Communication Efficient Primal-Dual Algorithm for Nonconvex Nonsmooth Distributed Optimization

Congliang Chen[1,2] congliangchen@link.cuhk.edu.cn
Jiawei Zhang[1,2] 216019001@link.cuhk.edu.cn
Li Shen[3] mathshenli@gmail.com
Peilin Zhao[3] masonzhao@tencent.com
Zhi-Quan Luo[1,2] luozq@cuhk.edu.cn
[1]The Chinese University of Hong Kong, Shenzhen [2]Shenzhen Research Institute of Big Data [3]Tencent AI Lab

Abstract

Decentralized optimization frequently appears in large scale machine learning problems. However, few works have been focused on solving the decentralized optimization problems under the difficult nonconvex nonsmooth setting. In this paper, we propose a distributed primal-dual algorithm to solve this type of problems in a decentralized manner and the proposed algorithm can achieve an $\mathcal{O}(1/\epsilon^2)$ iteration complexity to attain an ϵ-solution, which is the well-known lower iteration complexity bound for nonconvex optimization. Furthermore, to reduce communication overhead, we also modify our algorithm by compressing the vectors exchanged between nodes. The iteration complexity of the algorithm with compression is still $\mathcal{O}(1/\epsilon^2)$. To our knowledge, it is the first algorithm achieving this rate under a nonconvex, nonsmooth decentralized setting with compression. Besides, we apply the proposed algorithm to solve nonconvex linear regression problem and train a deep learning model, both of which demonstrate the efficacy of the proposed algorithms.

1 Introduction

In this work, we consider the following nonconvex distributed optimization problem:

$$\min_{w\in\mathbb{R}^n} \frac{1}{N}\sum_{i=1}^{N} f_i(w).$$

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130.

Usually a network with N nodes is involved to solve this problem. We represent the network by an undirected graph $G=(V,E)$, where V is a node set and E is the set of edges. We denote $|V|=N$ and $|E|=M$. The node i can only asscess its own local function f_i and communicate with its immediate neighbors j, i.e. $(i,j)\in E$. Moreover, to embed some prior knowledge or tackle over-fitting problem, some regularization terms, which may be nonsmooth functions, will come into the formulation, such as adding constraints in Convolutional Neural Network (Pathak et al., 2015). Besides, as the growing size of models such as deep neural network, people consider to reduce the model size (He et al., 2017), some nonsmooth terms (such as sparsity terms (Yun et al., 2019), group sparsity terms (Wen et al., 2016), etc.) will be also formulated in the objective function. Then, the problem becomes:

$$\min_{w\in\mathbb{R}^n} \frac{1}{N}\sum_{i=1}^{N} f_i(w)+h_i(w), \tag{1}$$

where h_i can be a non-smooth function(e.g. ℓ_1 regularization or indicator function of some constraints on x), and here we assume h_i is a convex function. To efficiently solve the above problem, a common way is to introduce N local variables x_1, x_2,$\cdots$,x_N and define $x=(x_1^T,x_2^T,\cdots,x_N^T)^T$, then problem (1) can be formulated as follows:

$$\begin{aligned}\min_{x_1,\cdots,x_N} \frac{1}{N}\sum_{i=1}^{N} f_i(x_i)+h_i(x_i)\\ s.t.\ x_i=x_j,\ \forall (i,j)\in E.\end{aligned} \tag{2}$$

In general, we can use different regularization terms in different nodes, however, because of the consensus constraint $x_i=x_j$, we can combine all the h_i's to the first node, and the rest of h_i are zeros in this paper.

To solve problem (2), lots of methods have been proposed including many second-order methods (Roosta-

Khorasani & Mahoney, 2016b,a). However, because of intolerable computation complexity for higher-order algorithms, first-order methods are more popular for solving large-scale problems. In general, to solve problem (2), there are two different types of first-order methods. One is the gradient descent based method, where each node performs gradient steps and then averages x_i with its neighbors. Some works change the gradient steps by adding momentum (Yu et al., 2019), adaptive learning rate (Nazari et al., 2019) or adding gradient tracking variables (Scaman et al., 2018) and some works change the average step by other average schemes of x_i (e.g. weighted average (Tang et al., 2019)). The other kind of method is the primal-dual-based method, where dual variables (y) are introduced into the algorithm and using primal-dual type methods to seeking the saddle points of its Lagrangian function. Generally speaking, primal-dual methods consist of two parts, the primal update, and the dual update. The primal step is to minimize the Lagrangian function by a local update of primal variables and the dual step is to perform a dual ascent by using the consensus residual, where communications with neighbors are involved. The primal-dual method is usually more efficient than the gradient-based method, which is well-studied in convex cases (Chang et al., 2014) and nonconvex smooth cases (Hong et al., 2017). Also in Hong et al. (2016), they show that the ADMM algorithm, which is a primal-dual method, converges for consensus problems with a nonsmooth term in the centralized setting, where a central node controls the consensus, and achieves the $\mathcal{O}(1/\epsilon^2)$ iteration complexity. However, it is still unknown whether we can use the primal-dual method to solve problem (2) in the decentralized setting. More importantly, to our knowledge, it is still unclear whether there exists a decentralized primal-dual optimization algorithm to solve problem (2) that can achieve the $\mathcal{O}(1/\epsilon^2)$ iteration complexity, which is the well-known lower bound for the iteration complexity for solving nonconvex optimization problems using first-order method (Carmon et al., 2019).

Furthermore, with the success of large models such as deep neural networks, communication among nodes becomes an important factor influencing the speed of the optimization algorithms. A popular strategy to reduce communication complexity is to compress the vectors exchanged by neighbor nodes. Various kinds of compression functions (Koloskova et al., 2019) are used in these scenarios such as quantization functions and sparsification functions. Therefore, another important problem is whether we can design a communication efficient algorithm for solving problem (2) with low communication overhead.

In this paper, we give affirmative answers to both of the above two problems. Concretely speaking, we propose a smoothed proximal-primal-dual algorithm for solving problem (2) under the nonconvex nonsmooth setting. The algorithm can achieve an ϵ-solution of problem (2) within $\mathcal{O}(1/\epsilon^2)$ iteration complexity in terms of KKT-residual. Furthermore, to reduce the communication cost, we use a compressor when nodes communicate with their neighbors. We prove that our algorithm with compression of information exchanged between neighbor nodes can also achieve an $\mathcal{O}(1/\epsilon^2)$ iteration complexity if the compression is in sufficiently high accuracy. To our knowledge, it is the first algorithm achieving the lower iteration complexity bound for nonconvex nonsmooth optimization with compression during communication.

2 Related Work

Distributed optimization methods have been studied for many years. For convex cases, many algorithms have been proposed to solve distributed optimization problems, including the distributed subgradient method (Nedic & Ozdaglar, 2009), consensus ADMM method (Chang et al., 2014; Shi et al., 2014). Recently, nonconvex distributed optimization problems have also attracted more attention. Yuan et al. (2016) extend the decentralized gradient descent algorithm to the nonconvex smooth case and it achieves $\mathcal{O}(1/\epsilon^4)$ iteration complexity. Lian et al. (2017) give the convergence analysis of a gradient-based algorithm under the nonconvex but smooth setting with $\mathcal{O}(1/\epsilon^3)$ iteration complexity. Di Lorenzo & Scutari (2016) propose a decentralized gradient-based algorithm for solving problems in the nonconvex nonsmooth setting, they show the convergence of their algorithm without establishing its iteration complexity.

On the other hand, it is well-known that in convex cases, primal-dual algorithms are usually more efficient than decentralized gradient-based methods (Lan et al., 2020; Scaman et al., 2018; Wei & Ozdaglar, 2012). Though primal-dual methods are well-studied for convex problems, the related convergence rate for nonconvex cases is still unclear for many problems. Recently, some papers analyze the primal-dual algorithms for nonconvex cases. Hong et al. (2017) give the primal-dual algorithm and show the convergence rate under the nonconvex setting with the optimal order. For nonconvex, nonsmooth settings, Hong et al. (2016) analyze ADMM algorithm for a special type of graph with a center point. They prove that the iteration complexity can also achieve the order of $\mathcal{O}(1/\epsilon^2)$. Also, the gradient tracking algorithm proposed by Sun et al. (2019) can solve nonconvex, nonsmooth decentralized problems and can achieve $\mathcal{O}(1/\epsilon^2)$ iteration complexity.

Methods	Smooth Obj.	Smooth Obj. Compressed Comm.	Nonsmooth Obj.	Nonsmooth Obj. Compressed Comm.
Yuan et al. (2016)	$O(\epsilon^{-4})$	-	-	-
Tang et al. (2019)	$O(\epsilon^{-3})$	$O(\epsilon^{-3})$	-	-
Koloskova et al. (2019)	$O(\epsilon^{-3})$	$O(\epsilon^{-3})$	-	-
Di Lorenzo & Scutari (2016)	$\surd$	-	$\surd$	-
Hong et al. (2017)	$O(\epsilon^{-2})$	-	-	-
Scaman et al. (2018)	$O(\epsilon^{-2})$	-	$O(\epsilon^{-2})$	-
Ours	$O(\epsilon^{-2})$	$O(\epsilon^{-2})$	$O(\epsilon^{-2})$	$O(\epsilon^{-2})$

Table 1: Comparison among different methods. '-' represents the setting is not considered. '$\surd$' represents the setting is considered but iteration complexity is not given.

Later, to reduce the overhead of communication, people usually add a compression when communicating. For example, Ye et al. (2020) and Nedic & Ozdaglar (2009) discuss using quantization function during communication. For nonconvex case, Tang et al. (2018a) introduce the compression functions into decentralized gradient descent, and still give the iteration complexities at the order of $\mathcal{O}(\epsilon^{-3})$, but the algorithm only works for carefully designed compressors. In addition, Tang et al. (2019) and Koloskova et al. (2019) give the algorithms working for more general compression functions. However, the iteration complexity merely achieves the order of $\mathcal{O}(\epsilon^{-3})$. To our knowledge, there is no work in the literature achieving the optimal iteration complexity $\mathcal{O}(1/\epsilon^2)$ iteration complexity for nonconvex, nonsmooth decentralized optimization using compression when doing communication.

Different from previous work, we propose a primal-dual method that can achieve the optimal rate under the nonconvex nonsmooth setting. Besides, by adding the compression function in the communication, we reduce the communication overhead and show it will not hurt the convergence speed. For a better comparison, we summarize the most related works in Table 1.

3 Algorithm

To solve problem (2), we first reformulate it into a more compact form. Specifically, we first define the edge-agent incident matrix $W \in \mathbb{R}^{M\times N}$. The k_{th} row of W represents the k_{th} edge of the graph G. If (i,j) is the k_{th} edge in the graph, we set $W_{k,i}=1$, $W_{k,j}=-1$, and the rest entries in the k_{th} row are zeros.

For example, if the network graph G contains 4 nodes which are connected as shown in Fig. 1, then

$$W=\begin{bmatrix} 1 & -1 & 0 & 0\\ 0 & 1 & -1 & 0\\ 0 & 0 & 1 & -1\\ -1 & 0 & 0 & 1\end{bmatrix}.$$

To match the dimension of x_i, we let $A = W \otimes I_n$.

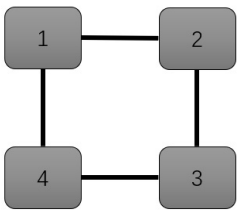


Figure 1: A network structure graph.

Then, we rewrite the consensus constraints of problem (2) as $Ax=0$. The problem (2) can be reformulated as the following form

$$\min_{x\in\mathbb{R}^{nN}} f(x)+h(x)$$
$$s.t.\ Ax=0,$$

where $f(x) = \frac{1}{N}\sum_{i=1}^N f_i(x_i)$ and $h(x) = \frac{1}{N}\sum_{i=1}^N h_i(x_i)$. The Lagrangian function of above problem is defined as follows:

$$L(x,y)=f(x)+h(x)+y^TAx.$$

Then, solving problem (2) is equivalent to finding the saddle points of this Lagrange function, which can be done simply by a primal-dual algorithm. When the strong convexity is absent, to solve the Lagrange function, people usually add (variant) augmented terms to give more stability of the primal-dual algorithm and the resulting algorithm is called the augmented Lagrangian method (ALM). However, the Augmented Lagrangian method (ALM) (Bertsekas, 1997) still does not guarantee to converge for nonconvex nonsmooth problems.

Inspired by Zhang & Luo (2020), we use the proximal-primal-dual framework to solve the Lagrangian function. For each iteration, we include a proximal term to the Lagrange function centered at an auxiliary sequence $\{z^t\}$, which is an exponentially weighted sequence of

the primal iterates. Then we use primal-dual type algorithm to approximately solve the saddle point of the proximal Lagrangian function and update z^t by taking a weighted average of z^t and x^{t+1}. Specifically, let

$$K(x,y,z) = f(x) + y^T Ax + \frac{p}{2}\|x-z\|^2. \qquad (3)$$

The steps of the algorithm can be written as:

$$\begin{aligned} x^{t+1} = \arg\min_x (&\langle \nabla_x K(x^t, y^t, z^t), x - x^t\rangle \\ &+h(x) + \tfrac{p}{2}\|x-z^t\|^2 + \tfrac{1}{2c}\|x-x^t\|^2); \\ y^{t+1} &= y^t + \alpha A x^{t+1}; \\ z^{t+1} &= z^t + \beta(x^{t+1} - z^t). \end{aligned}$$

Compared with ALM, here we do not use the augmented term in the Lagrangian function. The distributed implementation of the above steps is given in Algorithm 1. We need to see how to implement the above steps distributedly. However, because $y \in \mathbb{R}^{Mn}$, it can not be allocated to the N agents directly. Fortunately, when we update x^t using the gradient $\nabla_x K(x,y,z) = \nabla f(x) + A^T y + p(x-z)$, we only need the vector $A^T y$ instead of y itself. So we do not need to store y directly but store $\mu = A^T y$ instead. Further as $\mu \in \mathbb{R}^{Nn}$, we can easily allocate μ to the N agents. Let $\mu = (\mu_1^T, \cdots, \mu_N^T)^T$ and $K_i(x_i, \mu_i, z_i) = f_i(x_i) + \frac{p}{2}\|x_i - z_i\|^2 + \mu_i^T x_i$.

Algorithm 1 Distributed Primal-Dual Algorithm

1: Select $c > 0$, $\alpha > 0$, $0 < \beta \le 1$, and $p \ge 0$;
2: Initialize x_i^0, μ_i^0 and z_i^0;
3: **for** $t = 0, 1, 2, \cdots, T$ **do**
4: $\quad x_i^{t+1} = \arg\min_{x_i} \left(\langle \nabla_{x_i} K_i\left(x_i^t, \mu_i^t, z_i^t\right), x_i - x_i^t\rangle + h_i\left(x_i\right) + \frac{1}{2c}\|x_i - x_i^t\|^2\right)$;
5: $\quad$ Send x_i^{t+1} to $N(i)$ and receive x_j^{t+1} from $j \in N(i)$;
6: $\quad \mu_i^{t+1} = \mu_i^t + \alpha\left(d_i x_i^{t+1} - \sum_{j\in N(i)} x_j^{t+1}\right)$;
7: $\quad z_i^{t+1} = z_i^t + \beta\left(x_i^{t+1} - z_i^t\right)$;
8: **end for**

In each iteration, first, we update x_i^t in parallel by performing proximal gradient method with $K_i + h_i$. Then, we exchange x_i^{t+1}'s with its neighbors. After that, we update μ_i using x_i^{t+1} and the x_j^{t+1}'s from its neighbours. Lastly, we update z_i^t by taking average of z_i^t and x_i^{t+1}. Then, x_i, μ_i, z_i can be updated in distributed manner.

To reduce the communication overhead, we add compression at each iteration when the agents communicate with their neighbours. To keep track of the true x_i^t, any neighbour of i needs to control a tracking variable $\hat{x}_i^t$ of x_i^t and update $\hat{x}_i^t$ by using the compressed difference $C\left(x_i^{t+1} - \hat{x}_i^t\right)$. The compressed primal-dual algorithm is given in Algorithm 2.

Algorithm 2 Communication Efficient Distributed Primal-Dual Algorithm

1: Select $c > 0$, $\alpha > 0$, $0 < \beta \le 1$, $p \ge 0$ and compression function $C(\cdot)$;
2: Initialize x_i^0, $\hat{x}_i^0 = x_i^0$, μ_i^0 and z_i^0;
3: **for** $t = 0, 1, 2, \cdots, T$ **do**
4: $\quad x_i^{t+1} = \arg\min_{x_i} \left(\langle \nabla_{x_i} K_i\left(x_i^t, \mu_i^t, z_i^t\right), x_i - x_i^t\rangle + h_i(x_i) + \frac{1}{2c}\|x_i - x_i^t\|^2\right)$;
5: $\quad$ Send $C\left(x_i^{t+1} - \hat{x}_i^t\right)$ to $N(i)$ and receive $C\left(x_j^{t+1} - \hat{x}_j^t\right)$ from $j \in N(i)$;
6: $\quad \hat{x}_i^{t+1} = \hat{x}_i^t + C\left(x_i^{t+1} - \hat{x}_i^t\right)$;
7: $\quad \hat{x}_j^{t+1} = \hat{x}_j^t + C\left(x_j^{t+1} - \hat{x}_j^t\right)$, for $j \in N(i)$;
8: $\quad \mu_i^{t+1} = \mu_i^t + \alpha\left(d_i \hat{x}_i^{t+1} - \sum_{j\in N(i)} \hat{x}_j^{t+1}\right)$;
9: $\quad z_i^{t+1} = z_i^t + \beta\left(x_i^{t+1} - z_i^t\right)$;
10: **end for**

3.1 The intuition of our algorithms and the error-bound-based framework

To overcome the nonconvexity, we add a proximal term $\frac{p}{2}\|x-z\|^2$ to the objective function and then the following problem is equivalent to problem (2):

$$\begin{aligned} &\min_{x,z\in\mathbb{R}^{nN}} f(x) + \frac{p}{2}\|x-z\|^2 + h(x) \\ &s.t.\ Ax = 0. \end{aligned}$$

Let $M(z) = \min_{x:Ax=0} f(x) + \frac{p}{2}\|x-z\|^2 + h(x)$. Then solving (2) is equivalent to solve

$$\min_z P(z). \qquad (4)$$

We denote $x^*(z) = \arg\min_{x:Ax=0}(f(x) + h(x) + \frac{p}{2}\|x - z\|^2)$. By Danskin's Theorem, the gradient of $M(z)$ is given by

$$\nabla_z M(z) = p(z - x^*(z)).$$

Furthermore, in Algorithm 1, the updates for x^t and y^t can be viewed as approximately solving $x^*(z^t)$ with an error $x^{t+1} - x^*(z^t)$. Let $x(y,z) = \arg\min_x(K(x,y,z) + h(x))$, then the error can be further decomposed as the sum of a primal error $x^{t+1} - x(y^t, z^t)$ and a dual error $x(y^t, z^t) - x^*(z^t)$. Moreover, when updating the dual variables, we also suffer from a communication error $x^t - \hat{x}^t$ in Algorithm 1. The primal error occurs because we use only one step of proximal gradient when minimizing $K(x, y^t, z^t) + h(x)$ and the dual error appears since y^t is not an optimal dual vector. The communication error is because of the compression. According to the above analysis, when updating x, y, we can reduce some primal-dual potential function and when updating z, we approximately reduce $M(z)$. Let

$$d(y,z) = \min_x K(x,y,z) + h(x)$$

and define the potential function as

$$\phi(x,y,z) = K(x,y,z) + h(x)) - 2d(y,z) + 2M(z).$$

We want to prove that $\phi^t = \phi(x^t, y^t, z^t)$ is decreasing and bounded below. To prove the decrease of the potential function, we need to carefully bound the errors. Here, we give lemmas for the error bounds above.

Lemma 1 (The primal error bound). *Suppose $p > -\frac{L}{N}$ and $c \leq \frac{N}{L+Np}$, it holds that*

$$\|x^{t+1} - x(y^t, z^t)\| \leq \frac{1-\sigma_2}{\sigma_2}\|x^{t+1} - x^t\|,$$

where $\sigma_2 = \frac{c(Np-c)}{2N}$.

Lemma 2 (The dual error bound). *For all $y \in \mathbb{R}^{Mn}$ and all $z \in \mathbb{R}^{Nn}$, it holds that*

$$\|x(y,z) - x^*(z)\| \leq \sigma_4 \|Ax(y,z)\|,$$

where σ_4 is related to the p, L and the graph property on the first node.

The remaining part is to deal with the compression error. We bound the compression error by the difference on x^t and x^{t+1}, which we give in the following Lemma.

Lemma 3. *With the definition of x and $\hat{x}$ in Algorithm 2, the following equality always holds:*

$$\sum_{t=1}^{T} \|x^t - \hat{x}^t\|^2 \leq \frac{(1-\delta)^2}{\delta^2} \sum_{t=1}^{T} \|x^t - x^{t-1}\|^2.$$

Equipped with these error bounds, we can prove that our potential function is decreasing and this implies the convergence of our algorithm. The formal theorem will be given in the next section. Due to the space limitation, the proofs of the above lemmas are placed in the **supplementary file** (Lemma 12, Lemma 3 and Lemma 1).

4 Theoretical Analysis

In this section, we establish the convergence result of Algorithm 1 and Algorithm 2.

4.1 The stationary solution of problem (2)

First, we give the definition of stationary point and the approximate stationary points of problem (2).

We say that x is a (first-order) stationary point of problem (2) if x satisfies the first-order (KKT) condition, i.e. if there exists a y such that

$$0 \in \partial h(x) + \nabla f(x) + A^T y$$
$$Ax = 0$$

We then define the ϵ-stationary point as follows:

Definition 1. *(x, y) is called ϵ-stationary point if $\frac{L}{\sqrt{N\lambda_2}}\|Ax\| \leq \epsilon$ and there exists ν, such that $\nu \in \nabla f(x) + \partial h(x) + A^T y$ and $\sqrt{N}\|\nu\| \leq \epsilon$, where λ_2 is the smallest nonzero eigenvalue of $A^T A$. Also we let λ_1 to be the largest eigenvalue of $A^T A$ for later use.*

Remark 1. *The ϵ-stationary point in Definition 1 is an ϵ^2-solution in Hong et al. (2017); Tang et al. (2019), where the ϵ^2-solution in Hong et al. (2017) is defined to be an x with $\|\frac{1}{N}\sum_{i=1}^N \nabla f_i(x_i)\|^2 + \frac{L}{N\lambda_2}\sum_{(i,j)\in E}\|x_i - x_j\|^2 \leq \epsilon^2$, and the ϵ^2-solution in Tang et al. (2019) is defined to be the point with $\|\frac{1}{N}\sum_{i=1}^N \nabla f_i\left(\frac{1}{N}\sum_{i=1}^N x_i\right)\|^2 \leq \epsilon^2$. The detailed proof can be found in the* **supplementary file** *due to the limited space.*

4.2 Assumptions

In this subsection, we state our assumptions for the functions f and h, compression function $C(\cdot)$, and the connectivity of communication network, which will be used in the theoretical analysis.

Assumption 1. *For the functions f and h, we assume:*

1. *There exists an $w^* \in \mathbb{R}^n$ such that for all $w \in \mathbb{R}^n$, $\sum_{i=1}^N f_i(w^*) + \sum_{i=1}^N h_i(w^*) \leq \sum_{i=1}^N f_i(w) + \sum_{i=1}^N h_i(w)$. Let $\underline{f} = f(w^*) + h(w^*)$*
2. *Function f_i is differentiable function with L-Lipschitz continuous gradient, i.e., for all i, we have*

$$\|\nabla f_i(w) - \nabla f_i(w')\| \leq L\|w - w'\|, \ \forall w, w' \in \mathbb{R}^n.$$

3. *h_i is a proper convex, closed function.*

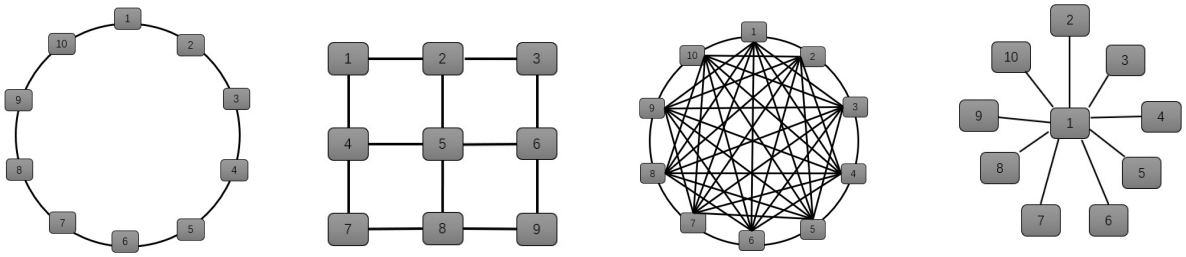
By the gradient Lipschitz continuity of f, it is easy to check that $K(x,y,z)$ defined in (3) also has a Lipschitz-continuous gradient with respect to variable x. Here, we denote its Lipschitz constant as $L_K = L/N + p$. Next, we give the assumption of the cmpression function $C(\cdot)$.

Assumption 2. *The compression function $C(\cdot)$ satisfies the following inequality:*

$$\|C(w) - w\| \leq (1-\delta)\|w\| \text{ for some } \delta > 0, \text{ and } \forall w \in \mathbb{R}^n.$$

Remark 2. *We give two examples of compression function:*

- *Top_k function, which preserves top_k values of vector x and sets the rest of entries to be zero, satisfies the Assumption 2 and has been used in Koloskova et al. (2019).*



Ring Grid Complete Star

Figure 2: Network structure in Table 2.

- *Define* $I(x,\Delta) = \begin{cases} -1, & x < -\Delta \\ 1, & x > \Delta \\ 0, & otherwise \end{cases}$. *Then, function* $\|x\|_\infty I(x, \|x\|_\infty/2)$ *is a compression function satisfying Assumption 2, which maps* $\mathbb{R}^n$ *to* $\mathbb{R} \times \{-1,0,1\}^n$ *and has been used in Wen et al. (2017).*

For the communication network, we assume that it is connected so that the information can be sent through different nodes.

Assumption 3. *The graph G is connected.*

4.3 The main theoretical result

Under the above assumptions, we have the following main theoretical result:

Theorem 1. *Suppose the parameters* $c \le \frac{1}{L_K}$, $p > -L/N$, *and* α, β *are chosen below some thresholds. Then it holds that for any* $T > 0$, *there exists* $s \in \{0,1,\cdots,T-1\}$ *such that* $\left(x^{s+1}, y^s\right)$ *is a* $B/\sqrt{T}$*-solution, where*

$$B = \mathcal{O}\left(\sqrt{\left(\Phi\left(x^0,y^0,z^0\right) - \underline{f}\right)}(f_1(c) + f_2(\alpha) + f_3(\beta))\right),$$

$$f_1(c) = \left(\left(\left(1 + \frac{\sqrt{\lambda_1}}{\sqrt{\lambda_2 c}} + p\right)\sqrt{N} + \frac{L}{\sqrt{N}}\right)\sqrt{c},\right.$$

$$f_2(\alpha) = \frac{L}{\sqrt{N\lambda_2\alpha}},\ \text{and}\ f_3(\beta) = \left(\sqrt{p} + \frac{\sqrt{\lambda_1}\beta L}{\sqrt{\lambda_2 p N}}\right)\frac{1}{\sqrt{\beta}}.$$

Remark 3. *The thresholds of* α *and* β *are related to p,* δ*, c, L and connectivity of the graph, which are defined in the proof of the theorem in the* **supplementary file**.

Remark 4. *Using the result in the Theorem 1, to achieve* ϵ*-stationary point,* $O(\frac{1}{\epsilon^2})$ *iterations are needed, which is the well-known lower bound of iteration complexity for the nonconvex case (Carmon et al., 2019).*

Corollary 1. *Suppose we choose the parameters* $c = \frac{N}{3L}$, $p = \frac{2L}{N}$, $\alpha = \frac{L\delta^2}{24\lambda_1(2\delta^2+(1-\delta)^2))}$ *and* $\beta = \frac{\delta^2}{1536\gamma((1-\delta)^2+2\delta^2)+12\delta^2}$, *where* γ *is a constant related to the structure of the graph. For any* $T > 0$, *there exists* $s \in \{0,1,\cdots,T-1\}$, *such that* (x^{s+1}, y^s) *is a* $B_1/\sqrt{T}$*-solution, where* $B_1 = \mathcal{O}\left(\sqrt{\left(\Phi\left(x^0,y^0,z^0\right) - \underline{f}\right)}\sqrt{\frac{L\gamma((1-\delta)^2+1)}{\delta^2}}\right)$, *where* γ *is a constant related to connection of graph.*

Remark 5. *For the smooth case where* $h(\cdot) = 0$, γ *is just the spectral gap of the Laplacian matrix defined as* λ_1/λ_2, *the division of the largest eigenvalue with the smallest non-zero eigenvalue which is widely used in the distributed optimization (Sun & Hong, 2019).*

For the nonsmooth case, we define $\tilde{\mathcal{L}}$ *as an* $(N-1)n \times (N-1)n$ *matrix generating by eliminating the rows and the columns corresponding to the first agent of the Laplacian matrix. Then the* γ *is the division of the largest eigenvalue of the Laplacian matrix with the smallest eigenvalue of* $\tilde{\mathcal{L}}$.

In the following Table 2, we calculate parameter γ *for four types of graph structures defined in the Fig. 2 as instance in smooth and nonsmooth settings, respectively.*

Graph Structure	Smooth	Nonsmooth
Ring(10 nodes)	3.9021	10.4721
Grid(3×3)	6	33.9973
Complete(10 nodes)	1	10
Star(10 nodes)	10	10

Table 2: Different γ under different settings.

Remark 6. *By taking* $\delta = 1$ *in Theorem 1 and Corollary 1, we get the convergence result of Algorithm 1.*

5 Experimental Results

In this section, we conduct extensive experiments to demonstrate the efficacy of the proposed Algorithms 1 and 2. We compare our proposed algorithms with a decentralized gradient descent based algorithm CHOCO(S)GD (Koloskova et al., 2019), a primal-dual algorithm Prox_PDA (Hong et al., 2017) and a gradient-tracking algorithm D^2 (Tang et al., 2018b) on two tasks. First, we implement the algorithms to solve a

nonconvex linear regression, and then we implement the algorithm to train a deep neural network.

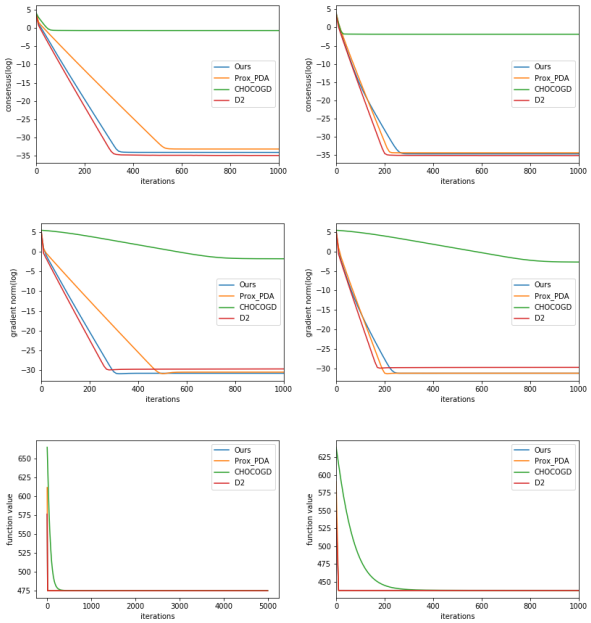
5.1 Nonconvex Linear Regression

To show the efficacy of our proposed algorithms, we start with a least square linear regression problem with a nonconvex regularizer. Specifically, we set

$$f_i(w) = \frac{1}{2}\|B_i w - b_i\|^2 + 5\sum_{j=1}^{n} \frac{w_j^2}{w_j^2+1},$$

where the regularizer is a sparsity-inducing term.

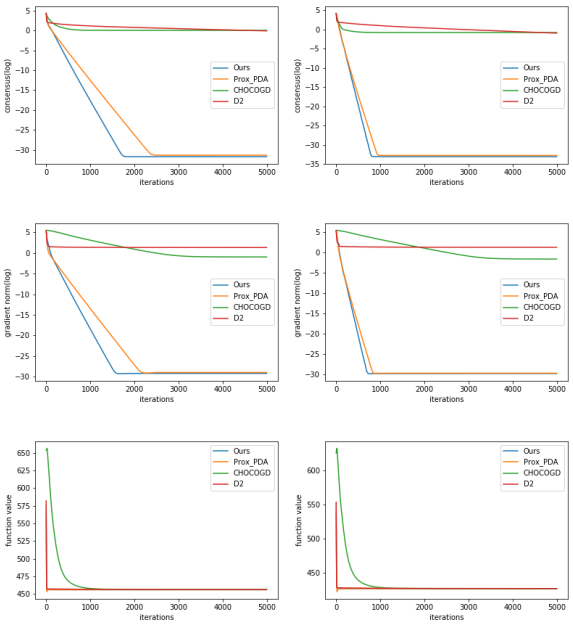
In the experiment we randomly generate data pairs $(B_i, b_i) \in \mathbb{R}^{100\times 100} \times \mathbb{R}^{100}$. Two graph network structures are used in the experiment: a ring with 10 nodes and a 3×3 grid graph in Figure 2. The hyperparameters of all compared algorithms are tuned by grid search. We report the experimental results with 10 times repeat.



ring grid

Figure 3: Results on Smooth and Full Precision Settings.

In Fig. 3, we only consider smooth objective function f, and give the results on the consensus residual $\sum_{i=1}^N \|x_i - \bar{x}\|^2$, the norm of gradient of $\sum_{i=1}^N f_i(\bar{x})$ and the function value evaluated at $\bar{x}$, where $\bar{x} = \frac{1}{N}\sum_{i=1}^N x_i$. In the view of consensus residual and gradient norm, the primal-dual methods and D^2 can converge much faster than gradient descent method (CHOCOGD), and can attain a more accurate solution. Besides, our algorithm converges with similiar speed to Prox_PDA but slower than D^2. For the function value, three methods can converge to the similar function value but CHOCOGD is slowest.



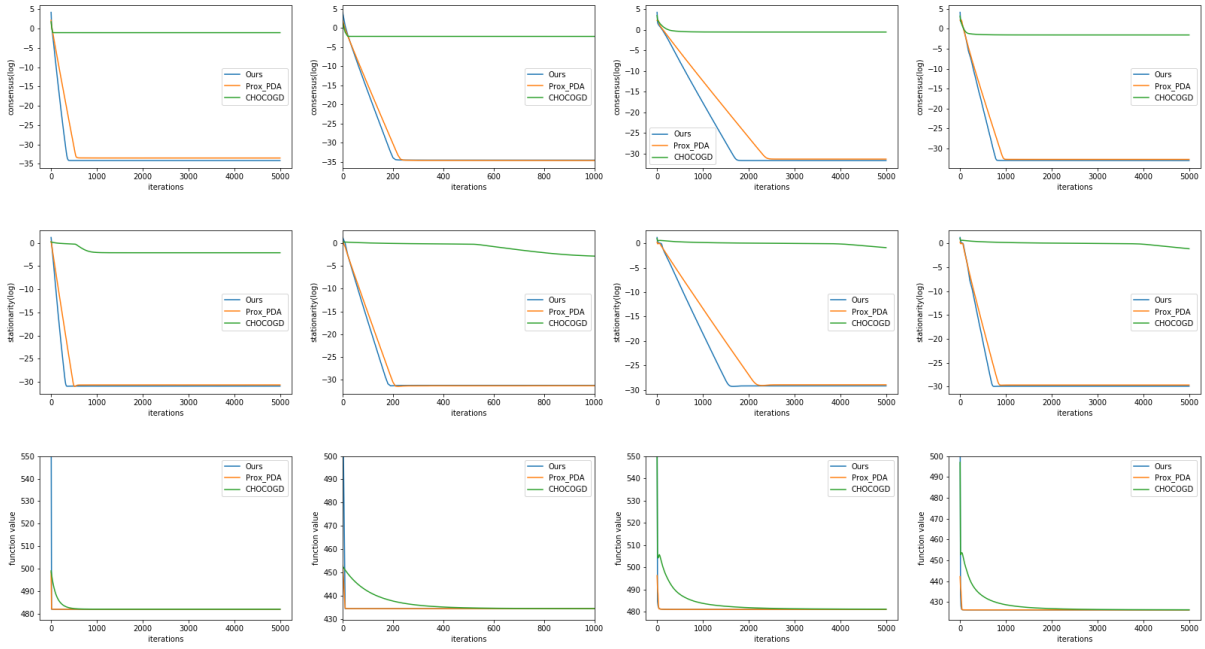
ring grid

Figure 4: Results on Smooth and Compressed Settings.

Next, we add compression to the communication variables by using top_k sparsification function (Koloskova et al., 2019). We select k to be 5 and give the experimental results of different algorithms in Fig. 4. Still, we evaluate the solution with consensus residuals, gradient norms, and the function values as before. As it is shown in Fig. 4, our methods and prox_PDA can attain better solutions than CHOCOGD and D^2. Moreover, due to our error-bound-based design, our algorithm performs fairly better than the compressed version of prox_PDA.

Next, we consider a nonsmooth objective, where the nonsmooth term $h(w)$ is chosen to be the indicator function: of the unit ball: $h(w) = \begin{cases} 0, & \|x\| \le 1 \\ \infty, & \|x\| > 1 \end{cases}$. In other word, we add a unit-ball constraint to the optimization variables. Because D^2 does not consider the nonsmooth case, we don't compare our algorithms with D^2.

For the CHOCOGD and Prox_PDA, because only smooth versions are proposed, we simply extend these two methods by replacing gradient descent step with gradient projection step. We still calculate the consensus residual of x_i's and the function values evaluated at $\bar{x}$. To estimate the stationarity of the solution $\bar{x}$, we plot the norm of the proximal gradient. Concretely speaking, we perform a gradient projection step on $\bar{x}$ with stepsize 1, and denote the point attained by gradient projection by $\hat{x}$. Then $\|\hat{x} - \bar{x}\|$ is used as a



ring (full precision) grid (full precision) ring (compressed) grid (compressed)

Figure 5: Results on Nonsmooth Settings.

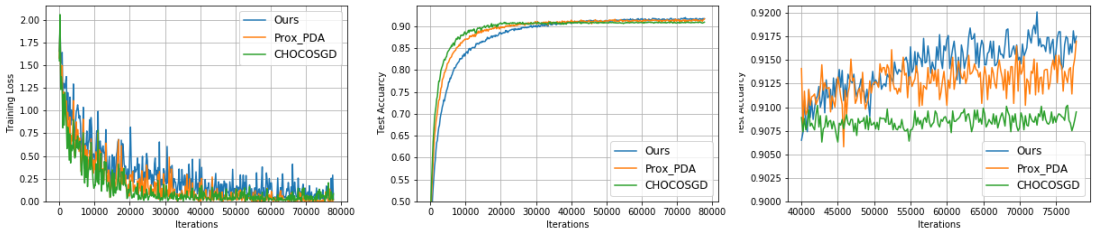


Figure 6: Results on CIFAR10 with Full Precision Communication.

measure of stationarity. We show the results for the nonconvex nonsmooth objective in Fig. 5. In Fig. 5, the upper row is the results on consensus, the middle row is the results on stationarity and the bottom row shows the results on function values. Although in our algorithm, the constraint is only added in the first node, our algorithm can converge faster than CHOCOGD and Prox_PDA. Moreover, for both the nonsmooth and compression setting, our algorithm converges much faster than Prox_PDA.

5.2 Neural Network Case

In this section, we show the results on training ResNet-18 (He et al., 2016) on the dataset CIFAR10 (Krizhevsky et al., 2009). Still, we use the top_k sparsification function as a compression function, and set k to be $0.1n$, where n is the dimension of weight x. We use a ring with 10 nodes as the communication network structure. For the hyper-parameter, all three algorithms use the learning rate $1e-2$, in our algorithm we use $p = 5$ for $\|x - z\|$ term, and the rest parameters are tuned by grid search with respect to the training accuracy. We use batch size as 32, and simply extend our method and Prox_PDA to the stochastic version by replacing the gradient with the stochastic gradient (mini-batch gradient). To measure the performance of the compared algorithms, we plot the training loss on the first node and the classification accuracy on the test set. We do not have a comparable result on algorithm D^2, we illustrate results of CHCOSGD, Prox_PDA, and our algorithm only. Experimental results are illustrated in Figures 6-7.

Fig. 6 shows the results with full precision communication. The left figure shows the training loss, the middle figure shows the test accuracy, and the right figure shows the zooming-in version of the accuracy curve in the last few iterations. It can be seen that our algorithm converges slower than SGD and Prox_PDA at the beginning iterations, because the proximal term in-

troduces more errors at the beginning. But finally, our algorithm becomes faster and can get a better solution than CHOCOSGD and Prox_PDA in 200 epochs.

Besides, in Fig. 7, we show the results of Algorithm 2 with compressed communication. The left figure is the training loss and the right figure is test accuracy. Unfortunately, Prox_PDA fails to get a good solution, we do not draw the curve of Prox_PDA in Fig. 7. Although our algorithm converges a little bit slower than CHOCOSGD because of the inexact of dual variables and large proximal error in the initial phase, our algorithm becomes faster than CHOCOSGD in about 20,000 iterations and finally gets higher accuracy in the 200 epochs compared to CHOCOSGD.

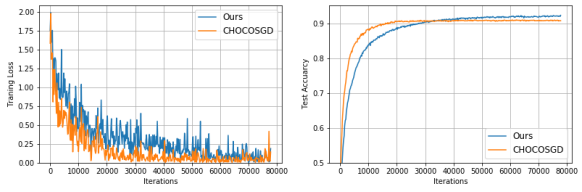


Figure 7: Results on CIFAR10 with Compressed Communication.

6 Conclusion

In this paper, we proposed a decentralized primal-dual algorithm to solve the nonconvex nonsmooth distributed optimization problems. To reduce the communication overhead we use the compression function during the communication. We show that for the nonconvex nonsmooth case the algorithm can converge to the ϵ-stationary point with $\mathcal{O}(\frac{1}{\epsilon^2})$ iterations, which is a well-known lower bound for nonconvex optimization. The experimental results on nonconvex linear regression and deep neural networks show the efficacy of the proposed algorithms.

Acknowledgements

This work is done when Congliang Chen and Jiawei Zhang are research interns at Tencent AI Lab, China. Li Shen is the corresponding author. The work is supported by the National Natural Science Foundation of China (No. 61731018) and the Shenzhen Fundamental Research Fund (No. KQTD201503311441545).

References

Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

Yair Carmon, John C Duchi, Oliver Hinder, and Aaron Sidford. Lower bounds for finding stationary points i. *Mathematical Programming*, pp. 1–50, 2019.

Tsung-Hui Chang, Mingyi Hong, and Xiangfeng Wang. Multi-agent distributed optimization via inexact consensus admm. *IEEE Transactions on Signal Processing*, 63(2):482–497, 2014.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1389–1397, 2017.

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

Mingyi Hong, Davood Hajinezhad, and Ming-Min Zhao. Prox-pda: The proximal primal-dual algorithm for fast distributed nonconvex optimization and learning over networks. In *International Conference on Machine Learning*, pp. 1529–1538, 2017.

Anastasia Koloskova, Sebastian U Stich, and Martin Jaggi. Decentralized stochastic optimization and gossip algorithms with compressed communication. *arXiv preprint arXiv:1902.00340*, 2019.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Guanghui Lan, Soomin Lee, and Yi Zhou. Communication-efficient algorithms for decentralized and stochastic optimization. *Mathematical Programming*, 180(1):237–284, 2020.

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

Parvin Nazari, Davoud Ataee Tarzanagh, and George Michailidis. Dadam: A consensus-based distributed adaptive gradient method for online optimization. *arXiv preprint arXiv:1901.09109*, 2019.

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

Deepak Pathak, Philipp Krahenbuhl, and Trevor Darrell. Constrained convolutional neural networks for weakly supervised segmentation. In *Proceedings of*

the IEEE international conference on computer vision, pp. 1796–1804, 2015.

Farbod Roosta-Khorasani and Michael W Mahoney. Sub-sampled newton methods ii: Local convergence rates. *arXiv preprint arXiv:1601.04738*, 2016a.

Farbod Roosta-Khorasani and Michael W. Mahoney. Sub-sampled newton methods i: Globally convergent algorithms, 2016b.

Kevin Scaman, Francis Bach, Sébastien Bubeck, Laurent Massoulié, and Yin Tat Lee. Optimal algorithms for non-smooth distributed optimization in networks. In *Advances in Neural Information Processing Systems*, pp. 2740–2749, 2018.

Wei Shi, Qing Ling, Kun Yuan, Gang Wu, and Wotao Yin. On the linear convergence of the admm in decentralized consensus optimization. *IEEE Transactions on Signal Processing*, 62(7):1750–1761, 2014.

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

Ying Sun, Amir Daneshmand, and Gesualdo Scutari. Convergence rate of distributed optimization algorithms based on gradient tracking. *arXiv preprint arXiv:1905.02637*, 2019.

Hanlin Tang, Shaoduo Gan, Ce Zhang, Tong Zhang, and Ji Liu. Communication compression for decentralized training. In *Advances in Neural Information Processing Systems*, pp. 7652–7662, 2018a.

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

Hanlin Tang, Xiangru Lian, Shuang Qiu, Lei Yuan, Ce Zhang, Tong Zhang, and Ji Liu. Deepsqueeze : Parallel stochastic gradient descent with double-pass error-compensated compression. *arXiv preprint arXiv:1907.07346*, 2019.

Ermin Wei and Asuman Ozdaglar. Distributed alternating direction method of multipliers. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pp. 5445–5450. IEEE, 2012.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*, pp. 2074–2082, 2016.

Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in neural information processing systems*, pp. 1509–1519, 2017.

Tian Ye, Peijun Xiao, and Ruoyu Sun. Deed: A general quantization scheme for communication efficiency in bits. *arXiv preprint arXiv:2006.11401*, 2020.

Hao Yu, Rong Jin, and Sen Yang. On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization. *arXiv preprint arXiv:1905.03817*, 2019.

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

Jihun Yun, Peng Zheng, Eunho Yang, Aurelie Lozano, and Aleksandr Aravkin. Trimming the l1 regularizer: Statistical analysis, optimization, and applications to deep learning. In *International Conference on Machine Learning*, pp. 7242–7251, 2019.

Jiawei Zhang and Zhi-Quan Luo. A proximal alternating direction method of multiplier for linearly constrained nonconvex minimization. *SIAM Journal on Optimization*, 30(3):2272–2302, 2020.