DESTRESS: Computation-Optimal and Communication-Efficient Decentralized Nonconvex Finite-Sum Optimization

Boyue Li BOYUEL@ANDREW.CMU.EDU
Carnegie Mellon University, USA

Zhize Li ZHIZE.LI@KAUST.EDU.SA
King Abdullah University of Science and Technology, Saudi Arabia

Yuejie Chi YUEJIEC@ANDREW.CMU.EDU
Carnegie Mellon University, USA

Abstract

Emerging applications in multi-agent environments call for decentralized algorithms for finite-sum optimizations that are resource-efficient in terms of both computation and communication. In this paper, we consider the setting that agents work collaboratively to minimize the sum of local loss functions by only communicating with their neighbors over a predetermined network topology. We develop a new algorithm, called DEcentralized STochastic REcurSive gradient methodS (DESTRESS) for nonconvex finite-sum optimization, which leverages several key algorithm design ideas including randomly activated stochastic recursive gradient updates with mini-batches and gradient tracking with extra mixing. DESTRESS matches the optimal incremental first-order oracle (IFO) complexity of state-of-the-art centralized algorithms for finding first-order stationary points, and significantly improves over existing decentralized algorithms. The communication complexity of DESTRESS also improves upon prior arts over a wide range of parameter regimes.

1. Introduction

The proliferation of multi-agent environments in emerging applications leads to a growing need of developing decentralized algorithms for optimizing finite-sum problems. Specifically, the goal is to minimize the global objective function $f(\boldsymbol{x}) := \frac{1}{N}\sum_{\boldsymbol{z}\in\mathcal{M}} \ell(\boldsymbol{x};\boldsymbol{z})$, where $\boldsymbol{x} \in \mathbb{R}^d$ denotes the parameter of interest, $\ell(\boldsymbol{x};\boldsymbol{z})$ denotes the sample loss of the sample $\boldsymbol{z}$, $\mathcal{M}$ denotes the entire dataset, and $N = |\mathcal{M}|$ denotes the number of data samples in the entire dataset. Of particular interest of this paper is the nonconvex setting, where $\ell(\boldsymbol{x};\boldsymbol{z})$ is nonconvex with respect to $\boldsymbol{x}$.Assuming the data are distributed equally among all agents,[1] each agent thus possesses $m := N/n$ samples, and $f(\boldsymbol{x})$ can be rewritten as $f(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^n f_i(\boldsymbol{x})$, where $f_i(\boldsymbol{x}) := \frac{1}{m}\sum_{\boldsymbol{z}\in\mathcal{M}_i} \ell(\boldsymbol{x};\boldsymbol{z})$ denotes the local objective function averaged over the local dataset $\mathcal{M}_i$ at the ith agent ($1 \le i \le n$) and $\mathcal{M} = \cup_{i=1}^n \mathcal{M}_i$. The communication pattern of the agents is specified via an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of all agents, and two agents can exchange information if and only if there is an edge in $\mathcal{E}$ connecting them.

The resource efficiency of a decentralized algorithm can often be measured in terms of its computation complexity and communication complexity. Achieving a desired level of resource effi-

1. It is straightforward to generalize to the unequal splitting case with a proper reweighting.

	Setting	Per-agent IFO Complexity	Communication Rounds
SVRG [1, 24]	centralized	$N + \frac{N^{2/3}L}{\epsilon}$	n/a
SCSG [12]	centralized	$N + \frac{N^{2/3}L}{\epsilon}$	n/a
SARAH/Spider/SpiderBoost [9, 20, 27]	centralized	$N + \frac{N^{1/2}L}{\epsilon}$	n/a
D-SARAH [6]	server/client	$m + \frac{mL}{\epsilon}$	$1 + \frac{L}{\epsilon}$
D-GET [25]	decentralized	$m + \frac{1}{(1-\alpha)^2} \cdot \frac{m^{1/2}L}{\epsilon}$	Same as IFO
GT-SARAH [31]	decentralized	$m + \max\left(\frac{1}{(1-\alpha)^2}, \left(\frac{m}{n}\right)^{1/2}, \frac{(m/n+1)^{1/3}}{1-\alpha}\right) \cdot \frac{L}{\epsilon}$	Same as IFO
DESTRESS (this paper)	decentralized	$m + \frac{(m/n)^{1/2}L}{\epsilon}$	$\frac{1}{(1-\alpha)^{1/2}} \cdot \left((mn)^{1/2} + \frac{L}{\epsilon}\right)$

Table 1: The per-agent IFO complexities and communication complexities to find ϵ-approximate first-order stationary points by stochastic variance-reduced algorithms for nonconvex finite-sum problems. SVRG, SCSG, SARAH, Spider and SpiderBoost are designed for the centralized setting, D-SARAH for the server/client setting, and D-GET, GT-SARAH and DESTRESS for the decentralized setting. The big-O notation and logarithmic terms are omitted for simplicity.

ciency for a decentralized algorithm often requires careful and delicate trade-offs between computation and communication, as these objectives are often conflicting in nature.

1.1. Our contributions

We propose DEcentralized STochastic RECurSive gradient methodS (DESTRESS), which provably finds first-order stationary points of the global objective function $f(\boldsymbol{x})$ with a optimal incremental first-order (IFO) oracle complexity, i.e. the complexity of evaluating sample gradients, matching state-of-the-art centralized algorithms, but at a much lower communication complexity compared to existing decentralized algorithms over a wide range of parameter regimes.

To save local computation, DESTRESS harnesses the finite-sum structure of the empirical risk function by performing stochastic variance-reduced recursive gradient updates at each agent [9, 20, 27] in a randomly activated manner. To save communication, DESTRESS employs gradient tracking [34] with a few mixing rounds per iteration, which helps accelerate the convergence through better information sharing [13]; the extra mixing scheme can be implemented using Chebyshev acceleration [2] to further improve the communication efficiency. In a nutshell, to find an ϵ-approximate first-order stationary points, DESTRESS requires: $O\big(m + (m/n)^{1/2}L/\epsilon\big)$ per-agent IFO calls, which is *network-independent*; and $O\Big(\frac{\log\left((n/m)^{1/2}+1\right)}{(1-\alpha)^{1/2}} \cdot \big((mn)^{1/2} + L/\epsilon\big)\Big)$ rounds of communication, where L is the smoothness parameter of the sample loss, $\alpha \in [0, 1)$ is the mixing rate of the network topology, n is the number of agents, and $m = N/n$ is the local sample size.

Comparisons with existing algorithms. Table 1 summarizes the convergence guarantees of representative stochastic variance-reduced algorithms for finding first-order stationary points across centralized, server/client and decentralized communication settings.

	E-R graph	Path graph	2-D Torus graph
$\frac{1}{(1-\alpha)}$	1	n^2	n
D-GET [25]	$m+\frac{m^{1/2}L}{\epsilon}$	$m+\frac{m^{1/2}n^4L}{\epsilon}$	$m+\frac{m^{1/2}n^2L}{\epsilon}$
GT-SARAH [31]	$m+\max\left((\frac{m}{n})^{1/2},(\frac{m}{n}+1)^{1/3}\right)\cdot\frac{L}{\epsilon}$	$m+\max\left(n^4,(\frac{m}{n})^{1/2},(\frac{m}{n}+1)^{1/3}n^2\right)\cdot\frac{L}{\epsilon}$	$m+\max\left(n^2,(\frac{m}{n})^{1/2},(\frac{m}{n}+1)^{1/3}n\right)\cdot\frac{L}{\epsilon}$
DESTRESS (this paper)	$(mn)^{1/2}+\frac{L}{\epsilon}$	$m^{1/2}n^{3/2}+\frac{nL}{\epsilon}$	$m^{1/2}n+\frac{n^{1/2}L}{\epsilon}$
Favorable range of parameters	$m\gtrsim n$ or $\epsilon\lesssim\frac{m^{1/2}L}{n^{3/2}}$	$m\gtrsim n^3$ or $\epsilon\lesssim\frac{n^{5/2}L}{m^{1/2}}$	$m\gtrsim n^2$ or $\epsilon\lesssim\frac{nL}{m^{1/2}}$

Table 2: Detailed comparisons of the communication complexities of D-GET, GT-SARAH and DESTRESS under three graph topologies, where the last row delineates the parameter ranges when DESTRESS is favorable. The complexities are simplified by plugging the bound on $1/(1-\alpha)$ from [18, Proposition 5]. The big-O notations and logarithmic terms are omitted for simplicity.

In terms of the computation complexity, the overall IFO complexity of DESTRESS—when summed over all agents—becomes $n\cdot O\big(m+(m/n)^{1/2}L/\epsilon\big)=O\big(mn+(mn)^{1/2}L/\epsilon\big)=O\big(N+N^{1/2}L/\epsilon\big)$, which matches the optimal IFO complexity of centralized algorithms and distributed server/client algorithms.

When it comes to the communication complexity, it is observed that the communication rounds of DESTRESS can be decomposed into the sum of an ϵ-independent term and an ϵ-dependent term:

$$\underbrace{\frac{1}{(1-\alpha)^{1/2}}\cdot(mn)^{1/2}}_{\epsilon-\text{independent}}+\underbrace{\frac{1}{(1-\alpha)^{1/2}}\cdot\frac{L}{\epsilon}}_{\epsilon-\text{dependent}};$$

similar decompositions also apply to competing decentralized algorithms. DESTRESS significantly improves the ϵ-dependent term of D-GET and GT-SARAH by at least a factor of $\frac{1}{(1-\alpha)^{3/2}}$.Further, the ϵ-independent term of DESTRESS is also smaller than that of D-GET/GT-SARAH as long as the local sample size is sufficient large, i.e. $m=O\big(\frac{n}{1-\alpha}\big)$.

To gain further insights in terms of the communication savings of DESTRESS, Table 2 compares the communication complexities of decentralized algorithms for finding first-order stationary points under some common network settings.It is clearly seen that the communication complexity of DESTRESS dominants the other two algorithms under a wide range of parameter regimes.

1.2. Additional related works

Decentralized optimization and learning have been studied extensively, with contemporary emphasis on the capabilities to scale gracefully to large-scale problems — both in terms of the size of the data and the size of the network. For the conciseness of the paper, we focus our discussions on the most relevant literature and refer interested readers to recent overviews [21, 29, 32] for further references.

Stochastic recursive gradient methods. Stochastic recursive gradients methods [9, 20, 27] achieve a optimal IFO complexity in the centralized setting for nonconvex finite-sum optimization.Many variants have been proposed for finite-sum optimization for finding first-order stationary points, including but not limited to SARAH [19, 20], Spider [9], Spiderboost [27] and SSRGD [14]. Several

algorithms, including SARAH, Spider, Spiderboost and SSRGD, adopt stochastic recursive gradients to improve the IFO complexity to $O(N + N^{1/2}L/\epsilon)$, which is optimal.

Decentralized stochastic non-convex optimization. There has been a flurry of recent activities in decentralized nonconvex optimization in both the server/client setting and the network setting. D-SARAH [6] extends SARAH to the server/client setting with a slightly worse IFO complexity and a sample-independent communication complexity. D-PSGD [15] and SGP [3] extend stochastic gradient descent (SGD) to solve the nonconvex decentralized expectation minimization problems with sub-optimal rates. D^2 [26] introduces a variance-reduced correction term to D-PSGD, which allows a constant step size and hence reaches a better convergence rate.

Gradient tracking Dynamic average consensus [34] proves to be extremely effective to track the dynamic average of local variables over the course of iterative algorithms, and has been applied to extend many central algorithms to decentralized settings, e.g. [8, 13, 17, 23]. This idea, also known as "gradient tracking" [23, 34], essentially adds a correction term to the naive information mixing.Gradient tracking provides a systematic approach to estimate the global gradient at each agent, which allows one to easily design decentralized optimization algorithms based on existing centralized algorithms. This idea is applied in [13, 25, 30, 31, 33].

Extra mixing Performing multiple mixing steps [4, 5, 10, 11, 13, 22] between local updates can greatly improve the dependence of the network in convergence rates, which is equivalent of communicating over a better-connected communication graph for the agents, which in turn leads to a faster convergence (and a better overall efficiency) due to better information mixing. Our algorithm also adopts the extra mixing steps, which leads to better IFO complexity and communication complexity.

Notations Define the stacked vector $\boldsymbol{x} \in \mathbb{R}^{nd}$ and its average over all agents $\overline{\boldsymbol{x}} \in \mathbb{R}^d$ as $\boldsymbol{x} := \left[\boldsymbol{x}_1^\top, \cdots, \boldsymbol{x}_n^\top\right]^\top$, $\quad \overline{\boldsymbol{x}} = \frac{1}{n}\sum_{i=1}^n \boldsymbol{x}_i$. The vectors $\boldsymbol{s}$, $\overline{\boldsymbol{s}}$, $\boldsymbol{u}$, $\overline{\boldsymbol{u}}$, $\boldsymbol{v}$ and $\overline{\boldsymbol{v}}$ are defined in the same fashion. In addition, for a stacked vector $\boldsymbol{x} \in \mathbb{R}^{nd}$, we introduce the distributed gradient $\nabla F(\boldsymbol{x}) \in \mathbb{R}^{nd}$ as $\nabla F(\boldsymbol{x}) := [\nabla f_1(\boldsymbol{x}_1)^\top, \cdots, \nabla f_n(\boldsymbol{x}_n)^\top]^\top$.

2. Preliminaries and Proposed Algorithm

Mixing. The information mixing between agents is conducted by updating the local information via a weighted sum of information from neighbors, which is characterized by a mixing (gossiping) matrix. Concerning this matrix is an important quantity called the mixing rate, defined in Theorem 1.

Definition 1 (Mixing matrix and mixing rate) *The* mixing matrix *is a matrix* $\boldsymbol{W} = [w_{ij}] \in \mathbb{R}^{n\times n}$*, such that* $w_{ij} = 0$ *if agent* i *and* j *are not connected according to the communication graph* $\mathcal{G}$*. Furthermore,* $\boldsymbol{W}\mathbf{1}_n = \mathbf{1}_n$ *and* $\boldsymbol{W}^\top\mathbf{1}_n = \mathbf{1}_n$*. The* mixing rate *of a mixing matrix* $\boldsymbol{W}$ *is defined as* $\alpha := \left\|\boldsymbol{W} - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\right\|_{\mathsf{op}}$.

The mixing rate indicates the speed of information shared across the network. For example, for a fully-connected network, choosing $\boldsymbol{W} = \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top$ leads to $\alpha = 0$. For general networks and mixing matrices, [18, Proposition 5] provides comprehensive bounds on $1/(1-\alpha)$ for various graphs.

2.1. The DESTRESS Algorithm

Detailed in Algorithm 1, we propose a novel decentralized stochastic optimization algorithm, dubbed DESTRESS, for finding first-order order stationary points of nonconvex finite-sum problems. Mo-

tivated by stochastic recursive gradient methods in the centralized setting, DESTRESS has a nested loop structure: 1) The outer loop adopts dynamic average consensus to estimate and track the global gradient $\nabla F(\boldsymbol{x}^{(t)})$ at each agent in (1), where $\boldsymbol{x}^{(t)}$ is the stacked parameters.2) The inner loop refines the parameter estimate $\boldsymbol{u}^{(t),0} = \boldsymbol{x}^{(t)}$ by by performing randomly activated stochastic recursive gradient updates in (2), where the stochastic recursive gradient $\boldsymbol{g}^{(t),s}$ is updated in (2b) via sampling mini-batches from activated agents' local datasets. Inspired by [13], we allow DESTRESS to perform a few rounds of mixing whenever communication takes place, to enable better information sharing and faster convergence. The extra mixing steps can be implemented by Chebyshev acceleration [2] with improved communication efficiency.

Algorithm 1 DESTRESS for decentralized nonconvex finite-sum optimization

input: initial parameter $\overline{\boldsymbol{x}}^{(0)}$, step size η, activation probability p, number of outer loops T, number of inner loops S and number of communication steps K_{in} and K_{out}.
initialization: set $\boldsymbol{x}_i^{(0)} = \overline{\boldsymbol{x}}^{(0)}$ and $\boldsymbol{s}_i^{(0)} = \nabla f(\overline{\boldsymbol{x}}^{(0)})$ for all agents $1 \le i \le n$.
for $t = 1, \ldots, T$ **do**
  Set the new parameter estimate $\boldsymbol{x}^{(t)} = \boldsymbol{u}^{(t-1),S}$.
  Update the global gradient estimate by aggregated local information and gradient tracking:

$$\boldsymbol{s}^{(t)} = (\boldsymbol{W}_{\mathsf{out}} \otimes \boldsymbol{I}_d)\Big(\boldsymbol{s}^{(t-1)} + \nabla F\big(\boldsymbol{x}^{(t)}\big) - \nabla F\big(\boldsymbol{x}^{(t-1)}\big)\Big) \tag{1}$$

  Set $\boldsymbol{u}^{(t),0} = \boldsymbol{x}^{(t)}$ and $\boldsymbol{v}^{(t),0} = \boldsymbol{s}^{(t)}$.
  for $s = 1, ..., S$ **do**
    Each agent i samples a mini-batch $\mathcal{Z}_i^{(t),s}$ of size b from $\mathcal{M}_i$ uniformly at random, sample $\lambda_i^{(t),s} \sim \mathcal{B}(p)$ where $\mathcal{B}(p)$ denotes the Bernoulli distribution with success probability p,[2] and then performs the following updates:

$$\boldsymbol{u}^{(t),s} = (\boldsymbol{W}_{\mathsf{in}} \otimes \boldsymbol{I}_d)(\boldsymbol{u}^{(t),s-1} - \eta \boldsymbol{v}^{(t),s-1}), \tag{2a}$$

$$\boldsymbol{g}_i^{(t),s} = \frac{\lambda_i^{(t),s}}{pb} \sum_{\boldsymbol{z}_i \in \mathcal{Z}_i^{(t),s}} \Big(\nabla \ell(\boldsymbol{u}_i^{(t),s}; \boldsymbol{z}_i) - \nabla \ell(\boldsymbol{u}_i^{(t),s-1}; \boldsymbol{z}_i)\Big) + \boldsymbol{v}_i^{(t),s-1}, \tag{2b}$$

$$\boldsymbol{v}^{(t),s} = (\boldsymbol{W}_{\mathsf{in}} \otimes \boldsymbol{I}_d)\boldsymbol{g}^{(t),s}. \tag{2c}$$

    end for
  end for
  output: $\boldsymbol{x}^{\mathsf{output}} \sim \mathrm{Uniform}(\{\boldsymbol{u}_i^{(t),s-1} | i \in [n], t \in [T], s \in [S]\})$.

3. Performance Guarantees

This section presents the performance guarantees of DESTRESS for finding first-order stationary points of the global objective function $f(\cdot)$.

We first introduce Assumption 1 and Assumption 2, which are standard assumptions imposed on the loss function. Assumption 1 implies that all local objective functions $f_i(\cdot)$ and the global

2. In practice, the stochastic gradients will not be computed if $\lambda_i^{(t),s} = 0$.

objective function $f(\cdot)$ also have Lipschitz gradients, and Assumption 2 guarantees there's no-trivial solutions.

Assumption 1 (Lipschitz gradient) *The sample loss function $\ell(\boldsymbol{x};\boldsymbol{z})$ has L-Lipschitz gradients for all $\boldsymbol{z}\in\mathcal{M}$ and $\boldsymbol{x}\in\mathbb{R}^d$, namely, $\left\|\nabla\ell(\boldsymbol{x};\boldsymbol{z})-\nabla\ell(\boldsymbol{x}';\boldsymbol{z})\right\|_2\leq L\|\boldsymbol{x}-\boldsymbol{x}'\|_2$, $\forall\boldsymbol{x},\boldsymbol{x}'\in\mathbb{R}^d$ and $\boldsymbol{z}\in\mathcal{M}$.*

Assumption 2 (Function boundedness) *The global objective function $f(\cdot)$ is bounded below, i.e., $f^*=\inf_{\boldsymbol{x}\in\mathbb{R}^d}f(\boldsymbol{x})>-\infty$.*

Due to the nonconvexity, first-order algorithms are generally guaranteed to converge to only first-order stationary points of the global loss function $f(\cdot)$, defined below in Theorem 2.

Definition 2 (First-order stationary point) *A point $\boldsymbol{x}\in\mathbb{R}^d$ is called an ϵ-approximate first-order stationary point of a differentiable function $f(\cdot)$ if $\|\nabla f(\boldsymbol{x})\|_2^2\leq\epsilon$.*

3.1. Main theorem

Theorem 3 shows that DESTRESS converges in expectation to an approximate first-order stationary point with an optimal per-agent IFO complexity under specific parameter choices.

Theorem 3 (Complexity for finding first-order stationary points) *Assume Assumption 1 and 2 hold. Set $S=\left\lceil\sqrt{mn}\right\rceil$, $b=\left\lceil\sqrt{m/n}\right\rceil$, $p=\frac{\sqrt{m/n}}{\left\lceil\sqrt{m/n}\right\rceil}$, $K_{\mathsf{out}}=\left\lceil\frac{\log(\sqrt{npb}+1)}{(1-\alpha)^{1/2}}\right\rceil$, $K_{\mathsf{in}}=\left\lceil\frac{\log(2/p)}{(1-\alpha)^{1/2}}\right\rceil$ and $\eta=\frac{1}{640L}$, implement the mixing steps using Chebyshev's acceleration [2], to reach an ϵ-approximate first-order stationary point, in expectation, DESTRESS takes $O\Big(m+\frac{(m/n)^{1/2}L}{\epsilon}\Big)$ IFO calls per agent, and $O\Big(\frac{\log\left((n/m)^{1/2}+1\right)}{(1-\alpha)^{1/2}}\cdot\left((mn)^{1/2}+\frac{L}{\epsilon}\right)\Big)$ rounds of communication.*

As elaborated in Section 1.1, DESTRESS achieves a network-independent IFO complexity that matches the optimal complexity in the centralized setting. When the accuracy $\epsilon\lesssim L/(mn)^{1/2}$, DESTRESS reaches a sample-independent communication complexity of $O\big(\frac{1}{(1-\alpha)^{1/2}}\cdot\frac{L}{\epsilon}\big)$.

It is worthwhile to further highlight the role of the random activation probability p in achieving the optimal IFO by allowing "fractional" batch size.When the local sample size is large ($m\geq n$), $b\approx\sqrt{m/n}$ and $p\approx 1$.However, when the number of agents is large ($n>m$), the batch size $b=1$ and $p=\sqrt{m/n}<1$, which reduces potential computation waste if we naively set $p=1$. Therefore, by introducing random activation, we can view $pb=\sqrt{m/n}$ as the effective batch size at each agent, which allows fractional values and leads to the optimal IFO complexity in all scenarios.

4. Conclusions

In this paper, we proposed DESTRESS for decentralized nonconvex finite-sum optimization, where both its theoretical convergence guarantees and empirical performances on real-world datasets were presented. In sum, DESTRESS matches the optimal IFO complexity of centralized SARAH for finding first-order stationary points, and improves both computation and communication complexities for a broad range of parameters regimes compared with existing approaches. A natural and important extension of this paper is to generalize and develop convergence guarantees of DESTRESS for finding second-order stationary points, which we leave to future works.

References

[1] Zeyuan Allen-Zhu and Elad Hazan. Variance reduction for faster non-convex optimization. In *International conference on machine learning*, pages 699–707. PMLR, 2016.

[2] Mario Arioli and Jennifer Scott. Chebyshev acceleration of iterative refinement. *Numerical Algorithms*, 66(3):591–608, 2014.

[3] Mahmoud Assran, Nicolas Loizou, Nicolas Ballas, and Mike Rabbat. Stochastic gradient push for distributed deep learning. In *International Conference on Machine Learning*, pages 344–353. PMLR, 2019.

[4] Albert S Berahas, Raghu Bollapragada, Nitish Shirish Keskar, and Ermin Wei. Balancing communication and computation in distributed optimization. *IEEE Transactions on Automatic Control*, 64(8):3141–3155, 2019. doi: 10.1109/TAC.2018.2880407.

[5] Albert S Berahas, Raghu Bollapragada, and Ermin Wei. On the convergence of nested decentralized gradient methods with multiple consensus and gradient steps. *IEEE Transactions on Signal Processing*, 2021.

[6] Shicong Cen, Huishuai Zhang, Yuejie Chi, Wei Chen, and Tie-Yan Liu. Convergence of distributed stochastic variance reduced methods without sampling extra data. *IEEE Transactions on Signal Processing*, 68:3976–3989, 2020.

[7] Li Deng. The MNIST database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[8] Paolo Di Lorenzo and Gesualdo Scutari. Next: In-network nonconvex optimization. *IEEE Transactions on Signal and Information Processing over Networks*, 2(2):120–136, 2016.

[9] Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. SPIDER: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In *Advances in Neural Information Processing Systems*, pages 687–697, 2018.

[10] Abolfazl Hashemi, Anish Acharya, Rudrajit Das, Haris Vikalo, Sujay Sanghavi, and Inderjit Dhillon. On the benefits of multiple gossip steps in communication-constrained decentralized optimization. *arXiv preprint arXiv:2011.10643*, 2020.

[11] Charikleia Iakovidou and Ermin Wei. S-NEAR-DGD: A flexible distributed stochastic gradient method for inexact communication. *arXiv preprint arXiv:2102.00121*, 2021.

[12] Lihua Lei, Cheng Ju, Jianbo Chen, and Michael I Jordan. Non-convex finite-sum optimization via SCSG methods. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[13] Boyue Li, Shicong Cen, Yuxin Chen, and Yuejie Chi. Communication-efficient distributed optimization in networks with gradient tracking and variance reduction. *Journal of Machine Learning Research*, 21(180):1–51, 2020.

[14] Zhize Li. SSRGD: Simple stochastic recursive gradient descent for escaping saddle points. In *Advances in Neural Information Processing Systems*, pages 1523–1533, 2019.

[15] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 5330–5340, 2017.

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

[17] Angelia Nedić, Alex Olshevsky, and Wei Shi. Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM Journal on Optimization*, 27(4):2597–2633, 2017.

[18] Angelia Nedić, Alex Olshevsky, and Michael G Rabbat. Network topology and communication-computation tradeoffs in decentralized optimization. *Proceedings of the IEEE*, 106(5):953–976, 2018.

[19] Lam M Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. SARAH: A novel method for machine learning problems using stochastic recursive gradient. In *International Conference on Machine Learning*, pages 2613–2621, 2017.

[20] Lam M Nguyen, Marten van Dijk, Dzung T Phan, Phuong Ha Nguyen, Tsui-Wei Weng, and Jayant R Kalagnanam. Finite-sum smooth optimization with SARAH. *arXiv preprint arXiv:1901.07648*, 2019.

[21] Matthew Nokleby, Haroon Raja, and Waheed U Bajwa. Scaling-up distributed processing of data streams for machine learning. *Proceedings of the IEEE*, 108(11):1984–2012, 2020.

[22] Taoxing Pan, Jun Liu, and Jie Wang. D-SPIDER-SFO: A decentralized optimization algorithm with faster convergence rate for nonconvex problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1619–1626, 2020.

[23] Guannan Qu and Na Li. Harnessing smoothness to accelerate distributed optimization. *IEEE Transactions on Control of Network Systems*, 5(3):1245–1260, 2018.

[24] Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning*, pages 314–323, 2016.

[25] Haoran Sun, Songtao Lu, and Mingyi Hong. Improving the sample and communication complexity for decentralized non-convex optimization: Joint gradient estimation and tracking. In *International Conference on Machine Learning*, pages 9217–9228. PMLR, 2020.

[26] Hanlin Tang, Xiangru Lian, Ming Yan, Ce Zhang, and Ji Liu. D^2: Decentralized training over decentralized data. In *International Conference on Machine Learning*, pages 4848–4856. PMLR, 2018.

[27] Zhe Wang, Kaiyi Ji, Yi Zhou, Yingbin Liang, and Vahid Tarokh. SpiderBoost and momentum: Faster variance reduction algorithms. In *Advances in Neural Information Processing Systems*, pages 2406–2416, 2019.

[28] Lin Xiao and Stephen Boyd. Fast linear iterations for distributed averaging. *Systems and Control Letters*, 53(1):65–78, 2004. ISSN 01676911. doi: 10.1016/j.sysconle.2004.02.022.

[29] Ran Xin, Soummya Kar, and Usman A Khan. Decentralized stochastic optimization and machine learning: A unified variance-reduction framework for robust performance and fast convergence. *IEEE Signal Processing Magazine*, 37(3):102–113, 2020.

[30] Ran Xin, Usman A Khan, and Soummya Kar. A fast randomized incremental gradient method for decentralized non-convex optimization. *arXiv preprint arXiv:2011.03853*, 2020.

[31] Ran Xin, Usman A Khan, and Soummya Kar. Fast decentralized non-convex finite-sum optimization with recursive variance reduction. *arXiv preprint arXiv:2008.07428*, 2020.

[32] Ran Xin, Shi Pu, Angelia Nedić, and Usman A Khan. A general framework for decentralized optimization with first-order methods. *Proceedings of the IEEE*, 108(11):1869–1889, 2020.

[33] Jiaqi Zhang and Keyou You. Decentralized stochastic gradient tracking for non-convex empirical risk minimization. *arXiv preprint arXiv:1909.02712*, 2019.

[34] Minghui Zhu and Sonia Martínez. Discrete-time dynamic average consensus. *Automatica*, 46 (2):322–329, 2010.

Appendix A. Baseline algorithms

Here, we list two related algorithms, DSGD [15, 16] (cf. Algorithm 2) and D-GET/GT-SARAH [25, 31] (cf. Algorithm 3), which are compared numerically against the proposed DESTRESS algorithm in Section C, for completeness.

Algorithm 2 Decentralized stochastic gradient descent (DSGD)

input: initial parameter $\overline{\boldsymbol{x}}^{(0)}$, step size η, number of outer loops T.
initialization: set $\boldsymbol{x}_i^{(0)} = \overline{\boldsymbol{x}}^{(0)}$.
for $t = 1, \ldots, T$ **do**
  Each agent i samples a data point $\boldsymbol{z}_i^{(t)}$ from $\mathcal{M}_i$ uniformly at random and compute the stochastic gradient:

$$\boldsymbol{g}_i^{(t)} = \nabla \ell(\boldsymbol{u}_i^{(t)}; \boldsymbol{z}_i^{(t)}).$$

  Update via local communication: $\boldsymbol{x}^{(t+1)} = (\boldsymbol{W} \otimes \boldsymbol{I}_d)(\boldsymbol{x}^{(t)} - \eta_t \boldsymbol{g}^{(t)})$.
end for
output: $\boldsymbol{x}^{\mathsf{output}} = \overline{\boldsymbol{x}}^{(T)}$.

Algorithm 3 D-GET/GT-SARAH

input: initial parameter $\overline{\boldsymbol{x}}^{(0)}$, step size η, communication frequency q.
initialization: set $\boldsymbol{v}^{(0)} = \boldsymbol{y}^{(0)} = \nabla F(\boldsymbol{x}^{(0)})$.
for $t = 1, \ldots, T$ **do**
  Update via local communication $\boldsymbol{x}^{(t)} = (\boldsymbol{W} \otimes \boldsymbol{I}_d)\boldsymbol{x}^{(t-1)} - \eta \boldsymbol{y}^{(t-1)}$.
  if $\mod (t, q) = 0$ **then**
    $\boldsymbol{v}^{(t)} = \nabla F(\boldsymbol{x}^{(t)})$.
  else
    Each agent i samples a data point $\mathcal{Z}_i^{(t)}$ from $\mathcal{M}_i$ uniformly at random, and then performs the following updates:

$$\boldsymbol{v}_i^{(t)} = \frac{1}{b} \sum_{\boldsymbol{z}_i \in \mathcal{Z}_i^{(t)}} \left(\nabla \ell(\boldsymbol{x}_i^{(t)}; \boldsymbol{z}_i) - \nabla \ell(\boldsymbol{x}_i^{(t-1)}; \boldsymbol{z}_i) \right) + \boldsymbol{v}_i^{(t-1)}.$$

  end if
  Update via local communication $\boldsymbol{y}^{(t)} = (\boldsymbol{W} \otimes \boldsymbol{I}_d)\boldsymbol{y}^{(t-1)} + \boldsymbol{v}^{(t)} - \boldsymbol{v}^{(t-1)}$.
end for
output: $\boldsymbol{x}^{\mathsf{output}} = \overline{\boldsymbol{x}}^{(T)}$.

Appendix B. Formal theorem statement

We state the main theorem in this section, as an addition to Theorem 3.

Theorem 4 (First-order optimality) *Assume Assumption 1 and 2 holds. Set $p \in (0,1]$, K_{in}, K_{out}, S, b and η to be positive and satisfy*

$$\alpha^{K_{\mathsf{in}}} \leq p \quad \text{and} \quad \eta L \leq \frac{(1-\alpha^{K_{\mathsf{in}}})^3(1-\alpha^{K_{\mathsf{out}}})}{10\big(1+\alpha^{K_{\mathsf{in}}}\alpha^{K_{\mathsf{out}}}\sqrt{npb}\big)\big(\sqrt{S/(npb)}+1\big)} \tag{3}$$

The output produced by Algorithm 1 satisfies

$$\mathbb{E}\big\|\nabla f(\boldsymbol{x}^{\mathsf{output}})\big\|_2^2 < \frac{4}{\eta TS}\Big(\mathbb{E}[f(\overline{\boldsymbol{x}}^{(0)})] - f^*\Big). \tag{4}$$

Appendix C. Numerical Experiments

This section provides numerical experiments on real datasets to evaluate our proposed algorithm DESTRESS with comparisons against two existing baselines: DSGD [15, 16] and GT-SARAH [31]. To allow for reproducibility, all codes can be found at

https://github.com/liboyue/Network-Distributed-Algorithm.

For all experiments, we set the number of agents $n = 20$, and split the dataset uniformly at random to each agent. In addition, since $m \gg n$ in all experiments, we set $p = 1$ for simplicity. We run each experiment on three communication graphs with the same data assignment and starting point: Erdös-Rènyi graph (the connectivity probability is set to 0.3), grid graph, and path graph. The mixing matrices are chosen as the symmetric fastest distributed linear averaging (FDLA) matrices [28] generated according to different graph topologies, and the extra mixing steps are implemented by Chebyshev's acceleration [2] to save communications as described earlier. To ensure convergence, DSGD adopts a diminishing step size schedule. All the parameters are tuned manually for best performance. We defer a detailed account of the baseline algorithms as well as parameter choices in Appendix A.

C.1. Regularized logistic regression

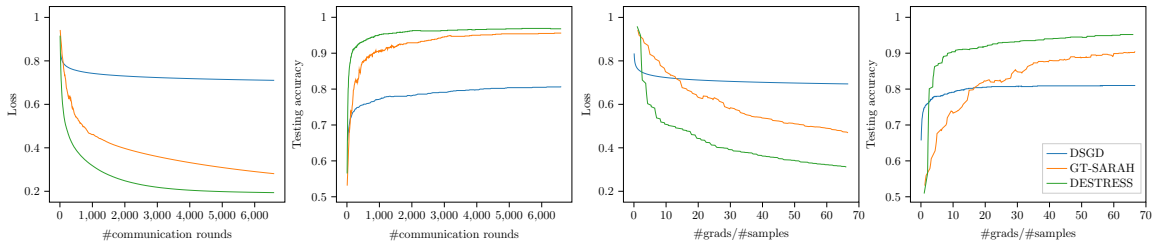
To begin with, we employ logistic regression with nonconvex regularization to solve a binary classification problem using the Gisette dataset.[3] We split the Gisette dataset to $n = 20$ agents, where each agent receives $m = 300$ training samples of dimension $d = 5000$. The sample loss function is given as

$$\ell(\boldsymbol{x}; \{\boldsymbol{f}, l\}) = -l \log\Big(\frac{1}{1+\exp(\boldsymbol{x}^\top \boldsymbol{f})}\Big) + (1-l)\log\Big(\frac{\exp(\boldsymbol{x}^\top \boldsymbol{f})}{1+\exp(\boldsymbol{x}^\top \boldsymbol{f})}\Big) + \lambda \sum_{i=1}^{d} \frac{x_i^2}{1+x_i^2},$$

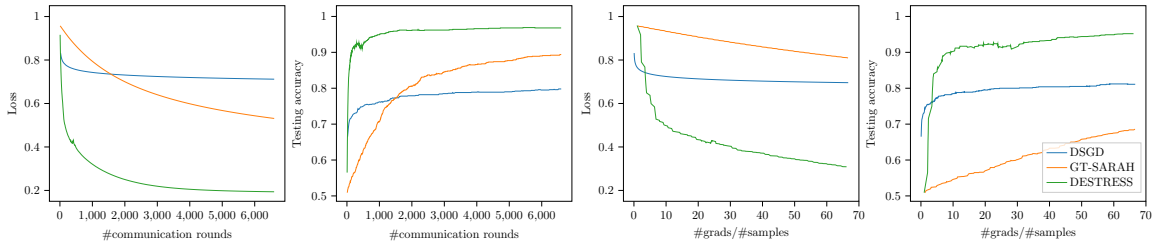
where $\{\boldsymbol{f}, l\}$ represents a training tuple, $\boldsymbol{f} \in \mathbb{R}^d$ is the feature vector and $l \in \{0,1\}$ is the label, and λ is the regularization parameter. For this experiment, we set $\lambda = 0.01$.

Figure 1 shows the loss and testing accuracy for all algorithms. DESTRESS significantly outperforms other algorithms both in terms of communication and computation. It is worth noting that, DSGD converges very fast at the beginning of training, but cannot sustain the progress due to the diminishing schedule of step sizes. On the contrary, the variance-reduced algorithms can

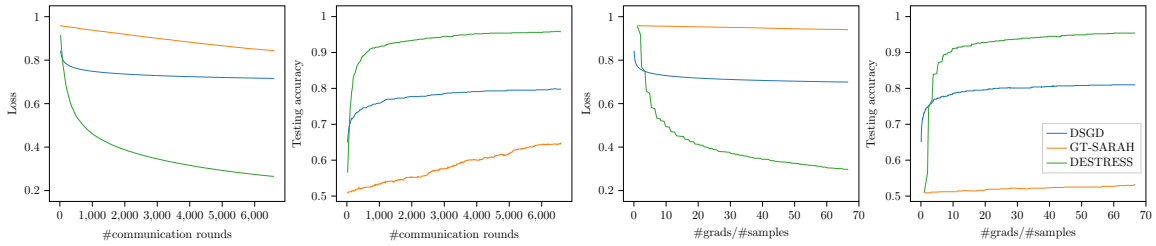
3. The dataset can be accessed at https://archive.ics.uci.edu/ml/datasets/Gisette.



(a) Erdös-Rènyi graph



(b) Grid graph



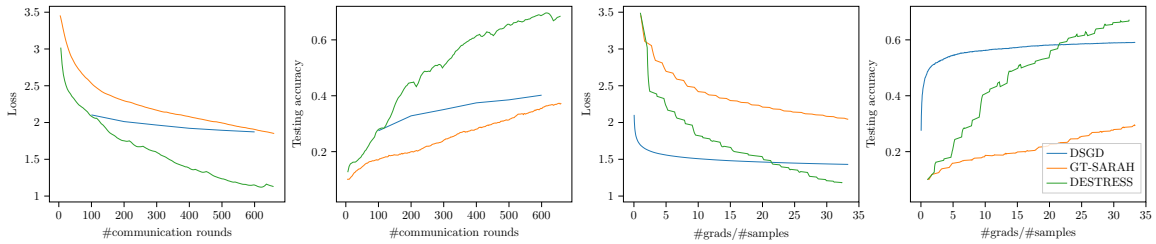
(c) Path graph

Figure 1: The training loss and testing accuracy with respect to the number of communication rounds (left two panels) and gradient evaluations (right two panels) for DSGD, GT-SARAH and DESTRESS when training a regularized logistic regression model on the Gisette dataset. Due to the initial full-gradient computation, the gradient evaluations of DESTRESS and GT-SARAH do not start from 0.

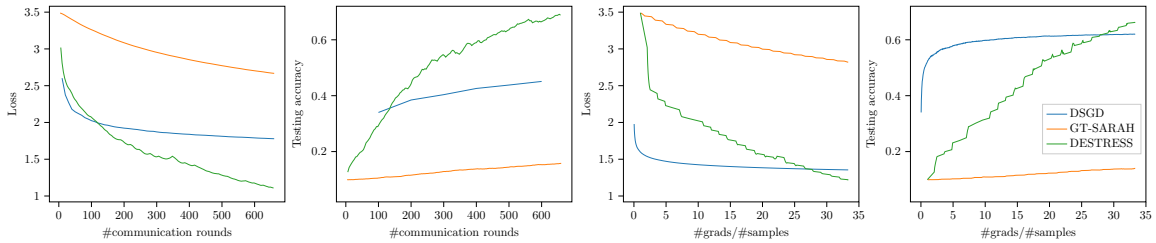
converge with a constant step size, and hence converge better overall. Moreover, due to the refined gradient estimation and information mixing designs, DESTRESS can bear a larger step size than GT-SARAH, which leads to the fastest convergence and best overall performance. In addition, a larger number of extra mixing steps leads to a better performance when the graph topology becomes less connected.

C.2. Neural network training

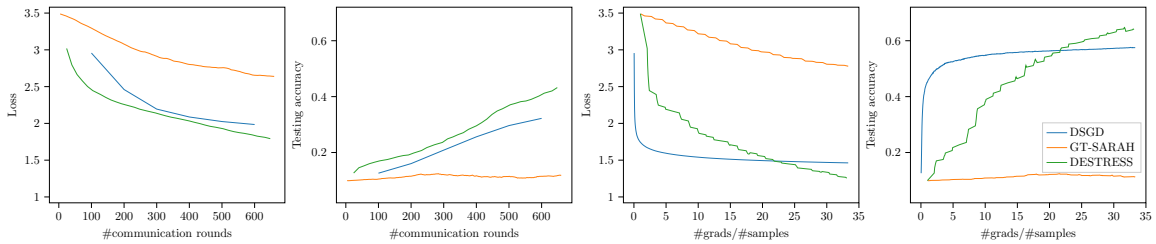
Next, we compare the performance of DESTRESS with comparisons to DSGD and GT-SARAH for training a one-hidden-layer neural network with 64 hidden neurons and sigmoid activations for classifying the MNIST dataset [7]. We evenly split $60,000$ training samples to 20 agents at random. Figure 2 plots the training loss and testing accuracy against the number of communication rounds and gradient evaluations for all algorithms. Again, DESTRESS significantly outperforms other



(a) Erdös-Rènyi graph



(b) Grid graph



(c) Path graph

Figure 2: The training loss and testing accuracy with respect to the number of communication rounds (left two panels) and gradient evaluations (right two panels) for DSGD, GT-SARAH and DESTRESS when training a one-hidden-layer neural network on the MNIST dataset. Due to the initial full-gradient computation, the gradient evaluations of DESTRESS and GT-SARAH do not start from 0.

algorithms in terms of computation and communication costs due to the larger step size and extra mixing, which validates our theoretical analysis.

Appendix D. Proof of Theorem 4

For notation simplicity, let

$$\alpha_{\mathsf{in}} = \alpha^{K_{\mathsf{in}}}, \qquad \alpha_{\mathsf{out}} = \alpha^{K_{\mathsf{out}}}$$

throughout the proof. In addition, with a slight abuse of notation, we define the global gradient $\nabla f(\boldsymbol{x}) \in \mathbb{R}^{nd}$ of an (nd)-dimensional vector $\boldsymbol{x} = \left[\boldsymbol{x}_1^\top, \cdots, \boldsymbol{x}_n^\top\right]^\top$, where $\boldsymbol{x}_i \in \mathbb{R}^d$, as follows

$$\nabla f(\boldsymbol{x}) := [\nabla f(\boldsymbol{x}_1)^\top, \cdots, \nabla f(\boldsymbol{x}_n)^\top]^\top. \tag{5}$$

The following fact is a straightforward consequence of our assumption on the mixing matrix $\boldsymbol{W}$ in Theorem 1.

Fact 1 *Let $\boldsymbol{x} = \left[\boldsymbol{x}_1^\top, \cdots, \boldsymbol{x}_n^\top\right]^\top$, and $\overline{\boldsymbol{x}} = \frac{1}{n}\sum_{i=1}^n \boldsymbol{x}_i$, where $\boldsymbol{x}_i \in \mathbb{R}^d$. For a mixing matrix $\boldsymbol{W} \in \mathbb{R}^{n\times n}$ satisfying Theorem 1, we have*

1. $\left(\frac{1}{n}\mathbf{1}_n^\top \otimes \boldsymbol{I}_d\right)(\boldsymbol{W}\otimes \boldsymbol{I}_d)\boldsymbol{x} = \left(\frac{1}{n}\mathbf{1}_n^\top \otimes \boldsymbol{I}_d\right)\boldsymbol{x} = \overline{\boldsymbol{x}}$;
2. $\left(\boldsymbol{I}_{nd} - (\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top)\otimes \boldsymbol{I}_d\right)(\boldsymbol{W}\otimes \boldsymbol{I}_d) = (\boldsymbol{W}\otimes \boldsymbol{I}_d - (\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top)\otimes \boldsymbol{I}_d)\left(\boldsymbol{I}_{nd} - (\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top)\otimes \boldsymbol{I}_d\right)$.

To begin with, we introduce a key lemma that upper bounds the norm of the gradient of the global loss function evaluated at the average local estimates over n agents, in terms of the function value difference at the beginning and the end of the inner loop, the gradient estimation error, and the norm of gradient estimates.

Lemma 5 (Inner loop induction) *Assume Assumption 1 holds. After $S \geq 1$ inner loops, one has*

$$\sum_{s=0}^{S-1}\|\nabla f(\overline{\boldsymbol{u}}^{(t),s})\|_2^2 \leq \frac{2}{\eta}\Big(f(\overline{\boldsymbol{u}}^{(t),0}) - f(\overline{\boldsymbol{u}}^{(t),S})\Big) + \sum_{s=0}^{S-1}\big\|\nabla f(\overline{\boldsymbol{u}}^{(t),s}) - \overline{\boldsymbol{v}}^{(t),s}\big\|_2^2 - (1-\eta L)\sum_{s=0}^{S-1}\big\|\overline{\boldsymbol{v}}^{(t),s}\big\|_2^2.$$

Proof [Proof of Theorem 5] The local update rule (2a), combined with Lemma 1, yields

$$\overline{\boldsymbol{u}}^{(t),s+1} = \overline{\boldsymbol{u}}^{(t),s} - \eta\overline{\boldsymbol{v}}^{(t),s}.$$

By Assumption 1, we have

$$\begin{aligned} f(\overline{\boldsymbol{u}}^{(t),s+1}) &= f(\overline{\boldsymbol{u}}^{(t),s} - \eta\overline{\boldsymbol{v}}^{(t),s}) \\ &\leq f(\overline{\boldsymbol{u}}^{(t),s}) - \big\langle\nabla f(\overline{\boldsymbol{u}}^{(t),s}), \eta\overline{\boldsymbol{v}}^{(t),s}\big\rangle + \frac{L}{2}\big\|\eta\overline{\boldsymbol{v}}^{(t),s}\big\|_2^2 \\ &= f(\overline{\boldsymbol{u}}^{(t),s}) - \frac{\eta}{2}\big\|\nabla f(\overline{\boldsymbol{u}}^{(t),s})\big\|_2^2 + \frac{\eta}{2}\big\|\nabla f(\overline{\boldsymbol{u}}^{(t),s}) - \overline{\boldsymbol{v}}^{(t),s}\big\|_2^2 - \Big(\frac{\eta}{2} - \frac{\eta^2 L}{2}\Big)\big\|\overline{\boldsymbol{v}}^{(t),s}\big\|_2^2, \end{aligned} \tag{6}$$

where the last equality is obtained by applying $-\langle\boldsymbol{a},\boldsymbol{b}\rangle = \frac{1}{2}\big(\|\boldsymbol{a}-\boldsymbol{b}\|_2^2 - \|\boldsymbol{a}\|_2^2 - \|\boldsymbol{b}\|_2^2\big)$. Summing over $s = 0, \ldots, S-1$ finishes the proof. ■

Because the output $\boldsymbol{x}^{\mathsf{output}}$ is chosen from $\big\{\boldsymbol{u}_i^{(t),s-1} | i \in [n], t \in [T], s \in [S]\big\}$ uniformly at random, we can compute the expectation of the output's gradient as follows:

$$
\begin{aligned}
nTS\mathbb{E}\big\|\nabla f(\boldsymbol{x}^{\mathsf{output}})\big\|_2^2 &= \sum_{i=1}^{n}\sum_{t=1}^{T}\sum_{s=0}^{S-1}\mathbb{E}\big\|\nabla f(\boldsymbol{u}_i^{(t),s})\big\|_2^2 \\
&\overset{\text{(i)}}{=} \sum_{t=1}^{T}\sum_{s=0}^{S-1}\mathbb{E}\big\|\nabla f(\boldsymbol{u}^{(t),s})\big\|_2^2 \\
&= \sum_{t=1}^{T}\sum_{s=0}^{S-1}\mathbb{E}\big\|\nabla f(\boldsymbol{u}^{(t),s}) - \nabla f(\mathbf{1}_n\otimes\overline{\boldsymbol{u}}^{(t),s}) + \nabla f(\mathbf{1}_n\otimes\overline{\boldsymbol{u}}^{(t),s})\big\|_2^2 \\
&\overset{\text{(ii)}}{\le} 2\sum_{t=1}^{T}\sum_{s=0}^{S-1}\Big(\mathbb{E}\big\|\nabla f(\boldsymbol{u}^{(t),s}) - \nabla f(\mathbf{1}_n\otimes\overline{\boldsymbol{u}}^{(t),s})\big\|_2^2 + \mathbb{E}\big\|\nabla f(\mathbf{1}_n\otimes\overline{\boldsymbol{u}}^{(t),s})\big\|_2^2\Big) \\
&\overset{\text{(iii)}}{\le} 2\sum_{t=0}^{T}\sum_{s=0}^{S-1}\Big(L^2\mathbb{E}\big\|\boldsymbol{u}^{(t),s} - \mathbf{1}_n\otimes\overline{\boldsymbol{u}}^{(t),s}\big\|_2^2 + n\mathbb{E}\big\|\nabla f(\overline{\boldsymbol{u}}^{(t),s})\big\|_2^2\Big), \qquad (7)
\end{aligned}
$$

where (i) follows from the change of notation using (5), (ii) follows from the Cauchy-Schwartz inequality, and (iii) follows from Assumption 1 and extending the summation to $t = 0, \ldots T$. Then, in view of Theorem 5, (7) can be further bounded by

$$
\begin{aligned}
nTS\mathbb{E}\big\|\nabla f(\boldsymbol{x}^{\mathsf{output}})\big\|_2^2 \le \frac{4n}{\eta}\Big(\mathbb{E}[f(\overline{\boldsymbol{x}}^{(0)})] - f^*\Big) + 2L^2\sum_{t=0}^{T}\sum_{s=0}^{S-1}\mathbb{E}\big\|\boldsymbol{u}^{(t),s} - \mathbf{1}_n\otimes\overline{\boldsymbol{u}}^{(t),s}\big\|_2^2 \\
+ 2n\sum_{t=0}^{T}\sum_{s=0}^{S-1}\Big(\mathbb{E}\big\|\nabla f(\overline{\boldsymbol{u}}^{(t),s}) - \overline{\boldsymbol{v}}^{(t),s}\big\|_2^2 - (1-\eta L)\mathbb{E}\big\|\overline{\boldsymbol{v}}^{(t),s}\big\|_2^2\Big), \qquad (8)
\end{aligned}
$$

where we use $\overline{\boldsymbol{u}}^{(t),0} = \overline{\boldsymbol{x}}^{(t)}$ and $f(\overline{\boldsymbol{u}}^{(t),S}) \ge f^*$.

Next, we present Theorem 6 and 7 to bound the double sum in (8), whose proofs can be found in **??** and **??**, respectively.

Lemma 6 (Sum of inner loop errors) *Assuming all conditions in Theorem 4 hold. For all $t > 0$, we can bound the summation of inner loop errors as*

$$
\begin{aligned}
&2L^2\sum_{s=0}^{S-1}\mathbb{E}\big\|\boldsymbol{u}^{(t),s} - \mathbf{1}_n\otimes\overline{\boldsymbol{u}}^{(t),s}\big\|_2^2 + 2n\sum_{s=0}^{S-1}\mathbb{E}\big\|\nabla f(\overline{\boldsymbol{u}}^{(t),s}) - \overline{\boldsymbol{v}}^{(t),s}\big\|_2^2 \\
&\qquad\le \frac{64L^2}{1-\alpha_{\mathsf{in}}}\cdot\Big(\frac{S}{npb}+1\Big)\mathbb{E}\big\|\boldsymbol{x}^{(t)} - \mathbf{1}_n\otimes\overline{\boldsymbol{x}}^{(t)}\big\|_2^2 + 2\alpha_{\mathsf{in}}^2\mathbb{E}\big\|\boldsymbol{s}^{(t)} - \mathbf{1}_n\otimes\overline{\boldsymbol{s}}^{(t)}\big\|_2^2 + \frac{2n}{25}\sum_{s=1}^{S}\mathbb{E}\big\|\overline{\boldsymbol{v}}^{(t),s-1}\big\|_2^2.
\end{aligned}
$$

Lemma 7 (Sum of outer loop gradient estimation error and consensus error) *Assuming all conditions in Theorem 4 hold. We have*

$$
\frac{64L^2}{1-\alpha_{\mathsf{in}}}\cdot\Big(\frac{S}{npb}+1\Big)\sum_{t=0}^{T}\mathbb{E}\big\|\boldsymbol{x}^{(t)} - \mathbf{1}_n\otimes\overline{\boldsymbol{x}}^{(t)}\big\|_2^2 + 2\alpha_{\mathsf{in}}^2\sum_{t=0}^{T}\mathbb{E}\big\|\boldsymbol{s}^{(t)} - \mathbf{1}_n\otimes\overline{\boldsymbol{s}}^{(t)}\big\|_2^2 \le \frac{11n}{25}\sum_{t=1}^{T}\sum_{s=0}^{S-1}\mathbb{E}\big\|\overline{\boldsymbol{v}}^{(t),s}\big\|_2^2.
$$

Using Theorem 6, (8) can be bounded as follows:

$$nTS\mathbb{E}\left\|\nabla f(\boldsymbol{x}^{\mathsf{output}})\right\|_2^2 < \frac{4n}{\eta}\Big(\mathbb{E}[f(\overline{\boldsymbol{x}}^{(t),0})] - f^*\Big) - 2n\Big(\frac{24}{25} - \eta L\Big)\sum_{t=0}^{T}\sum_{s=1}^{S}\mathbb{E}\left\|\overline{\boldsymbol{v}}^{(t),s-1}\right\|_2^2 + \frac{64L^2}{1-\alpha_{\mathsf{in}}}\cdot\Big(\frac{S}{npb}+1\Big)\sum_{t=0}^{T}\mathbb{E}\left\|\boldsymbol{x}^{(t)} - \mathbf{1}_n\otimes\overline{\boldsymbol{x}}^{(t)}\right\|_2^2 + 2\alpha_{\mathsf{in}}^2\sum_{t=0}^{T}\mathbb{E}\left\|\boldsymbol{s}^{(t)} - \mathbf{1}_n\otimes\overline{\boldsymbol{s}}^{(t)}\right\|_2^2, \tag{9}$$

where we bound the sum of inner loop errors $L^2\sum_{s=0}^{S-1}\mathbb{E}\left\|\boldsymbol{u}^{(t),s}-\mathbf{1}_n\otimes\overline{\boldsymbol{u}}^{(t),s}\right\|_2^2$ and $n\sum_{s=0}^{S-1}\mathbb{E}\left\|\nabla f(\overline{\boldsymbol{u}}^{(t),s})-\overline{\boldsymbol{v}}^{(t),s}\right\|_2^2$ by the initial value of each inner loop $\mathbb{E}\left\|\boldsymbol{x}^{(t)} - \mathbf{1}_n\otimes\overline{\boldsymbol{x}}^{(t)}\right\|_2^2$ and $\mathbb{E}\left\|\boldsymbol{s}^{(t)} - \mathbf{1}_n\otimes\overline{\boldsymbol{s}}^{(t)}\right\|_2^2$, and the summation of the norm of average inner loop gradient estimator $n\sum_{s=1}^{S}\mathbb{E}\left\|\overline{\boldsymbol{v}}^{(t),s-1}\right\|_2^2$.

By Theorem 7, (9) can be further bounded as

$$\begin{aligned} nTS\mathbb{E}\left\|\nabla f(\boldsymbol{x}^{\mathsf{output}})\right\|_2^2 &\le \frac{4n}{\eta}\Big(\mathbb{E}[f(\overline{\boldsymbol{x}}^{(t),0})] - f^*\Big) - 2n\Big(\frac{37}{50} - \eta L\Big)\sum_{t=1}^{T}\sum_{s=0}^{S-1}\mathbb{E}\left\|\overline{\boldsymbol{v}}^{(t),s}\right\|_2^2 \\ &< \frac{4n}{\eta}\Big(\mathbb{E}[f(\overline{\boldsymbol{x}}^{(t),0})] - f^*\Big), \end{aligned}$$

which concludes the proof.

Appendix E. Proof of Corollary 3

Without loss of generality, we assume $n \ge 2$. Otherwise, the problem reduces to the centralized setting with a single agent $n = 1$, and the bound holds trivially. We will confirm the choice of parameters in Theorem 3 in the following paragraphs, and finally obtain the IFO complexity and communication complexity.

Step size η. We first assume $\alpha_{\mathsf{in}} \le \frac{p}{2} \le \frac{1}{2}$ and $\alpha_{\mathsf{out}} \le \frac{1}{\sqrt{npb}+1} \le \frac{1}{2}$, which will be proved to hold shortly, then we can verify the step size choice meets the requirement in (3) as:

$$\frac{(1-\alpha_{\mathsf{in}})^3(1-\alpha_{\mathsf{out}})}{1+\alpha^{K_{\mathsf{in}}}\alpha^{K_{\mathsf{out}}}\sqrt{pnb}}\cdot\frac{1}{10L\big(\sqrt{S/(npb)}+1\big)} \ge \frac{(1/2)^4}{2}\cdot\frac{1}{20L} = \frac{1}{640L}.$$

Mixing steps K_{in} and K_{out}. Using Chebyshev's acceleration [2] to implement the mixing steps, it amounts to an improved mixing rate of $\alpha_{\mathsf{cheb}} \asymp 1-\sqrt{2(1-\alpha)}$, when the original mixing rate α is close to 1. Set $K_{\mathsf{in}} = \left\lceil\frac{\log(2/p)}{\sqrt{1-\alpha}}\right\rceil$ and $K_{\mathsf{out}} = \left\lceil\frac{\log(\sqrt{npb}+1)}{\sqrt{1-\alpha}}\right\rceil$. We are now positioned to examine the effective mixing rate $\alpha_{\mathsf{in}} = \alpha_{\mathsf{cheb}}^{K_{\mathsf{in}}}$ and $\alpha_{\mathsf{out}} = \alpha_{\mathsf{cheb}}^{K_{\mathsf{out}}}$, as follows

$$\alpha_{\mathsf{out}} = \alpha_{\mathsf{cheb}}^{K_{\mathsf{out}}} \overset{\text{(i)}}{\le} \alpha_{\mathsf{cheb}}^{\frac{\log(\sqrt{npb}+1)}{\sqrt{1-\alpha}}} \asymp \alpha_{\mathsf{cheb}}^{\frac{\sqrt{2}\log(\sqrt{npb}+1)}{1-\alpha_{\mathsf{cheb}}}} \overset{\text{(ii)}}{\le} \alpha_{\mathsf{cheb}}^{\frac{\sqrt{2}\log(\sqrt{npb}+1)}{-\log\alpha_{\mathsf{cheb}}}} < \frac{1}{\sqrt{npb}+1} \overset{\text{(iii)}}{\le} \frac{1}{2},$$

where (i) follows from $K_{\mathsf{out}} = \left\lceil\frac{\log(\sqrt{npb}+1)}{\sqrt{1-\alpha}}\right\rceil$, (ii) follows from $\log x \le x-1$, $\forall x > 0$, and (iii) follows from $n \ge 1$ and $b \ge 1$. By a similar argument, we have $\alpha_{\mathsf{in}} = \alpha_{\mathsf{cheb}}^{K_{\mathsf{in}}} \le \frac{p}{2}$.

Complexity. Plugging in the selected parameters into (4) in Theorem 4, We have

$$\mathbb{E}\big\|\nabla f(\boldsymbol{x}^{\mathsf{output}})\big\|_2^2 \leq \frac{4}{\eta TS}\Big(\mathbb{E}[f(\overline{\boldsymbol{x}}^{(t),0})] - f^*\Big) = O\Big(\frac{L}{T\sqrt{mn}}\Big).$$

Consequently, the outer iteration complexity is $T = O\Big(1 + \frac{L}{(mn)^{1/2}\epsilon}\Big)$. With this in place, we summarize the communication and IFO complexities as follows:

- The communication complexity is $T\cdot(SK_{\mathsf{in}}+K_{\mathsf{out}}) = O\Big(\frac{(mn)^{1/2}\log\big((n/m)^{1/2}+1\big)+\log\big((mn)^{1/4}+1\big)}{\sqrt{1-\alpha}}\cdot\Big(1+\frac{L}{(mn)^{1/2}\epsilon}\Big)\Big) = O\Big(\frac{\log\big((n/m)^{1/2}+1\big)}{\sqrt{1-\alpha}}\cdot\big((mn)^{1/2}+\frac{L}{\epsilon}\big)\Big)$, where we use $2/p = \frac{2\lceil\sqrt{m/n}\rceil}{\sqrt{m/n}} \leq \frac{2(\sqrt{m/n}+1)}{\sqrt{m/n}} = 2(\sqrt{n/m}+1)$ to bound K_{in}.
- The IFO complexity is $T\cdot(Spb+2m) = O\Big(m + \frac{(m/n)^{1/2}L}{\epsilon}\Big)$.