8-2022

DESTRESS: Computation-optimal and communication-efficient decentralized nonconvex finite-sum optimization

Boyue LI

Zhize LI
Singapore Management University, zhizeli@smu.edu.sg

Yuejie CHI

Citation

LI, Boyue; LI, Zhize; and CHI, Yuejie. DESTRESS: Computation-optimal and communication-efficient decentralized nonconvex finite-sum optimization. (2022). *SIAM Journal on Mathematics of Data Science*. 4, (3), 1031-1051.
Available at: https://ink.library.smu.edu.sg/sis_research/8691

SIAM J. MATH. DATA SCI.
Vol. 4, No. 3, pp. 1031–1051

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

Boyue Li†, Zhize Li†, and Yuejie Chi†

Abstract. Emerging applications in multiagent environments such as internet-of-things, networked sensing, autonomous systems, and federated learning, 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 prototypical setting where the 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 matches the optimal incremental first-order oracle complexity of centralized algorithms for finding first-order stationary points, while maintaining communication efficiency. Detailed theoretical and numerical comparisons corroborate that the resource efficiencies of DESTRESS improve upon prior decentralized algorithms over a wide range of parameter regimes. DESTRESS leverages several key algorithm design ideas including stochastic recursive gradient updates with minibatches for local computation, gradient tracking with extra mixing (i.e., multiple gossiping rounds) for periteration communication, together with careful choices of hyperparameters and new analysis frameworks to provably achieve a desirable computation-communication trade-off.

Key words. decentralized optimization, nonconvex finite-sum optimization, stochastic recursive gradient methods

MSC codes. 68Q25, 68Q11, 68T09

DOI. 10.1137/21M1450677

1. Introduction.

The proliferation of multiagent environments in emerging applications such as internet-of-things (IoT), networked sensing, and autonomous systems, together with the necessity of training machine learning models using distributed systems in federated learning, leads to a growing need for developing decentralized algorithms for optimizing finite-sum problems. Specifically, the goal is to minimize the global objective function,

$$\underset{\boldsymbol{x}\in\mathbb{R}^d}{\text{minimize}} \quad f(\boldsymbol{x}) := \frac{1}{N}\sum_{\boldsymbol{z}\in\mathcal{M}} \ell(\boldsymbol{x};\boldsymbol{z}), \tag{1.1}$$

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

*Received by the editors October 5, 2021; accepted for publication (in revised form) May 24, 2022; published electronically August 4, 2022.
https://doi.org/10.1137/21M1450677
Funding: This work is supported in part by ONR N00014-19-1-2404, by AFRL under FA8750-20-2-0504, and by NSF under CCF-1901199 and CCF-2007911. The first author is also gratefully supported by Wei Shen and Xuehong Zhang Presidential Fellowship at Carnegie Mellon University.

†Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA USA (boyuel@andrew.cmu.edu, zhizel@andrew.cmu.edu, yuejiec@andrew.cmu.edu).

entire dataset. Of particular interest to this paper is the nonconvex setting, where $\ell(\boldsymbol{x};\boldsymbol{z})$ is nonconvex with respect to $\boldsymbol{x}$, due to its ubiquity across problems in machine learning and signal processing, including but not limited to nonlinear estimation, neural network training, and so on.

In a prototypical decentralized environment, however, each agent only has access to a disjoint subset of the data samples, and aims to work collaboratively to optimize $f(\boldsymbol{x})$, by only exchanging information with its neighbors over a predetermined network topology. 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. Unlike the server/client setting, the decentralized setting, sometimes also called the network setting, does not admit a parameter server to facilitate global information sharing, therefore it is much more challenging to understand and delineate the impact of the network graph.

Roughly speaking, in a typical decentralized algorithm, the agents alternate between (1) communication, which propagates local information and enforces consensus, and (2) computation, which updates individual parameter estimates and improves convergence using information received from the neighbors. The resource efficiency of a decentralized algorithm can often be measured in terms of its computation complexity and communication complexity. For example, communication can be extremely time consuming and become the top priority when the bandwidth is limited. On the other hand, minimizing computation, especially at resource-constrained agents (e.g., power-hungry IoT or mobile devices), is also critical to ensure the overall efficiency. Achieving a desired level of resource efficiency 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. The central contribution of this paper lies in the development of a new resource-efficient algorithm for nonconvex finite-sum optimization problems in a decentralized environment, dubbed DEcentralized STochastic REcurSive gradient methodS (DESTRESS). DESTRESS provably finds first-order stationary points of the global objective function $f(\boldsymbol{x})$ with the optimal incremental first-order (IFO) oracle complexity, i.e., the complexity of evaluating sample gradients, matching state-of-the-art centralized algorithms, but

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

at a much lower communication complexity compared to existing decentralized algorithms over a wide range of parameter regimes.

To achieve resource efficiency, DESTRESS leverages several key ideas in the algorithm design. To reduce local computation, DESTRESS harnesses the finite-sum structure of the empirical risk function by performing stochastic variance-reduced recursive gradient updates [29, 10, 40, 19, 22, 20, 49]—an approach that is shown to be optimal in terms of IFO complexity in the centralized setting—in a randomly activated manner to further improve computational efficiency when the local sample size is limited. To reduce communication, DESTRESS employs gradient tracking [50] with a few mixing rounds per iteration, which helps accelerate the convergence through better information sharing [17]; 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 point, i.e. $\mathbb{E}\|\nabla f(\boldsymbol{x}^{\mathsf{output}})\|_2^2 \le \epsilon$, where $\boldsymbol{x}^{\mathsf{output}}$ is the output of DESTRESS, and the expectation is taken with respect to the randomness of the algorithm, DESTRESS requires

- $O(m + (m/n)^{1/2}L/\epsilon)$ per-agent IFO calls,[2] which is *network-independent*; and
- $O(\frac{\log((n/m)^{1/2}+2)}{(1-\alpha)^{1/2}} \cdot ((mn)^{1/2} + \frac{L}{\epsilon}))$ 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 and decentralized communication settings.

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. The algorithms listed in the first three rows are designed for the centralized setting, and the remaining D-GET, GT-SARAH, and our DESTRESS are in the decentralized setting. Here, n is the number of agents, $m = N/n$ is the local sample size, L is the smoothness parameter of the sample loss, and $\alpha \in [0, 1)$ is the mixing rate of the network topology. The big-O notations and logarithmic terms are omitted for simplicity.

Algorithms	Setting	Per-agent IFO Complexity	Communication Rounds
SVRG [1, 33]	centralized	$N + \frac{N^{2/3}L}{\epsilon}$	n/a
SCSG/SVRG+ [16, 21]	centralized	$N + \frac{N^{2/3}L}{\epsilon}$	n/a
SNVRG [49]	centralized	$N + \frac{N^{1/2}L}{\epsilon}$	n/a
SARAH/SPIDER/SpiderBoost [29, 10, 40]	centralized	$N + \frac{N^{1/2}L}{\epsilon}$	n/a
SSRGD/ZeroSARAH/PAGE [19, 22, 20]	centralized	$N + \frac{N^{1/2}L}{\epsilon}$	n/a
D-GET [38]	decentralized	$m + \frac{1}{(1-\alpha)^2} \cdot \frac{m^{1/2}L}{\epsilon}$	Same as IFO
GT-SARAH [44]	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	$m + \frac{(m/n)^{1/2}L}{\epsilon}$	$\frac{1}{(1-\alpha)^{1/2}} \cdot \left((mn)^{1/2} + \frac{L}{\epsilon}\right)$

[2]The big-O notation is defined in subsection 1.3.

- In terms of the computation complexity, the overall IFO complexity of DESTRESS—when summed over all agents—becomes

$$n \cdot O\big(m + (m/n)^{1/2} L/\epsilon\big) = O\big(mn + (mn)^{1/2} L/\epsilon\big) = O\big(N + N^{1/2} L/\epsilon\big),$$

matching the optimal IFO complexity of centralized algorithms (e.g., SPIDER [10], PAGE [20]) and distributed server/client algorithms (e.g., D-ZeroSARAH [22]). However, the state-of-the-art decentralized algorithm GT-SARAH [44] is not able to achieve this optimal IFO complexity for all situations (see Table 1). To the best of our knowledge, DESTRESS is the first algorithm to achieve the optimal IFO complexity for the decentralized setting regardless of network topology and sample size.

- 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 (up to a logarithmic factor), i.e.,

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

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 $\frac{1}{(1-\alpha)^{3/2}}$, and, therefore, saves more communications over poorly connected networks. 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 sufficiently large, i.e., $m = \Omega\big(\frac{n}{1-\alpha}\big)$, which also holds for a wide variety of application scenarios. To gain further insights in terms of the communication savings of DESTRESS, Table 2 further compares the communication complexities of decentralized algorithms for finding first-order stationary points under three common network settings.

Table 2

Detailed comparisons of the communication complexities of D-GET, GT-SARAH, and DESTRESS under three graph topologies, where the last two rows delineate the improved factors of DESTRESS over existing algorithms. The communication savings become significant especially when $m = \Omega\big(\frac{n}{1-\alpha}\big)$. The complexities are simplified by plugging the bound on the spectral gap $1-\alpha$ from [25, Proposition 5]. *Here, n is the number of agents, $m = N/n$ is the local sample size, L is the smoothness parameter of the sample loss, and $\alpha \in [0, 1)$ is the mixing rate of the network topology. The big-O notations and logarithmic terms are omitted for simplicity.*

	Erdős-Rényi graph	2-dimensional grid graph	Path graph
$1-\alpha$ (spectral gap)	1	$\frac{1}{n \log n}$	$\frac{1}{n^2}$
D-GET [38]	$m + \frac{m^{1/2} L}{\epsilon}$	$m + \frac{m^{1/2} n^2 L}{\epsilon}$	$m + \frac{m^{1/2} n^4 L}{\epsilon}$
GT-SARAH [44]	$m + \max\Big\{1,\ \big(\frac{m}{n}\big)^{1/3},\ \big(\frac{m}{n}\big)^{1/2}\Big\} \cdot \frac{L}{\epsilon}$	$m + \max\Big\{n^2,\ m^{1/3} n^{2/3},\ \big(\frac{m}{n}\big)^{1/2}\Big\} \cdot \frac{L}{\epsilon}$	$m + \max\Big\{n^4,\ m^{1/3} n^{5/3},\ \big(\frac{m}{n}\big)^{1/2}\Big\} \cdot \frac{L}{\epsilon}$
DESTRESS (this paper)	$(mn)^{1/2} + \frac{L}{\epsilon}$	$m^{1/2} n + \frac{n^{1/2} L}{\epsilon}$	$(mn^3)^{1/2} + \frac{nL}{\epsilon}$
Improvement factors for ϵ-independent term	$\big(\frac{m}{n}\big)^{1/2}$	$\frac{m^{1/2}}{n}$	$\frac{m^{1/2}}{n^{3/2}}$
Improvement factors for ϵ-dependent term	$\max\Big\{1,\ \big(\frac{m}{n}\big)^{1/3},\ \big(\frac{m}{n}\big)^{1/2}\Big\}$	$\max\Big\{n^{3/2},\ m^{1/3} n^{1/6},\ \frac{m^{1/2}}{n}\Big\}$	$\max\Big\{n^3,\ m^{1/3} n^{2/3},\ \frac{m^{1/2}}{n^{3/2}}\Big\}$

In sum, DESTRESS harnesses the ideas of random client activation, variance reduction, gradient tracking, and extra mixing in a sophisticated manner to achieve a scalable decentralized algorithm for nonconvex empirical risk minimization that is competitive in both computation and communication over existing approaches.

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 [30, 45, 42] for further references.

Stochastic variance-reduced methods. Many variants of stochastic variance-reduced gradient based methods have been proposed for finite-sum optimization for finding first-order stationary points, including but not limited to SVRG [14, 1, 33], SCSG [16], SVRG+ [21], SAGA [7], SARAH [28, 29], SPIDER [10], SpiderBoost [40], SSRGD [19], ZeroSARAH [22], and PAGE [20]. SVRG/SVRG+/SCSG/SAGA utilize stochastic variance-reduced gradients as a corrected estimator of the full gradient, but can only achieve a suboptimal IFO complexity of $O(N + N^{2/3}L/\epsilon)$. Other algorithms such as SARAH, SPIDER, SpiderBoost, SSRGD, and PAGE adopt stochastic recursive gradients to improve the IFO complexity to $O(N+N^{1/2}L/\epsilon)$, which is optimal indicated by the lower bound provided in [10, 20]. DESTRESS also utilizes the stochastic recursive gradients to perform variance reduction, which results in the optimal IFO complexity for finding first-order stationary points.

Decentralized stochastic nonconvex optimization. There has been a flurry of recent activity in decentralized nonconvex optimization in both the server/client setting and the network setting. In the server/client setting, [6] simplifies the approaches in [15] for distributing stochastic variance-reduced algorithms without requiring sampling extra data. In particular, D-SARAH [6] extends SARAH to the server/client setting but with a slightly worse IFO complexity and a sample-independent communication complexity. D-ZeroSARAH [22] obtains the optimal IFO complexity in the server/client setting. In the network setting, D-PSGD [23] and SGP [3] extend stochastic gradient descent (SGD) to solve the nonconvex decentralized expectation minimization problems with suboptimal rates. However, due to the noisy stochastic gradients, D-PSGD can only use diminishing step size to ensure convergence, and SGP uses a small step size on the order of $1/K$, where K denotes the total iterations. D^2 [39] introduces a variance-reduced correction term to D-PSGD, which allows a constant step size and hence reaches a better convergence rate.

Gradient tracking [50, 32] 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 [47] to extend SGD to the decentralized setting, and in [17] to extend quasi-Newton algorithms as well as stochastic variance-reduced algorithms, with performance guarantees for optimizing strongly convex functions. GT-SAGA [43] further uses SAGA-style updates and reaches a convergence rate that matches SAGA [7, 34]. However, GT-SAGA requires one to store a variable table, which leads to a high memory complexity. D-GET [38] and GT-SARAH [44] adopt equivalent recursive local gradient estimators to enable the use of constant step sizes without extra memory usage. The IFO complexity of GT-SARAH is optimal in the re-

strictive range $m \gtrsim \frac{n}{(1-\alpha)^6}$, while DESTRESS achieves the optimal IFO over all parameter regimes.

In addition to variance reduction techniques, performing multiple mixing steps between local updates can greatly improve the dependence of the network in convergence rates, which is equivalent to 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. This technique is applied by a number of recent works including [4, 31, 5, 17, 12, 13, 35, 36, 18, 46, 11, 24], and its effectiveness is verified both in theory and experiments. Our algorithm also adopts the extra mixing steps, which leads to better IFO complexity and communication complexity.

1.3. Paper organization and notation. Section 2 introduces preliminary concepts and the algorithm development, section 3 shows the theoretical performance guarantees for DESTRESS, section 4 provides numerical evidence to support the analysis, and section 5 concludes the paper. Proofs and experiment settings are postponed to appendices.

Throughout this paper, we use boldface letters to represent matrices and vectors. We use $\|\cdot\|_{\mathsf{op}}$ for the matrix operator norm, $\otimes$ for the Kronecker product, $\boldsymbol{I}_n$ for the n-dimensional identity matrix, and $\mathbf{1}_n$ for the n-dimensional all-ones vector. For two real functions $f(\cdot)$ and $g(\cdot)$ defined on $\mathbb{R}^+$, we say $f(x) = O\big(g(x)\big)$ or $f(x) \lesssim g(x)$ if there exists some universal constant $M > 0$ such that $f(x) \le Mg(x)$. The notation $f(x) = \Omega\big(g(x)\big)$ or $f(x) \gtrsim g(x)$ means $g(x) = O\big(f(x)\big)$.

2. Preliminaries and proposed algorithm.

We start by describing a few useful preliminary concepts and definitions in subsection 2.1, then present the proposed algorithm in subsection 2.2.

2.1. Preliminaries.

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. Related to this matrix is an important quantity called the mixing rate, defined in Definition 2.1.

Definition 2.1 (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*

$$\alpha := \Big\|\boldsymbol{W} - \tfrac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\Big\|_{\mathsf{op}}. \tag{2.1}$$

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, [25, Proposition 5] provides comprehensive bounds on $1-\alpha$—also known as the spectral gap—for various graphs. In practice, fastest distributed linear averaging (FDLA) matrices [41] are more favorable because they can achieve a much smaller mixing rate, but they usually contain negative elements and are not symmetric. Different from other algorithms that require the mixing matrix to be doubly stochastic, our analysis can handle arbitrary mixing matrices as long as their row/column sums equal to one.

Dynamic average consensus. It has been well understood by now that using a naive mixing of local information merely, e.g. the local gradients of neighboring agents, does not lead to fast convergence of decentralized extensions of centralized methods [27, 37]. This is due to the fact that the quantity of interest in solving decentralized optimization problems is often iteration varying, which naive mixing is unable to track; consequently, an accumulation of errors leads to either slow convergence or poor accuracy. Fortunately, the general scheme of dynamic average consensus [50] proves to be extremely effective in this regard for tracking 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., [26, 32, 9, 17]. This idea, also known as "gradient tracking" in the literature, essentially adds a correction term to the naive information mixing, which we will employ in the communication stage of the proposed algorithm to track the dynamic average of local gradients.

Stochastic recursive gradient methods. Stochastic recursive gradients methods [29, 10, 40, 19] achieve the optimal IFO complexity in the centralized setting for nonconvex finite-sum optimization, which make it natural to adapt them to the decentralized setting with the hope of maintaining the appealing IFO complexity. Roughly speaking, these methods use a nested loop structure to iteratively refine the parameter, where (1) a global gradient evaluation is performed at each outer loop, and (2) a stochastic recursive gradient estimator is used to calculate the gradient and update the parameter at each inner loop. In the proposed DESTRESS algorithm, this nested loop structure lends itself to a natural decentralized scheme, as will be seen momentarily.

Additional notation. For convenience of presentation, define the stacked vector $\boldsymbol{x} \in \mathbb{R}^{nd}$ and its average over all agents $\overline{\boldsymbol{x}} \in \mathbb{R}^d$ as

$$\boldsymbol{x} := \left[\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_n^\top\right]^\top, \quad \overline{\boldsymbol{x}} = \frac{1}{n}\sum_{i=1}^n \boldsymbol{x}_i. \tag{2.2}$$

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

$$\nabla F(\boldsymbol{x}) := [\nabla f_1(\boldsymbol{x}_1)^\top, \ldots, \nabla f_n(\boldsymbol{x}_n)^\top]^\top. \tag{2.3}$$

2.2. The DESTRESS algorithm. Detailed in Algorithm 2.1, we propose a novel decentralized stochastic optimization algorithm, dubbed DESTRESS, for finding first-order stationary points of nonconvex finite-sum problems. Motivated by stochastic recursive gradient methods in the centralized setting, DESTRESS has a nested loop structure:

1. The inner loop refines the parameter estimate $\boldsymbol{u}^{(t),0} = \boldsymbol{x}^{(t-1)}$ by performing randomly activated stochastic recursive gradient updates (2.4), where the stochastic recursive gradient $\boldsymbol{v}^{(t),s}$ is updated in (2.4b) and (2.4c) via mixing minibatch stochastic gradients from activated agents' local datasets.
2. The outer loop adopts dynamic average consensus to estimate and track the global gradient $\nabla F(\boldsymbol{x}^{(t)})$ at each agent by $\boldsymbol{s}^{(t)}$ in (2.5), which allows the next inner loop to start from a less noisy starting gradient $\boldsymbol{v}^{(t+1),0} = \boldsymbol{s}^{(t)}$. A key property of (2.5)—which is a direct consequence of dynamic average consensus—is that the average of $\boldsymbol{s}^{(t)}$ equals the dynamic average of local gradients, i.e., $\overline{\boldsymbol{s}}^{(t)} = \frac{1}{n}\sum_{i\in[n]} \boldsymbol{s}_i^{(t)} = \frac{1}{n}\sum_{i\in[n]} \nabla f_i(\boldsymbol{x}_i^{(t)})$.

To enable better information sharing and faster convergence, inspired by [17], we allow DESTRESS to perform a few rounds of mixing or gossiping whenever communication takes place. Specifically, DESTRESS performs K_{out} and K_{in} mixing steps for the outer and inner loops, respectively, per iteration, which is equivalent to using

$$\boldsymbol{W}_{\mathsf{out}} = \boldsymbol{W}^{K_{\mathsf{out}}} \qquad \text{and} \qquad \boldsymbol{W}_{\mathsf{in}} = \boldsymbol{W}^{K_{\mathsf{in}}}$$

as mixing matrices, and correspondingly a network with better connectivity; see (2.5), (2.4a), and (2.4c). Note that Algorithm 2.1 is written in matrix notation, where the mixing steps are described by $\boldsymbol{W}_{\mathsf{in}} \otimes \boldsymbol{I}_n$ or $\boldsymbol{W}_{\mathsf{out}} \otimes \boldsymbol{I}_n$ and applied to all agents simultaneously. The extra mixing steps can be implemented by Chebyshev acceleration [2] with improved communication efficiency.

Algorithm 2.1 DESTRESS for decentralized nonconvex finite-sum optimization.

input: initial parameter $\overline{\boldsymbol{x}}^{(0)}$, step size η, activation probability p, batch size b, number of outer loops T, number of inner loops S, and number of communication (extra mixing) 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 inner loop initial parameters $\boldsymbol{u}^{(t),0} = \boldsymbol{x}^{(t-1)}$ and $\boldsymbol{v}^{(t),0} = \boldsymbol{s}^{(t-1)}$.

  for $s = 1, ..., S$ **do**

    Each agent i samples a minibatch $\mathcal{Z}_i^{(t),s}$ of size b from $\mathcal{M}_i$ uniformly at random, $\lambda_i^{(t),s} \sim \mathcal{B}(p)$, where $\mathcal{B}(p)$ denotes the Bernoulli distribution with parameter p,[3] and then performs the following updates:

$$\boldsymbol{u}^{(t),s} = (\boldsymbol{W}_{\mathsf{in}} \otimes \boldsymbol{I}_d)(\boldsymbol{u}^{(t),s-1} - \eta \boldsymbol{v}^{(t),s-1}), \tag{2.4a}$$

$$\boldsymbol{g}_i^{(t),s} = \frac{\lambda_i^{(t),s}}{pb} \sum_{\boldsymbol{z}_i \in \mathcal{Z}_i^{(t),s}} \Big(\nabla \ell(\boldsymbol{u}_i^{(t),s}; \boldsymbol{z}_i) - \nabla \ell(\boldsymbol{u}_i^{(t),s-1}; \boldsymbol{z}_i) \Big) + \boldsymbol{v}_i^{(t),s-1}, \tag{2.4b}$$

$$\boldsymbol{v}^{(t),s} = (\boldsymbol{W}_{\mathsf{in}} \otimes \boldsymbol{I}_d)\boldsymbol{g}^{(t),s}. \tag{2.4c}$$

  end for

  Set the new parameter estimate $\boldsymbol{x}^{(t)} = \boldsymbol{u}^{(t),S}$.

  Update the global gradient estimate by aggregated local information and gradient tracking:

$$\boldsymbol{s}^{(t)} = (\boldsymbol{W}_{\mathsf{out}} \otimes \boldsymbol{I}_d)\Big(\boldsymbol{s}^{(t-1)} + \nabla F\big(\boldsymbol{x}^{(t)}\big) - \nabla F\big(\boldsymbol{x}^{(t-1)}\big) \Big). \tag{2.5}$$

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] The stochastic gradients will not be computed if $\lambda_i^{(t),s} = 0$.

Compared with existing decentralized algorithms based on stochastic variance-reduced algorithms such as D-GET [38] and GT-SARAH [44], DESTRESS utilizes different gradient estimators and communication protocols: First, DESTRESS produces a sequence of reference points $\{\boldsymbol{x}^{(t)}\}$ that converge to a global first-order stationary point and corresponding global gradient estimates $\{\boldsymbol{s}^{(t)}\}$ that are updated by full gradient computations, so that inner loops can refine $\boldsymbol{x}^{(t)}$ using stochastic recursive gradients based on accurate gradient estimates; second, the communication and computation in DESTRESS are paced differently due to the introduction of extra mixing, which allow more flexible trade-off schemes between different types of resources; last but not least, the random activation of stochastic recursive gradient updates further saves local computation, especially when the local sample size is small compared to the number of agents.

3. Performance guarantees.

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

3.1. Assumptions.

We first introduce Assumptions 1 and 2, which are standard assumptions imposed on the loss function. Assumption 1 implies that all local objective functions $f_i(\cdot)$ and the global objective function $f(\cdot)$ also have Lipschitz gradients, and Assumption 2 guarantees the absence of 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 Definition 3.1.

Definition 3.1 (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*

$$\|\nabla f(\boldsymbol{x})\|_2^2 \leq \epsilon.$$

3.2. Main theorem.

Theorem 3.2, whose proof is deferred to Appendix B, shows that DESTRESS converges in expectation to an approximate first-order stationary point, under suitable parameter choices.

Theorem 3.2 (first-order optimality). *Assume Assumptions 1 and 2 hold. Set $p \in (0,1]$, K_{in}, K_{out}, S, b, and η to be positive and satisfy*

$$\alpha^{K_{\mathsf{in}}} \leq p \quad \text{and} \quad \eta L \leq \frac{(1-\alpha^{K_{\mathsf{in}}})^3(1-\alpha^{K_{\mathsf{out}}})}{10\big(1+\alpha^{K_{\mathsf{in}}}\alpha^{K_{\mathsf{out}}}\sqrt{npb}\big)\big(\sqrt{S/(npb)}+1\big)}. \tag{3.1}$$

The output produced by Algorithm 2.1 satisfies

$$\mathbb{E}\left\|\nabla f(\boldsymbol{x}^{\mathsf{output}})\right\|_2^2 < \frac{4}{\eta TS}\Big(\mathbb{E}[f(\overline{\boldsymbol{x}}^{(0)})] - f^*\Big). \tag{3.2}$$

If there is only one agent, i.e., $n = 1$, the mixing rate will be $\alpha = 0$; we can choose $K_{\mathsf{in}} = K_{\mathsf{out}} = p = 1$, and Theorem 3.2 reduces to [29, Theorem 1], its counterpart in the centralized setting. For general decentralized settings with arbitrary mixing schedules, Theorem 3.2 provides a comprehensive characterization of the convergence rate, where an ϵ-approximate first-order stationary point can be found in expectation in a total of

$$TS = O\left(\frac{\mathbb{E}[f(\overline{\boldsymbol{x}}^{(0)})] - f^*}{\eta\epsilon}\right)$$

iterations; here, T is the number of outer iterations and S is the number of inner iterations. Clearly, a larger step size η, as allowable by (3.1), hints on a smaller iteration complexity, and hence a smaller IFO complexity.

There are two conditions in (3.1). On one end, K_{in} needs to be large enough (i.e., perform more rounds of extra mixing) to counter the effect when p is small (i.e., we compute fewer stochastic gradients every iteration), or when α is close to 1 (i.e., the network is poorly connected). On the other end, the step size η needs to be small enough to account for the requirement of the step size in the centralized setting, as well as the effect of imperfect communication due to decentralization. For well-connected networks, where $\alpha \ll 1$, the terms introduced by the decentralized setting will diminish—indicating the iteration complexity is close to that of the centralized setting. For poorly connected networks, carefully designing the mixing matrix and other parameters can ensure a desirable trade-off between convergence speed and communication cost. The following corollary provides specific parameter choices for DESTRESS to achieve the optimal per-agent IFO complexity. The proof is deferred to Appendix C.

Corollary 3.3 (complexity for finding first-order stationary points). *Under conditions of Theorem 3.2, set* $S = \lceil\sqrt{mn}\rceil$, $b = \lceil\sqrt{m/n}\rceil$, $p = \frac{\sqrt{m/n}}{\lceil\sqrt{m/n}\rceil}$, $K_{\mathsf{out}} = \lceil\frac{\log(\sqrt{npb}+1)}{(1-\alpha)^{1/2}}\rceil$, $K_{\mathsf{in}} = \lceil\frac{\log(2/p)}{(1-\alpha)^{1/2}}\rceil$, $\eta = \frac{1}{640L}$, *and implement the mixing steps using Chebyshev's acceleration [2]. To reach an* ϵ*-approximate first-order stationary point, in expectation, DESTRESS then takes* $O(m + \frac{(m/n)^{1/2}L}{\epsilon})$ *IFO calls per agent, and* $O(\frac{\log((n/m)^{1/2}+2)}{(1-\alpha)^{1/2}} \cdot ((mn)^{1/2} + \frac{L}{\epsilon}))$ *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. In addition, when the accuracy $\epsilon \lesssim L/(mn)^{1/2}$, DESTRESS reaches a communication complexity of $O(\frac{1}{(1-\alpha)^{1/2}} \cdot \frac{L}{\epsilon})$, which is independent of the sample size.

It is worthwhile to further highlight the role of the random activation probability p in achieving the optimal IFO by allowing "fractional" batch size. Note that the batch size is set as $b = \lceil\sqrt{m/n}\rceil$, where m is the local sample size, and n is the number of agents.

1. When the local sample size is large, i.e., $m \geq n$, we can approximate $b \approx \sqrt{m/n}$ and $p \approx 1$. In fact, Corollary 3.3 continues to hold with $p = 1$ in this regime.
2. However, when the number of agents is large, i.e., $n > m$, the batch size $b = 1$ and $p = \sqrt{m/n} < 1$, which mitigates the potential computation waste by only selecting a subset of agents to perform local computation, compared to the case when 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. Numerical experiments.

This section provides numerical experiments on real datasets to evaluate our proposed algorithm DESTRESS with comparisons against two existing baselines: DSGD [27, 23] and GT-SARAH [44]. To allow for reproducibility, we fix random seeds for each experiment, and all code can be found at https://github.com/liboyue/Network-Distributed-Algorithm.

For all experiments, we shuffle the datasets and normalize the samples by subtracting the mean and dividing the standard deviation. We set the number of agents $n = 20$, and split all datasets uniformly 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–Rényi graph (the connectivity probability is set to 0.3), grid graph, and path graph. The mixing matrices are chosen as the symmetric FDLA matrices [41] 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 parameters are tuned manually for the best performance. We defer detailed descriptions of baseline algorithms as well as parameter choices in Appendix A.

4.1. Logistic regression with nonconvex regularization.

To begin with, we employ logistic regression with nonconvex regularization to solve a binary classification problem using the Gisette dataset.[4] We split the Gisette dataset into $n = 20$ agents, where each agent receives $m = 300$ training samples. The sample loss function is given as

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

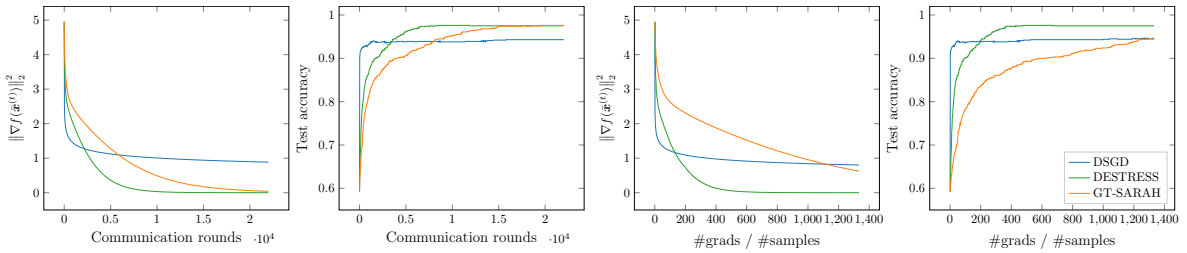
where $\{\boldsymbol{f}, l\}$ represents a training tuple, $\boldsymbol{f} \in \mathbb{R}^d$ is the feature vector, and $l \in \{0, 1\}$ is the label, and λ is the regularization parameter. For this experiment, we set $\lambda = 0.1$.

Figure 1 shows the train gradient norm 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 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 communication graph is less connected.

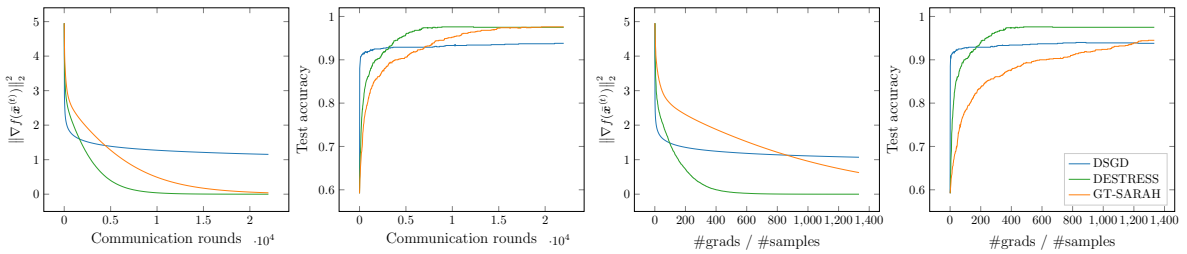
4.2. Neural network training.

Next, we compare the performance of DESTRESS 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 [8]. We evenly split the MNIST

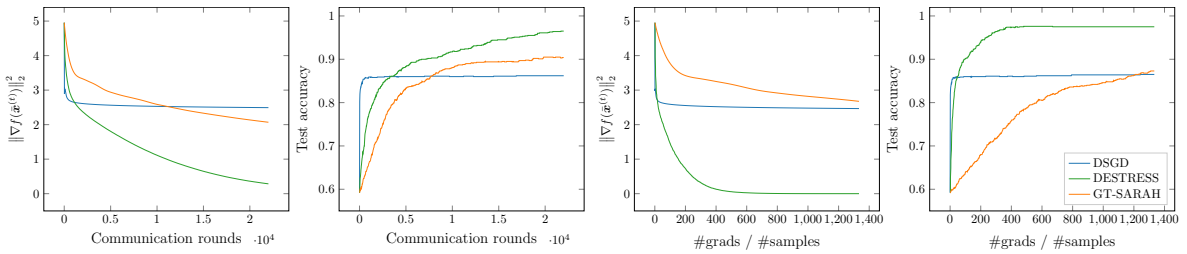
[4] The dataset can be accessed at https://archive.ics.uci.edu/ml/datasets/Gisette.



(a) Erdös–Rényi graph



(b) Grid graph

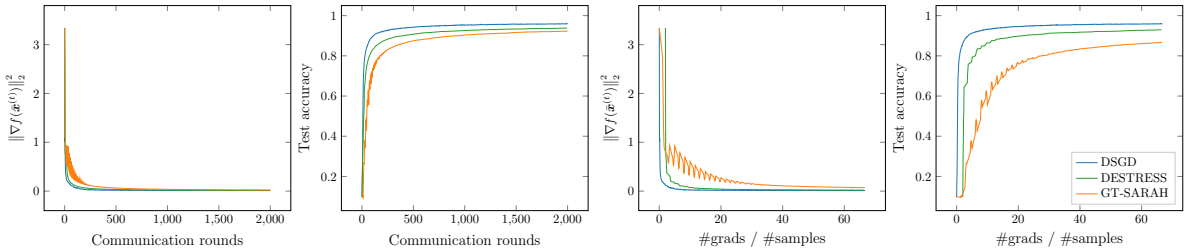


(c) Path graph

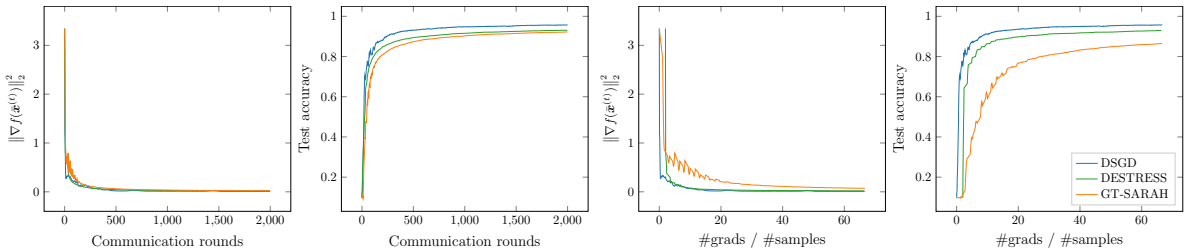
Figure 1. *The train gradient norm and testing accuracy with respect to the number of communication rounds (left two panels) and gradient evaluations (right two panels) for DSGD, GT-SARAH, and DESTRESS when training logistic regression model with nonconvex regularization on the Gisette dataset. Due to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

dataset into $n = 20$ agents, where each agent receives $m = 3,000$ training samples. Figure 2 plots the training gradient norm and testing accuracy against the number of communication rounds and gradient evaluations for all algorithms. DESTRESS significantly outperforms GT-SARAH in terms of computation and communication costs due to the larger step size and extra mixing. Differently from the previous experiment, DSGD performs the best for the Erdös–Rényi graph and grid graph that are well-connected, while it converges slower than DESTRESS on the path graph.

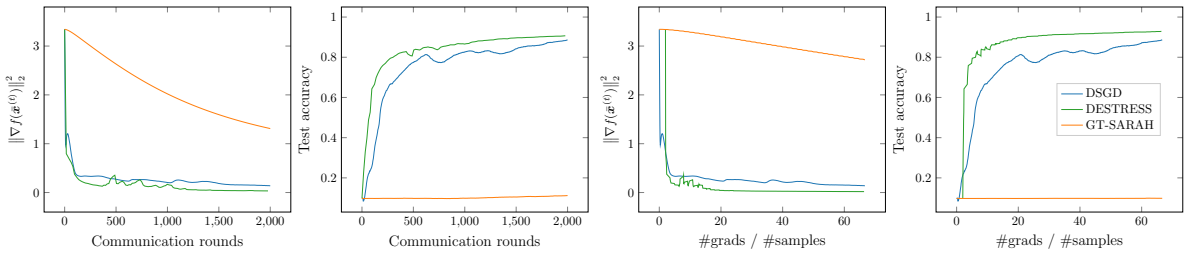
The last experiment investigates the convergence precision $1/\epsilon$ of DESTRESS with respect to the number of gradient evaluations. Under the same experimental setup, we conduct 64 different runs where each run starts from a different initial point. The convergence precision is computed by the inverse of the running average of the squared gradient norms. The results, including mean and variance, are shown in Figure 3, which numerically validate the linear relation indicated by Corollary 3.3.



(a) Erdös–Rényi graph



(b) Grid graph



(c) Path graph

Figure 2. *The train gradient norm 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.

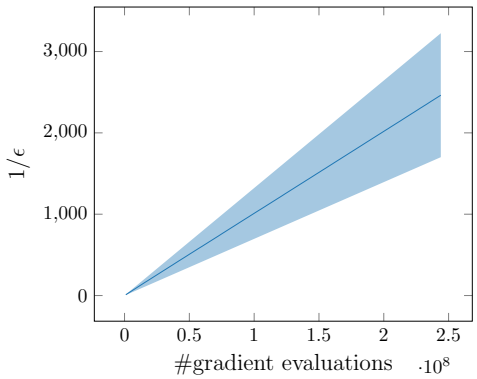


Figure 3. *The convergence precision* $1/\epsilon$ *with respect to the number of total gradient evaluations for neural network training averaged over 64 experiments. The shade shows the variance.*

Algorithm A.1 Decentralized stochastic gradient descent (DSGD).

input: initial parameter $\overline{\boldsymbol{x}}^{(0)}$, initial step size η_0, number of iterations T.
initialization: set $\boldsymbol{x}_i^{(0)} = \overline{\boldsymbol{x}}^{(0)}$.
for $t = 1, \ldots, T$ **do**
  Each agent i samples a minibatch $\mathcal{Z}_i^{(t)}$ from $\mathcal{M}_i$ uniformly at random, and then performs the following updates:

$$\boldsymbol{g}_i^{(t)} = \frac{1}{b} \sum_{\boldsymbol{z}_i \in \mathcal{Z}_i^{(t)}} \nabla \ell(\boldsymbol{u}_i^{(t)}; \boldsymbol{z}_i).$$

  Update via local communication: $\boldsymbol{x}^{(t+1)} = (\boldsymbol{W} \otimes \boldsymbol{I}_d) \left(\boldsymbol{x}^{(t)} - \frac{\eta_0}{\sqrt{t}} \boldsymbol{g}^{(t)} \right)$.
end for
output: $\boldsymbol{x}^{\mathsf{output}} = \overline{\boldsymbol{x}}^{(T)}$.

5. 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-type methods for finding first-order stationary points, and improves both computation and communication complexities for a broad range of parameter 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. The use of communication compression to further reduce the communication cost is also of interest [48]. We leave these interesting directions to future works.

Appendix A. Experiment details. For completeness, we list two baseline algorithms, DSGD [27, 23] (cf. Algorithm A.1) and GT-SARAH [44] (cf. Algorithm A.2), which are compared numerically against the proposed DESTRESS algorithm in section 4. Furthermore, the detailed hyperparameter settings for the experiments in sections 4.1 and 4.2 are listed in Tables 3 and 4, respectively.

Appendix B. Proof of Theorem 3.2.

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 $\boldsymbol{x} = \left[\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_n^\top\right]^\top$, where $\boldsymbol{x}_i \in \mathbb{R}^d$, as follows

$$\nabla f(\boldsymbol{x}) := [\nabla f(\boldsymbol{x}_1)^\top, \ldots, \nabla f(\boldsymbol{x}_n)^\top]^\top. \tag{B.1}$$

The following fact is a straightforward consequence of our assumption on the mixing matrix $\boldsymbol{W}$ in Definition 2.1.

Algorithm A.2 GT-SARAH.

input: initial parameter $\overline{\boldsymbol{x}}^{(0)}$, step size η, number of outer loops T, number of inner loops 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 minibatch $\mathcal{Z}_i^{(t)}$ 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: $\boldsymbol{x}^{\mathsf{output}} = \overline{\boldsymbol{x}}^{(T)}$.

Table 3
Parameter settings for the experiments on regularized logistic regression in section 4.1.

Algorithms	DSGD		DESTRESS						GT-SARAH		
Parameters	η_0	b	η	p	K_{in}	K_{out}	b	S	η	b	S
Erdös–Rènyi	1	10	0.01	1	2	2	10	10	0.001	10	10
Grid	1	10	0.01	1	2	3	10	10	0.001	10	10
Path	0.1	10	0.01	1	8	8	10	10	0.0001	10	10

Table 4
Parameter settings for the experiments on neural network training in section 4.2.

Algorithms	DSGD		DESTRESS						GT-SARAH		
Parameters	η_0	b	η	p	K_{in}	K_{out}	b	S	η	b	S
Erdös–Rènyi	1	100	1	1	2	2	100	10	0.1	100	10
Grid	1	100	1	1	3	4	100	10	0.1	100	10
Path	0.1	100	1	1	8	10	100	10	0.0001	100	10

Fact 1. *Let* $\boldsymbol{x} = \left[\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_n^\top\right]^\top$, *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 Definition 2.1, 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 B.1 (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 of Lemma B.1. The local update rule (2.4a), 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

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

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:

$$\begin{aligned} nTS\mathbb{E}\big\|\nabla f(\boldsymbol{x}^{\mathsf{output}})\big\|_2^2 &= \sum_{i=1}^{n}\sum_{t=1}^{T}\sum_{s=0}^{S-1}\mathbb{E}\big\|\nabla f(\boldsymbol{u}_i^{(t),s})\big\|_2^2 \\ &\overset{\text{(i)}}{=} \sum_{t=1}^{T}\sum_{s=0}^{S-1}\mathbb{E}\big\|\nabla f(\boldsymbol{u}^{(t),s})\big\|_2^2 \\ &= \sum_{t=1}^{T}\sum_{s=0}^{S-1}\mathbb{E}\big\|\nabla f(\boldsymbol{u}^{(t),s}) - \nabla f(\mathbf{1}_n \otimes \overline{\boldsymbol{u}}^{(t),s}) + \nabla f(\mathbf{1}_n \otimes \overline{\boldsymbol{u}}^{(t),s})\big\|_2^2 \\ &\overset{\text{(ii)}}{\leq} 2\sum_{t=1}^{T}\sum_{s=0}^{S-1}\Big(\mathbb{E}\big\|\nabla f(\boldsymbol{u}^{(t),s}) - \nabla f(\mathbf{1}_n \otimes \overline{\boldsymbol{u}}^{(t),s})\big\|_2^2 + \mathbb{E}\big\|\nabla f(\mathbf{1}_n \otimes \overline{\boldsymbol{u}}^{(t),s})\big\|_2^2\Big) \\ &\overset{\text{(iii)}}{\leq} 2\sum_{t=1}^{T}\sum_{s=0}^{S-1}\Big(L^2\mathbb{E}\big\|\boldsymbol{u}^{(t),s} - \mathbf{1}_n \otimes \overline{\boldsymbol{u}}^{(t),s}\big\|_2^2 + n\mathbb{E}\big\|\nabla f(\overline{\boldsymbol{u}}^{(t),s})\big\|_2^2\Big), \end{aligned} \tag{B.3}$$

where (i) follows from the change of notation using (B.1), (ii) follows from the Cauchy–Schwartz inequality, and (iii) follows from Assumption 1. Then, in view of Lemma B.1, (B.3) can be further bounded by

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

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

Next, we present Lemmas B.2 and B.3 to bound the double sum in (B.4), whose proofs can be found in sections SM1 and SM2, respectively.

Lemma B.2 (sum of inner loop errors). *Assume all conditions in Theorem 3.2 hold. For all $t > 0$, we can bound the summation of inner loop errors as*

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

Lemma B.3 (sum of outer loop gradient estimation error and consensus error). *Assume all conditions in Theorem 3.2 hold. We have*

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

Using Lemma B.2, (B.4) can be bounded as follows:

$$nTS\mathbb{E}\big\|\nabla f(\boldsymbol{x}^{\mathsf{output}})\big\|_2^2 < \frac{4n}{\eta}\Big(\mathbb{E}[f(\overline{\boldsymbol{x}}^{(t),0})] - f^*\Big) - 2n\Big(\frac{24}{25} - \eta L\Big)\sum_{t=1}^{T}\sum_{s=0}^{S-1}\mathbb{E}\big\|\overline{\boldsymbol{v}}^{(t),s}\big\|_2^2$$
$$+ \frac{64L^2}{1-\alpha_{\mathsf{in}}}\cdot\Big(\frac{S}{npb}+1\Big)\sum_{t=0}^{T-1}\mathbb{E}\big\|\boldsymbol{x}^{(t)} - \mathbf{1}_n\otimes\overline{\boldsymbol{x}}^{(t)}\big\|_2^2 + 2\alpha_{\mathsf{in}}^2\sum_{t=0}^{T-1}\mathbb{E}\big\|\boldsymbol{s}^{(t)} - \mathbf{1}_n\otimes\overline{\boldsymbol{s}}^{(t)}\big\|_2^2, \tag{B.5}$$

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

and $\mathbb{E}\|\boldsymbol{s}^{(t)} - \mathbf{1}_n \otimes \overline{\boldsymbol{s}}^{(t)}\|_2^2$, and the summation of the norm of average inner loop gradient estimator $n\sum_{s=1}^{S}\mathbb{E}\|\overline{\boldsymbol{v}}^{(t),s-1}\|_2^2$.

By Lemma B.3, (B.5) can be further bounded as

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

which concludes the proof.

Appendix C. Proof of corollary 3.3.

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 Corollary 3.3 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 (3.1) 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 $K_{\mathsf{in}} = \lceil\frac{\log(2/p)}{\sqrt{1-\alpha}}\rceil$ and $K_{\mathsf{out}} = \lceil\frac{\log(\sqrt{npb}+1)}{\sqrt{1-\alpha}}\rceil$. 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 $K_{\mathsf{out}} = \lceil\frac{\log(\sqrt{npb}+1)}{\sqrt{1-\alpha}}\rceil$, (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 the selected parameters into (3.2) in Theorem 3.2, we have

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

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

$$T \cdot (SK_{\mathsf{in}} + K_{\mathsf{out}}) = O\left(\frac{(mn)^{1/2}\log\left(2(n/m)^{1/2}+2\right)+\log\left((mn)^{1/4}+1\right)}{\sqrt{1-\alpha}} \cdot \left(1+\frac{L}{(mn)^{1/2}\epsilon}\right)\right)$$
$$= O\left(\frac{\log\left((n/m)^{1/2}+2\right)}{\sqrt{1-\alpha}} \cdot \left((mn)^{1/2}+\frac{L}{\epsilon}\right)\right),$$

  where we use $2/p = \frac{2\lceil\sqrt{m/n}\rceil}{\sqrt{m/n}} \le \frac{2(\sqrt{m/n}+1)}{\sqrt{m/n}} = 2(\sqrt{n/m}+1)$ to bound K_{in}.
- The IFO complexity is $T \cdot (Spb + 2m) = O\Big(m + \frac{(m/n)^{1/2}L}{\epsilon}\Big)$.

REFERENCES

[1] Z. Allen-Zhu and E. Hazan, *Variance reduction for faster non-convex optimization*, Proc. Mach. Learn. Res. (PMLR), 48 (2016), pp. 699–707.

[2] M. Arioli and J. Scott, *Chebyshev acceleration of iterative refinement*, Numer. Algorithms, 66 (2014), pp. 591–608.

[3] M. Assran, N. Loizou, N. Ballas, and M. Rabbat, *Stochastic gradient push for distributed deep learning*, Proc. Mach. Learn. Res. (PMLR), 97 (2019), pp. 344–353.

[4] A. S. Berahas, R. Bollapragada, N. S. Keskar, and E. Wei, *Balancing communication and computation in distributed optimization*, IEEE Trans. Automat. Control, 64 (2019), pp. 3141–3155.

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

[6] S. Cen, H. Zhang, Y. Chi, W. Chen, and T.-Y. Liu, *Convergence of distributed stochastic variance reduced methods without sampling extra data*, IEEE Trans. Signal Process., 68 (2020), pp. 3976–3989.

[7] A. Defazio, F. Bach, and S. Lacoste-Julien, *SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives*, in Advances in Neural Information Processing Systems, Curran Associates, Red Hook, NY, 2014, pp. 1646–1654.

[8] L. Deng, *The MNIST database of handwritten digit images for machine learning research [best of the web]*, IEEE Signal Process. Mag., 29 (2012), pp. 141–142.

[9] P. Di Lorenzo and G. Scutari, *Next: In-network nonconvex optimization*, IEEE Trans. Signal Inform. Process. Netw., 2 (2016), pp. 120–136.

[10] C. Fang, C. J. Li, Z. Lin, and T. Zhang, *SPIDER: Near-optimal non-convex optimization via stochastic path-integrated differential estimator*, in Advances in Neural Information Processing Systems, Curran Associates, Red Hook, NY, 2018, pp. 687–697.

[11] L. V. Gambuzza and M. Frasca, *Distributed control of multiconsensus*, IEEE Trans. Automat. Control, 66 (2020), pp. 2032–2044.

[12] A. Hashemi, A. Acharya, R. Das, H. Vikalo, S. Sanghavi, and I. S. Dhillon, *On the benefits of multiple gossip steps in communication-constrained decentralized federated learning*, IEEE Trans. Parallel Distrib. Systems, 33 (2021), pp. 2727–2739.

[13] C. Iakovidou and E. Wei, *S-NEAR-DGD: A flexible distributed stochastic gradient method for inexact communication*, IEEE Trans. Automat. Control, to appear.

[14] R. Johnson and T. Zhang, *Accelerating stochastic gradient descent using predictive variance reduction*, in Advances in Neural Information Processing Systems, Curran Associates, Red Hook, NY, 2013, pp. 315–323.

[15] J. D. Lee, Q. Lin, T. Ma, and T. Yang, *Distributed stochastic variance reduced gradient methods by sampling extra data with replacement*, J. Mach. Learn. Res., 18 (2017), pp. 4404–4446.

[16] L. Lei, C. Ju, J. Chen, and M. I. Jordan, *Non-convex finite-sum optimization via SCSG methods*, in Advances in Neural Information Processing Systems, Curran Associates, Red Hook, NY, 2017, pp. 2345–2355.

[17] B. Li, S. Cen, Y. Chen, and Y. Chi, *Communication-efficient distributed optimization in networks with gradient tracking and variance reduction*, J. Mach. Learn. Res., 21 (2020), pp. 1–51.
[18] H. Li, C. Fang, W. Yin, and Z. Lin, *Decentralized accelerated gradient methods with increasing penalty parameters*, IEEE Trans. Signal Process., 68 (2020), pp. 4855–4870.
[19] Z. Li, *SSRGD: Simple stochastic recursive gradient descent for escaping saddle points*, in Advances in Neural Information Processing Systems, Curran Associates, Red Hook, NY, 2019, pp. 1523–1533.
[20] Z. Li, H. Bao, X. Zhang, and P. Richtárik, *Page: A simple and optimal probabilistic gradient estimator for nonconvex optimization*, Proc. Mach. Learn. Res. (PMLR), 139 (2021), pp. 6286–6295.
[21] Z. Li and J. Li, *A simple proximal stochastic gradient method for nonsmooth nonconvex optimization*, in Advances in Neural Information Processing Systems, Curran Associates, Red Hook, NY, 2018, pp. 5569–5579.
[22] Z. Li and P. Richtárik, *ZeroSARAH: Efficient Nonconvex Finite-Sum Optimization with Zero Full Gradient Computation*, preprint, arXiv:2103.01447, 2021.
[23] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, *Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent*, in Advances in Neural Information Processing Systems, 2017, pp. 5330–5340.
[24] Y. Lu and C. De Sa, *Optimal complexity in decentralized training*, Proc. Mach. Learn. Res. (PMLR), 139 (2021), pp. 7111–7123.
[25] A. Nedić, A. Olshevsky, and M. G. Rabbat, *Network topology and communication-computation tradeoffs in decentralized optimization*, Proc. IEEE, 106 (2018), pp. 953–976.
[26] A. Nedić, A. Olshevsky, and W. Shi, *Achieving geometric convergence for distributed optimization over time-varying graphs*, SIAM J. Optim., 27 (2017), pp. 2597–2633.
[27] A. Nedic and A. Ozdaglar, *Distributed subgradient methods for multi-agent optimization*, IEEE Trans. Automat. Control, 54 (2009), pp. 48–61.
[28] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč, *SARAH: A novel method for machine learning problems using stochastic recursive gradient*, Proc. Mach. Learn. Res. (PMLR), 70 (2017), pp. 2613–2621.
[29] L. M. Nguyen, M. van Dijk, D. T. Phan, P. H. Nguyen, T.-W. Weng, and J. R. Kalagnanam, *Finite-sum smooth optimization with SARAH*, Comput. Optim. Appl., 82 (2022), pp. 1–33.
[30] M. Nokleby, H. Raja, and W. U. Bajwa, *Scaling-up distributed processing of data streams for machine learning*, Proc. IEEE, 108 (2020), pp. 1984–2012.
[31] T. Pan, J. Liu, and J. Wang, *D-SPIDER-SFO: A decentralized optimization algorithm with faster convergence rate for nonconvex problems*, in Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 1619–1626.
[32] G. Qu and N. Li, *Harnessing smoothness to accelerate distributed optimization*, IEEE Trans. Control Netw. Syst., 5 (2018), pp. 1245–1260.
[33] S. J. Reddi, A. Hefny, S. Sra, B. Poczos, and A. Smola, *Stochastic variance reduction for nonconvex optimization*, Proc. Mach. Learn. Res. (PMLR), 48 (2016), pp. 314–323.
[34] S. J. Reddi, S. Sra, B. Póczos, and A. Smola, *Fast incremental method for smooth nonconvex optimization*, in 2016 IEEE 55th Conference on Decision and Control, IEEE, Piscataway, NJ, 2016, pp. 1971–1977.
[35] K. Scaman, F. Bach, S. Bubeck, Y. T. Lee, and L. Massoulié, *Optimal algorithms for smooth and strongly convex distributed optimization in networks*, Proc. Mach. Learn. Res. (PMLR), 70 (2017), pp. 3027–3036.
[36] K. Scaman, F. Bach, S. Bubeck, L. Massoulié, and Y. T. Lee, *Optimal algorithms for non-smooth distributed optimization in networks*, in Advances in Neural Information Processing Systems, Curran Associates, Red Hook, NY, 2018, pp. 2740–2749.
[37] W. Shi, Q. Ling, G. Wu, and W. Yin, *EXTRA: An exact first-order algorithm for decentralized consensus optimization*, SIAM J. Optim., 25 (2015), pp. 944–966.
[38] H. Sun, S. Lu, and M. Hong, *Improving the sample and communication complexity for decentralized non-convex optimization: Joint gradient estimation and tracking*, Proc. Mach. Learn. Res. (PMLR), 119 (2020), pp. 9217–9228.
[39] H. Tang, X. Lian, M. Yan, C. Zhang, and J. Liu, *D^2: Decentralized training over decentralized data*, Proc. Mach. Learn. Res. (PMLR), 80 (2018), pp. 4848–4856.

[40] Z. Wang, K. Ji, Y. Zhou, Y. Liang, and V. Tarokh, *SpiderBoost and momentum: Faster variance reduction algorithms*, in Advances in Neural Information Processing Systems, Curran Associates, Red Hook, NY, 2019, pp. 2406–2416.

[41] L. Xiao and S. Boyd, *Fast linear iterations for distributed averaging*, Systems Control Lett., 53 (2004), pp. 65–78.

[42] R. Xin, S. Kar, and U. A. Khan, *Decentralized stochastic optimization and machine learning: A unified variance-reduction framework for robust performance and fast convergence*, IEEE Signal Processing Mag., 37 (2020), pp. 102–113.

[43] R. Xin, U. A. Khan, and S. Kar, *A fast randomized incremental gradient method for decentralized non-convex optimization*, IEEE Trans. Automat. Control, to appear.

[44] R. Xin, U. A. Khan, and S. Kar, *Fast decentralized nonconvex finite-sum optimization with recursive variance reduction*, SIAM J. Optim., 32 (2022), pp. 1–28.

[45] R. Xin, S. Pu, A. Nedić, and U. A. Khan, *A general framework for decentralized optimization with first-order methods*, Proc. IEEE, 108 (2020), pp. 1869–1889.

[46] H. Ye, Z. Zhou, L. Luo, and T. Zhang, *Decentralized accelerated proximal gradient descent*, in Advances in Neural Information Processing Systems, Curran Associates, Red Hook, NY, 2020, pp. 18308–18317.

[47] J. Zhang and K. You, *Decentralized Stochastic Gradient Tracking for Non-convex Empirical Risk Minimization*, preprint, arXiv:1909.02712, 2019.

[48] H. Zhao, B. Li, Z. Li, P. Richtárik, and Y. Chi, *BEER: Fast $O(1/T)$ Rate for Decentralized Nonconvex Optimization with Communication Compression*, preprint, arXiv:2201.13320, 2022.

[49] D. Zhou, P. Xu, and Q. Gu, *Stochastic nested variance reduction for nonconvex optimization*, in Advances in Neural Information Processing Systems, Curran Associates, Red Hook, NY, 2018, pp. 3921–3932.

[50] M. Zhu and S. Martínez, *Discrete-time dynamic average consensus*, Automatica J. IFAC, 46 (2010), pp. 322–329.