Multi-consensus decentralized primal-dual fixed point algorithm for distributed learning

Kejie Tang[y], Weidong Liu[z], and Xiaojun Mao[x]

Abstract. Decentralized distributed learning has recently attracted significant attention in many applications in machine learning and signal processing. To solve a decentralized optimization with regularization, we propose a Multi-consensus Decentralized Primal-Dual Fixed Point (MD-PDFP) algorithm. We apply multiple consensus steps with the gradient tracking technique to extend the primal-dual fixed point method over a network. The communication complexities of our procedure are given under certain conditions. Moreover, we show that our algorithm is consistent under general conditions and it enjoys global linear convergence under strong convexity. With some particular choices of regularizations, our algorithm can be applied to decentralized machine learning applications. Finally, several numerical experiments and real data analyses are conducted to demonstrate the effectiveness of the proposed algorithm.

Key words. Decentralized optimization, multi-consensus, gradient tracking, proximal gradient

MSC codes. 65K10, 49M15

1. Introduction.

Decentralized distributed learning has become increasingly popular in multi-agent decision-making in wireless networks[28], multi-robot coordination[7], smart power grids[16], the Internet of Things (IoT)[3], and others[34, 6, 51]. In decentralized learning, one essential problem is solving the decentralized optimization

$$\min_{x\in\mathbb{R}^p} \frac{1}{n}\sum_{i=1}^{n} f_i(x) + (h \square B)(x), \tag{1.1}$$

where $f_i(x)$, $i \in V = \{1, 2, \ldots, n\}$ denote n local loss functions distributed in the vertex set V of an undirected and connected network $G = (V, E)$, h is a common regularization function, $B \in \mathbb{R}^{q\times p}$ is a matrix of the linear transformation in the regularization, and $\square$ indicates the compound operator. Let $\Gamma_0(\mathbb{R}^p)$ be a set of proper, lower semi-continuous convex functions on $\mathbb{R}^p$. Throughout this study, we assume $f(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x) \in \Gamma_0(\mathbb{R}^p)$ and $h \in \Gamma_0(\mathbb{R}^q)$.

Optimization (1.1) covers a wide class of popular machine learning methods. For example, when B is the identity operator, (1.1) includes the Lasso[48], penalized logistic regression[13], penalized M estimation[22], and many other machine learning methods. It also includes

Corresponding authors: W. Liu and X. Mao.

Funding: Weidong Liu's research is supported by National Program on Key Basic Research Project (2018AAA0100704), NSFC Grant No. 11825104 and 11690013, Youth Talent Support Program, Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102). Xiaojun Mao's research is supported by NSFC Grant No. 12001109, the Science and Technology Commission of Shanghai Municipality grant 20dz1200600.

[y]School of Mathematical Sciences, Shanghai Jiao Tong University, Shanghai 200240, China (tangkj00@sjtu.edu.cn).

[z]School of Mathematical Sciences, MoE Key Lab of Artificial Intelligence, Shanghai Jiao Tong University, Shanghai 200240, China (weidongl@sjtu.edu.cn).

[x]School of Mathematical Sciences, Ministry of Education Key Laboratory of Scientific and Engineering Computing, Shanghai Jiao Tong University, Shanghai 200240, China (maoxj@sjtu.edu.cn).

This manuscript is for review purposes only.

2.1. Group Lasso. The group Lasso penalty [50, 21] is typically used to estimate parameters with a group structure. Consider the following optimization:

$$\min_{x\in\mathbb{R}^p} \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\|y_i - A_i x\|_2^2 + \lambda_N \sum_{j=1}^{m}\|[x]_{G_j}\|_2, \tag{2.6}$$

where $y_i \in \mathbb{R}^d$; $A_i \in \mathbb{R}^{d\times p}$, are the observed responses/covariates, G_j is the set of the j-th group and $[x]_{G_j}$ is the sub-vector of x with elements in G_j. Let $\{a:b\}$ be the index set from a to b and set $S_j = \{\sum_{k=1}^{j-1}|G_k|+1 : \sum_{k=1}^{j}|G_k|\}$. Now, we take f_i and h in (1.1) by $f_i(x) = 1/2\|y_i - A_i x\|_2^2$ and $h(v) = \lambda_N \sum_{j=1}^{m}\|[v]_{S_j}\|_2$. The group Lasso penalty can be written as $\lambda_N \sum_{j=1}^{m}\|[x]_{S_j}\|_2 = h(Bx)$, where $B = (b_{i,j})$ is a $\sum_{j=1}^{m}|G_j| \times p$ matrix, with the sub-matrices satisfying $[B]_{S_j,G_j} = I_{|G_j|}$, for $j = 1,\ldots,m$. It is easy to see that the proximity operator prox_h has an explicit form

$$[\mathrm{prox}_h(v)]_{S_j} = \max\{\|[v]_{S_j}\|_2 - \lambda_N, 0\}\frac{[v]_{S_j}}{\|[v]_{S_j}\|_2},$$

for $j = 1,\ldots,m$.

If $A = (A_1^\top,\ldots,A_n^\top)^\top \in \mathbb{R}^{nd\times p}$ has full column rank, then $f = 1/n\sum_{i=1}^{n} f_i$ is strongly convex. If the m groups $\{G_j : 1 \le j \le m\}$ are disjoint, then B is full row rank with $\lambda_{\min}(BB^\top) > 0$. By Theorem 3.4, with a fixed K round of communications in each iteration, the algorithm converges to the true solution with linear rates. On the other hand, for general A and the general structure of groups, our algorithm still exhibits a consistency rate; see Theorem 3.8.

2.2. Fused Lasso. Consider the following optimization:

$$\min_{x\in\mathbb{R}^p} \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\|y_i - A_i x\|_2^2 + \lambda_N \|Bx\|_1. \tag{2.7}$$

With some appropriate choices of B, $\|Bx\|_1$ will become to the Fused Lasso penalty [49, 50]. In particular, let

$$B = B_{1d} = \begin{pmatrix} 1 & -1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & -1 \end{pmatrix}.$$

Then $\|Bx\|_1 = \sum_{j=1}^{p-1}|x_j - x_{j+1}|$.

The above regularization function is called the fused penalty (also known as 1d Fused Lasso), which is designed for problems with features that can be ordered in some meaningful way; see [49]. We now take $f_i(x) = 1/2\|y_i - A_i x\|_2^2$ and $h(v) = \lambda_N \|v\|_1$ in (1.1) with $v \in \mathbb{R}^{q\times p}$ and $q = p - 1$. In this example, the proximity operator of h becomes the soft thresholding. It can be easily seen that the matrix $B_{1d}B_{1d}^\top$ is a tri-diagonal matrix and $\lambda_{\min}(B_{1d}B_{1d}^\top) > 0$. Under

the full column rank condition on $A \in R^{nd \times p}$, the convergence of our algorithm with a linear convergence rate is given in Theorem 3.4.

We next consider the 2d Fused Lasso in [49], which can be used to denoise images with a possible piecewise constant structure. Let $p = p_1 \times p_2$ and $x \in R^p$ by $\{x_{j;k}; j = 1; \ldots; p_1; k = 1; \ldots; p_2\}$ be the signals in an image. Let $B = B_{2d}$ be a $(2p - p_1 - p_2) \times p$ matrix such that

$$B_{2d} = \begin{pmatrix} D_{p_1} \otimes I_{p_2} \\ I_{p_1} \otimes D_{p_2} \end{pmatrix}; \quad \text{with} \quad D_{p_l} = \begin{pmatrix} -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -1 & 1 \end{pmatrix} \in R^{(p_l - 1) \times p_l};$$

$l = \{1; 2\}$, where $\otimes$ denotes the Kronecker product. Then

$$\|Bx\|_1 = \sum_{j=1}^{p_1} \sum_{k=1}^{p_2 - 1} |x_{j;k} - x_{j;k+1}| + \sum_{k=1}^{p_2} \sum_{j=1}^{p_1 - 1} |x_{j;k} - x_{j+1;k}|.$$

The 2d Fused Lasso can determine the piecewise constant structure in each row/column in the image matrix. Note that in 2d Fused Lasso, the matrix $B_{2d} B_{2d}^\top$ is no longer full rank. In this case, the proposed algorithm remains operational.

2.3. Network Lasso. Let $G = \{V; E\}$ be a graph, where V is the vertex set and E denotes the set of edges. [18] proposed the network Lasso over G by the following problem:

$$\min_{X \in R^{n \times p}} \frac{1}{n} \sum_{i \in V} f_i(x_i) + \lambda_N \sum_{(j;k) \in E} w_{jk} \|x_j - x_k\|_2; \tag{2.8}$$

where $|V| = n; |E| = m; X = (x_1^\top; \ldots; x_n^\top)^\top \in R^{n \times p}$ and w_{jk} are the relative weights among the network edges. [18] discussed many common problems in control systems and statistical learning that can be formulated as an instance of the network Lasso.

In [18, 61], the authors proposed ADMM-type algorithms to solve the problem (2.8) and its variants. In each iteration, every node needs to solve an optimization problem. As a special case with $f_i(x_i) = 1/2 \|y_i - A_i x_i\|_2^2$, every node involves the computation of the inverse matrix $(I_p + A_i A_i^\top)^{-1}$ which is computationally expensive.

In this paper, we apply the proposed procedure Algorithm 2.1 to solve the problem (2.8). Let $H = (h_{l;i}) \in R^{m \times n}$ be the incident matrix of the graph G with m edges and n nodes. And the matrix is defined as

$$h_{l;i} = \begin{cases} w_{jk}; & \text{if } i = s(l); \\ -w_{jk}; & \text{if } i = e(l); \\ 0; & \text{otherwise;} \end{cases}$$

where $s(l)$ and $e(l)$ denote the starting and ending node indices of edge l respectively, with $s(l) < e(l)$ and $\{s(l); e(l)\} = \{j; k\}$.

Using H, the optimization problem (2.8) can be reformulated as

$$\min_{X \in R^{n \times p}} \frac{1}{n} \sum_{i=1}^{n} f_i((e_i^\top \otimes I_p)X) + h((H \otimes I_p)X);$$

where $e_i = (0, \ldots, 0, 1, 0, \ldots, 0)^\top \in \mathbb{R}^n$ with one non-zero element on the i-th entry, $\otimes$ is the Kronecker product and the non-smooth function h is defined as $h(v) = \sum_{j=1}^{m} \|v_j\|_2$ for $v = (v_1^\top, \ldots, v_m^\top)^\top \in \mathbb{R}^{mp}$.

When the graph is connected and $m = n - 1$, which means that the graph is a tree, the matrix $B = H \otimes I_p \in \mathbb{R}^{(mp) \times (np)}$ has full row rank with $\lambda_{\min}(BB^\top) > 0$. If $f = 1/n \sum_{i=1}^{n} f_i$ is strongly convex, then linear convergence is guaranteed in Theorem 3.4. In [61], a tree graph was constructed based on the ordinary least squares estimates for better clustering, and the optimization problem is similar to (2.8). Subsequently, Algorithm 2.1 can also be used for this problem with the linear convergence guarantee. As for generally connected graphs, our algorithm is still consistent with Theorem 3.8.

3. Convergence Analysis.

To establish the convergence results for our procedure, we start from some common assumptions as follows.

Assumption 3.1. The network $G = \{V, E\}$ is undirected and connected. The nodes set is $V = \{1, 2, \ldots, n\}$ and the edge set is $E \subseteq V \times V$.

In decentralized learning, undirected and connected network topologies are common assumptions in the literature. Under this assumption, many consensus and averaging algorithms satisfy $\mathcal{W}$-average consensus in Definition 2.1; see Remark 2.2.

Assumption 3.2. For each $i = 1, 2, \ldots, n$, f_i is differential with L_i-smoothness, satisfying that for $x, y \in \mathbb{R}^p$

$$\|\nabla f_i(x) - \nabla f_i(y)\|_2 \le L_i \|x - y\|_2,$$

where $L_i > 0$ is constant. Then $f(x) = 1/n \sum_{i=1}^{n} f_i(x)$ is smooth with $L = \max_{i=1, \ldots, n} L_i$.

Many loss functions satisfy the L-smoothness condition, for example, the quadratic loss, logistic loss, and Huber loss. For regularization, we allow h to be non-smooth, such that it covers many commonly used regularizers, see Section 2. The next assumption is on the existence of a solution to the problem (1.1).

Assumption 3.3. Problem (1.1) has at least one solution.

This is a fairly weak condition and holds when the target function is coercive (see Proposition 3.1 in [10]); that is, $\lim_{\|x\|_2 \to +\infty} (f(x) + (h \circ B)(x)) = +\infty$. We are now ready to present the main results. We first state the linear convergence rate under the strong convexity of $f(x)$. A function $f(x)$ is μ-strongly convex if there exists some $\mu > 0$ such that

$$\langle \nabla f(x) - \nabla f(y), x - y \rangle \ge \mu \|x - y\|_2^2 \quad \text{for } x, y \in \mathbb{R}^p.$$

Furthermore, if B has full row rank with $\lambda_{\min} BB^\top > 0$, then there is a unique solution x^* of (1.1); see Chen et al. [9].

Theorem 3.4. Suppose that Assumptions 3.1-3.3 hold. In addition, we assume that $f = 1/n \sum_{i=1}^{n} f_i$ is μ-strongly convex. Matrix B has full row rank with $\lambda_{\min}^2$, $\lambda_{\min} BB^\top > 0$. Let $0 < \alpha < 2/L$, $0 < \beta \le 1/\lambda_{\max} BB^\top$, $1/\lambda_{\max}^2$, and K satisfy

$$K = \frac{\log(1/\varepsilon) + \log(7 + 6L + 2\alpha\lambda_{\max} + 3\beta(1 + \alpha\lambda_{\max}) + 4\alpha\lambda_{\max}(1 + L))}{\log(1/\rho)}; \tag{3.1}$$

where $\quad 6\ (1 \quad {}^2)=(48(1+C_1)^2), \quad = \quad \sqrt{\quad 1 \quad \min f \quad (2 \quad L\ );\ {}^2_{\min} g}$ and $C_1 = 6+2= {}_{\max} + L+4=L$.

Subsequently, for $t > 1$,

$$\text{(3.2)} \qquad kx_t \quad x\ k_2^2 + \quad kv_t \quad v\ k_2^2\ 6\ \frac{1+\ {}^2}{2}^{\ t} C_0^2; \qquad \text{and}$$

$$\text{(3.3)} \qquad \max\left\{ V_t \quad 1v_t^{>}\ {}_2;\ X_t \quad 1x_t^{>}\ {}_2;\ G_t \quad 1g_t^{>}\ {}_2 \right\}^{o} \quad 6\ \frac{1+\ {}^2}{2}^{\ t=2}\ p\overline{n}\,(2(1+C_1)=C_2+12)\ C_0;$$

where

$$C_0^2 = kx_0 \quad x\ k_2^2 + \quad kv_0 \quad v\ k_2^2 + \frac{C_2^2}{n} \max\left\{ V_0 \quad 1v_0^{>}\ {}_2;\ X_0 \quad 1x_0^{>}\ {}_2;\ G_0 \quad 1g_0^{>}\ {}_2 \right\}^{o_2};$$

$$C_2 = \sqrt{\frac{2(3 \quad {}^2)}{1 \quad {}^4}}(6+2= {}_{\max} + L+4=L):$$

To achieve $X_T \quad 1x^{>}\ {}^2_2 = O(\ )$, let $\quad = 1=L$ and $\quad = \min f\ 1= {}^2_{\max};\ =(L \quad {}^2_{\min})g$, then the computational and communication complexities of Algorithm 2.1 are

$$T = O\ \max f\ L=;\ {}^2_{\max}= {}^2_{\min} g \log \quad {}^{1} \quad ; \qquad \text{and}$$

$$Q = O\ \frac{\max f\ L=;\ {}^2_{\max}= {}^2_{\min} g}{1} \log \frac{g_1(L;\ ;\ {}_{\max};\ {}_{\min})}{g_2(L;\ ;\ {}_{\max};\ {}_{\min})} \log \quad \frac{1}{} \quad ;$$

respectively, where g_1 and g_2 are polynomials of $L;\ ;\ {}_{\max};\ {}_{\min}$.

Remark 3.5. The formula for $g_1(L;\ ;\ {}_{\max};\ {}_{\min})$ and $g_2(L;\ ;\ {}_{\max};\ {}_{\min})$ is

$$\frac{g_1(L;\ ;\ {}_{\max};\ {}_{\min})}{g_2(L;\ ;\ {}_{\max};\ {}_{\min})} = \max\ \frac{L}{}; \frac{{}^2_{\max}}{{}^2_{\min}} \quad 7+\frac{2}{{}_{\max}} + L + \frac{4}{L}^{\ 2} \quad 7+6L+2\ {}_{\max} + \frac{3(1+\ {}_{\max})}{L} + \frac{4(1+L)}{{}_{\max}} \ :$$

Remark 3.6. From the inequalities (3.2) and (3.3), we can see that the values of X_t in every node converge to the solution x with linear rates. This rate depends on the condition number $L=$ of the loss function f and the condition number $({}_{\max}= {}_{\min})^2$ of the matrix $BB^{>}$. Speci cally, for $B = I_p$, the rate becomes to $O(\ 1 \quad \frac{}{2L}^{\ t})$. This covers the convergence rate of the popular proximal gradient method in a centralized setting. Using the Nesterov acceleration technique to obtain a faster rate, as in Ye et al. [57] is also possible. This requires more detailed analysis and is left for future work.

Remark 3.7. We discuss the number of communication rounds between nodes for each iteration. The main factor for K in (3.1) is $1=\log(1=\ )\quad(1\quad)^{\ 1}$. Note that $\ $ is the contraction coe cient of A. The size of $\ $ depends on the network topology of G. In particular, $\ =\ _2$ for (2:4) is used, and $\ =1\quad\sqrt{1\quad_2}$ for (2.5). If we take W as the lazy Metropolis matrix (see (4.1)), then $_2<1\quad1=(71n^2)$ for all undirected and connected networks [35]. For networks with special structures, such as the Erdos-Renyi random network, expander graph, and complete graph, $_2$ can be much smaller than 1, see [32]. For these networks, a small number of communication rounds is required for each iteration. Finally, it is also worth noting that K is independent of the iteration number t.

When the strong convexity on f and the full row rank condition on B are not satis ed, the following theorem shows that our procedure is still consistent with a slightly larger K.

Theorem 3.8. Suppose that Assumptions 3.1-3.3 hold. Let $\ $ and $\ $ be the same as those in Theorem 3.4. Take any $xed\ >0$ and choose

$$K_t=\frac{\log\frac{(t+1)^{\ 2}}{}+\log(7+6\ L+2\ _{max}+3\ \ (1+\ _{max})+4\ \ _{max}(1+\ L))}{\log(1=\ )}:$$

Let $(v_i(t);x_i(t);g_i(t))$ for all $i\ 2\ V$ be generated using Algorithm 2.1. Then, $(v_i(t);x_i(t);g_i(t))$ converges to some $(v\ ;x\ ;g\ )$ for all $i\ 2\ V$. Moreover, $x\ $ is a solution to problem (1.1).

Among the multi-consensus algorithms, [25] has taken a similar choice for K_t for solving (1.1) without the regularizer, i.e., $h=0$. However, there are some di erences between our results and theirs. First, for general convex loss functions, the convergence rate in [25] was established for the regret function instead of the solution to (1.1). Second, for $h\ 6=0$, they only considered the case $B=I_p$; similarly, the rate was only established for the regret function. In addition, they did not provide a convergence rate for the strongly convex loss function ($\ 6=0$).

4. Experimental Evaluation. In this section, we investigate the numerical performance of proposed Multi-consensus Decentralized Primal-Dual Fixed Point(MD-PDFP) algorithm for di erent optimization problems under di erent networks. In all the experiments, we choose the FastMix algorithm proposed by [29] with $\ =1\quad\sqrt{1\quad_2}$ as the $\ $-average consensus algorithm A in Algorithm 2.1. For the double stochastic matrix $W=(w_{i;j})$ associated with the network $G=fV;Eg$ we adopt the lazy Metropolis matrix [35, 32] de ned as follows:

$$w_{ij}=\begin{cases}\frac{1}{2\max f\deg(i);\deg(j)g}; & \text{if } (i;j)\ 2\ E;\\ 0; & \text{if } (i;j)\ 2\ E \text{ and } i\ 6=\ j;\\ 1\quad\sum_{k2V;k6=i}w_{ik}; & \text{if } i=\ j;\end{cases}\tag{4.1}$$

where $\deg(i)$ is the degree of node i throughout all the experiments in this paper. Notice that this lazy Metropolis matrix can be computed in parallel because each node only needs to know its own degree and the degrees of its neighbors. For all the experiments, we rst run the centralized algorithm PDFP^{2}O 10000 rounds to be the benchmark result $x\ $. To evaluate the performance of our proposed method, we adopt the Related Error(RE) of the decentralized

algorithm

$$\mathrm{RE}(t) = \frac{\| X_t - 1 x^{\top} \|_2^2}{\| X_0 - 1 x^{\top} \|_2^2}$$

as the criterion at iteration t.

4.1. Comparisons with existing methods. In this section, we investigate the performance of our proposed MD-PDFP by comparing it with several popular distributed algorithms, such as the Inexact Consensus ADMM(IC-ADMM)[8] and the Proximal Gradient EXact rsT-ordeR Algorithm(PG-EXTRA)[44] under the regularizer $(h \square B)(x) = \lambda_N \|x\|_1$.

We consider the decentralized problem for the two types of networks. One is the Erdős-Rényi network with a connectivity probability p_c. It is well known that the Erdős-Rényi network is a random network whose edges are generated with probability p_c. In this experiment, we take the connecting probability $p_c = 0.3$; thus, the second-largest eigenvalue of the double stochastic matrix W induced by this network is $\lambda_2 = 0.81$. Another decentralized network we considered is the n nodes two-star network, which consists of two stars whose centers are connected, and each star is a tree that includes $n/2$ nodes with one node having degree $n/2 - 1$ and the other $n/2 - 1$ nodes with degree one. In our numerical experiments, we fix these decentralized network systems with $n = 50$ nodes generated by the above networks. Then, the second-largest eigenvalue of the matrix W induced by the two-star network is $\lambda_2 = 0.9985$. To ensure the convergence of the consensus algorithm FastMix[29] in our proposed MD-PDFP, we set the number of communication K as $K = 1, 2$ in the Erdős-Rényi network and $K = 4, 5$ in the two-star network. In terms of the stepsize parameters τ and σ, we have constraints $0 < \tau < 2/L$ and $0 < \sigma \le 1/\lambda_{\max}(B^{\top} B) = 1$ in Theorems 3.8 and 3.4. We set $\tau = 1.99/L$ where L can be computed by $L = \max_i \lambda_{\max}(A_i^{\top} A_i)$ numerically and $\sigma = 1$.

In this comparison, we focus on sparse learning over different decentralized networks:

$$\min_{x \in \mathbb{R}^p} \frac{1}{2n} \sum_{i=1}^{n} \| y_i - A_i x \|_2^2 + \lambda_N \|x\|_1, \tag{4.2}$$

where $y_i \in \mathbb{R}^d$ are the responses and $A_i \in \mathbb{R}^{d \times p}$ represents the corresponding covariates on node i. The classical ℓ_1 penalty is considered to encourage the sparsity of the solution. For fair comparisons, we choose the regulation parameter λ_N using the Akaike information criterion(AIC)[1] for the centralized form of problem (4.2). We generate each row of the covariates A_i i.i.d. from a normal distribution $\mathcal{N}(0_p, \Sigma_{p \times p})$, where $\Sigma_{p \times p} = (0.5^{|i-j|})_{ij} \in \mathbb{R}^{p \times p}$. The responses y_i are generated by the model $y_i = A_i x + \varepsilon_i$ on node i, where $x = (1, \ldots, 1, 0, \ldots, 0)^{\top} \in \mathbb{R}^p$ with the first 20 non-zero entries all equal to one and the remaining entries are all zero. Independent random noises ε_i are drawn from $\mathcal{N}(0_d, I_d)$. We consider the dimension $p = 500$ and the local sample size $d = 20$ on each node.

Figure 1 shows the results of the decentralized algorithms. With a small λ_2, the number of communication K does not need to be large over the networks to achieve the consensus in each iteration. Thus, the communication cost for each iteration is low. The proposed method outperforms the other two decentralized algorithms.

(a) (b)

Figure 1. Relative error versus iterations and the number of communication of di erent algorithms under the Erdos-Renyi network.

(a) (b)

Figure 2. Relative error versus iterations and the number of communication of di erent algorithms under the two-star network.

It can be shown in Figure 2 that our algorithm MD-PDFP enjoys linear convergence. With a large $_2$, a large number of communication K can accelerate the convergence. Our algorithm performs well in terms of both the number of iterations and the number of communication.

4.2. Decentralized Fused Lasso. In this section, we work on the decentralized Fused Lasso to investigate the e ect of di erent networks and di erent loss functions. All simulations are repeated with 100 replications in these experiments.

Experiment (a) E ect of the networks. To evaluate the e ect of di erent networks, we consider three kinds of network structures: the Erdos-Renyi network with connectivity probability p_c, the path network, and the two-star network. As described in the above experiment, the Erdos-Renyi network is a random network whose edges are generated with probability p_c, while the two-star network is a network that consists of two centers, and the other nodes are connected to the centers separately. In particular, the path network is a straight line

with the first and the last nodes of degree one and the other nodes of degree two. In our simulations, we consider two probabilities, $p_c = 0.3$ and $p_c = 0.8$, to indicate Erdős-Rényi networks with different densities. In our numerical experiments, we fix these decentralized network systems with $n = 50$ nodes generated by the above four types of networks. We generate the data samples $y_i \in \mathbb{R}^d$; $A_i \in \mathbb{R}^{d \times p}$ in the same method as the above experiments. We fix the dimension $p = 500$. The total sample size N is fixed as $N = nd$, and each node collects part of the data. As for the local sample size d, two scenarios are considered in this experiment: (i) take $d = 20$ in 1d Fused Lasso problem, and thus B_{1d} is a $(p-1) \times p$ matrix, which implies $\lambda_{\min}(B_{1d}B_{1d}^\top) > 0$; (ii) take $d = 5$ in 2d Fused Lasso problem. By constructing the grid graph is $25 \times 20 = p_1 \times p_2$ with $p_1 = 25$ and $p_2 = 20$, we have $B_{2d} \in \mathbb{R}^{955 \times 500}$ which is no longer full rank. It is noteworthy that $A = [A_1^\top; \ldots; A_n^\top]^\top \in \mathbb{R}^{nd \times p}$ is not full rank, and $f(x) = 1/n \sum_{i=1}^n f_i(x) = 1/(2n) \sum_{i=1}^n \|A_i x - y_i\|_2^2$ is not strongly convex when $d = 5$. The smoothness is computed by $L = \max_i \lambda_{\max}(A_i^\top A_i)$. For ease of presentation, we choose the tunning parameter λ_N using the commonly used 5-fold cross-validation[15].

Table 1 and 2 show the average iterations to achieve the specific relative error with different parameters K, θ, L, and η. It can be observed that the different choices of K are due to the network's topology. With a larger second-largest eigenvalue λ_2, we need more communication rounds K to achieve a consensus. With the proper stepsize parameters θ and L, the proposed algorithm requires fewer iterations to reach certain convergence. However, the parameters η and $\lambda_{\max}$ in the proximal operator do not significantly influence the convergence rate.

Table 1

Scenario (i): 1d Fused Lasso, the average number of iterations versus the related error with different parameters and networks.

Graph	λ_2	K	L	$\theta^2/\lambda_{\max}$	RE=10^{-2}	RE=10^{-4}	RE=10^{-6}
Erdős-Rényi($p_c = 0.3$)	0.785	1	2.0	1.0	100.86	256.00	436.73
	0.789	1	1.0	1.0	203.00	546.43	950.08
	0.798	1	2.0	0.5	98.89	256.46	445.48
	0.826	5	2.0	1.0	94.27	245.44	441.72
Erdős-Rényi($p_c = 0.8$)	0.614	1	2.0	1.0	100.50	250.86	445.13
	0.606	1	1.0	1.0	205.76	532.64	901.70
	0.595	1	2.0	0.5	102.91	262.54	466.87
	0.605	5	2.0	1.0	100.87	251.11	430.72
Path	0.9990	25	2.0	1.0	108.09	273.00	459.90
	-	25	1.0	1.0	205.05	542.01	912.58
	-	25	2.0	0.5	110.89	276.11	464.17
	-	30	2.0	1.0	100.47	255.86	436.34
Two-Star	0.9985	25	2.0	1.0	104.26	275.95	466.39
	-	25	1.0	1.0	205.35	554.10	967.23
	-	25	2.0	0.5	102.17	265.41	454.01
	-	30	2.0	1.0	98.04	240.52	423.17

Figure 3 denotes the results of the experiments on the two-star network. For the 1d Fused Lasso scenario, Figures 3(a) and (b) illustrate the effects of the parameters θ and η. It should be noted that a larger θ and η that fulfill the conditions in Theorem 3.8 can accelerate the convergence of the proposed algorithm. When these parameters do not satisfy certain conditions, they degrade the performance of the proposed algorithm. Figure 3(c) shows that

Table 2

Scenario (ii): 2d Fused Lasso, the average number of iterations versus the related error with different parameters and networks.

Graph	$_2$	K	L	$^2_{max}$	RE=10 ^{2}	RE=10 ^{4}	RE=10 ^{6}
Erdos-Renyi($p_c = 0:3$)	0.796	1	2.0	1.0	171.76	411.19	712.71
	0.772	1	1.0	1.0	334.65	708.20	1201.05
	0.819	1	2.0	0.5	175.98	428.20	780.99
	0.811	5	2.0	1.0	168.08	375.47	664.39
Erdos-Renyi($p_c = 0:8$)	0.599	1	2.0	1.0	179.47	419.00	705.43
	0.640	1	1.0	1.0	334.57	718.29	1190.19
	0.598	1	2.0	0.5	176.34	423.90	721.91
	0.603	5	2.0	1.0	167.95	422.30	767.00
Path	0.9990	25	2.0	1.0	173.08	405.46	695.33
	-	25	1.0	1.0	319.63	692.75	1143.60
	-	25	2.0	0.5	174.40	417.34	721.13
	-	30	2.0	1.0	158.10	364.22	621.37
Two-Star	0.9985	25	2.0	1.0	166.39	407.70	705.42
	-	25	1.0	1.0	335.38	744.10	1258.63
	-	25	2.0	0.5	174.74	410.62	733.17
	-	30	2.0	1.0	165.14	395.42	694.93

once the communication rounds K is large enough, say $K = 23$ in this case, there is no need to increase K for fast convergence. As for the 2d Fused Lasso, Figures 3(d), (e), and (f) show similar patterns, and we can obtain similar conclusions.

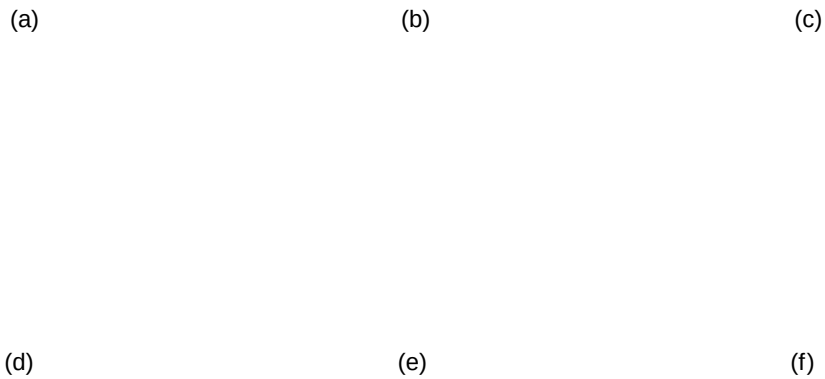
Experiment (b) Effect of the loss functions. To evaluate the effect of different loss functions, we consider Huber regression [20] and the logistic regression with the Fused Lasso penalty. The proposed MD-PDFP algorithm is applied to the two loss functions with both 1d and 2d Fused Lasso penalties. We fix the dimension $p = 500$ and the same transfer matrices B_{1d} and B_{2d} as in Experiment (a). The Erdos-Renyi networks are considered with probabilities $p_c = 0:3$ and $p_c = 0:8$. The tuning parameter $_N$ is also chosen by 5-fold cross-validation. For the smoothness parameters L of different loss functions $f(x)$, we compute them by taking the second derivative of $f(x)$ respectively. We generate jth row a_{ij} of the covariates A_i i.i.d. sampled from a normal distribution $N(0_p; \ _{p\ p})$, where $_{p\ p} = (0:5^{|i\ j|})\ 2\ R^{p\ p}$. For Huber regression, the data samples y_i are generated by $y_i = A_i x + \ _i\ 2\ R^d$, where $x = (1; \ ; 1; 0; \ ; 0)\ 2\ R^p$ with 20 non-zero entries and the noise $_i$ is i.i.d. generated from a mixture of normal distributions $0:9N(0_d; I_d) + 0:1N(0_d; 100I_d)$.

In the Huber regression, the Huber loss is adopted for robust estimation, which is given by

$$L\ (x) = \begin{cases} \frac{1}{2}|x|^2; & |x| < \ ; \\ |x| \ \frac{1}{2} \ ; & |x| > \ : \end{cases} \tag{4.3}$$

where $$ is a positive constant that controls the level of the outliers. Here we set $ = 1:345$ according to the suggestion given by [20]. We then consider the Huber loss with a penalty in the decentralized form:

$$\min_{x\,2\,R^p} \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{d} L\ \ y_{ij} \ \ a_{ij}^{>} x \ + \ _N\ kBx\ k_1 :$$



(a) (b) (c)

(d) (e) (f)

Figure 3. The logarithm of relative error versus iterations with different parameters in two scenarios. The default parameters are $= 2/L$; $= 1/\lambda_{max}(BB^\top)$; $K = 25$. (a) and (d) are RE versus iterations for $\lambda_{max}(BB^\top) = 0.1; 0.5; 1; 1.5$. (b) and (e) are RE versus iterations for $L = 0.5; 1.0; 2; 2.5$. (c) and (f) are RE versus iterations for $K = 23; 24; 25; 30$.

Image classification problems have drawn considerable attention in real applications [18]. Logistic regression is a widely used classification model. In the logistic regression, we generate binary responses $y_{i,j}$ as follows:

$$y_{i,j} = \begin{cases} 1, & \text{w. p. } \frac{\exp(a_{i,j}^\top x)}{1+\exp(a_{i,j}^\top x)}; \\ 0, & \text{w. p. } \frac{1}{1+\exp(a_{i,j}^\top x)}. \end{cases}$$

In this paper, we adopt a penalized logistic regression in a decentralized form:

$$\min_{x \in R^p} \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{d} \log\left(1 + e^{-y_{ij} a_{ij}^\top x}\right) + \lambda_N \|Bx\|_1.$$

The average number of iterations versus the related error with different parameters and loss functions are shown in Tables 3-6. Our algorithm achieves convergence on two penalized regressions in two scenarios. The conclusions regarding the parameters and are the same as those in the experiment above. In the Erdős-Rényi network with $p_c = 0.3$, more communication rounds K usually accelerate the consensus, but only slightly. In the Erdős-Rényi network with $p_c = 0.8$, with a smaller second-largest eigenvalue $_2$, $K = 1$ is sufficient to guarantee the consensus.

Table 3

Scenario (i): 1d Fused Lasso, the average number of iterations versus the related error with different parameters with Huber loss.

Graph	$_2$	K	L	$^2_{max}$	RE=10 ^{2}	RE=10 ^{4}	RE=10 ^{6}
Erdos-Renyi($p_c = 0:3$)	0.791	1	2.0	1.0	166.73	387.56	642.91
	0.774	1	1.0	1.0	324.65	751.51	1286.30
	0.829	1	2.0	0.5	169.39	382.78	651.56
	0.775	5	2.0	1.0	163.04	385.00	629.52
Erdos-Renyi($p_c = 0:8$)	0.590	1	2.0	1.0	163.52	365.13	602.69
	0.604	1	1.0	1.0	345.08	835.39	1373.30
	0.600	1	2.0	0.5	165.27	357.65	617.73
	0.619	5	2.0	1.0	165.52	386.60	640.08

Table 4

Scenario (i): 1d Fused Lasso, the average number of iterations versus the related error with different parameters with logistic loss function.

Graph	$_2$	K	L	$^2_{max}$	RE=10 ^{2}	RE=10 ^{4}	RE=10 ^{6}
Erdos-Renyi($p_c = 0:3$)	0.793	1	2.0	1.0	1434.11	3212.14	3962.47
	0.771	1	1.0	1.0	2540.31	5877.41	9222.52
	0.756	1	2.0	0.5	1373.86	3213.40	5055.26
	0.761	5	2.0	1.0	1374.23	3175.07	4980.84
Erdos-Renyi($p_c = 0:8$)	0.604	1	2.0	1.0	1361.11	2933.69	4905.00
	0.599	1	1.0	1.0	2709.26	6308.79	9928.22
	0.598	1	2.0	0.5	1370.30	3143.72	4929.59
	0.599	5	2.0	1.0	1373.40	3176.13	5004.81

Table 5

Scenario (ii): 2d Fused Lasso, the average number of iterations versus the related error with different parameters with Huber loss.

Graph	$_2$	K	L	$^2_{max}$	RE=10 ^{2}	RE=10 ^{4}	RE=10 ^{6}
Erdos-Renyi($p_c = 0:3$)	0.806	1	2.0	1.0	618.39	988.86	1397.73
	0.784	1	1.0	1.0	1251.08	1919.95	2589.60
	0.773	1	2.0	0.5	647.82	1039.86	1436.69
	0.789	5	2.0	1.0	614.30	942.04	1286.13
Erdos-Renyi($p_c = 0:8$)	0.601	1	2.0	1.0	598.65	931.91	1315.47
	0.614	1	1.0	1.0	1215.57	1934.29	2673.63
	0.606	1	2.0	0.5	634.34	1043.13	1476.39
	0.645	5	2.0	1.0	628.39	1000.08	1373.13

Table 6

Scenario (ii): 2d Fused Lasso, the average number of iterations versus the related error with different parameters with logistic loss function.

Graph	$_2$	K	L	$^2_{max}$	RE=10 ^{2}	RE=10 ^{4}	RE=10 ^{6}
Erdos-Renyi($p_c = 0:3$)	0.805	1	2.0	1.0	313.68	713.28	1105.99
	0.780	1	1.0	1.0	538.13	1232.31	1937.53
	0.770	1	2.0	0.5	344.01	749.62	1227.34
	0.845	5	2.0	1.0	287.34	759.01	1159.06
Erdos-Renyi($p_c = 0:8$)	0.598	1	2.0	1.0	317.10	745.37	1181.93
	0.619	1	1.0	1.0	470.21	1000.01	1648.74
	0.616	1	2.0	0.5	369.31	809.68	1357.91
	0.614	5	2.0	1.0	285.74	678.52	983.73

4.3. Network Lasso. In this experiment, we aim to apply MD-PDFP to the network Lasso introduced in Subsection 2.3 and demonstrate the e ciency of our algorithm. We solve the optimization problem in the network:

$$\min_{(x_1^\top;\ ;x_n^\top)^\top} \frac{1}{2n}\sum_{i=1}^{n} \|y_i - A_i x_i\|_2^2 + \sum_{(j,k)\in E} {}_N\, {}_{j,k}\, \|x_j - x_k\|_2;$$

where $y_i \in R^p; A_i \in R^{p\ p}; x_i \in R^p$. We x the dimension $p = 1000$. The network G we considered here is a two-star network with $n = 50$ nodes. Then, the second-largest eigenvalue ${}_2 = 0.9985$. For the choice of weights ${}_{j,k}$, we adopt the suggestion in [61], and the weights ${}_{j,k}$ are given by ${}_{j,k} = 1/\|\hat{x}_j - \hat{x}_k\|_2$, where $\hat{x}_i = (A_i^\top A_i)^{-1} A_i^\top y_i$ is the local ordinary least squares estimator on node i. Each row of A_i is i.i.d. generated from a normal distribution $N(0_p;\ _{p\ p})$, where ${}_{p\ p} = (0.5^{|i-j|})_{i,j} \in R^{p\ p}$. We assume that two clusters exist in the network. The centroids of these two clusters are x_0 and x_1 respectively. Speci cally, we set $x_0 = (1; \ ; 1; 0; \ ; 0)^\top$ with 20 non-zero elements on the rst 20 entries and $x_1 = (0; \ ; 0; 1; \ ; 1)^\top$ with 20 non-zero elements on the last 20 entries. We generate the data $y_i = A_i x_j + {}_i$, where ${}_i \sim N(0_p; I_p)$ and $j \in \{0; 1\}$ is binary random variable with equal probability.

We set the tuning parameter ${}_N = {}_{\text{critical}}$ as de ned in [18]. For our algorithm, the number of communication K is set to be $K = 25$. And the step size parameter $= 1.99/L; = 1/{}_{\max}(BB^\top)$, where $L = \max_i {}_{\max}(A_i^\top A_i)$ and $B = H \otimes I_p$ de ned in Subsection 2.3.

(a) Relative Error (b) Average time for one round.

Figure 4. Relative error versus iterations when $p = 1000$ and the average time for one round on two algorithms.

Figure 4(a) shows the linear convergence of two algorithms. The ADMM-based algorithm in [18] converges faster than our algorithm, achieving a $10^{-5.8}$ relative error after only 1485 update iterations, whereas our algorithm requires 7463 update iterations. However, the ADMM-based algorithm spends more time to achieve the same relative error. As shown in Figure 4(b), in high dimensions, the ADMM-based algorithm requires more time per update iteration because of the computation of the inverse matrix $(D_i I_p + A_i^\top A_i)^{-1}$ for some positive scale D_i.

522 4.4. Real Data. In this section, we apply our proposed algorithm, MD-PDFP, to real
523 image reconstruction problems. Some works have developed distributed algorithms to re-
524 cover high-resolution images, with each node processing a subset of views [23, 4]. In image
525 reconstruction[27], people developed several methods in a parallel way to fast the reconstruction
526 process. In a distributed setting, the entire image x can be split into multiple pieces x_i. Our
527 goal is to reconstruct the entire image $x = (x_1^\top; \ldots; x_n^\top)^\top \in \mathbb{R}^p$. Typically, we observe a
528 noisy version of the image after a certain transformation A. In general, this problem can be
529 described as follows:

530 (4.4)
$$\min_{x \in \mathbb{R}^p} \frac{1}{2n} \sum_{i=1}^{n} \|y_i - A_i x_i\|_2^2 + \lambda_N \|Bx\|_1;$$
531

532 where λ_N is the tuning parameter, A_i is a linear operator representing the subsampling x_i is
533 a subgraph of the image x, and y_i is the noisy subsampling image. We adopted the widely
534 used total variation penalty in image processing [12] as the regularizer. Thus, the operator
535 $B : \mathbb{R}^p \to \mathbb{R}^{2p}$ is defined as a discrete gradient operator to denoise. The definition of B can be
536 found in [31, 12], which is similar to the 2d Fused Lasso.
537 Experiment (a). The first experiment is implemented on a self-taken photo of $p = 1216 \times 1824$
538 pixels with three color channels. We divide 1216×1824 pixels x_{img} into 24 small pictures,
539 where each picture consists of 304×304 pixels. In the distributed setting, we contribute evenly
540 to $n = 4 \times 6 = 24$ nodes. The subgraphs x_i on each node have dimensions of 304×304. We
541 vectorize them as vectors of length 92416.
542 The subsampling operator A_i averages the image every 4×4 pixels to obtain a low-resolution
543 image with 76×76 size. y_i is the observed low-resolution image with noise relative to the node
544 i, that is, $y_i = A_i x_{img}^i + \varepsilon_i$ where $\varepsilon_i \sim N(0; I_{5776})$ and $x_{img} = (x_{img}^{1\top}; \ldots; x_{img}^{n\top})^\top \in \mathbb{R}^p$ is the
545 original image.

(a) The images are divided into 24 sub-images and placed on each node.

(b) Zooming of the original image.

Figure 5. The original image and the result of the averaging images $A_i x_{img}^i$.

546 Higher-resolution images are considered to simulate higher-load scenarios. Each camera
547 can capture only a small portion of the environment, and the image is enormous, with more
548 than two million pixels. Parallel computing is used to process three color channels on each