D-SPIDER-SFO: A Decentralized Optimization Algorithm with Faster Convergence Rate for Nonconvex Problems

Taoxing Pan,[1] **Jun Liu,**[2] **Jie Wang**[1*]
[1]University of Science and Technology of China, [2]Infinia ML, Inc.
tx1997@mail.ustc.edu.cn
jun.liu@infiniaml.com
jiewangx@ustc.edu.cn

Abstract

Decentralized optimization algorithms have attracted intensive interests recently, as it has a balanced communication pattern, especially when solving large-scale machine learning problems. Stochastic Path Integrated Differential Estimator Stochastic First-Order method (SPIDER-SFO) nearly achieves the algorithmic lower bound in certain regimes for nonconvex problems. However, whether we can find a decentralized algorithm which achieves a similar convergence rate to SPIDER-SFO is still unclear. To tackle this problem, we propose a decentralized variant of SPIDER-SFO, called **d**ecentralized SPIDER-SFO (D-SPIDER-SFO). We show that D-SPIDER-SFO achieves a similar gradient computation cost—that is, $\mathcal{O}(\epsilon^{-3})$ for finding an ϵ-approximate first-order stationary point—to its centralized counterpart. To the best of our knowledge, D-SPIDER-SFO achieves the state-of-the-art performance for solving nonconvex optimization problems on decentralized networks in terms of the computational cost. Experiments on different network configurations demonstrate the efficiency of the proposed method.

Introduction

Distributed optimization is a popular technique for solving large scale machine learning problems (Li et al. 2014), ranging from visual object recognition (Huang et al. 2017; He et al. 2016) to natural language processing (Vaswani et al. 2017; Devlin et al. 2019). For distributed optimization, a set of workers form a connected computational network, and each worker is assigned a portion of the computing task. The centralized network topology, like parameter server (Jianmin et al. 2016; Dean et al. 2012; Li et al. 2014; Zinkevich et al. 2010), consists of a central worker connected with all other workers. This communication mechanism could degrade the performance significantly in scenarios where the underlying network has low bandwidth or high latency (Lian et al. 2017).

In contrast, the decentralized network topology offers better network load balance—as all nodes in the network only communicate with their neighbors instead of the central node—which implies that they may be able to outperform their centralized counterparts. These motivate many works on decentralized algorithms. Nedić and Ozdaglar (2009) studied distributed subgradient method for optimizing a sum of convex objective functions. Shi et al. (2014) analyzed the linear convergence rate of the ADMM in decentralized consensus optimization. Yuan, Ling, and Yin (2016) studied the convergence properties of the decentralized gradient descent method (DGD). They proved that the local solutions and the mean solution converge to a neighborhood of the global minimizer at a linear rate for strongly convex problems. Mokhtari and Ribeiro (2016) studied decentralized double stochastic averaging gradient algorithm (DSA) and Wei et al. (2015) proposed decentralized exact first-order algorithm (EXTRA). Both of these two algorithms converge to an optimal solution at a linear rate for strongly convex problems. Lian et al. (2017) studied decentralized PSGD (D-PSGD) and showed that decentralized algorithms could be faster than their centralized counterparts. Tang et al. (2018) proposed D^2 algorithm which is less sensitive to the data variance across workers. Scaman et al. (2018) provided two optimal decentralized algorithms, called multi-step primal-dual (MSPD) and distributed randomized smoothing (DRS), and their corresponding optimal convergence rate for convex problems in certain regimes. Assran et al. (2019) proposed Stochastic Gradient Push (SGP) and proved that SGP converges to a stationary point of smooth and nonconvex objectives at the sub-linear rate.

On the other hand, to achieve a faster convergence rate, researchers have also proposed many nonconvex optimization algorithms. Stochastic Gradient Descent (SGD) (Robbins and Monro 1951) achieves an ϵ-approximate stationary point with a gradient cost of $\mathcal{O}(\epsilon^{-4})$ (Ghadimi and Lan 2013). To improve the convergence rate of SGD, researchers have proposed variance-reduction methods (Roux, Schmidt, and Bach 2012; Defazio, Bach, and Lacoste-Julien 2014). Specifically, the finite-sum Stochastic Variance Reduced Gradient method (SVRG) (Johnson and Zhang 2013; Reddi et al. 2016) and online Stochastically Controlled Stochastic Gradient method (SCSG) (Lei et al. 2017) achieve a gradient cost of $\mathcal{O}(\min(m^{2/3}\epsilon^{-2}, \epsilon^{-10/3}))$, where m is the number of samples. SNVRG (Zhou, Xu, and Gu 2018) achieves a gradient cost of $\widetilde{\mathcal{O}}(\epsilon^{-3})$, while SPIDER-SFO (Fang et

*Corresponding author

Table 1: Comparision of D-PSGD, D^2 and D-SPIDER-SFO and their centralized competitors.

Algorithm	Communication cost on the busiest node	Gradient Computation Cost	Bounded data variance among workers
C-PSGD (Dekel et al. 2012)	$\mathcal{O}(n)$	$\mathcal{O}(\epsilon^{-4})$	$\times$
D-PSGD (Lian et al. 2017)	$\mathcal{O}$ (Deg(network))	$\mathcal{O}(\epsilon^{-4})$	need
D^2(Tang et al. 2018)	$\mathcal{O}$ (Deg(network))	$\mathcal{O}(\epsilon^{-4})$	no need
C-SPIDER-SFO (Fang et al. 2018)	$\mathcal{O}(n)$	$\mathcal{O}(\epsilon^{-3})$	$\times$
D-SPIDER-SFO	$\mathcal{O}$ (Deg(network))	$\mathcal{O}(\epsilon^{-3})$	no need

al. 2018) and SARAH (Nguyen et al. 2017; 2019) achieve a gradient cost of $\mathcal{O}(\epsilon^{-3})$. Moreover, Fang et al. (2018) showed that SPIDER-SFO nearly achieves the algorithmic lower bound in certain regimes for nonconvex problems. Though these works have made significant progress, convergence properties of faster optimization algorithms for nonconvex problems in *the decentralized settings* are unclear.

In this paper, we propose **d**ecentralized SPIDER-SFO (D-SPIDER-SFO) for faster convergence rate for nonconvex problems. We theoretically analyze that D-SPIDER-SFO achieves an ϵ-approximate stationary point in gradient cost of $\mathcal{O}(\epsilon^{-3})$, which achieves the state-of-the-art performance for solving nonconvex optimization problems in the decentralized settings. Moreover, this result indicates that D-SPIDER-SFO achieves a similar gradient computation cost to its centralized competitor, called centralized SPIDER-SFO (C-SPIDER-SFO). To give a quick comparison of our algorithm and other existing first-order algorithms for nonconvex optimization in the decentralized settings, we summarize the gradient cost and communication complexity of the most relevant algorithms in Table1. Table 1 shows that D-SPIDER-SFO converges faster than D-PSGD and D^2 in terms of the gradient computation cost. Moreover, compared with C-SPIDER-SFO, D-SPIDER-SFO reduces much communication cost on the busiest worker. Therefore, D-SPIDER-SFO can outperform C-SPIDER-SFO when the communication becomes the bottleneck of the computational network. Our main contributions are as follows.

1. We propose D-SPIDER-SFO for finding approximate first-order stationary points for nonconvex problems in the decentralized settings, which is a decentralized parallel version of SPIDER-SFO.
2. We theoretically analyze that D-SPIDER-SFO achieves the gradient computation cost of $\mathcal{O}(\epsilon^{-3})$ to find an ϵ-approximate first-order stationary point, which is similar to SPIDER-SFO in the centralized network topology. To the best of our knowledge, D-SPIDER-SFO achieves the state-of-the-art performance for solving nonconvex optimization problems in the decentralized settings.

Notation: Let $\|\cdot\|$ be the vector and the matrix ℓ_2 norm and $\|\cdot\|_F$ be the matrix Frobenius norm. $\nabla f(\cdot)$ denotes the gradient of a function f. Let $\mathbf{1}_n$ be the column vector in $\mathbb{R}^n$ with 1 for all elements and e_i be the column vector with a 1 in the ith coordinate and 0's elsewhere. We denote by f^* the optimal solution of f. For a matrix $A \in \mathbb{R}^{n\times n}$, let $\lambda_i(A)$ be the i-th largest eigenvalue of a matrix. For any fixed integer $j \geq i \geq 0$, let $[i:j]$ be the set $\{i, i+1, \ldots, j\}$ and $\{x\}_{i:j}$ be the sequence $\{x_i, x_{i+1}, \ldots, x_j\}$.

Basics and Motivation

Decentralized Optimization Problems

In this section, we briefly review some basics of the decentralized optimization problem. We represent the decentralized communication topology with a weighted directed graph: (V, W). V is the set of all computational nodes, that is, $V := \{1, 2, \ldots, n\}$. W is a matrix and $W_{i,j}$ represents how much node j can affect node i, while $W_{ij} = 0$ means that node i and j are disconnected. Therefore, $W_{ij} \in [0, 1]$, for all i, j. Moreover, in the decentralized optimization settings, we assume that W is symmetric and doubly stochastic, which means that W satisfies (i) $W_{ij} = W_{ji}$ for all i, j, and (ii) $\sum_j W_{ij} = 1$ for all i and $\sum_i W_{ij} = 1$ for all j.

Throughout this paper, we consider the following decentralized optimization problem:

$$\min_{x\in\mathbb{R}^N} f(x) := \frac{1}{n}\sum_{i=1}^{n} \underbrace{\mathbb{E}_{\xi\in\mathcal{D}_i} F_i(x;\xi)}_{=:f_i(x)}, \tag{1}$$

where n is the number of workers, $\mathcal{D}_i$ is a predefined distribution of the local data for worker i, and ξ is a random data sample. Decentralized problems require that the graph of the computational network is connected and each worker can only exchange local information with its neighbors.

In the i-th node, $x_i, \xi_i, f_i(x_i), F_i(x_i;\xi_i)$ is the local optimization variables, random sample, target function and stochastic component function. Let $\mathcal{S}$ be a subset that samples S elements in the dataset. For simplicity, we denote by $\xi_{k,i}$ the subset that i-th node samples at iterate k, that is, $\nabla F_i(x_{k,i};\xi_{k,i}) = \nabla F_i(x_{k,i};\mathcal{S}_{k,i}) = \frac{1}{S_{k,i}}\sum_{\xi_j\in\mathcal{S}_{k,i}} \nabla F_i(x_{k,i};\xi_j)$. In order to present the core idea more clearly, at iterate k, we define the concatenation of all local optimization variables, estimators of full gradients, stochastic gradients, and full gradients by matrix $X_k, G_k, \partial F(X_k;\xi_k), \partial f(X_k) \in \mathbb{R}^{N\times n}$ respectively:

$$\begin{aligned}
X_k &:= [x_{k,1}, \ \cdots \ , x_{k,n}],\\
G_k &:= [g_{k,1}, \ \cdots \ , g_{k,n}],\\
\partial F(X_k, \xi_k) &:= [\nabla F_1(x_{k,1};\xi_{k,1}), \ \cdots \ , \nabla F_n(x_{k,n};\xi_{k,n})],\\
\partial f(X_k) &:= [\nabla f_1(x_{k,1}), \ \cdots \ , \nabla f_n(x_{k,n})].
\end{aligned}$$

In general, at iterate k, let the stepsize be η_k. We define $\eta_k V_k$ as the update, where $V_k \in \mathbb{R}^{N\times n}$. Therefore, we can view the update rule as:

$$X_{k+1} \leftarrow X_k W - \eta_k V_k. \tag{2}$$

D-SPIDER-SFO

In this section, we introduce the basic settings, assumptions, and the flow of D-SPIDER-SFO in the first subsection. Then, we compare D-SPIDER-SFO with D-PSGD and D^2 in a special scenario to show our core idea. In the final subsection, we propose the error-bound theorems for finding an ϵ-approximate first-order stationary point.

Algorithm 1 D-SPIDER-SFO on the ith node

```
Input: Require initial point $X_0$, weighted matrix $W$, number of iterations $K$, learning rate $\eta$, constant $q$, and two sample sizes $S^{(1)}$ and $S^{(2)}$
Initialize: $X_{-1} = X_0, G_{-1} = 0$
for $k = 0, \ldots, K-1$ do
    if mod$(k, q) = 0$ then
        Draw $S^{(1)}$ samples and compute the stochastic gradient $\nabla F_i(x_{k,i}; \mathcal{S}^{(1)}_{k,i})$
        $g_{k,i} = \nabla F_i(x_{k,i}; \mathcal{S}^{(1)}_{k,i})$
        $x_{k+\frac{1}{2},i} = 2x_{k,i} - x_{k-1,i} - \eta(g_{k,i} - g_{k-1,i})$
    else
        Draw $S^{(2)}$ samples, and compute two stochastic gradient $\nabla F_i(x_{k,i}; \mathcal{S}^{(2)}_{k,i})$ and $\nabla F_i(x_{k-1,i}; \mathcal{S}^{(2)}_{k,i})$
        $g_{k,i} = \nabla F_i(x_{k,i}; \mathcal{S}^{(2)}_{k,i}) - \nabla F_i(x_{k-1,i}; \mathcal{S}^{(2)}_{k,i}) + g_{k-1,i}$
        $x_{k+\frac{1}{2},i} = 2x_{k,i} - x_{k-1,i} - \eta(\nabla F_i(x_{k,i}; \mathcal{S}^{(2)}_{k,i}) - \nabla F_i(x_{k-1,i}; \mathcal{S}^{(2)}_{k,i}))$
    end if
    $x_{k+1,i} = \sum_{j=1}^{n} W_{j,i} x_{k+\frac{1}{2},j}$
end for
Return $\widetilde{x} = \frac{X_K \mathbf{1}_n}{n}$
```

Settings and Assumptions

In this subsection, we introduce the formal definition of an ϵ-approximate first-order stationary point and commonly used assumptions for decentralized optimization problems. Moreover, we briefly introduce the key steps at iterate k for worker i in D-SPIDER-SFO algorithm.

Definition 1 *We call $\widetilde{x} \in \mathbb{R}^N$ an ϵ-approximate first-order stationary point, if*

$$\|\nabla f(\widetilde{x})\| \leq \epsilon. \tag{3}$$

Assumption 1 *We make the following commonly used assumptions for the convergence analysis.*

1. ***Lipschitz gradient:*** *All local loss functions $f_i(\cdot)$ have L-Lipschitzian gradients.*
2. ***Average Lipschitz gradient:*** *In each fixed node i, the component function $F_i(x_i; \xi_i)$ has an average L-Lipschitz gradient, that is,*
$$\mathbb{E}\|\nabla F_i(x;\xi_i) - \nabla F_i(y;\xi_i)\|^2 \leq L^2\|x-y\|^2, \forall x, y.$$
3. ***Spectral gap:*** *Given the symmetric doubly stochastic matrix W. Let the eigenvalues of $W \in \mathbb{R}^{n\times n}$ be $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. We denote by λ the second largest value of the set of eigenvalues, i.e.,*
$$\lambda = \max_{i\in\{2,\cdots,n\}} \lambda_i = \lambda_2.$$
We assume $\lambda < 1$ and $\lambda_n > -\frac{1}{3}$.
4. ***Bounded variance:*** *Assume the variance of stochastic gradient within each worker is bounded, which implies there exists a constant σ, such that*
$$\mathbb{E}_{\xi\sim\mathcal{D}_i}\|\nabla F_i(x;\xi) - \nabla f_i(x)\|^2 \leq \sigma^2, \forall i, \forall x.$$
5. *(For D-PSGD Algorithm only)* ***Bounded data variance among workers:*** *Assume the variance of full gradient among all workers is bounded, which implies that there exists a constant ζ, such that*
$$\mathbb{E}_{i\sim\mathcal{U}([n])}\|\nabla f_i(x) - \nabla f(x)\|^2 \leq \zeta^2, \forall i, \forall x.$$

Remark 1 *The eigenvalues of W measure the speed of information spread across the network (Lian et al. 2017). D-SPIDER-SFO requires $\lambda_2 < 1$ and $\lambda_n > -\frac{1}{3}$, which is the same as the assumption in D^2 (Tang et al. 2018), while D-PSGD only needs the former condition. D-PSGD needs bounded data variance among workers assumption additionally, as it is sensitive to such data variance.*

D-SPIDER-SFO algorithm is a synchronous decentralized parallel algorithm. Each node repeats these four key steps at iterate k concurrently:

1. Each node computes a local stochastic gradient on their local data. When $\mod(k,q) \neq 0$, all nodes compute $\nabla F_i(x_{k,i}; \mathcal{S}^{(2)}_{k,i})$ and $\nabla F_i(x_{k-1,i}; \mathcal{S}^{(2)}_{k,i})$ using the local models at both iterate k and the last iterate; otherwise, they compute $\nabla F_i(x_{k,i}; \mathcal{S}^{(1)}_{k,i})$.
2. Each node updates its local estimator of the full gradient $g_{k,i}$. When $\mod(k,q) \neq 0$, all nodes compute $g_{k,i} \leftarrow \nabla F_i(x_{k,i}; \mathcal{S}^{(2)}_{k,i}) - \nabla F_i(x_{k-1,i}; \mathcal{S}^{(2)}_{k,i}) + g_{k-1,i}$; else they compute $g_{k,i} \leftarrow \nabla F_i(x_{k,i}; \mathcal{S}^{(1)}_{k,i})$.
3. Each node updates their local model. When $\mod(k,q) \neq 0$, all nodes compute $x_{k+\frac{1}{2},i} \leftarrow 2x_{k,i} - x_{k-1,i} - \eta(\nabla F_i(x_{k,i}; \mathcal{S}^{(2)}_{k,i}) - \nabla F_i(x_{k-1,i}; \mathcal{S}^{(2)}_{k,i}))$; else they compute $x_{k+\frac{1}{2},i} \leftarrow 2x_{k,i} - x_{k-1,i} - \eta(g_{k,i} - g_{k-1,i})$.
4. When meeting the synchronization barrier, each node takes weighted average with its and neighbors' local optimization variables: $x_{k+1,i} = \sum_{j=1}^{n} W_{j,i} x_{k+\frac{1}{2},j}$.

To understand D-SPIDER-SFO, we consider the update rule of global optimization variable $\frac{X_k \mathbf{1}_n}{n}$. Let $k_0 = \lfloor k/q \rfloor \cdot q$. For convenience, we define

$$\overline{\Delta}_k = \frac{(X_{k+1} - X_k)\mathbf{1}_n}{n} = \frac{1}{n}\sum_{i=1}^{n}(x_{k+1,i} - x_{k,i}),$$

$$\overline{H}_k(X) = \partial F(X; \xi_k)\frac{\mathbf{1}_n}{n} = \frac{1}{n}\sum_{i=1}^{n}\nabla F_i(x_i; \xi_{k,i}),$$

where ξ_k denotes the samples at the k-th iterate. Therefore,

$$\begin{aligned}
&\overline{\Delta}_k \\
=&\overline{\Delta}_{k-1} - \eta(\overline{H}_k(X_k) - \overline{H}_k(X_{k-1})) \\
=&-\eta \overline{H}_{k_0}(X_{k_0}) - \eta \sum_{s=k_0+1}^{k} (\overline{H}_s(X_s) - \overline{H}_s(X_{s-1})).
\end{aligned}$$

As for centralized SPIDER-SFO, we have

$$\begin{aligned}
&x_{k+1} - x_k \\
=&-\eta_k(\nabla F(x_k;\xi_k) - \nabla F(x_{k-1};\xi_k) + v_{k-1}) \\
=&-\eta_{k_0}\nabla F(x_{k_0};\xi_{k_0}) \\
&-\sum_{s=k_0+1}^{k} \eta_s(\nabla F(x_s;\xi_s) - \nabla F(x_{s-1};\xi_s)).
\end{aligned}$$

Remark 2 *Nguyen et al. propose SARAH for (strongly) convex optimization problems. SPIDER-SFO adopts a similar recursive stochastic gradient update framework and nearly matches the algorithmic lower bound in certain regimes for nonconvex problems. Moreover, Wang et al. [2] propose SpiderBoost and show that SpiderBoost, a variant of SPIDER-SFO with fixed step size, achieves a similar convergence rate to SPIDER-SFO for nonconvex problems. Inspired by these algorithms, we propose decentralized SPIDER-SFO (D-SPIDER-SFO). As we can see, the update rule of D-SPIDER-SFO is similar to its centralized counterpart with fixed step size.*

Core Idea

The convergence property of decentralized parallel stochastic algorithms is related to the variance of stochastic gradients and the data variance across workers. In this subsection, we present in detail the underlying idea to reduce the gradient complexity behind the algorithm design.

The general update rule (2) shows that $\mathbb{E}[\|V_k\|_F^2]$ affects the convergence, especially when we approach a solution. For showing the improvement of D-SPIDER-SFO, we will compare the upper bound of $\mathbb{E}[\|V_k\|_F^2]$ of three algorithms, which are D-PSGD, D^2, and D-SPIDER-SFO.

The update rule of D-PSGD is $X_{k+1} = X_k W - \eta \partial F(X_k;\xi_k)$, that is, $V_k = \partial F(X_k;\xi_k)$. Then, we have

$$\begin{aligned}
&\mathbb{E}\|\partial F(X_k;\xi_k)\|_F^2 \\
\leq&4\mathbb{E}\|\partial F(X_k;\xi_k) - \partial f(X_k)\|_F^2 \\
&+4\mathbb{E}\left\|\partial f(X_k) - \partial f\left(\frac{X_k\mathbf{1}_n}{n}\mathbf{1}_n^T\right)\right\|_F^2 \\
&+4\mathbb{E}\left\|\partial f\left(\frac{X_k\mathbf{1}_n}{n}\mathbf{1}_n^T\right) - \nabla f\left(\frac{X_k\mathbf{1}_n}{n}\right)\mathbf{1}_n^T\right\|_F^2 \\
&+4\mathbb{E}\left\|\nabla f\left(\frac{X_k\mathbf{1}_n}{n}\right)\mathbf{1}_n^T\right\|_F^2 \\
\leq&4n\sigma^2 + 4n\zeta^2 + 4L^2\sum_{i=1}^{n}\left\|x_{k,i} - \frac{X_k\mathbf{1}_n}{n}\right\|^2 \\
&+4\mathbb{E}\left\|\nabla f\left(\frac{X_k\mathbf{1}_n}{n}\right)\mathbf{1}_n^T\right\|_F^2.
\end{aligned}$$

Moreover, the update rule of D^2 is $X_{k+1} = [2X_k - X_{k-1} - \eta(\partial F(X_k;\xi_k) - \partial F(X_{k-1};\xi_{k-1}))]\,W$. For convenience, we define $Q_k = \frac{X_k - X_{k-1}}{\eta}$. Therefore, we can conclude the upper bound of $\mathbb{E}\|V_k\|_F^2$.

$$\begin{aligned}
&\mathbb{E}\,\|[-Q_k + (\partial F(X_k;\xi_k) - \partial F(X_{k-1};\xi_{k-1}))]\,W\|_F^2 \\
\leq&2\mathbb{E}\,\|Q_k\|_F^2 + 2\mathbb{E}\|\partial F(X_k;\xi_k) - \partial F(X_{k-1};\xi_{k-1})\|_F^2 \\
\leq&2\mathbb{E}\,\|Q_k\|_F^2 + 6\mathbb{E}\|\partial F(X_k;\xi_k) - \partial f(X_k)\|_F^2 \\
&+6\mathbb{E}\|\partial F(X_{k-1};\xi_{k-1}) - \partial f(X_{k-1})\|_F^2 \\
&+6\mathbb{E}\|\partial f(X_k) - \partial f(X_{k-1})\|_F^2 \\
\leq&\frac{2}{\eta^2}\sum_{i=1}^{n}\mathbb{E}\,\|x_{k,i} - x_{k-1,i}\|^2 \\
&+6\sum_{i=1}^{n}\mathbb{E}\|\nabla F_i(x_{k,i};\xi_{k,i}) - \nabla f_i(x_{k,i})\|^2 \\
&+6\sum_{i=1}^{n}\mathbb{E}\|\nabla F_i(x_{k-1,i};\xi_{k-1,i}) - \nabla f_i(x_{k-1,i})\|^2 \\
&+6\sum_{i=1}^{n}\mathbb{E}\|\nabla f_i(x_{k,i}) - \nabla f_i(x_{k-1,i})\|^2 \\
\leq&2\left(\eta^{-2} + 3L^2\right)\sum_{i=1}^{n}\mathbb{E}\|x_{k,i} - x_{k-1,i}\|^2 + 12n\sigma^2.
\end{aligned}$$

Since the update rule of D-SPIDER-SFO has two different patterns, we discuss them seperately. If $\mod(k,q) \neq 0$, we have $G_k = [-Q_k - (\partial F(X_k;\xi_k) - \partial F(X_{k-1};\xi_k))]W$.

$$\begin{aligned}
&\mathbb{E}\,\|[Q_k + (\partial F(X_k;\xi_k) - \partial F(X_{k-1};\xi_k))]\,W\|_F^2 \\
\leq&2\mathbb{E}\,\|Q_k\|_F^2 + 2\mathbb{E}\|\partial F(X_k;\xi_k) - \partial F(X_{k-1};\xi_k)\|_F^2 \\
\leq&\frac{2}{\eta^2}\sum_{i=1}^{n}\mathbb{E}\,\|x_{k,i} - x_{k-1,i}\|^2 \\
&+2\sum_{i=1}^{n}\mathbb{E}\|\nabla F_i(x_{k,i};\xi_{k,i}) - \nabla F_i(x_{k-1,i};\xi_{k,i})\|_F^2 \\
\leq&2\left(\eta^{-2} + L^2\right)\sum_{i=1}^{n}\mathbb{E}\|x_{k,i} - x_{k-1,i}\|^2.
\end{aligned}$$

If $\mod(k,q) = 0$ and $k > 0$, we have $G_k = [-Q_k - (\partial F(X_k;\xi_k) - G_{k-1})]W$. Let $k_0 = k - q$, and we have

$$\begin{aligned}
&\mathbb{E}\,\|[-Q_k - (\partial F(X_k;\xi_k) - G_{k-1})]\,W\|_F^2 \\
\leq&2\mathbb{E}\|Q_k\|_F^2 + 2\mathbb{E}\|\partial F(X_k;\xi_k) - G_{k-1}\|_F^2 \\
\leq&2\mathbb{E}\|Q_k\|_F^2 + 4\mathbb{E}\|\partial F(X_k;\xi_k) - \partial F(X_{k_0};\xi_{k_0})\|_F^2 \\
&+4\mathbb{E}\left\|\sum_{j=k_0}^{k-2}(\partial F(X_{j+1};\xi_{j+1}) - \partial F(X_j;\xi_{j+1})\right\|_F^2, \qquad (4)
\end{aligned}$$

where

$$\begin{aligned}
G_{k-1} = &\sum_{j=k_0}^{k-2}(\partial F(X_{j+1};\xi_{j+1}) - \partial F(X_j;\xi_{j+1})) \\
&+\partial F(X_{k_0};\xi_{k_0}).
\end{aligned}$$

Assume that for any $j \in [k_0 : k]$, X_j has achieved the optimum $X^* := x^* \mathbf{1}_n^T$ with all local models equal to the optimum x^*. Then, $\mathbb{E}[\|V_k\|_F^2]$ of D-PSGD, and D^2, is bounded by $\mathcal{O}(\sigma^2+\zeta^2)$, $\mathcal{O}(\sigma^2)$, which is similar to Tang et al. (2018). For convenience, considering the finite-sum case, if we set the batch size S_1 equal to the size m of the dataset, that is, we compute the full gradient at iteration k and k_0. Moreover, as for any $j \in [k_0 : k]$, $X_j = X^*$, then each term of (4) is zero, that is, $\mathbb{E}[\|V_k\|_F^2]$ is bounded by zero. D-SPIDER-SFO will stop at the optimum, while D-PSGD and D^2 will escape from the optimum because of the variance of stochastic gradients or data variance across workers. If we need D^2 stops at the optimum, D^2 should compute the full gradient at each iteration, which is similar to EXTRA (Wei et al. 2015), while D-SPIDER-SFO needs to compute full gradient per q iteration. This is the key ingredient for the superior performance of D-SPIDER-SFO. By this sight, D-SPIDER-SFO achieves a faster convergence rate. In the following analysis, we show that the gradient cost of D-SPIDER-SFO is $\mathcal{O}(\frac{1}{\epsilon^3})$.

Convergence Rate Analysis

In this subsection, we analyze the convergence properties of the D-SPIDER-SFO algorithm. We propose the error bound of the gradient estimation in Lemma 1, which is critical in convergence analysis. Then, based on Lemma 1, we present the upper bound of gradient cost for finding an ϵ approximate first-order stationary point, which is the state-of-the-art for decentralized nonconvex optimization problems.

Before analyzing the convergence properties, we consider the update rule of global optimization variables as follows,

$$\frac{X_{k+1}\mathbf{1}_n}{n} = \frac{(X_k W - \eta V_k)\mathbf{1}_n}{n} = \frac{(X_k - \eta V_k)\mathbf{1}_n}{n}.$$

To analyze the convergence rate of D-SPIDER-SFO, we conclude the following Lemma 1 which bounds the error of the gradient estimator $\frac{V\mathbf{1}_n}{n}$.

Lemma 1 *Under the Assumption 1, we have*

$$\begin{aligned}
&\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left\|\nabla f\left(\frac{X_k\mathbf{1}_n}{n}\right) - \frac{V_k\mathbf{1}}{n}\right\|^2 \\
\le&\left[\frac{4L^2C_1}{nD} + \frac{24L^4C_1q}{nDS_2}\right]\frac{\mathbb{E}\|X_1\|_F^2}{K} + \frac{2\sigma^2}{S_1} \\
&+\frac{1}{K}\left[\frac{4L^4\eta^4C_2}{Dn} + \frac{24L^4\eta^4C_2q}{DS_2}\right]\sum_{k=0}^{K-1}\mathbb{E}\left\|\frac{V_k\mathbf{1}_n}{n}\right\|^2,
\end{aligned}$$

where

$$\begin{aligned}
C_1 &= \max\left\{\frac{1}{1-|b_n|^2}, \frac{1}{(1-\lambda_2)^2}\right\},\\
C_2 &= \max\left\{\frac{\lambda_n^2}{(1-|b_n|^2)}, \frac{\lambda_2^2}{(1-\sqrt{\lambda_2})^2(1-\lambda_2)}\right\},\\
b_n &= \lambda_n - \sqrt{\lambda_n^2 - \lambda_n},\\
D &= 1 - \frac{48C_2 q\eta^2 L^2}{S_2}.
\end{aligned}$$

In Appendix, we will give the upper bound of $\mathbb{E}\|X_1\|_F^2$. Lemma 1 shows that the error bound of the gradient estimator is related to the second moment of $\left\|\frac{X_k\mathbf{1}_n}{n}\right\|$. Then, we give the analysis of the convergence rate. W.l.o.g., we assume the algorithm starts from 0, that is $X_0 = 0$, and define $\zeta_0 = \frac{1}{n}\sum_{i=1}^n \|\nabla f_i(\mathbf{0}) - \nabla f(\mathbf{0})\|$.

Theorem 1 *For the online case, set parameters S_1, S_2, η, and q as constants, and C_1, C_2, and D as in Lemma 1. Then, under the Assumption 1, for Algorithm 1, we have*

$$\begin{aligned}
&\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}\left\|\nabla f\left(\frac{X_k\mathbf{1}_n}{n}\right)\right\|^2 + \frac{M}{K}\sum_{k=0}^{K-1}\mathbb{E}\left\|\frac{V_k\mathbf{1}_n}{n}\right\|^2 \\
\le&\frac{2\mathbb{E}[f(\frac{X_0\mathbf{1}_n}{n}) - f^*]}{\eta K} \\
&+\left(1 + \frac{32C_2L^2\eta^2}{nqD} + \frac{192C_2L^2\eta^2}{nS_2D}\right)\frac{2\sigma^2}{S_1} \\
&+\frac{3\eta^2}{K}\left(\frac{4L^2C_1}{D} + \frac{24L^2C_1q}{DS_2}\right)(\sigma^2 + \zeta_0^2 + \|\nabla f(0)\|^2),
\end{aligned}$$

where

$$M := 1 - L\eta - \frac{6qL^2\eta^2}{S_2}\left[1 + \frac{4C_2L^2\eta^2}{D}\left(1 + \frac{6q}{S_2}\right)\right].$$

By appropriately specifying the batch size S_1, S_2, the step size η, and the parameter q, we reach the following corollary. In the online learning case, we let the input parameters be

$$S_1 = \frac{\sigma^2}{\epsilon^2}, S_2 = \frac{\sigma}{\epsilon}, q = \frac{\sigma}{\epsilon}, \tag{5}$$

$$\eta < \min\left(\frac{-1+\sqrt{13}}{12L}, \frac{1}{4\sqrt{3C_2}L}\right). \tag{6}$$

Corollary 1 *Set the parameters S_1, S_2, q, η as in* (5) *and* (6), *and set $K = \lfloor \frac{l}{\epsilon^2} \rfloor + 1$. Then under the Assumption 1, running Algorithm 1 for K iterations, we have*

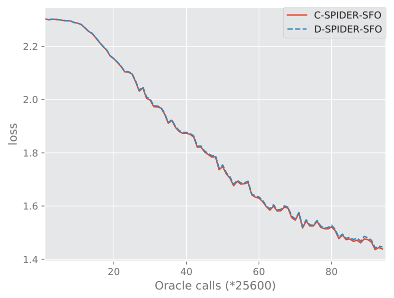
$$\begin{aligned}
&\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left\|\nabla f\left(\frac{X_k\mathbf{1}}{n}\right)\right\|^2 \\
\le&3\epsilon^2 + \frac{448C_2L^2\eta^2\epsilon^3}{nD\sigma},
\end{aligned}$$

where

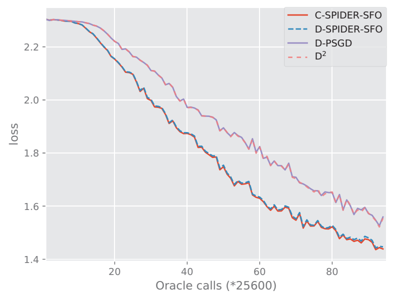
$$\begin{aligned}
l :=&\frac{2\mathbb{E}[f(0) - f^*]}{\eta} \\
&+\frac{84C_1L^2\eta^2}{D}(\sigma^2 + \zeta_0^2 + \|\nabla f(0)\|^2).
\end{aligned}$$

The gradient cost is bounded by $2l\sigma\epsilon^{-3} + 2\sigma^2\epsilon^{-2}$.

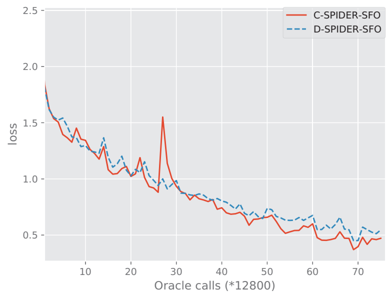
Remark 3 *Corollary 1 shows that measured by gradient cost, D-SPIDER-SFO achieves the convergence rate of $\mathcal{O}(\epsilon^{-3})$, which is similar to its centralized counterparts. Due to properties of decentralized optimization problems, the coefficient in Corollary 1 of the term ϵ^{-3} depends on the network topology W and the data variance among workers ζ^2 in addition, while compared with the centralized competitor (Fang et al. 2018). Although the differences exist, we conduct experiments to show that D-SPIDER-SFO converges with a similar speed to C-SPIDER-SFO.*



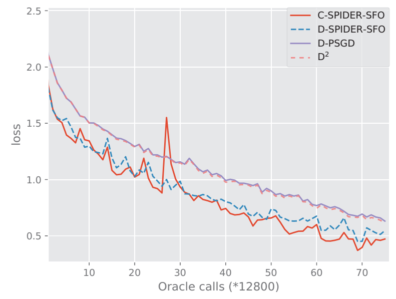
(a) SPIDER, LeNet5



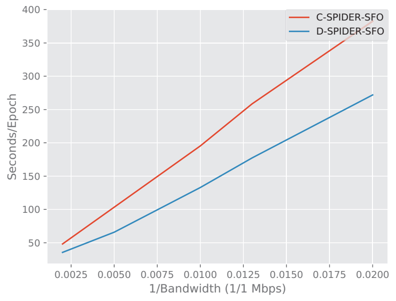
(b) SPIDER and SGD, LeNet5



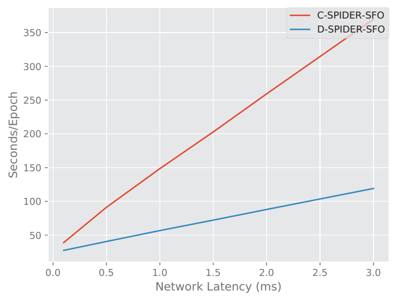
(c) SPIDER, ResNet-18



(d) SPIDER and SGD, ResNet-18



(e) Impact of Network Bandwidth



(f) Impact of Network Latency

Figure 1: In the experiments, we train two convolutional neural network, LeNet5 and ResNet18. Fig. 1(a) and 1(c) are the comparisons between D-SPIDER-SFO and C-SPIDER-SFO. Fig. 1(b) and 1(d) are the comparisons between D-SPIDER-SFO, C-SPIDER-SFO, D-PSGD, and D^2. Fig. 1(e) and 1(d) show the impact of the bandwidth and latency.

Experiments

In this section, we conduct extensive experiments to validate our theory. We introduce our experiment settings in the first subsection. Then in the second subsection, we conduct the experiments to demonstrate that D-SPIDER-SFO can get a similar convergence rate to C-SPIDER-SFO and converges faster than D-PSGD and D^2. Moreover, we validate that D-SPIDER-SFO outperforms its centralized counterpart, C-SPIDER-SFO, on the networks with low bandwidth or high latency. In the final, we show that D-SPIDER-SFO is robust to the data variance among workers. The code of D-SPIDER-SFO is available on GitHub at https://github.com/MIRALab-USTC/D-SPIDER-SFO.

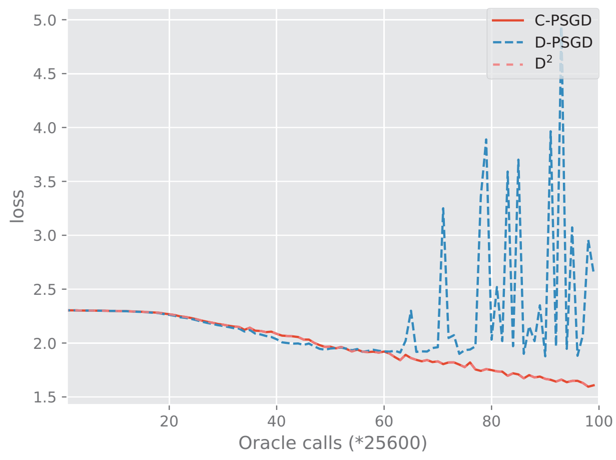
Experiment setting

Datasets and models We conduct our experiments on the image classification task. In our experiments, we train our models on CIFAR-10 (Krizhevsky and Hinton 2009). The CIFAR-10 dataset consists of 60,000 32x32 color images in 10 classes when the training set has 50,000 images. For image classification, we train two convolution neural network models on CIFAR-10. The first one is LeNet5 (Lecun et al. 1998), which consists of a 6-filter 5×5 convolution layer, a 2×2 max-pooling layer, a 16-filter 5×5 convolution layer and two fully connected layers with 120, 84 neurons respectively. The second one is ResNet-18 (He et al. 2015).
Implementations and setups We implement our code on framework PyTorch. All implementations are compiled with PyTorch1.3 with gloo. We conduct experiments both on the CPU server and GPU server. CPU cluster is a machine with four CPUs, each of which is an Intel(R) Xeon(R) Gold 6154 CPU @ 3.00GHz with 18 cores. GPU server is a machine with 8 GPUs, each of which is a Nvidia GeForce GTX 2080Ti. In the experiments, we use the ring network topology, seeing each core or GPU as a node, with corresponding symmetric doubly stochastic matrix W in the form of

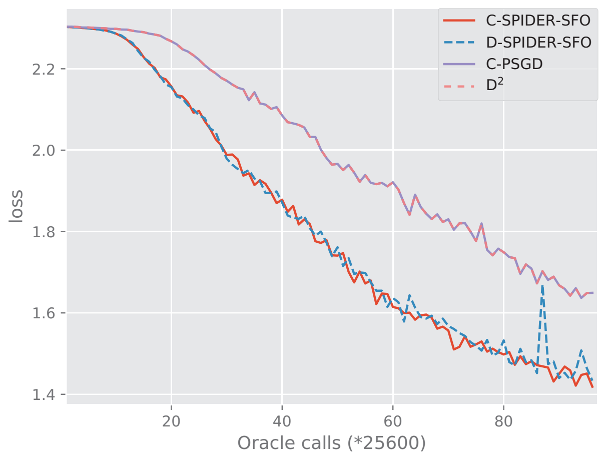
$$W = \begin{pmatrix} 1/2 & 1/4 & & & & 1/4 \\ 1/4 & 1/2 & 1/4 & & & \\ & 1/4 & 1/2 & \ddots & & \\ & & \ddots & \ddots & 1/4 & \\ & & & 1/4 & 1/2 & 1/4 \\ 1/4 & & & & 1/4 & 1/2 \end{pmatrix} \in \mathbb{R}^{n \times n}.$$

Experiments of D-SPIDER-SFO

To show that D-SPIDER-SFO can get a similar convergence rate to its centralized version, we choose the computational complexity as metrics instead of the wall clock speed. In the experiments of training LeNet5, for D-PSGD and D^2, we use the constant learning rate $\frac{\eta_0}{\sqrt{K/n}}$ and tune η_0 from $\{0.1, 0.05, 0.01, 0.005, 0.001\}$ and set the batch size 16 for each node, where K is the number of iterates and n is the number of workers. For D-SPIDER-SFO and C-SPIDER-SFO, we set $S_1 = 256, S_2 =$



(a) D-PSGD vs C-PSGD



(b) D-SPIDER-SFO vs C-SPIDER-SFO

Figure 2: Fig. 2(a) is the comparison between D-PSGD and its centralized parallel version when the data is unshuffled. Fig. 2(b) is the comparison between D-SPIDER-SFO with its centralized counterpart when the data is unshuffled.

$16, q = 16$ for each node, and tune the learning rate from $\{0.1, 0.05, 0.01, 0.005, 0.001\}$. When we conduct experiments on ResNet-18, for D-PSGD and D^2, we tune η_0 from $\{0.03, 0.01, 0.003, 0.001, 0.0003\}$, and also tune the learning rate of D-SPIDER-SFO and C-SPIDER-SFO from the same set $\{0.03, 0.01, 0.003, 0.001, 0.0003\}$. We conduct experiments on a computational network with eight nodes. Due to the space limitation, we show the experiments of training convolutional neural network models, LetNet5 and ResNet-18, on 8 GPUs in this paper and list the experiments on the CPU cluster in Supplement Material.

The gradient computation cost of both D-PSGD and D^2 is $\mathcal{O}(\epsilon^{-4})$ for finding an ϵ approximated stationary point, while D-SPIDER-SFO achieves $\mathcal{O}(\epsilon^{-3})$. Figure 1(b) and 1(d) validates our theoretical analysis and shows that D-SPIDER-SFO converges faster than D-PSGD and D^2. Moreover, figure 1(a) and 1(c) also shows that D-SPIDER-SFO achieves a similar convergence rate to its centralized competitor.

As the decentralized network has more balanced communication patterns, D-SPIDER-SFO should outperform its centralized counterpart, when the communication becomes the bottleneck of the computational network. To demonstrate the above statement, we use the wall clock time as the metrics. In this experiment, we train LeNet5 on a cluster with 8 GPUs. We adopt the same parameters and experiment settings as what we use to train LeNet5. We use the tc command to control the bandwidth and latency of the network. Figure 1(e) and 1(f) shows the wall clock time to finish one epoch on different network configurations. When the bandwidth becomes smaller, or the latency becomes higher, D-SPIDER-SFO can be even one order of magnitude faster than its centralized counterpart. The experiments demonstrate that the balanced communication pattern improves the efficiency of D-SPIDER-SFO.

Tang et al. (2018) proposed D^2 algorithm is less sensitive to the data variance across workers. From the theoretical analysis, D-SPIDER-SFO is also robust to that variance. The experiments demonstrate the statement and show that D-SPIDER-SFO converges faster than D^2 when the data variance across workers is maximized.

We follow the method proposed in (Tang et al. 2018) to create a data distribution with large data variance for the comparison between D-SPIDER-SFO and D^2. We conduct the experiments on a server with 5 GPUs and choose the computational complexity as metrics. Each worker only has access to two classes of the whole dataset, called the unshuffled case, and we tune the learning rate of D^2 as before.

Figure 2(a) shows that D-PSGD does not converge in the unshuffled case, which is consistent with the original work (Tang et al. 2018). Figure 2(b) shows that D-SPIDER-SFO converges faster than D^2, and even it has a similar computing complexity as its centralized implementation. The experiments demonstrate the theoretical statement that D-SPIDER-SFO is robust to the data variance across workers.

Conclusion

In this paper, we propose D-SPIDER-SFO as a decentralized parallel variant of SPIDER-SFO for a faster convergence rate for nonconvex problems. We theoretically analyze that D-SPIDER-SFO achieves an ϵ-approximate stationary point in the gradient cost of $\mathcal{O}(\epsilon^{-3})$. To the best of our knowledge, D-SPIDER-SFO achieves the state-of-the-art performance for solving nonconvex optimization problems on decentralized networks. Experiments on different network configurations demonstrate the efficiency of the proposed method.

Acknowledgements

We would like to thank the anonymous reviewers for their constructive suggestions for the paper. This work was supported in part by NSFC (61822604, 61836006).

References

Assran, M.; Loizou, N.; Ballas, N.; and Rabbat, M. 2019. Stochastic gradient push for distributed deep learning. In *Proceedings of the 36th International Conference on Machine Learning - Volume 97*, 344–353.

Dean, J.; Corrado, G.; Monga, R.; Chen, K.; Devin, M.; Mao, M.; aurelio Ranzato, M.; Senior, A.; Tucker, P.; Yang, K.; Le, Q. V.; and Ng, A. Y. 2012. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems 25*, 1223–1231.

Defazio, A.; Bach, F.; and Lacoste-Julien, S. 2014. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems 27*, 1646–1654.

Dekel, O.; Gilad-Bachrach, R.; Shamir, O.; and Xiao, L. 2012. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research* 165–202.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Fang, C.; Li, C. J.; Lin, Z.; and Zhang, T. 2018. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In *Advances in Neural Information Processing Systems 31*, 689–699.

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

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Jianmin, C.; Monga, R.; Bengio, S.; and Jozefowicz, R. 2016. Revisiting distributed synchronous sgd. In *International Conference on Learning Representations Workshop Track*.

Johnson, R., and Zhang, T. 2013. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems 26*, 315–323.

Krizhevsky, A., and Hinton, G. 2009. Learning multiple layers of features from tiny images. In *Technical Report*.

Lecun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, 2278–2324.

Lei, L.; Ju, C.; Chen, J.; and Jordan, M. I. 2017. Non-convex finite-sum optimization via scsg methods. In *Advances in Neural Information Processing Systems 30*, 2348–2358.

Li, M.; Andersen, D. G.; Park, J. W.; Smola, A. J.; Ahmed, A.; Josifovski, V.; Long, J.; Shekita, E. J.; and Su, B.-Y. 2014. Scaling distributed machine learning with the parameter server. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, 583–598.

Lian, X.; Zhang, C.; Zhang, H.; Hsieh, C.-J.; Zhang, W.; and Liu, J. 2017. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems 30*, 5330–5340.

Mokhtari, A., and Ribeiro, A. 2016. Dsa: Decentralized double stochastic averaging gradient algorithm. *Journal of Machine Learning Research* 2165–2199.

Nedić, A., and Ozdaglar, A. 2009. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control* 48–61.

Nguyen, L. M.; Liu, J.; Scheinberg, K.; and Takáč, M. 2017. Sarah: A novel method for machine learning problems using stochastic recursive gradient. In *Proceedings of the 34th International Conference on Machine Learning*, 2613–2621.

Nguyen, L. M.; Dijk, M. v.; Phan, D. T.; Nguyen, P. H.; Weng, T.-W.; and Kalagnanam, J. R. 2019. Finite-sum smooth optimization with sarah. In *arXiv preprint arXiv: 1901.07648*.

Reddi, S. J.; Hefny, A.; Sra, S.; Póczós, B.; and Smola, A. J. 2016. Stochastic variance reduction for nonconvex optimization. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, 314–323.

Robbins, H., and Monro, S. 1951. A stochastic approximation method. *The Annals of Mathematical Statistics* 400–407.

Roux, N. L.; Schmidt, M.; and Bach, F. R. 2012. A stochastic gradient method with an exponential convergence _rate for finite training sets. In *Advances in Neural Information Processing Systems 25*, 2663–2671.

Scaman, K.; Bach, F.; Bubeck, S.; Massoulié, L.; and Lee, Y. T. 2018. Optimal algorithms for non-smooth distributed optimization in networks. In *Advances in Neural Information Processing Systems 31*, 2740–2749.

Shi, W.; Ling, Q.; Yuan, K.; Wu, G.; and Yin, W. 2014. On the linear convergence of the admm in decentralized consensus optimization. *IEEE Transactions on Signal Processing* 1750–1761.

Tang, H.; Lian, X.; Yan, M.; Zhang, C.; and Liu, J. 2018. D^2: Decentralized training over decentralized data. In *Proceedings of the 35th International Conference on Machine Learning*, 4848–4856.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, 5998–6008.

Wei, S.; Qing, L.; Gang, W.; and Wotao, Y. 2015. Extra: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization* 944–966.

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

Zhou, D.; Xu, P.; and Gu, Q. 2018. Stochastic nested variance reduced gradient descent for nonconvex optimization. In *Advances in Neural Information Processing Systems 31*, 3921–3932.

Zinkevich, M.; Weimer, M.; Li, L.; and Smola, A. J. 2010. Parallelized stochastic gradient descent. In *Advances in Neural Information Processing Systems 23*, 2595–2603.