastic Decentralized (GNSD) Algorithm* [18]: Each node keeps two variables $\mathbf{x}_{i,t}$ and $\mathbf{y}_{i,t}$. The local variable $\mathbf{x}_{i,t}$ is updated as $\mathbf{x}_{i,t+1} = \sum_{j\in\mathcal{N}_i}[\mathbf{W}]_{ij}\mathbf{x}_{j,t} - \eta\mathbf{y}_{i,t}$ and the tracked gradient $\mathbf{y}_{i,t}$ is updated as $\mathbf{y}_{i,t+1} = \sum_{j\in\mathcal{N}_i}[\mathbf{W}]_{ij}\mathbf{y}_{j,t} + \nabla f_i(\mathbf{x}_{i,t+1};\zeta_{i,t+1}) - \nabla f_i(\mathbf{x}_{i,t};\zeta_{i,t})$.
- *Decentralized Gradient Estimation and Tracking (D-GET)* [20]: Each node keeps three variables $\mathbf{x}_{i,t}$, $\mathbf{y}_{i,t}$ and $\mathbf{v}_{i,t}$. The local variable $\mathbf{x}_{i,t}$ and the tracked gradient $\mathbf{y}_{i,t}$ are

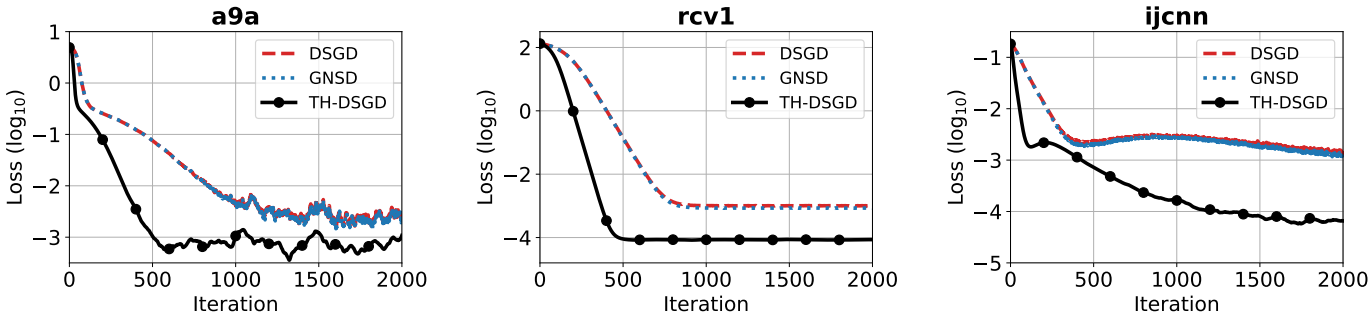


Fig. 1: Results of combined loss with models trained by three different algorithms. The curves are averaged over 10 trials.

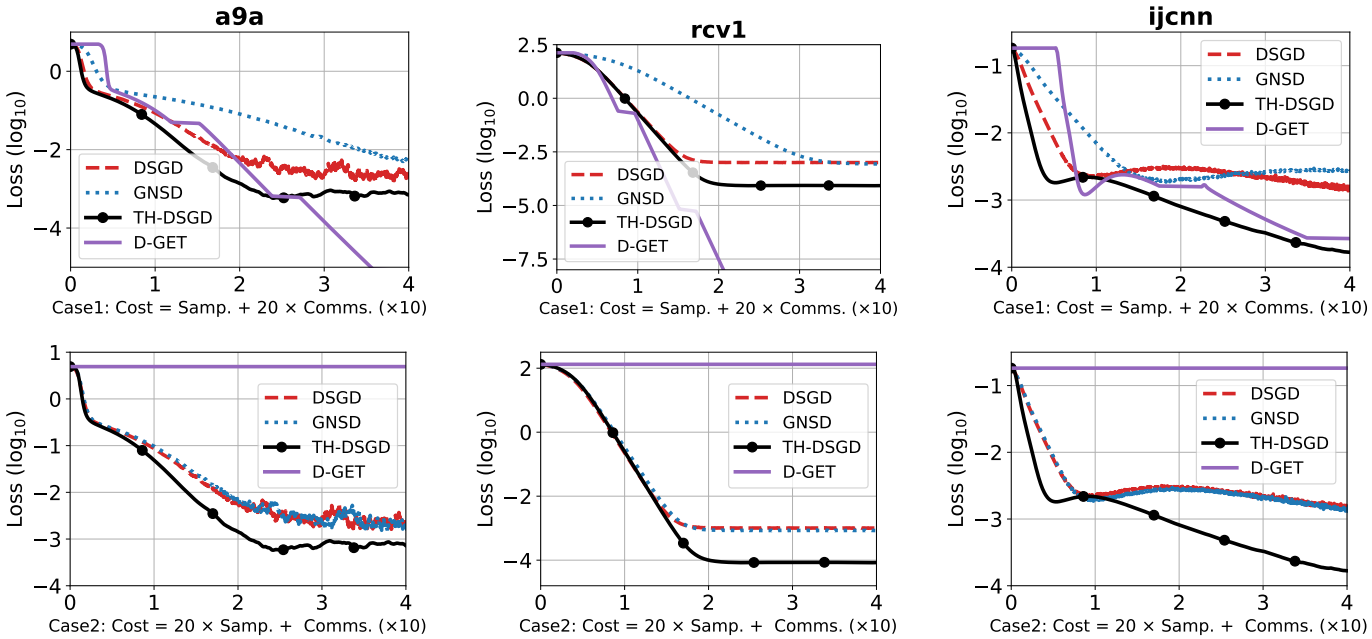


Fig. 2: Results of two different costs vs combined loss. The curves are averaged over 10 trials.

updated as in GNSD, and $\mathbf{v}_{i,t}$ is the local SpiderBoost estimator: $\mathbf{v}_{i,t} = \mathbf{v}_{i,t-1} + \nabla f(\mathbf{x}_{i,t}|\zeta_{i,t}) - \nabla f(\mathbf{x}_{i,t-1}|\zeta_{i,t})$, if $\text{mod}(t,q) \neq 0$; otherwise, we use the local full gradient.

Network Model: The graph $\mathcal{G}$ of the communication network is generated by the Erdös-Rènyi graph with different edge connection probability p_c and node number m. The consensus matrix is chosen as $\mathbf{W} = \mathbf{I} - \frac{2}{3\lambda_{\max}(\mathbf{L})}\mathbf{L}$, where $\mathbf{L}$ is the Laplacian matrix of $\mathcal{G}$ and $\lambda_{\max}(\mathbf{L})$ denotes the largest eigenvalue of $\mathbf{L}$.

Learning Model: We consider the binary logistic regression with the following non-convex regularizer, which is a widely used learning model for non-convex empirical risk minimization problems in the literature (see, e.g., [44], [30], [16]):

$$\min_{\mathbf{x}\in\mathbb{R}^d} -\frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}[y_{ij}\log\big(\frac{1}{1+e^{-\mathbf{x}^\top\zeta_{ij}}}\big) + (1-y_{ij})\log\big(\frac{e^{-\mathbf{x}^\top\zeta_{ij}}}{1+e^{-\mathbf{x}^\top\zeta_{ij}}}\big)] + \rho\sum_{i=1}^{d}\frac{\mathbf{x}_i^2}{1+\mathbf{x}_i^2}, \quad (29)$$

where the label $y_{ij} \in \{0,1\}$, the feature $\zeta_{ij} \in \mathbb{R}^d$ and we set $\rho = 0.1$ in our experiments.

Datasets: We consider three commonly used binary classification datasets from LibSVM [19]: $a9a$ dataset, $rcv1.binary$ dataset and $ijcnn1$ dataset. More specifically, $a9a$ dataset has 32561 samples and 123 features, $rcv1.binary$ dataset has 20242 samples and 47236 features, and $ijcnn1$ dataset has 49990 samples and 22 features. For each dataset, we evenly divide the the samples into m sub-datasets corresponding to the m nodes in the network.

Parameters: We set the total number of iterations to 2000 and run 10 trials for each algorithm. The mini-batch size in each iteration is one for DSGD, GNSD and our proposed TH-DSGD. In addition, the initial mini-batch size for TH-DSGD is $12 \approx \sqrt[3]{2000}$. For D-GET, following the suggestions in [20], we choose the inner loop iteration and the mini-batch size as $\lfloor\sqrt{n}\rfloor$, and all samples are used in the outer loop iterations to compute full gradients. For all four algorithms, the step-size η is tuned by searching over the grid $\{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0\}$ and picking the best one. We set $\beta = 0.99$ for TH-DSGD.

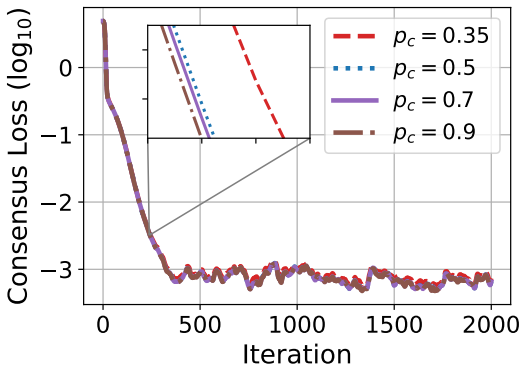
Results: We first compare the convergence rates among three *single-loop* algorithms:[2] DSGD, GNSD and TH-DSGD (recall that the mini-batch sizes for all algorithms are set to one for fair comparisons). The network has 10 nodes and the edge connectivity probability p_c is 0.5. We adopt the combined loss defined in left-hand-side (LHS) of Eq. (3) as the criterion. The results are shown in Figure 1. We can see that TH-DSGD convergences faster and achieve a smaller combined loss than the other two methods. For example, for the $a9a$ dataset, TH-DSGD converges in approximately 500 iterations and reaches the error neighborhood of size 10^{-3}, while the DSGD and GNSD algorithms took almost 1000 iterations and their loss values remain approximately at the level of $10^{-2.5}$.

For convenience in evaluating and comparing both sample and communication complexities of the algorithms, we define the following total weighted cost:

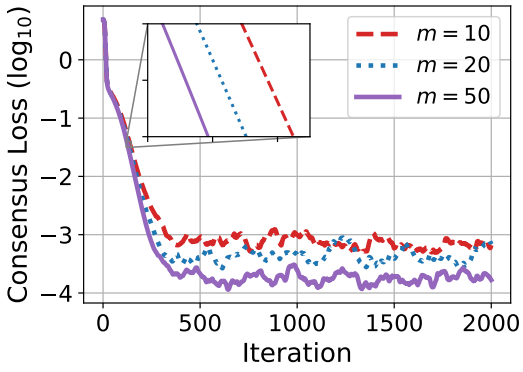
$$\text{cost} = c_s \times \text{sample complexity} + c_c \times \text{communication complexity}, \quad (30)$$

where c_s and c_c represent the costs of per-unit sample complexity and per-unit communication complexity, respectively. In our comparisons, we consider two cases: *Case 1:* $c_s = 1$, $c_c = 20$; and *Case 2:* $c_s = 20$, $c_c = 1$. Case 1 represents the networks have high communication cost, while Case 2 represents the each node has high computation cost. All four algorithms are compared and the results are shown in Figure 2. In Case 1 where communication is more costly compared to computation, our TH-DSGD algorithm have a better performance for the most part, while D-GET reaches a better learning loss result as the cost increases. This is because D-GET has $O(\epsilon^{-2})$ communication complexity, which is better than the $O(\epsilon^{-3})$ communication complexity of TH-DSGD. Thus, D-GET tends to outperform in scenarios with high communication costs. Note also that, in this case, D-GET can reach a more accurate learning result: for example, the error is smaller than 10^{-7} for $rcv1$ dataset. This is not surprising because D-GET periodically obtains a full gradient, which is affordable in Case 1. However, for Case 2 where computation is more expensive, our TH-DSGD outperforms all other three algorithms. We can see that both DSGD and GNSD have similar performances but cannot reach the same learning accuracy as the proposed TH-DSGD algorithm. Note that, in Case 2, the cost for computing the full gradient for D-GET is so expensive that it cannot even finish the first step within the given cost constraint. To summarize, our TH-DSGD outperforms the other algorithms especially in the cases with the large datasets and low computational speed.

Lastly, we examine the impacts of the network topology. We first set the number of nodes in the network to $m = 10$ and vary the edge connectivity probability p_c by choosing from the discrete set $\{0.35, 0.5, 0.7, 0.9\}$. Note a smaller p_c implies a sparser network. Theoretically, it should be harder for TH-DSGD to converge (recall the discussions on Theorem 1 in Section III-B). We fix the step-size as 0.1. The results are shown in Figure 3 (a). Under different p_c values, TH-DSGD has very similar performance in terms of the convergence speed and accuracy. With a larger p_c (a denser network), the speed is slightly faster (see the zoom-in view in Figure 3 (a)), which confirms our theoretical analysis and shows that TH-DSGD is *stable* with respect to network sparsity. Next, we fix $p_c = 0.5$ and vary the network size m from 10 to 20, and to 50. We illustrate the results in Figure 3 (b). We can see that with more nodes in the network, the TH-DSGD algorithm converges faster and reaches a better accuracy, which suggests that our proposed algorithm tends to perform better in the large network regime.



(a)



(b)

Fig. 3: Results of TH-DSGD convergence under different network topology on $a9a$ dataset.

V. Conclusion

In this paper, we proposed a triple hybrid decentralized stochastic gradient descent (TH-DSGD) algorithm for network consensus optimization. Compared with existing methods, the proposed method improves both the sample and communication complexities by adopting variance reduction and gradient tracking techniques. In particular, to attain an ϵ^2-stationary solution, the total sample complexity of our algorithm is $O(m\epsilon^{-3})$ and the communication complexity is $O(\epsilon^{-3})$, which is better than $O(\epsilon^{-4})$ of the traditional decentralized stochastic gradient algorithms [5], [18]. We conducted extensive experiments on a variety of real world datasets to verify our theoretical findings. It is shown that our algorithm outperforms the existing methods when training on the large-scale datasets.

[2]Note that the D-GET algorithm has a double-loop structure and its convergence rate is not directly comparable.

REFERENCES

[1] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild: A lock-free approach to parallelizing stochastic gradient descent," in *Advances in neural information processing systems*, 2011, pp. 693–701.

[2] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola, "Parallelized stochastic gradient descent," in *Advances in neural information processing systems*, 2010, pp. 2595–2603.

[3] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang *et al.*, "Large scale distributed deep networks," in *Advances in neural information processing systems*, 2012, pp. 1223–1231.

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

[5] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," in *Advances in Neural Information Processing Systems*, 2017, pp. 5330–5340.

[6] J. Chen, Z. J. Towfic, and A. H. Sayed, "Dictionary learning over distributed models," *IEEE Transactions on Signal Processing*, vol. 63, no. 4, pp. 1001–1016, 2014.

[7] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Transactions on Industrial informatics*, vol. 9, no. 1, pp. 427–438, 2012.

[8] W. Ren, R. W. Beard, and E. M. Atkins, "Information consensus in multivehicle cooperative control," *IEEE Control systems magazine*, vol. 27, no. 2, pp. 71–82, 2007.

[9] K. Zhou and S. I. Roumeliotis, "Multirobot active target tracking with combinations of relative observations," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 678–695, 2011.

[10] W. Wang, J. Wang, M. Kolar, and N. Srebro, "Distributed stochastic multi-task learning with graph regularization," *arXiv preprint arXiv:1802.03830*, 2018.

[11] X. Zhang, J. Liu, and Z. Zhu, "Distributed linear model clustering over networks: A tree-based fused-lasso admm approach," *arXiv preprint arXiv:1905.11549*, 2019.

[12] K. Ali and W. Van Stam, "Tivo: making show recommendations using a distributed collaborative filtering architecture," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004, pp. 394–401.

[13] Z. Jiang, K. Mukherjee, and S. Sarkar, "On consensus-disagreement tradeoff in distributed optimization," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 571–576.

[14] J. N. Tsitsiklis, "Problems in decentralized decision making and computation." Massachusetts Inst of Tech Cambridge Lab for Information and Decision Systems, Tech. Rep., 1984.

[15] S. H. Rhee, H.-S. Kim, and S.-W. Sohn, "The effect of decentralized resource allocation in network-centric warfare," in *The International Conference on Information Network 2012*. IEEE, 2012, pp. 478–481.

[16] Q. Tran-Dinh, N. H. Pham, D. T. Phan, and L. M. Nguyen, "Hybrid stochastic gradient descent algorithms for stochastic nonconvex optimization," *arXiv preprint arXiv:1905.05920*, 2019.

[17] Z. Jiang, A. Balu, C. Hegde, and S. Sarkar, "Collaborative deep learning in fixed topology networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 5904–5914.

[18] S. Lu, X. Zhang, H. Sun, and M. Hong, "Gnsd: a gradient-tracking based nonconvex stochastic algorithm for decentralized optimization," in *2019 IEEE Data Science Workshop, DSW 2019*. Institute of Electrical and Electronics Engineers Inc., 2019, pp. 315–321.

[19] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, pp. 1–27, 2011.

[20] H. Sun, S. Lu, and M. Hong, "Improving the sample and communication complexity for decentralized non-convex optimization: A joint gradient estimation and tracking approach," in *International conference on machine learning*, 2020.

[21] S. Ghadimi and G. Lan, "Stochastic first-and zeroth-order methods for nonconvex stochastic programming," *SIAM Journal on Optimization*, vol. 23, no. 4, pp. 2341–2368, 2013.

[22] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *Siam Review*, vol. 60, no. 2, pp. 223–311, 2018.

[23] P. Zhou, X. Yuan, and J. Feng, "New insight into hybrid stochastic gradient descent: Beyond with-replacement sampling and convexity," in *Advances in Neural Information Processing Systems*, 2018, pp. 1234–1243.

[24] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Advances in neural information processing systems*, 2013, pp. 315–323.

[25] S. J. Reddi, A. Hefny, S. Sra, B. Póczos, and A. Smola, "Stochastic variance reduction for nonconvex optimization," in *International conference on machine learning*, 2016, pp. 314–323.

[26] A. Defazio, F. Bach, and S. Lacoste-Julien, "Saga: A fast incremental gradient method with support for non-strongly convex composite objectives," in *Advances in neural information processing systems*, 2014, pp. 1646–1654.

[27] L. Lei, C. Ju, J. Chen, and M. I. Jordan, "Non-convex finite-sum optimization via scsg methods," in *Advances in Neural Information Processing Systems*, 2017, pp. 2348–2358.

[28] C. Fang, C. J. Li, Z. Lin, and T. Zhang, "Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator," in *Advances in Neural Information Processing Systems*, 2018, pp. 689–699.

[29] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takác, "Sarah: A novel method for machine learning problems using stochastic recursive gradient," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2613–2621.

[30] Z. Wang, K. Ji, Y. Zhou, Y. Liang, and V. Tarokh, "Spiderboost and momentum: Faster variance reduction algorithms," in *Advances in Neural Information Processing Systems*, 2019, pp. 2403–2413.

[31] 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.

[32] W. Shi, Q. Ling, G. Wu, and W. Yin, "Extra: An exact first-order algorithm for decentralized consensus optimization," *SIAM Journal on Optimization*, vol. 25, no. 2, pp. 944–966, 2015.

[33] P. Di Lorenzo and G. Scutari, "Next: In-network nonconvex optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 2, pp. 120–136, 2016.

[34] H. Sun and M. Hong, "Distributed non-convex first-order optimization and information processing: Lower complexity bounds and rate optimal algorithms," *IEEE Transactions on Signal processing*, vol. 67, no. 22, pp. 5912–5928, 2019.

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

[36] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro, "A decentralized second-order method with exact linear convergence rate for consensus optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 4, pp. 507–522, 2016.

[37] M. Eisen, A. Mokhtari, and A. Ribeiro, "Decentralized quasi-newton methods," *IEEE Transactions on Signal Processing*, vol. 65, no. 10, pp. 2613–2628, 2017.

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

[39] T.-H. Chang, M. Hong, H.-T. Wai, X. Zhang, and S. Lu, "Distributed learning in the non-convex world: From batch to streaming data, and beyond," *arXiv preprint arXiv:2001.04786*, 2020.

[40] J. Zeng and W. Yin, "On nonconvex decentralized gradient descent," *IEEE Transactions on signal processing*, vol. 66, no. 11, pp. 2834–2848, 2018.

[41] A. Mokhtari and A. Ribeiro, "Dsa: Decentralized double stochastic averaging gradient algorithm," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2165–2199, 2016.

[42] K. Yuan, B. Ying, J. Liu, and A. H. Sayed, "Variance-reduced stochastic learning by networked agents under random reshuffling," *IEEE Transactions on Signal Processing*, vol. 67, no. 2, pp. 351–366, 2018.

[43] R. Xin, U. A. Khan, and S. Kar, "Variance-reduced decentralized stochastic optimization with gradient tracking," *arXiv preprint arXiv:1909.11774*, 2019.

[44] Z. Wang, Y. Zhou, Y. Liang, and G. Lan, "Cubic regularization with momentum for nonconvex optimization," *arXiv preprint arXiv:1810.03763*, 2018.