Improving the Sample and Communication Complexity for Decentralized Non-Convex Optimization: Joint Gradient Estimation and Tracking

Haoran Sun [1] **Songtao Lu** [2] **Mingyi Hong** [1]

Abstract

Many modern large-scale machine learning problems benefit from decentralized and stochastic optimization. Recent works have shown that utilizing both decentralized computing and local stochastic gradient estimates can outperform state-of-the-art centralized algorithms, in applications involving highly non-convex problems, such as training deep neural networks. In this work, we propose a decentralized stochastic algorithm to deal with certain smooth non-convex problems where there are m nodes in the system, and each node has a large number of samples (denoted as n). Differently from the majority of the existing decentralized learning algorithms for either stochastic or finite-sum problems, our focus is given to *both* reducing the total communication rounds among the nodes, while accessing the minimum number of local data samples. In particular, we propose an algorithm named D-GET (decentralized gradient estimation and tracking), which jointly performs decentralized gradient estimation (which estimates the local gradient using a subset of local samples) *and* gradient tracking (which tracks the global full gradient using local estimates). We show that, to achieve certain ϵ stationary solution of the deterministic finite sum problem, the proposed algorithm achieves an $\mathcal{O}(mn^{1/2}\epsilon^{-1})$ sample complexity and an $\mathcal{O}(\epsilon^{-1})$ communication complexity. These bounds significantly improve upon the best existing bounds of $\mathcal{O}(mn\epsilon^{-1})$ and $\mathcal{O}(\epsilon^{-1})$, respectively. Similarly, for online problems, the proposed method achieves an $\mathcal{O}(m\epsilon^{-3/2})$ sample complexity and an $\mathcal{O}(\epsilon^{-1})$ communication complexity.

[1]Department of ECE, University of Minnesota Twin Cities, Minneapolis, MN USA [2]IBM Research AI, IBM Thomas J. Watson Research Center, Yorktown Heights, NY USA. Correspondence to: Haoran Sun <sun00111@umn.edu>, Songtao Lu <songtao@ibm.com>, Mingyi Hong <mhong@umn.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

1. Introduction

Recent advances of decentralized optimization enable us to utilize distributed resources to significantly improve the computation efficiency (Boyd et al., 2011; Lian et al., 2017). Compared to the typical parameter-server type distributed system with a fusion center, decentralized optimization has its unique advantages in preserving data privacy, enhancing network robustness, and improving the computation efficiency (Lian et al., 2017; Nedic & Ozdaglar, 2009; Chen & Sayed, 2012; Yuan et al., 2016). Furthermore, in many emerging applications such as collaborative filtering (Ali & Van Stam, 2004), federated learning (Konečnỳ et al., 2016) and dictionary learning (Chen et al., 2014), the data is naturally collected in a decentralized setting, and it is not possible to transfer the distributed data to a central location. Therefore, decentralized computation has sparked considerable interest in both academia and industry.

Motivated by these facts, in this paper we consider the following decentralized optimization problem,

$$\min_{\mathbf{x}\in\mathbb{R}^{md}} f(\mathbf{x}) = \frac{1}{m}\sum_{i=1}^{m} f^i(\mathbf{x}_i), \qquad (1)$$
$$\text{s.t.} \quad \mathbf{x}_i = \mathbf{x}_k, \quad \forall (i,k) \in \mathcal{E}.$$

where $f^i(\cdot) : \mathbb{R}^d \to \mathbb{R}$ denotes the loss function which is smooth (possibly non-convex), and m is the total number of such functions. We consider the scenario where each node $i \in [m] := \{1, \cdots, m\}$ can only access its local function $f^i(\cdot)$, and can communicate with its neighbors via an undirected and unweighted graph $\mathcal{G} = \{\mathcal{E}, \mathcal{V}\}$. And $\mathbf{x}$ stacks all the variables: $\mathbf{x} := [\mathbf{x}_1; \mathbf{x}_2; \cdots; \mathbf{x}_m] \in \mathbb{R}^{md}$.

In this work, we consider two typical representations of the local cost functions:

1. **Finite-Sum Setting:** Each $f^i(\cdot)$ is defined as the average cost of n local samples, that is:

$$f^i(\cdot) = \frac{1}{n}\sum_{j=1}^{n} f_j^i(\cdot), \forall i \qquad (2)$$

   where n is the total number of local samples at node i, $f_j^i(\cdot)$ denotes the cost for jth data sample at ith node.

2. **Online Setting:** Each $f^i(\cdot)$ is defined as:

$$f^i(\cdot) = \mathbb{E}_{\xi\sim\mathcal{D}_i}[f^i_\xi(\cdot)], \forall i \tag{3}$$

where $\mathcal{D}_i$ denotes the data distribution at node i.

For the above decentralized non-convex problem (1), one essential task is to find an ϵ stationary solution $\mathbf{x}^* := [\mathbf{x}_1^*; \cdots; \mathbf{x}_m^*] \in \mathbb{R}^{md}$ such that the optimality gap h^* satisfies

$$\left\| \frac{1}{m}\sum_{i=1}^m \nabla f^i(\mathbf{x}_i^*) \right\|^2 + \frac{1}{m}\sum_i^m \left\| \mathbf{x}_i^* - \frac{1}{m}\sum_i^m \mathbf{x}_i^* \right\|^2 \leq \epsilon. \tag{4}$$

This solution quality measure encodes both the size of local gradient error for classical centralized non-convex problems and the consensus error for decentralized optimization.

Many modern decentralized methods can be applied to obtain the above mentioned ϵ stationary solution for problem (1). In the finite-sum setting (2), deterministic decentralized methods (Hong et al., 2017; Di Lorenzo & Scutari, 2016; Sun et al., 2019; Sun & Hong, 2019), which process the local dataset in full batches, typically achieve $\mathcal{O}(\epsilon^{-1})$ communication complexity (i.e., $\mathcal{O}(\epsilon^{-1})$ rounds of message exchanges are required to obtain ϵ stationary solution), and $\mathcal{O}(mn\epsilon^{-1})$ sample complexity.[1] Meanwhile, stochastic methods (Lian et al., 2017; Tang et al., 2018; Assran et al., 2019; Lu et al., 2019), which randomly pick subsets of local samples, achieve $\mathcal{O}(m\epsilon^{-2})$ sample and $\mathcal{O}(\epsilon^{-2})$ communication complexity. These complexity bounds indicate that, when the sample size is large (i.e., $\epsilon^{-1} = o(n)$), the stochastic methods are preferred for lower sample complexity, but the deterministic methods still achieve lower communication complexity. On the other hand, in the online setting (3), only stochastic methods can be applied, and those methods again achieve $\mathcal{O}(m\epsilon^{-2})$ sample and $\mathcal{O}(\epsilon^{-2})$ communication complexity (Tang et al., 2018).

1.1. Our Contribution

Compared with the majority of the existing decentralized learning algorithms for either stochastic or deterministic problems, the focus of this work is given to *both* reducing the total communication and sample complexity. Specifically, we propose a decentralized gradient estimation and tracking (D-GET) approach, which uses a subset of samples to estimate the local gradients (by utilizing modern variance reduction techniques (Fang et al., 2018; Nguyen et al., 2017)), while using the differences of past local gradients to track the global gradients (by leveraging the idea of decentralized gradient tracking (Di Lorenzo & Scutari, 2016; Pu & Nedić, 2018)). Remarkably, the proposed approach enjoys a sample complexity of $\mathcal{O}(mn^{1/2}\epsilon^{-1})$ and communication complexity of $\mathcal{O}(\epsilon^{-1})$ for finite sum problem (2), which outperforms all existing decentralized methods. The sample complexity rate is $\sqrt{m}$ worse than the known sample complexity lower bound for centralized problem (Fang et al., 2018), and the communication complexity matches the existing communication lower bound (Sun & Hong, 2019) for decentralized non-convex optimization (in terms of the dependency in ϵ). Furthermore, the proposed approach is also able to achieve $\mathcal{O}(m\epsilon^{-3/2})$ sample complexity and $\mathcal{O}(\epsilon^{-1})$ communication complexity for the online problem (3), reducing the best existing bounds (such as those obtained in (Tang et al., 2018; Lu et al., 2019; Lu & Wu, 2020)) by factors of $\mathcal{O}(\epsilon^{-1/2})$ and $\mathcal{O}(\epsilon^{-1})$, respectively, through a more restrictive mean-squared smoothness assumption (Arjevani et al., 2019). The rate $\mathcal{O}(m\epsilon^{-3/2})$ is m worse than the centralized stochastic lower bound $\mathcal{O}(\epsilon^{-3/2})$ for non-convex problems (Arjevani et al., 2019). We illustrate the main results of this work and compare the gradient and communication cost for state-of-the-art decentralized non-convex optimization algorithms in Table 1.[2] Note that in Table 1, by *constant step-size* we mean that it is not dependent on the target accuracy ϵ, nor the iteration number.

1.2. Related Works

Decentralized Optimization. Decentralized optimization has been extensively studied for convex problems and can be traced back to the 1980s (Bertsekas, 1989). We refer the readers to the recent survey (Nedić et al., 2018) and the references therein for a complete review. When the problem becomes non-convex, many algorithms such as primal-dual based methods (Hong et al., 2016; 2017), gradient tracking based methods (Di Lorenzo & Scutari, 2016; Daneshmand et al., 2016), and non-convex extensions of DGD methods (Zeng & Yin, 2018) have been proposed, where the $\mathcal{O}(\epsilon^{-1})$ iteration and communication complexity have been shown. Note that the above algorithms all require $\mathcal{O}(1)$ full gradient evaluations per iteration, so when directly applied to solve problems where each $f^i(\cdot)$ takes the form in (2), they all require $\mathcal{O}(mn\epsilon^{-1})$ local data samples.

However, due to the requirement that each iteration of the algorithm needs a full gradient evaluation, the above batch methods can be computationally very demanding. One natural solution is to use the stochastic gradient to approximate the true gradient. Stochastic decentralized non-convex methods can be traced back to (Bianchi & Jakubowicz, 2013; Bianchi et al., 2013), and recent advances including DSGD (Jiang et al., 2017), PSGD (Lian et al., 2017), D^2 (Tang et al., 2018), GNSD (Lu et al., 2019) and stochastic gradient push (Assran et al., 2019). However, the large variance coming from the stochastic gradient estimator and the use of

[1] Note that for the finite sum problem (2), the "sample complexity" refers to the total number of samples *accessed* by the algorithms to compute sample gradient $\nabla f_j^i(\mathbf{x}_i)$'s. If the same sample $j \in [n_i]$ is accessed k times and each time the evaluated gradients are different, then the sample complexity increases by k.

[2] For batch algorithms DGD, NEXT, Prox-PDA and xFILTER, the bounds are obtained by multiplying their convergence rates with $m \times n$, since when applied to solve finite-sum problems, each iteration requires $\mathcal{O}(1)$ full gradient evaluation.

Table 1. Comparison of algorithms on decentralized non-convex optimization

ALGORITHM	CONSTANT STEPSIZE	FINITE-SUM	ONLINE	COMMUNICATION
DGD (ZENG & YIN, 2018)	✗	$\mathcal{O}(mn\epsilon^{-2})$	✗	$\mathcal{O}(\epsilon^{-2})$
SONATA (SUN ET AL., 2019)	✓	$\mathcal{O}(mn\epsilon^{-1})$	✗	$\mathcal{O}(\epsilon^{-1})$
PROX-PDA (HONG ET AL., 2017)	✓	$\mathcal{O}(mn\epsilon^{-1})$	✗	$\mathcal{O}(\epsilon^{-1})$
XFILTER (SUN & HONG, 2019)	✓	$\mathcal{O}(mn\epsilon^{-1})$	✗	$\mathcal{O}(\epsilon^{-1})$
PSGD (LIAN ET AL., 2017)	✗	$\mathcal{O}(m\epsilon^{-2})$	$\mathcal{O}(m\epsilon^{-2})$	$\mathcal{O}(\epsilon^{-2})$
D^2 (TANG ET AL., 2018)	✓	$\mathcal{O}(m\epsilon^{-2})$	$\mathcal{O}(m\epsilon^{-2})$	$\mathcal{O}(\epsilon^{-2})$
GNSD (LU ET AL., 2019)	✓	$\mathcal{O}(m\epsilon^{-2})$	$\mathcal{O}(m\epsilon^{-2})$	$\mathcal{O}(\epsilon^{-2})$
D-GET (THIS WORK)	✓	$\mathcal{O}(m\sqrt{n}\epsilon^{-1})$	$\mathcal{O}(m\epsilon^{-3/2})$	$\mathcal{O}(\epsilon^{-1})$
Lower Bound (FANG ET AL., 2018; SUN & HONG, 2019)		$\mathcal{O}(\sqrt{mn}\epsilon^{-1})$	-	$\mathcal{O}(\epsilon^{-1})$

diminishing step-size slow down the convergence, resulting at least $\mathcal{O}(m\epsilon^{-2})$ sample and $\mathcal{O}(\epsilon^{-2})$ communication cost.

Variance Reduction. Consider the following non-convex finite sum problem: $\min_{\mathbf{w}\in\mathbb{R}^d} f(\mathbf{w}) = \frac{1}{mn}\sum_{j=1}^{mn} f_j(\mathbf{w})$. If we assume that $f(\cdot)$ has Lipschitz gradient, and directly apply the vanilla gradient descent (GD) method on $f(\mathbf{w})$, then it requires $\mathcal{O}(mn\epsilon^{-1})$ gradient evaluations to reach $\|\nabla f(\mathbf{w})\|^2 \leq \epsilon$ (Nesterov, 1998). When $m \times n$ is large, it is usually preferable to process a subset of data each time. In this case, stochastic gradient descent (SGD) can be used to achieve an $\mathcal{O}(\epsilon^{-2})$ convergence rate (Ghadimi & Lan, 2013). To bridge the gap between the GD and SGD, many variance reduced gradient estimators have been proposed, including SAGA (Defazio et al., 2014) and SVRG (Johnson & Zhang, 2013). The idea is to reduce the variance of the stochastic gradient estimators and substantially improves the convergence rate. In particular, the above approaches have been shown to achieve sample complexities of $\mathcal{O}((mn)^{2/3}\epsilon^{-1})$ for finite sum problems (Reddi et al., 2016; Allen-Zhu & Hazan, 2016; Lei et al., 2017) and $\mathcal{O}(\epsilon^{-5/3})$ for online problem (Lei et al., 2017). Recent works further improve the above gradient estimators and achieve $\mathcal{O}((mn)^{1/2}\epsilon^{-1})$ sample complexity for finite sum problems (Nguyen et al., 2019; Fang et al., 2018; Wang et al., 2019; Zhou et al., 2018) and $\mathcal{O}(\epsilon^{-3/2})$ sample complexity for online problems (Fang et al., 2018; Wang et al., 2019). At the same time, the $\mathcal{O}((mn)^{1/2}\epsilon^{-1})$ sample complexity is shown to be optimal when $m \times n \leq \mathcal{O}(\epsilon^{-2})$ (Fang et al., 2018).

Decentralized Variance Reduction. The variance reduced decentralized optimization has been extensively studied for convex problems. The DSA proposed in (Mokhtari & Ribeiro, 2016) combines the algorithm design ideas from EXTRA (Shi et al., 2015) and SAGA (Defazio et al., 2014), and achieves the first expected linear convergence for decentralized stochastic optimization. Recent works also include the DSBA (Shen et al., 2018), diffusion-AVRG (Yuan et al., 2018), ADFS (Hendrikx et al., 2019), SAL-Edge (Wang & Li, 2019), GT-SAGA (Xin et al., 2019), Network-DANE (Li et al., 2019), and Cen et al. (2019), just to name a few. However, when the problem becomes non-convex, to the best of our knowledge, no algorithms with provable guarantees are available.

2. The Finite Sum Setting

In this section, we consider the non-convex decentralized optimization problem (1) with finite number of samples as defined in (2), which is restated below:

$$\min_{\mathbf{x}\in\mathbb{R}^{md}} f(\mathbf{x}) = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} f_j^i(\mathbf{x}_i), \qquad \text{(P1)}$$
$$\text{s.t.} \quad \mathbf{x}_i = \mathbf{x}_k, \quad \forall (i,k) \in \mathcal{E}.$$

We make the following standard assumptions on the above problem:

Assumption 1. *The objective function has Lipschitz continuous gradient with constant L:*

$$\|\nabla f^i(\mathbf{x}_i) - \nabla f^i(\mathbf{x}_i')\| \leq L\|\mathbf{x}_i - \mathbf{x}_i'\|, \forall i \qquad (5)$$

while the component function has average Lipschitz continuous gradient with constant L:

$$\mathbb{E}_j\|\nabla f_j^i(\mathbf{x}_i) - \nabla f_j^i(\mathbf{x}_i')\| \leq L\|\mathbf{x}_i - \mathbf{x}_i'\|, \forall i \qquad (6)$$

Assumption 2. *The mixing matrix $\mathbf{W}$ is symmetric, and satisfying the following*

$$|\lambda_{\max}(\mathbf{W})| := \eta < 1, \quad \mathbf{W}\mathbf{1} = \mathbf{1}, \qquad (7)$$

where $\lambda_{\max}(\mathbf{W})$ denotes the second largest eigenvalue of $\mathbf{W} \in \mathbb{R}^{m\times m}$.[3] *Note that many choices of mixing matrices satisfy the above condition, see Appendix A.*

Next, let us formally define our communication and sample complexity measures.

Definition 1. (Sample Complexity) *The Incremental First-order Oracle (IFO) is defined as an operation in which, one node $i \in [m]$ takes a data sample $j \in [n]$, a point $\mathbf{w} \in \mathbb{R}^d$, and returns the pair $(f_j^i(\mathbf{w}), \nabla f_j^i(\mathbf{w}))$. The sample complexity is defined as the total number of IFO calls required across the entire network to achieve an ϵ stationary solution defined in (4).*

[3] For notation simplicity when dealing with mixing matrix multiplication, but without loss of generality, we will assume that the optimization variable $\mathbf{x}_i$ in (1) is a scalar. The results can be extended to vector case via the Kronecker product.

Definition 2. (Communication Complexity) *In one round of communication, each node $i \in [m]$ is allowed to broadcast and received one d-dimensional vector to and from its neighbors, respectively. Then the communication complexity is defined as the total rounds of communications required to achieve an ϵ stationary solution defined in (4).*

2.1. Algorithm Design

In this section, we introduce the proposed algorithm named Decentralized Gradient Estimation and Tracking (D-GET), for solving problem (P1). To motivate our algorithm design, we can observe from our discussion in Section 1.2 that, the existing deterministic decentralized methods typically suffer from the high sample complexity, while the decentralized stochastic algorithms suffer from the high communication cost. Such a phenomenon inspires us to find a solution in between, which could simultaneously reduce the sample and the communication costs.

One natural solution is to incorporate the modern variance reduction techniques into the classical decentralized methods. Our idea is to use some variance reduced gradient estimator to track the full gradient of the entire problem, then perform decentralized gradient descent update. The gradient tracking step gives us fast convergence with a constant step-size, while the variance reduction method significantly reduces the variation of the estimated gradient.

Unfortunately, the decentralized methods and variance reduction techniques cannot be directly combined. Compared with the existing decentralized and variance reduction techniques in the literature, the key challenges in the algorithm design and analysis are given below:

1) Due to the decentralized nature of the problem, none of the nodes can access the full gradient of the original objective function. The (possibly uncontrollable) consensus error always exists during the whole process of implementing the decentralized algorithm. Therefore, it is not clear that the existing variance reduction methods could be applied at each individual node effectively, since all of those require accurate global gradient evaluation from time to time.

2) It is then natural to integrate some procedure that is able to approximate the global gradient. For example, one straightforward way to perform gradient tracking is to introduce a new auxiliary variable $\mathbf{y}$ as the following (Di Lorenzo & Scutari, 2016; Lu et al., 2019), which is updated by only using local estimated gradient and neighbors' parameters:

$$\mathbf{y}_i^r = \sum_{k\in\mathcal{N}_i} \mathbf{W}_{ik}\mathbf{y}_k^{r-1} + \frac{1}{|S_2^r|}\sum_{j\in S_2^r} \nabla f_j^i(\mathbf{x}_i^r) \quad (8)$$
$$- \frac{1}{|S_2^{r-1}|}\sum_{j\in S_2^{r-1}} \nabla f_j^i(\mathbf{x}_i^{r-1}),$$

where S_2^r and S_2^{r-1} are the samples selected at the r and $r-1$th iterations, respectively. If the tracked $\mathbf{y}_i$'s were used in the (local) variance reduction procedure, there would be at least two main issues of decreasing the variance resulted from the tracked gradient as follows: *i*) at the early stage of implementing the decentralized algorithm, the consensus/tracking error may dominate the variance of the tracked gradient, since the message of the full gradient has not been sufficiently propagated through the network. Consequently, performing variance reduction on $\mathbf{y}_i$'s will not be able to increase the quality of the full gradient estimation; *ii*) even assuming that there was no consensus error. Since only the stochastic gradients, i.e., $\sum_{j\in\mathcal{S}_2^r}\nabla f_j^i(\mathbf{x}_i^r)$, were used in the tracking, the $\mathbf{y}_i^r$'s themselves had high variance, resulting that such (possibly low-quality) full gradient estimates may not be compatible to variance reduction methods as developed in the current literature (which often require full gradient evaluation from time to time).

The challenges discussed above suggest that it is non-trivial to design an algorithm that can be implemented in a fully decentralized manner, while still achieving the superior sample complexity and convergence rate achieved by state-of-the-art variance reduction methods. In this work, we propose an algorithm which uses a novel decentralized gradient estimation and tracking strategy, together with a number of other design choices, to address the issues raised above.

To introduce the algorithm, let us first define two auxiliary local variables $\mathbf{v}_i$ and $\mathbf{y}_i$, where $\mathbf{v}_i$ is designed to estimate the local full batch gradient $\frac{1}{n}\sum_{j=1}^n \nabla f_j^i(\mathbf{x}_i)$ by only using sample gradient $\nabla f_j^i(\mathbf{x}_i)'s$, while $\mathbf{y}_i$ is designed to track the global average gradient $\frac{1}{mn}\sum_{i=1}^m\sum_{j=1}^n \nabla f_j^i(\mathbf{x}_i)$ by utilizing $\mathbf{v}_i$'s. After the local and global gradient estimates are obtained, the algorithm performs local update based on the direction of $\mathbf{y}_i$; see the main steps below.

1) Local update using estimated gradient ($\mathbf{x}$ update): Each local node i first combines its previous iterates $\mathbf{x}_i^{r-1}$ with its local neighbors $\mathbf{x}_k^{r-1},\ k\in\mathcal{N}_i$ (by using the kth row of weight matrix $\mathbf{W}$), then makes a prediction based on the gradient estimate $\mathbf{y}_i^{r-1}$, i.e.,

$$\mathbf{x}_i^r = \sum_{k\in\mathcal{N}_i}\mathbf{W}_{ik}\mathbf{x}_k^{r-1} - \alpha\mathbf{y}_i^{r-1}. \quad (9)$$

2) Estimate local gradients ($\mathbf{v}$ update): Depending on the iteration r, each local node i either directly calculates the full local gradient $\nabla f^i(\mathbf{x}_i^r)$ when $\mathrm{mod}(r,q)=0$

$$\mathbf{v}_i^r = \nabla f^i(\mathbf{x}_i^r), \quad (10)$$

or estimates its local gradient via an estimator $\mathbf{v}$ using $|S_2|$ random samples otherwise,

$$\mathbf{v}_i^r = \frac{1}{|S_2|}\sum_{j\in S_2}\left[\nabla f_j^i(\mathbf{x}_i^r) - \nabla f_j^i(\mathbf{x}_i^{r-1})\right] + \mathbf{v}_i^{r-1}, \quad (11)$$

where $q>0$ is the interval in which local full gradient will be evaluated once.

3) Track global gradients ($\mathbf{y}$ update): Each local node i combines its previous local estimate $\mathbf{y}_i^{r-1}$ with its local neighbors $\mathbf{y}_k^{r-1}$, $k \in \mathcal{N}_i$, then makes a new estimation based on the fresh information $\mathbf{v}_i^r$, i.e.,

$$\mathbf{y}_i^r = \sum_{k\in\mathcal{N}_i} \mathbf{W}_{ik}\mathbf{y}_k^{r-1} + \mathbf{v}_i^r - \mathbf{v}_i^{r-1}. \tag{12}$$

In the following table, we summarize the proposed algorithm in a more compact form. Note that we use $\mathbf{x} \in \mathbb{R}^{md}$, $\mathbf{v} \in \mathbb{R}^{md}$, $\mathbf{y} \in \mathbb{R}^{md}$, $\nabla f(\mathbf{x}) \in \mathbb{R}^{md}$ and $\nabla f_j(\mathbf{x}) \in \mathbb{R}^{md}$ to denote the concatenation of the $\mathbf{x}_i \in \mathbb{R}^d$, $\mathbf{v}_i \in \mathbb{R}^d$, $\mathbf{y}_i \in \mathbb{R}^d$, $\nabla f^i(\mathbf{x}_i) \in \mathbb{R}^d$ and $\nabla f_j^i(\mathbf{x}_i) \in \mathbb{R}^d$ across all nodes.

Algorithm 1 D-GET Algorithm for finite sum problem (P1)

Input: $\mathbf{x}^0, \alpha, q, |S_2|$
$\mathbf{v}^0 = \nabla f(\mathbf{x}^0)$, $\mathbf{y}^0 = \nabla f(\mathbf{x}^0)$
for $r = 1, 2, \ldots, T$ **do**
  $\mathbf{x}^r = \mathbf{W}\mathbf{x}^{r-1} - \alpha\mathbf{y}^{r-1}$
  if $\mathrm{mod}(r, q) = 0$ **then**
    Calculate the full gradient
    $\mathbf{v}^r = \nabla f(\mathbf{x}^r)$
  else
    Each node draws S_2 samples from $[n]$ with replacement
    $\mathbf{v}^r = \frac{1}{|S_2|}\sum_{j\in S_2}\left[\nabla f_j(\mathbf{x}^r) - \nabla f_j(\mathbf{x}^{r-1})\right] + \mathbf{v}^{r-1}$
  end if
  $\mathbf{y}^r = \mathbf{W}\mathbf{y}^{r-1} + \mathbf{v}^r - \mathbf{v}^{r-1}$
end for
Output: $\mathbf{x}^R$ where $R \in [0, T]$ is the uniformly distributed random variable.

Remark 1. This is a "double loop" algorithm, where each outer iteration (i.e., mod $(r, q) = 0$) is followed by $q - 1$ inner iterations (i.e., mod $(r, q) \neq 0$). The inner loop estimates the local gradient via stochastic sampling at every iteration r, while the outer loop aims to reduce the estimation variance by recalculating the full batch gradient at every q iterations. The local communication, update, and tracking steps are performed at both inner and outer iterations.

Remark 2. In D-GET, the total communication rounds is in the same order as the total number of iterations, since only two rounds of communications are performed per iteration, via broadcasting the local variable $\mathbf{x}_i^{r-1}$ and $\mathbf{y}_i^{r-1}$ to their neighbors, and combining local $\mathbf{x}_k^{r-1}$ and $\mathbf{y}_k^{r-1}$'s, $k \in \mathcal{N}_i$. On the other hand, the total number of samples used per iteration is either $m|S_2|$ (where inner iterations are executed) or mn (where outer iterations are executed).

Remark 3. Note that our $\mathbf{x}$ and $\mathbf{y}$ updates are reminiscent of the classical gradient tracking methods (Di Lorenzo & Scutari, 2016), and $\mathbf{v}$ update takes a similar form as the SARAH/SPIDER estimator (Nguyen et al., 2017; Fang et al., 2018). However, it is non-trivial to directly combine the gradient tracking and the variance reduction together, as we mentioned at the beginning of Section 2.1. The proposed D-GET uses a number of design choices to address these challenges. For example, two vectors $\mathbf{v}$ and $\mathbf{y}$ are used to respectively estimate the local and global gradients, in a way that the local gradient estimates do not depend on the (potentially inaccurate) global tracked gradients; to reduce the variance in $\mathbf{y}$, we occasionally use the full local gradient to perform tracking, etc. Nevertheless, the key challenge in the analysis is to properly bound the accumulated errors from the two estimates $\mathbf{v}$ and $\mathbf{y}$.

2.2. Convergence Analysis

To facilitate our analysis, we first define the average iterates $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$ among all m nodes,

$$\bar{\mathbf{x}}^r := \frac{1}{m}\sum_i \mathbf{x}_i^r, \quad \bar{\mathbf{v}}^r := \frac{1}{m}\sum_i \mathbf{v}_i^r, \tag{13a}$$

$$\bar{\mathbf{y}}^r := \frac{1}{m}\sum_i \mathbf{y}_i^r. \tag{13b}$$

We use r to denote the overall iteration number. The total number of outer iterations until iteration r as below:

$$n_r := \lfloor r/q \rfloor + 1, \quad \text{such that} \quad (n_r - 1)q \le r \le n_r q - 1.$$

Next, we outline the proof steps of the convergence rate analysis.

Step 1. We first show that the variance of our local and global gradient estimators can be bounded via $\mathbf{x}$ and $\mathbf{y}$ iterates. The bounds to be given below is tighter than the classical analysis of decentralized stochastic methods, which assume the variance are bounded by some universal constant (Tang et al., 2018; Lian et al., 2017; Jiang et al., 2017). This is an important step to obtain lower sample/communication complexity, since later we can show that the right-hand-side (RHS) of our bound shrinks as the iteration progresses.

Lemma 1. *(Bounded Variance) Under Assumption 1 - 2, the sequence generated by the inner loop of Algorithm 1 satisfies the following inequalities (for all $(n_r-1)q \le r \le n_r q-1$):*

$$\begin{aligned}
&\mathbb{E}\|\bar{\mathbf{y}}^r - \frac{1}{m}\sum_{i=1}^m \nabla f^i(\mathbf{x}_i^r)\|^2 \qquad (14)\\
\le&\frac{8L^2}{m|S_2|}\sum_{t=(n_r-1)q}^{r}\mathbb{E}\|\mathbf{x}^t - \mathbf{1}\bar{\mathbf{x}}^t\|^2\\
+&\frac{4\alpha^2L^2}{m|S_2|}\sum_{t=(n_r-1)q}^{r}\mathbb{E}\|\mathbf{y}^t - \mathbf{1}\bar{\mathbf{y}}^t\|^2\\
+&\frac{4\alpha^2L^2}{|S_2|}\sum_{t=(n_r-1)q}^{r}\mathbb{E}\|\bar{\mathbf{y}}^t\|^2\\
+&\mathbb{E}\|\bar{\mathbf{y}}^{(n_r-1)q} - \frac{1}{m}\sum_{i=1}^m \nabla f^i(\mathbf{x}_i{}^{(n_r-1)q})\|^2.
\end{aligned}$$

$$\mathbb{E}\|\mathbf{v}^r - \nabla f(\mathbf{x}^r)\|^2 \leq \frac{L^2}{|S_2|}\sum_{t=(n_r-1)q}^{r}\mathbb{E}\|\mathbf{x}^{t+1}-\mathbf{x}^t\|^2 + \mathbb{E}\|\mathbf{v}^{(n_r-1)q} - \nabla f(\mathbf{x}^{(n_r-1)q})\|^2. \quad (15)$$

Step 2. We then study the descent on $\mathbb{E}[f(\bar{\mathbf{x}}^r)]$, which is the expected value of the cost function evaluated on the average iterates.

Lemma 2. *(Descent Lemma) Suppose Assumptions 1 - 2 hold, and for any $r \geq 0$ satisfying mod$(r,q) = 0$, the following holds for some $\epsilon_1 \geq 0$:*

$$\mathbb{E}\left[\|\bar{\mathbf{y}}^r - \frac{1}{mn}\sum_{i=1}^{m}\sum_{k=1}^{n}\nabla f_k^i(\mathbf{x}_i^r)\|^2\right] \leq \epsilon_1. \quad (16)$$

Algorithm 1 ensures the following relation for all $r \geq 0$,

$$\begin{aligned}\mathbb{E}[f(\bar{\mathbf{x}}^{r+1})] \leq{}& \mathbb{E}[f(\bar{\mathbf{x}}^0)] \quad (17)\\ &- \left(\frac{\alpha}{2} - \frac{\alpha^2 L}{2} - \frac{4\alpha^3 L^2 q}{|S_2|}\right)\sum_{t=0}^{r}\mathbb{E}\|\bar{\mathbf{y}}^t\|^2\\ &+ \left(\frac{\alpha L^2}{m} + \frac{8\alpha L^2 q}{m|S_2|}\right)\sum_{t=0}^{r}\mathbb{E}\|\mathbf{x}^t - \mathbf{1}\bar{\mathbf{x}}^t\|^2\\ &+ \frac{4\alpha^3 L^2 q}{m|S_2|}\sum_{t=0}^{r}\mathbb{E}\|\mathbf{y}^t - \mathbf{1}\bar{\mathbf{y}}^t\|^2 + \alpha(r+1)\epsilon_1.\end{aligned}$$

A key observation from Lemma 2 is that, in the RHS of (17), besides the negative term in $\mathbb{E}\|\bar{\mathbf{y}}^k\|^2$, we also have several extra terms (such as $\mathbb{E}\|\mathbf{x}^k - \mathbf{1}\bar{\mathbf{x}}^k\|^2$ and $\mathbb{E}\|\mathbf{y}^k - \mathbf{1}\bar{\mathbf{y}}^k\|^2$) that cannot be made negative. Therefore, we need to find some potential function that is strictly descending per iteration.

Note that ϵ_1 in (16) comes from the variance of $\mathbf{v}^r$ in estimating the full local gradient at each outer loop n_r. For Algorithm 1, where we calculate a full batch gradient per outer loop in step (33), $\epsilon_1 = 0$. However, we still would like to include ϵ_1 in the above result because, later when we analyze the online version (where such a variance will no longer be zero), we can re-use the above result.

Step 3. Next, we introduce the contraction property, which combined with $\mathbb{E}[f(\bar{\mathbf{x}}^r)]$ will be used to construct the potential function.

Lemma 3. *(Iterates Contraction) Using the Assumption 2 on $\mathbf{W}$ and applying Algorithm 1, we have the following contraction property of the iterates:*

$$\begin{aligned}&\mathbb{E}\|\mathbf{x}^{r+1} - \mathbf{1}\bar{\mathbf{x}}^{r+1}\|^2 \quad (18)\\ \leq{}&(1+\beta)\eta^2\mathbb{E}\|\mathbf{x}^r - \mathbf{1}\bar{\mathbf{x}}^r\|^2 + (1+\frac{1}{\beta})\alpha^2\mathbb{E}\|\mathbf{y}^r - \mathbf{1}\bar{\mathbf{y}}^r\|^2,\\ &\mathbb{E}\|\mathbf{y}^{r+1} - \mathbf{1}\bar{\mathbf{y}}^{r+1}\|^2 \quad (19)\\ \leq{}&(1+\beta)\eta^2\mathbb{E}\|\mathbf{y}^r - \mathbf{1}\bar{\mathbf{y}}^r\|^2 + (1+\frac{1}{\beta})\mathbb{E}\|\mathbf{v}^{r+1} - \mathbf{v}^r\|^2,\end{aligned}$$

where β is some constant such that $(1+\beta)\eta^2 < 1$.

If we further assume for all $r \geq 0$ satisfying mod$(r,q) = 0$, the following holds for some $\epsilon_2 \geq 0$:

$$\mathbb{E}\|\mathbf{v}^r - \nabla f(\mathbf{x}^r)\|^2 \leq \epsilon_2. \quad (20)$$

Then we have the following bound:

$$\begin{aligned}&\sum_{t=0}^{r}\mathbb{E}\|\mathbf{v}^{t+1} - \mathbf{v}^t\|^2 \leq 48L^2\sum_{t=0}^{r}\mathbb{E}\|\mathbf{x}^t - \mathbf{1}\bar{\mathbf{x}}^t\|^2\\ &+ 24L^2\alpha^2\sum_{t=0}^{r}\mathbb{E}\|\mathbf{y}^t - \mathbf{1}\bar{\mathbf{y}}^t\|^2\\ &+ 24L^2\alpha^2\sum_{t=0}^{r}\mathbb{E}\|\mathbf{1}\bar{\mathbf{y}}^t\|^2 + 6(r+1)\epsilon_2,\ \forall\, r \geq 0. \quad (21)\end{aligned}$$

Again, ϵ_2 comes from the variance of the estimating the local gradient in each outer loop, and we have $\epsilon_2 = 0$ for Algorithm 1. Note that (18) can also be written as following

$$\begin{aligned}&\mathbb{E}\|\mathbf{x}^{r+1} - \mathbf{1}\bar{\mathbf{x}}^{r+1}\|^2 - \mathbb{E}\|\mathbf{x}^r - \mathbf{1}\bar{\mathbf{x}}^r\|^2\\ \leq{}&\left((1+\beta)\eta^2 - 1\right)\mathbb{E}\|\mathbf{x}^r - \mathbf{1}\bar{\mathbf{x}}^r\|^2\\ &+ (1+\frac{1}{\beta})\alpha^2\mathbb{E}\|\mathbf{y}^r - \mathbf{1}\bar{\mathbf{y}}^r\|^2. \quad (22)\end{aligned}$$

One key observation here is that we have $(1+\beta)\eta^2 - 1 < 0$ by properly choosing β. Therefore, the RHS of the above equation can be made negative by properly selecting the step-size α.

Step 4. This step combines the descent estimates obtained in Step 2-3 to construct a potential function, by using a conic combination of $\mathbb{E}[f(\bar{\mathbf{x}}^r)]$, $\mathbb{E}\|\mathbf{x}^r - \mathbf{1}\bar{\mathbf{x}}^r\|^2$ and $\mathbb{E}\|\mathbf{y}^r - \mathbf{1}\bar{\mathbf{y}}^r\|^2$.

Lemma 4. *(Potential Function) Constructing the following potential function*

$$H(\mathbf{x}^r) := \mathbb{E}[f(\bar{\mathbf{x}}^r)] + \frac{1}{m}\mathbb{E}\|\mathbf{x}^r - \mathbf{1}\bar{\mathbf{x}}^r\|^2 + \frac{\alpha}{m}\mathbb{E}\|\mathbf{y}^r - \mathbf{1}\bar{\mathbf{y}}^r\|^2.$$

Under Assumption 1 - 2 and Algorithm 1, if we further pick $q = |S_2|$ and define ϵ_1 and ϵ_2 as in (16) and (20), we have

$$\begin{aligned}&H(\mathbf{x}^{r+1}) - H(\mathbf{x}^0)\\ \leq{}&-C_1\sum_{t=0}^{r}\mathbb{E}\|\bar{\mathbf{y}}^t\|^2 - C_2\sum_{t=0}^{r}\frac{1}{m}\mathbb{E}\|\mathbf{x}^t - \mathbf{1}\bar{\mathbf{x}}^t\|^2\\ &- C_3\sum_{t=0}^{r}\frac{1}{m}\mathbb{E}\|\mathbf{y}^t - \mathbf{1}\bar{\mathbf{y}}^t\|^2 + \epsilon_3,\end{aligned}$$

where

$$C_1 := \left(\frac{1}{2} - \frac{\alpha L}{2} - 4\alpha^2 L^2 - 24(1+\frac{1}{\beta})\alpha^2 L^2\right)\alpha, \tag{23a}$$

$$C_2 := \left(1 - (1+\beta)\eta^2 - 48\alpha(1+\frac{1}{\beta})L^2 - 9\alpha L^2\right), \tag{23b}$$

$$C_3 := \alpha - (1+\beta)\alpha\eta^2 - (1+\frac{1}{\beta})\alpha^2 - 24(1+\frac{1}{\beta})\alpha^3 L^2 - 4\alpha^3 L^2, \tag{23c}$$

$$\epsilon_3 := \alpha(r+1)(\epsilon_1 + 6\frac{1}{m}(1+\frac{1}{\beta})\epsilon_2). \tag{23d}$$

Step 5. We can then properly choose the step-size α, and make C_1, C_2, C_3 to be positive. Therefore, our solution quality measure $\mathbb{E}\|\frac{1}{m}\sum_{i=1}^m \nabla f^i(\mathbf{x}_i^r)\|^2 + \frac{1}{m}\mathbb{E}\|\mathbf{x}^r - \mathbf{1}\bar{\mathbf{x}}^r\|^2$ can be expressed as the difference of the potential functions and the proof is complete.

Theorem 1. *Consider problem (P1) and under Assumption 1 - 2, if we pick $\alpha = \min\{K_1, K_2, K_3\}$ and $q = |S_2| = \sqrt{n}$, then we have following results by applying Algorithm 1,*

$$\frac{1}{T}\sum_{t=0}^{T}\mathbb{E}\|\frac{1}{m}\sum_{i=1}^m \nabla f^i(\mathbf{x}_i^t)\|^2 + \frac{1}{T}\sum_{t=0}^{T}\frac{1}{m}\mathbb{E}\|\mathbf{x}^t - \mathbf{1}\bar{\mathbf{x}}^t\|^2 \leq C_0 \cdot \frac{\mathbb{E}f(\mathbf{x}^0) - \underline{f}}{T},$$

where $\underline{f}$ denotes the lower bound of f, and the constants are defined as following

$$K_1 := \frac{-\frac{L}{2} + \sqrt{(\frac{L}{2})^2 + 48(1+\frac{1}{\beta})L^2 + 8L^2}}{48(1+\frac{1}{\beta})L^2 + 8L^2},$$

$$K_2 := \frac{1-(1+\beta)\eta^2}{48(1+\frac{1}{\beta})L^2 + 9L^2},$$

$$K_3 := \frac{-(1+\frac{1}{\beta})}{48(1+\frac{1}{\beta})L^2 + 8L^2} + \frac{\sqrt{(1+\frac{1}{\beta})^2 + 4(1-(1+\beta)\eta^2)(24(1+\frac{1}{\beta}) + 4L^2)}}{48(1+\frac{1}{\beta})L^2 + 8L^2},$$

$$C_0 := \left(\frac{8\alpha^2L^2+2}{C_1} + \frac{16L^2+1}{mC_2} + \frac{8\alpha^2L^2}{mC_3}\right),$$

in which η denotes the second largest eigenvalue of the mixing matrix from (7), β denotes a constant satisfying $1-(1+\beta)\eta^2 > 0$, for example, $\beta = (1-\eta^2)/(2\eta^2)$, and C_1, C_2, C_3 are defined in (23a)-(23c).

By directly applying the above result, we have the upper bound on gradient and communication cost by properly choosing T based on ϵ.

Corollary 1. *To achieve the following ϵ stationary solution of problem (P1) by Algorithm 1,*

$$\frac{1}{T}\sum_{t=0}^{T}\mathbb{E}\|\frac{1}{m}\sum_{i=1}^m \nabla f^i(\mathbf{x}_i^t)\|^2 + \frac{1}{T}\sum_{t=0}^{T}\frac{1}{m}\mathbb{E}\|\mathbf{x}^t - \mathbf{1}\bar{\mathbf{x}}^t\|^2 \leq \epsilon,$$

the total number of iterations T and communication rounds required are both in the order of $\mathcal{O}(\epsilon^{-1})$, and the total number of samples evaluated across the network is in the order of $\mathcal{O}(mn + mn^{1/2}\epsilon^{-1})$.

Note that our metric is evaluated on individual variable $\mathbf{x}_i$ and our algorithm also output $\mathbf{x}_i$ as each agent i's final solution. If we choose to use the average $\bar{\mathbf{x}} = \frac{1}{m}\sum \mathbf{x}_i$ as our final solution, one can show that the metric on average iterates $\bar{\mathbf{x}}$ is also small through simple derivations.

Corollary 2. *To achieve the following ϵ stationary solution of problem (P1) by Algorithm 1,*

$$\frac{1}{T}\sum_{t=0}^{T}\mathbb{E}\|\frac{1}{m}\sum_{i=1}^m \nabla f^i(\bar{\mathbf{x}}^t)\|^2 + \frac{1}{T}\sum_{t=0}^{T}\frac{1}{m}\mathbb{E}\|\mathbf{x}^t - \mathbf{1}\bar{\mathbf{x}}^t\|^2 \leq \epsilon,$$

the total number of iterations T and communication rounds required are both in the order of $\mathcal{O}(\epsilon^{-1})$, and the total number of samples evaluated across the network is in the order of $\mathcal{O}(mn + mn^{1/2}\epsilon^{-1})$.

If we further take expectation over the iteration t, we can have the convergence guarantee on our algorithm output $\mathbf{x}_i^R$ (or $\bar{\mathbf{x}}^R$ similarly), where $R \in [0, T]$ is the uniformly distributed random variable.

Corollary 3. *To achieve the following ϵ stationary solution of problem (P1) by Algorithm 1,*

$$\mathbb{E}\|\frac{1}{m}\sum_{i=1}^m \nabla f^i(\mathbf{x}_i^R)\|^2 + \frac{1}{m}\mathbb{E}\|\mathbf{x}^R - \mathbf{1}\bar{\mathbf{x}}^R\|^2 \leq \epsilon,$$

the total number of iterations T and communication rounds required are both in the order of $\mathcal{O}(\epsilon^{-1})$, and the total number of samples evaluated across the network is in the order of $\mathcal{O}(mn + mn^{1/2}\epsilon^{-1})$. The expectation $\mathbb{E}$ here is taken over the iteration R and the randomness from the random sampling step (11).

3. The Online Setting

In this section, we discuss the online setting (3) for solving problem (1), where the problem can either be expressed as the following

$$\min_{\mathbf{x}\in\mathbb{R}^{md}} f(\mathbf{x}) = \frac{1}{m}\sum_{i=1}^m \mathbb{E}_{\xi\sim\mathcal{D}_i}\left[f_\xi^i(\mathbf{x}_i)\right], \quad \text{s.t.} \quad \mathbf{x}_i = \mathbf{x}_j, \quad \forall (i,j)\in\mathcal{E}, \tag{P2}$$

where ξ represents the data drawn from the data distribution $\mathcal{D}_i$ at the ith node, or in form (P1) such that the number of samples n is too large to calculate the full batch even occasionally. In either one of these scenarios, full batch evaluations at the local nodes are no longer performed for each outer iteration.

The above setting has been well-studied for the centralized problem (with large or even infinite number of samples). For example, in SCSG (Lei et al., 2017), a batch size $\mathcal{O}(\epsilon^{-1})$ is used when the sample size is large or the target accuracy $\mathcal{O}(\epsilon)$ is moderate, improving the rate to $\mathcal{O}(\epsilon^{-5/3})$ from $\mathcal{O}(\epsilon^{-2})$ compared to the vanilla SGD (Ghadimi & Lan, 2013). Recently, SPIDER (Fang et al., 2018) further improves the results to $\mathcal{O}(\epsilon^{-3/2})$, while the SpiderBoost (Wang et al., 2019) uses a constant step-size and is amenable to solve non-smooth problem at this rate.

3.1. The Proposed Algorithm

To begin with, we first introduce two additional commonly used assumptions in the online learning setting, together with our Assumption 1 and 2.

Assumption 3. *At each iteration, samples are independently collected, and the stochastic gradient is an unbiased estimate of the true gradient:*

$$\mathbb{E}_\xi[\nabla f_\xi^i(\mathbf{x}_i)] = \nabla f^i(\mathbf{x}_i), \forall i. \tag{24}$$

Assumption 4. *The variance between the stochastic gradient and the true gradient is bounded:*

$$\mathbb{E}_\xi[\|\nabla f_\xi^i(\mathbf{x}_i) - \nabla f^i(\mathbf{x}_i)\|^2] \le \sigma^2, \forall i. \tag{25}$$

To present our algorithms, note that compared to problem (P1), the main difference of having the expectation in (P2) is that the full batch gradient evaluation is no longer feasible. Therefore, we need to slightly revise our algorithm in Section 2 and redesign the local gradient estimation step (i.e., the $\mathbf{v}$ update). Specifically, different from (10) where we sample the full batch, here we randomly draw S_1 samples, the size of which is inversely proportional to the desired accuracy ϵ. We have the following updates on $\mathbf{v}$:

Depending on the iteration r, each local node i either estimates its local gradient using $|S_1|$ random samples when $\text{mod}(r, q) = 0$,

$$\mathbf{v}_i^r = \frac{1}{|S_1|}\sum_{\xi\in S_1}\nabla f_\xi^i(\mathbf{x}_i^r), \tag{26}$$

or uses $|S_2|$ random samples otherwise,

$$\mathbf{v}_i^r = \frac{1}{|S_2|}\sum_{\xi\in S_2}\left[\nabla f_\xi^i(\mathbf{x}_i^r) - \nabla f_\xi^i(\mathbf{x}_i^{r-1})\right] + \mathbf{v}_i^{r-1}. \tag{27}$$

It is easy to check that the following relation on average iterates is obvious when $\text{mod}(r, q) = 0$ and $\bar{\mathbf{y}}^0 = \bar{\mathbf{v}}^0$,

$$\bar{\mathbf{y}}^r = \bar{\mathbf{v}}^r = \frac{1}{m|S_1|}\sum_{i=1}^m\sum_{\xi\in S_1}\nabla f_\xi^i(\mathbf{x}_i^r). \tag{28}$$

The rest of the updates on $\mathbf{x}$ and $\mathbf{y}$ are same as the finite sum setting; see Algorithm 2 below for details.

Algorithm 2 D-GET Algorithm (global view) (online)

Input: $\mathbf{x}^0, \alpha, q, |S_1|, |S_2|$
Draw S_1 samples with replacement
$\mathbf{v}^0 = \frac{1}{|S_1|}\sum_{\xi\in S_1}\nabla f_\xi(\mathbf{x}^0)$, $\mathbf{y}^0 = \mathbf{v}^0$
for $r = 1, 2, \ldots$ **do**
  $\mathbf{x}^r = \mathbf{W}\mathbf{x}^{r-1} - \alpha\mathbf{y}^{r-1}$
  if $\text{mod}(r, q) = 0$ **then**
    Draw S_1 samples with replacement
    $\mathbf{v}^r = \frac{1}{|S_1|}\sum_{\xi\in S_1}\nabla f_\xi(\mathbf{x}^r)$
  else
    Draw S_2 samples with replacement
    $\mathbf{v}^r = \frac{1}{|S_2|}\sum_{\xi\in S_2}\left[\nabla f_\xi(\mathbf{x}^r) - \nabla f_\xi(\mathbf{x}^{r-1})\right] + \mathbf{v}^{r-1}$
  end if
  $\mathbf{y}^r = \mathbf{W}\mathbf{y}^{r-1} + \mathbf{v}^r - \mathbf{v}^{r-1}$
end for
Output: $\mathbf{x}^R$ where $R \in [0, T]$ is the uniformly distributed random variable.

3.2. Convergence Analysis

The analysis follows the same steps as described in Section 2.2 and it is easy to verify that our Lemma 1 to Lemma 4 still hold true for Algorithm 2. However, for online setting where we no longer sample a full batch, the variance ϵ_1 and ϵ_2 cannot be eliminated. The lemma given below provides the bounds on ϵ_1 and ϵ_2.

Lemma 5. *(Bounded Variance) Under Assumption 1 to 4, the sequence generated by the outer loop of Algorithm 2 satisfies the following relations (for all r such that mod(r, q)=0)*

$$\mathbb{E}\|\mathbf{v}^r - \nabla f(\mathbf{x}^r)\|^2 \le \frac{m\sigma^2}{|S_1|},$$
$$\mathbb{E}\|\bar{\mathbf{y}}^r - \frac{1}{m}\sum_{i=1}^m\nabla f^i(\mathbf{x}_i^r)\|^2 \le \frac{\sigma^2}{|S_1|}.$$

By using the above lemma, we can then choose the sample size inversely proportional to the targeted accuracy and obtain our final results.

Theorem 2. *Suppose Assumption 1 - 4 hold, and pick the following parameters for problem (P2):*

$$\alpha = \min\{K_1, K_2, K_3\}, \quad q = |S_2| = \sqrt{|S_1|},$$
$$|S_1| = (4C_0\alpha(7 + \frac{6}{\beta})\sigma^2 + 8\sigma^2)/\epsilon.$$

Then we have the following result by applying Algorithm 2,

$$\frac{1}{T}\sum_{t=0}^T\mathbb{E}\|\frac{1}{m}\sum_{i=1}^m\nabla f^i(\mathbf{x}_i^t)\|^2 + \frac{1}{T}\sum_{t=0}^T\frac{1}{m}\mathbb{E}\|\mathbf{x}^t - \mathbf{1}\bar{\mathbf{x}}^t\|^2$$
$$\le C_0\cdot\frac{\mathbb{E}f(\mathbf{x}^0) - \underline{f}}{T} + \frac{\epsilon}{2}.$$

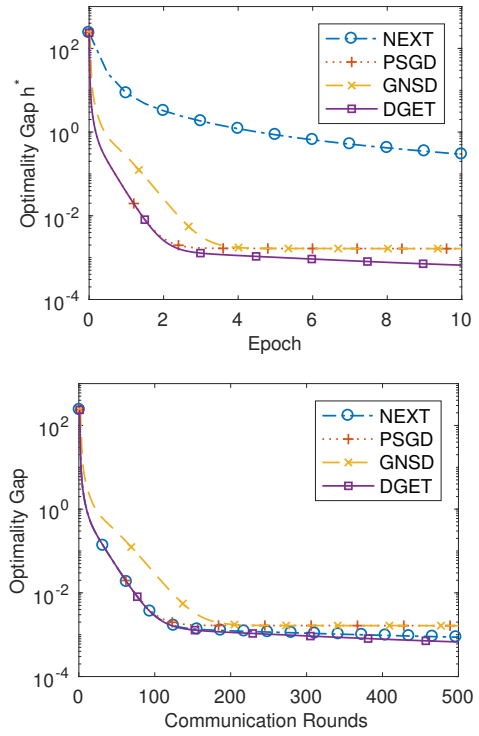

Figure 1. Logistic regression with non-convex regularizer

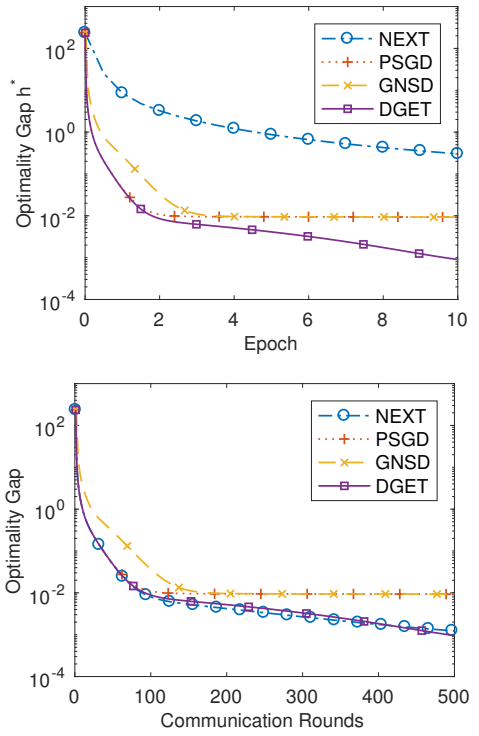

Figure 2. Non-convex robust linear regression

Corollary 4. *By using Algorithm 2, to achieve the ϵ stationary solution of problem (1), i.e.,*

$$\frac{1}{T}\sum_{t=0}^{T}\mathbb{E}\|\frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^t)\|^2+\frac{1}{T}\sum_{t=0}^{T}\frac{1}{m}\mathbb{E}\|\mathbf{x}^t-\mathbf{1}\bar{\mathbf{x}}^t\|^2\leq\epsilon,$$

the total number of iterations T and communication rounds required are both in the order of $\mathcal{O}(\epsilon^{-1})$, and the total sample complexity is in the order of $\mathcal{O}(m\epsilon^{-3/2})$.

4. Experimental Results

In this section, we demonstrate the performance of the proposed algorithms on two classical smooth non-convex problems: a) decentralized logistic regression with non-convex regularizer and b) non-convex robust linear regression, the detailed objective functions are given in Appendix C.

We use the dataset a9a ($n = 32561, d = 123$) from the LIBSVM (Chang & Lin, 2011), and we distribute the data so each node contains 3256 data points with 123 features. Then we compare the proposed D-GET with the NEXT (Sun et al., 2019), PSGD (Lian et al., 2017) and GNSD (Lu et al., 2019) over the path communication graph $\mathcal{E}$.

Simulation results in terms of both sample complexity and the communication complexity averaged over 10 realizations are shown in Fig. 1 and Fig. 2, where the x-axis denotes total number of required (a) epochs and (b) communication rounds, and the y-axis denotes the quality measure (4). It can be observed that the proposed D-GET could achieve much faster convergence in terms of sample complexity, while matches the communication complexity as the deterministic algorithms, as claimed in Theorem 1 and 2. Additional simulations results on different datasets in terms of both the optimality gap and loss functions are available in Appendix C due to space limit.

5. Concluding Remarks

In this work, we proposed a joint gradient estimation and tracking approach (D-GET) for fully decentralized non-convex optimization problems. By utilizing modern variance reduction and gradient tracking techniques, the proposed method improves the sample and/or communication complexities compared with existing methods. In particular, for decentralized finite sum problems, the proposed approach requires only $\mathcal{O}(mn^{1/2}\epsilon^{-1})$ sample complexity and $\mathcal{O}(\epsilon^{-1})$ communication complexity to reach the ϵ stationary solution. For online problem, our approach achieves an $\mathcal{O}(m\epsilon^{-3/2})$ sample and an $\mathcal{O}(\epsilon^{-1})$ communication complexity, which significantly improves upon the best existing bounds of $\mathcal{O}(m\epsilon^{-2})$ and $\mathcal{O}(\epsilon^{-2})$ as derived in (Tang et al., 2018).

6. Acknowledgments

The authors were supported by NSF under the grant CIF-1910385 and in part by an AFOSR grant 19RT0424, and an ARO grant W911NF-19-1-0247.

References

Ali, K. and Van Stam, W. TiVo: making show recommendations using a distributed collaborative filtering architecture. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 394–401, 2004.

Allen-Zhu, Z. and Hazan, E. Variance reduction for faster non-convex optimization. In *Proceedings of International Conference on Machine Learning (ICML)*, pp. 699–707, 2016.

Arjevani, Y., Carmon, Y., Duchi, J. C., Foster, D. J., Srebro, N., and Woodworth, B. Lower bounds for non-convex stochastic optimization. *arXiv preprint arXiv:1912.02365*, 2019.

Assran, M., Loizou, N., Ballas, N., and Rabbat, M. Stochastic gradient push for distributed deep learning. In *Proceedings of International Conference on Machine Learning (ICML)*, pp. 344–353, 2019.

Bertsekas, D. P. *Parallel and distributed computation: numerical methods*, volume 23. Prentice hall Englewood Cliffs, NJ, 1989.

Bianchi, P. and Jakubowicz, J. Convergence of a multi-agent projected stochastic gradient algorithm for non-convex optimization. *IEEE Transactions on Automatic Control*, 58(2):391–405, 2013.

Bianchi, P., Fort, G., and Hachem, W. Performance of a distributed stochastic approximation algorithm. *IEEE Transactions on Information Theory*, 59(11):7405–7418, 2013.

Boyd, S., Diaconis, P., and Xiao, L. Fastest mixing markov chain on a graph. *SIAM Review*, 46(4):667–689, 2004.

Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

Cen, S., Zhang, H., Chi, Y., Chen, W., and Liu, T.-Y. Convergence of distributed stochastic variance reduced methods without sampling extra data. *arXiv preprint arXiv:1905.12648*, 2019.

Chang, C.-C. and Lin, C.-J. Libsvm: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

Chen, J. and Sayed, A. H. Diffusion adaptation strategies for distributed optimization and learning over networks. *IEEE Transactions on Signal Processing*, 60(8):4289–4305, 2012.

Chen, J., Towfic, Z. J., and Sayed, A. H. Dictionary learning over distributed models. *IEEE Transactions on Signal Processing*, 63(4):1001–1016, 2014.

Daneshmand, A., Scutari, G., and Facchinei, F. Distributed dictionary learning. In *Proceedings of the 50th Asilomar Conference on Signals, Systems and Computers*, pp. 1001–1005, 2016.

Defazio, A., Bach, F., and Lacoste-Julien, S. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1646–1654, 2014.

Di Lorenzo, P. and Scutari, G. NEXT: In-network non-convex optimization. *IEEE Transactions on Signal and Information Processing over Networks*, 2(2):120–136, 2016.

Fang, C., Li, C. J., Lin, Z., and Zhang, T. SPIDER: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 689–699, 2018.

Ghadimi, S. and Lan, G. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

Hendrikx, H., Bach, F., and Massoulie, L. An accelerated decentralized stochastic proximal algorithm for finite sums. *arXiv preprint arXiv:1905.11394*, 2019.

Hong, M., Luo, Z.-Q., and Razaviyayn, M. Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. *SIAM Journal on Optimization*, 26(1):337–364, 2016.

Hong, M., Hajinezhad, D., and Zhao, M.-M. Prox-PDA: The proximal primal-dual algorithm for fast distributed nonconvex optimization and learning over networks. In *Proceedings of International Conference on Machine Learning (ICML)*, pp. 1529–1538, 2017.

Jiang, Z., Balu, A., Hegde, C., and Sarkar, S. Collaborative deep learning in fixed topology networks. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 5904–5914, 2017.

Johnson, R. and Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 315–323, 2013.

Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Lei, L., Ju, C., Chen, J., and Jordan, M. I. Non-convex finite-sum optimization via SCSG methods. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2348–2358, 2017.

Li, B., Cen, S., Chen, Y., and Chi, Y. Communication-efficient distributed optimization in networks with gradient tracking. *arXiv preprint arXiv:1909.05844*, 2019.

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

Lu, S. and Wu, C. W. Decentralized stochastic non-convex optimization over weakly connected time-varying digraphs. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5770–5774, 2020.

Lu, S., Zhang, X., Sun, H., and Hong, M. GNSD: a gradient-tracking based nonconvex stochastic algorithm for decentralized optimization. In *Proceedings of IEEE Data Science Workshop (DSW)*, pp. 315–321, June 2019.

Mokhtari, A. and Ribeiro, A. DSA: Decentralized double stochastic averaging gradient algorithm. *The Journal of Machine Learning Research*, 17(1):2165–2199, 2016.

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

Nedić, A., Olshevsky, A., and Rabbat, M. G. Network topology and communication-computation tradeoffs in decentralized optimization. *Proceedings of the IEEE*, 106 (5):953–976, 2018.

Nesterov, Y. Introductory lectures on convex programming volume i: Basic course. *Lecture notes*, 3(4):5, 1998.

Nguyen, L. M., Liu, J., Scheinberg, K., and Takáč, M. SARAH: A novel method for machine learning problems using stochastic recursive gradient. In *Proceedings of International Conference on Machine Learning (ICML)*, pp. 2613–2621, 2017.

Nguyen, L. M., van Dijk, M., Phan, D. T., Nguyen, P. H., Weng, T.-W., and Kalagnanam, J. R. Optimal finite-sum smooth non-convex optimization with sarah. *arXiv preprint arXiv:1901.07648*, 2019.

Pu, S. and Nedić, A. A distributed stochastic gradient tracking method. In *Proceedings of the Conference on Decision and Control (CDC)*, pp. 963–968, 2018.

Reddi, S. J., Hefny, A., Sra, S., Poczos, B., and Smola, A. Stochastic variance reduction for nonconvex optimization. In *Proceedings of International Conference on Machine Learning (ICML)*, pp. 314–323, 2016.

Shen, Z., Mokhtari, A., Zhou, T., Zhao, P., and Qian, H. Towards more efficient stochastic decentralized learning: Faster convergence and sparse communication. *arXiv preprint arXiv:1805.09969*, 2018.

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

Sun, H. and Hong, M. Distributed non-convex first-order optimization and information processing: Lower complexity bounds and rate optimal algorithms. *IEEE Transactions on Signal Processing*, 67(22):5912–5928, 2019.

Sun, Y., Daneshmand, A., and Scutari, G. Convergence rate of distributed optimization algorithms based on gradient tracking. *arXiv preprint arXiv:1905.02637*, 2019.

Tang, H., Lian, X., Yan, M., Zhang, C., and Liu, J. D^2: Decentralized training over decentralized data. In *Proceedings of International Conference on Machine Learning (ICML)*, pp. 4855–4863, 2018.

Wang, Z. and Li, H. Edge-based stochastic gradient algorithm for distributed optimization. *IEEE Transactions on Network Science and Engineering*, 2019.

Wang, Z., Ji, K., Zhou, Y., Liang, Y., and Tarokh, V. Spiderboost and momentum: Faster variance reduction algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2403–2413. 2019.

Xiao, L. and Boyd, S. Fast linear iterations for distributed averaging. *Systems and Control Letters*, 53(1):65–78, 2004.

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

Yuan, K., Ling, Q., and Yin, W. On the convergence of decentralized gradient descent. *SIAM Journal on Optimization*, 26(3):1835–1854, 2016.

Yuan, K., Ying, B., Liu, J., and Sayed, A. H. Variance-reduced stochastic learning by networked agents under random reshuffling. *IEEE Transactions on Signal Processing*, 67(2):351–366, 2018.

Zeng, J. and Yin, W. On nonconvex decentralized gradient descent. *IEEE Transactions on Signal Processing*, 66 (11):2834–2848, 2018.

Zhou, D., Xu, P., and Gu, Q. Stochastic nested variance reduced gradient descent for nonconvex optimization. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 3921–3932, 2018.