A Decentralized Primal-Dual Method with Quasi-Newton Tracking

Liping Wang, Hao Wu, and Hongchao Zhang

***Abstract*—This paper considers the decentralized optimization problem of minimizing a finite sum of strongly convex and twice continuously differentiable functions over a fixed-connected undirected network. A fully decentralized primal-dual method (DPDM) and its generalization (GDPDM), which allows for multiple primal steps per iteration, are proposed. In our methods, both primal and dual updates use second-order information obtained by quasi-Newton techniques which only involve matrix-vector multiplication. Specifically, the primal update applies a Jacobi relaxation step using the BFGS approximation for both computation and communication efficiency. The dual update employs a new second-order correction step. We show that the decentralized local primal updating direction on each node asymptotically approaches the centralized quasi-Newton direction. Under proper choice of parameters, GDPDM including DPDM has global linear convergence for solving strongly convex decentralized optimization problems. Our numerical results show both GDPDM and DPDM are very efficient compared with other state-of-the-art methods for solving decentralized optimization.**

***Index Terms*—Decentralized optimization, primal-dual method, quasi-Newton method, BFGS update, BB method, global convergence, linear convergence rate.**

I. INTRODUCTION

In this paper, we consider the following decentralized optimization problem over an undirected and connected network containing n nodes

$$\mathbf{z}^* = \arg\min_{\mathbf{z}\in\mathbb{R}^p} \sum_{i=1}^{n} f_i(\mathbf{z}), \tag{1}$$

where the local objective function $f_i : \mathbb{R}^p \to \mathbb{R}$, $i = 1, \ldots, n$, on each node is strongly convex and twice continuously differentiable. Consider the underlying network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, n\}$ is the set of nodes, and $\mathcal{E}$ is the collection of unordered edges. We denote two nodes as neighbors if they are connected by an edge. In a decentralized setting, there does not exist one central server to gather local information from all nodes, compute shared global information, and broadcast it back to the whole network. Each local function f_i is only known to node i. All the nodes collaborate with their neighbors through information exchange (i.e., communication) to finally obtain the global solution $\mathbf{z}^*$. Decentralized optimization has wide applications including decentralized resources control [1], wireless networks [2], decentralized machine learning [3], power systems [4], federated learning [5], etc..

As more important practical applications surge up, the decentralized optimization methods have been extensively studied in recent years, where first-order methods gain attention very rapidly due to their simple iterative schemes and low computational cost per iteration. The decentralized gradient descent (DGD) [6]–[8] methods are a class of the most well-known first-order methods for solving decentralized optimization. However, DGD methods usually converge to the optimal solution only with a diminishing step size, while a small constant stepsize often leads the iterate to a neighborhood of the minimizer [7]. There are many recent works devoted to a constant step size with guaranteed convergence. By significantly increasing the number of communication steps, the DGD method [9] achieves global convergence with R-linear rate. To ensure global convergence, EXTRA [10], [11] as well as its modification, called NIDS [12], adopts different mixing matrices at odd and even iterations. The method given in [13] combines the diffusion strategy with an important subclass of left-stochastic matrices to obtain wider stability region and enhanced performance than EXTRA. Gradient Tracking (GT) methods [14]–[18] track the global average gradient to design local search directions at any node. An effective approach to achieve global convergence is to design the methods in the primal-dual domain. A general method given in [19] unified EXTRA and GT into a primal-dual framework. A more flexible first-order primal-dual framework is proposed in [20], where multiple primal steps per iteration are allowed. In addition, there are also some special classes of primal-dual methods based on alternating direction approaches [21]–[23].

Although first-order methods are more simple and easily implementable, their asymptotic convergence speed is often slow for more accurate solutions. So, many second-order methods have been proposed recently to accelerate the convergence rate. Some methods focus on penalized approaches for solving a constrained problem, where a consensus constraint is introduced to reformulate the problem (1). NN [24] uses the Newton's method to solve the penalty problem. DQN [25] develops a diagonal correction technique to overcome the challenge that the Hessian inverse of the penalty function can not be computed in a decentralized way. DBFGS [26] is a decentralized quasi-Newton method which only uses the local neighbor information. However, NN, DQN, and DBFGS are inexact penalty methods in the sense that the penalty parameter needs to go to infinity for ensuring global convergence. ESOM [27], PMM-DQN [25], and PD-

Liping Wang and Hao Wu are in School of Mathematics, Nanjing University of Aeronautics and Astronautics (e-mail: wlpmath@nuaa.edu.cn; wuhoo104@nuaa.edu.cn).

Hongchao Zhang is in Department of Mathematics, Louisiana State University (e-mail: hozhang@math.lsu.edu).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org., provided by the authors. The material includes additional mathematical derivations and experiments.

QN [28] improve NN, DQN, and DBFGS, respectively, in the primal-dual framework. Newton Tracking (NT) [29] is also developed in the primal-dual domain, where the local search directions asymptotically track the global centralized Newton direction. A decentralized ADMM [30] incorporates BFGS quasi-Newton techniques to improve computation efficiency, and the quadratically approximated ADMM given by [31] was improved in [32] to promote communication efficiency. Two decentralized stochastic quasi-Newton methods are proposed in [33], where average gradient approximations are used with regularization or damping techniques to construct the Hessian inverse approximations. Another method [34] with the same name as DQN is relevant to methods in [33] but extended to the equality-constrained optimization setting.

Among all the previously developed primal-dual methods, there are rarely methods applying second-order information in the dual update. Moreover, when combining the quasi-Newton techniques into the decentralized methods in the primal-dual framework, some critical issues may arise. First, the local quasi-Newton matrices generated by the BFGS approximation are not necessarily positive definite although the augmented Lagrangian function across all nodes is strongly convex. Second, when the primal problem is solved inexactly, the exact dual gradient is not obtained. Extracting useful Hessian information from inexact dual gradients also needs to be studied. Our goal of this paper is to explore second-order information in both primal and dual domains, and propose a new fully decentralized primal-dual quasi-Newton method with both theoretical global linear convergence and numerical efficiency. Our main contributions are as follows.

1. In the primal domain, we combine the Jacobi relaxation technique with the BFGS approximation for updating the primal iterates. Multiple adaptive primal updates per iteration are also allowed in our approach to balance the accuracy obtained in both the primal and dual domains.
2. In the dual domain, by applying the Newton's method to the dual problem and making some critical reformulations, we obtain a novel dual updating step, which can be viewed as adding a second-order correction term to the usual dual ascent step. To maintain computational efficiency, we apply BB-approximation techniques to capture the spectral information of the dual Hessian. To the best of our knowledge, we are the first to give a primal-dual method using quasi-Newton dual update with guaranteed convergence.
3. Our proposed method is a quasi-Newton tracking method, where the local search direction on each node tracks the global centralized quasi-Newton direction. Our quasi-Newton tracking can be considered as a generalization of the Newton tracking [29]. In addition, our approaches only involve matrix-vector multiplications to avoid inverting a matrix. The numerical results show that our methods are very efficient compared with other state-of-the-art methods for solving decentralized optimization.

The paper organized is as follows: In Section 2, we reformulate the problem as a constrained decentralized optimization, and develop our new methods. Global convergence as well as linear convergence rate are established in Section 3. Numerical experiments are performed in Section 4 to compare our method with other well-established first and second-order methods for solving decentralized optimization. Finally, we draw some conclusions in Section 5.

A. Notation

We use uppercase and lowercase boldface letters to denote matrices and vectors, respectively. We let $\mathbf{x}_i$ denote the local copy of the global variable $\mathbf{z}$ at node i and define $\mathcal{N}_i$ as the set consisting of the neighbors of node i (we treat node i itself as one of its neighbors for convenience.). $\bar{\mathbf{x}}$ denotes $\frac{1}{n}\sum_{i=1}^n \mathbf{x}_i$. Kronecker Product is denoted as $\otimes$. $\mathrm{Proj}_{[a,b]}(\cdot)$ is the projection operator onto interval $[a,b]$. Given a vector $\mathbf{v}$ and a symmetric matrix $\mathbf{M}$, we let $\mathrm{span}(\mathbf{v})$ denote the linear subspace spanned by $\mathbf{v}$ and $\|\mathbf{v}\|_{\mathbf{M}}^2$ denote $\mathbf{v}^\mathsf{T}\mathbf{M}\mathbf{v}$. $\lambda_{\min}(\mathbf{M})$, $\lambda_2(\mathbf{M})$, and $\lambda_{\max}(\mathbf{M})$ denote the smallest, second smallest, and largest eigenvalues of $\mathbf{M}$, respectively. $\mathrm{Null}(\mathbf{M})$ denotes the null space of $\mathbf{M}$. The trace and determinant of $\mathbf{M}$ is denoted as $\mathrm{tr}(\mathbf{M})$ and $\det(\mathbf{M})$, respectively. $\mathbf{M}^\mathsf{T}$ denotes transpose of $\mathbf{M}$ and $\mathbf{M}^\dagger$ denotes its pseudo inverse. For matrices $\mathbf{M}_1$ and $\mathbf{M}_2$ with same dimension, $\mathbf{M}_1 \succeq \mathbf{M}_2$ means $\mathbf{M}_1 - \mathbf{M}_2$ is positive definite. We use $\mathbf{I}_p$ to denote the $p \times p$ identity matrix. Especially, $\mathbf{I}$ denotes $\mathbf{I}_{np}$. We let $\mathrm{diag}\{a_1, \ldots, a_p\}$ denote a $p \times p$ diagonal matrix whose diagonal elements are $a_1, \ldots, a_p$, and $\log(\cdot)$ denote $\log_{10}(\cdot)$. For any scalar $\theta \in \mathbb{R}$, if $\theta \neq 0$, $I_{R\setminus\{0\}}(\theta) = 1$; otherwise, $I_{R\setminus\{0\}}(\theta) = 0$.

II. PROBLEM FORMULATION AND ALGORITHM DEVELOPMENT

A. Problem Formulation

Since the network is connected, the constraints $\mathbf{x}_i = \mathbf{x}_j, i, j = 1, \ldots, n$, are equivalent to $\mathbf{x}_i = \mathbf{x}_j, j \in \mathcal{N}_i, i = 1, \ldots, n$. Therefore, the problem (1) can be reformulated as the following constrained optimization

$$\{\mathbf{x}_i^*\}_{i=1}^n = \arg\min_{\{\mathbf{x}_i\}_{i=1}^n} \sum_{i=1}^n f_i(\mathbf{x}_i), \quad s.t. \mathbf{x}_i = \mathbf{x}_j, \forall j \in \mathcal{N}_i, i = 1, \ldots, n. \tag{2}$$

The consensus constraints ensure the solution of problem (2) is equal to the solution of (1) and $\mathbf{x}_1^* = \mathbf{x}_2^* = \ldots = \mathbf{z}^*$.

To reveal the network structure, we introduce a mixing matrix $\tilde{\mathbf{W}} \in R^{n\times n}$ which has the following standard features.
1. $\tilde{\mathbf{W}}$ is nonnegative and $\tilde{W}_{ij}$ characterizes the active link (i,j), i.e., $\tilde{W}_{ij} > 0$ if $j \in \mathcal{N}_i$, $\tilde{W}_{ij} = 0$ otherwise.
2. $\tilde{\mathbf{W}}$ is symmetric and doubly stochastic, i.e., $\tilde{\mathbf{W}} = \tilde{\mathbf{W}}^\mathsf{T}$ and $\tilde{\mathbf{W}}\mathbf{1}_n = \mathbf{1}_n$.

It has a few common choices for the mixing matrix $\tilde{\mathbf{W}}$, such as Laplacian-based constant edge weight matrix [35] and Metropolis constant edge weight matrix [36]. From the second feature, we have $\mathrm{Null}(\mathbf{I}_n - \tilde{\mathbf{W}}) = \mathrm{span}(\mathbf{1}_n)$. Using the mixing matrix $\tilde{\mathbf{W}}$, we can rewrite problem (2) in an equivalent

compact form. Let us denote

$$\mathbf{x} = [\mathbf{x}_1; \mathbf{x}_2; \ldots; \mathbf{x}_n] \in \mathbb{R}^{np}, \quad f(\mathbf{x}) = \sum_{i=1}^{n} f_i(\mathbf{x}_i),$$
$$\nabla f(\mathbf{x}) = [\nabla f_1(\mathbf{x}_1); \nabla f_2(\mathbf{x}_2); \ldots, \nabla f_n(\mathbf{x}_n)] \in \mathbb{R}^{np},$$
$$\mathbf{W} = \tilde{\mathbf{W}} \otimes \mathbf{I}_p \in \mathbb{R}^{np \times np} \text{ and } \mathbf{I} - \mathbf{W} = (\mathbf{I}_n - \tilde{\mathbf{W}}) \otimes \mathbf{I}_p.$$

We can easily know that the equation $(\mathbf{I} - \mathbf{W})\mathbf{x} = \mathbf{0}$ holds if and only if $\mathbf{x}_1 = \mathbf{x}_2 = \ldots = \mathbf{x}_n$. According to the Perron-Frobenius theorem [37], the eigenvalues of $\tilde{\mathbf{W}}$ lie in $(-1, 1]$ and 1 is the single eigenvalue. Since $\mathbf{I} - \mathbf{W}$ is symmetric positive semidefinite, $(\mathbf{I} - \mathbf{W})^{1/2}$ has the same null space as $\mathbf{I} - \mathbf{W}$. Hence, problem (2) can be reformulated as

$$\begin{aligned} \mathbf{x}^* = &\arg \min_{\mathbf{x} \in \mathbb{R}^{np}} f(\mathbf{x}), \\ &s.t. (\mathbf{I} - \mathbf{W})^{1/2} \mathbf{x} = \mathbf{0}. \end{aligned} \tag{3}$$

In this paper, we have the following assumptions on the objective function.

Assumption 1. *The local gradients $\{\nabla f_i(\mathbf{z})\}_{i=1}^n$ are Lipschitz continuous with constant $L > 0$, i.e.,*

$$\|\nabla f_i(\mathbf{z}) - \nabla f_i(\tilde{\mathbf{z}})\| \leq L \|\mathbf{z} - \tilde{\mathbf{z}}\|, \tag{4}$$

$\forall \mathbf{z}, \tilde{\mathbf{z}} \in \mathbb{R}^p, i = 1, \ldots, n.$

Assumption 2. *The local objective functions $\{f_i(\mathbf{z})\}_{i=1}^n$ are strongly convex with modulus $\mu > 0$, i.e.,*

$$f_i(\tilde{\mathbf{z}}) \geq f_i(\mathbf{z}) + \nabla f_i(\mathbf{z})^\mathsf{T}(\tilde{\mathbf{z}} - \mathbf{z}) + \frac{\mu}{2}\|\tilde{\mathbf{z}} - \mathbf{z}\|^2, \tag{5}$$

$\forall \mathbf{z}, \tilde{\mathbf{z}} \in \mathbb{R}^p, i = 1, \ldots, n.$

Combining **Assumption 1** with **Assumption 2**, we have

$$\mu \mathbf{I}_p \preceq \nabla^2 f_i(\mathbf{z}) \preceq L \mathbf{I}_p, \ \forall \mathbf{z} \in \mathbb{R}^p, i = 1, \ldots, n. \tag{6}$$

Since the Hessian $\nabla^2 f(\mathbf{x})$ is a block diagonal matrix whose i-th diagonal block is $\nabla^2 f_i(\mathbf{x}_i)$, the above bounds also hold for $\nabla^2 f(\mathbf{x})$, that is

$$\mu \mathbf{I} \preceq \nabla^2 f(\mathbf{x}) \preceq L \mathbf{I}, \ \forall \mathbf{x} \in \mathbb{R}^{np}. \tag{7}$$

As discussed previously, primal-dual methods are effective approaches to solving the problem (3). However, most existing methods only use first-order information to update the dual variables by employing a dual ascent step, such as

$$\boldsymbol{\lambda}^{t+1} = \boldsymbol{\lambda}^t + \alpha(\mathbf{I} - \mathbf{W})^{1/2} \mathbf{x}^{t+1}, \tag{8}$$

where $\boldsymbol{\lambda}^t$ is the Lagrangian multiplier, also called dual variable. In the following, we propose decentralized primal-dual methods whose primal and dual updates will use second-order information approximated by certain quasi-Newton techniques.

B. Algorithm Development

The augmented Lagrangian function of the problem (3) is

$$\tilde{L}_\alpha(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \left\langle \boldsymbol{\lambda}, (\mathbf{I} - \mathbf{W})^{1/2} \mathbf{x} \right\rangle + \frac{\alpha}{2} \mathbf{x}^\mathsf{T} (\mathbf{I} - \mathbf{W}) \mathbf{x}, \tag{9}$$

where $\boldsymbol{\lambda} = [\boldsymbol{\lambda}_1; \boldsymbol{\lambda}_2; \ldots; \boldsymbol{\lambda}_n] \in \mathbb{R}^{np}$, $\alpha > 0$ is the penalty parameter. (9) can be also viewed as the Lagrangian function of the following penalized optimization

$$\begin{aligned} \mathbf{x}^* = &\arg \min_{\mathbf{x} \in \mathbb{R}^{np}} f(\mathbf{x}) + \frac{\alpha}{2} \mathbf{x}^\mathsf{T} (\mathbf{I} - \mathbf{W}) \mathbf{x}, \\ &s.t. (\mathbf{I} - \mathbf{W})^{1/2} \mathbf{x} = \mathbf{0}. \end{aligned} \tag{10}$$

Obviously, the problem (10) is equivalent to the problem (3). Let $\mathbf{x}^*(\boldsymbol{\lambda})$ denote the minimizer of $\tilde{L}_\alpha(\cdot, \boldsymbol{\lambda})$, i.e.,

$$\mathbf{x}^*(\boldsymbol{\lambda}) = \arg \min_{\mathbf{x} \in \mathbb{R}^{np}} \tilde{L}_\alpha(\mathbf{x}, \boldsymbol{\lambda}). \tag{11}$$

The optimality condition of (11) gives

$$\nabla f(\mathbf{x}^*(\boldsymbol{\lambda})) + (\mathbf{I} - \mathbf{W})^{1/2} \boldsymbol{\lambda} + \alpha(\mathbf{I} - \mathbf{W}) \mathbf{x}^*(\boldsymbol{\lambda}) = \mathbf{0}. \tag{12}$$

For any $\boldsymbol{\lambda}^* \in \mathbb{R}^{np}$ that satisfies $(\mathbf{I} - \mathbf{W})^{1/2} \mathbf{x}^*(\boldsymbol{\lambda}^*) = \mathbf{0}$, $(\mathbf{x}^*(\boldsymbol{\lambda}^*), \boldsymbol{\lambda}^*)$ is an primal-dual solution of (10). We now apply Netwon's method to solve the feasibility system

$$(\mathbf{I} - \mathbf{W})^{1/2} \mathbf{x}^*(\boldsymbol{\lambda}) = \mathbf{0}. \tag{13}$$

The t-th iteration of Netwon's method gives

$$\begin{aligned} &(\mathbf{I} - \mathbf{W})^{1/2} \mathbf{x}^*(\boldsymbol{\lambda}^t) \\ &+ (\mathbf{I} - \mathbf{W})^{1/2} \left(\frac{\partial \mathbf{x}^*(\boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}} \right)^\mathsf{T}_{\boldsymbol{\lambda} = \boldsymbol{\lambda}^t} \left(\boldsymbol{\lambda}^{t+1} - \boldsymbol{\lambda}^t \right) = \mathbf{0}. \end{aligned} \tag{14}$$

Differentiating (12) with respect to $\boldsymbol{\lambda}$, we have

$$\left(\nabla^2 f(\mathbf{x}^*(\boldsymbol{\lambda})) + \alpha(\mathbf{I} - \mathbf{W}) \right) \left(\frac{\partial \mathbf{x}^*(\boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}} \right)^\mathsf{T} = -(\mathbf{I} - \mathbf{W})^{1/2},$$

which gives

$$\left(\frac{\partial \mathbf{x}^*(\boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}} \right)^\mathsf{T}_{\boldsymbol{\lambda} = \boldsymbol{\lambda}^t} = -\left[\nabla^2_{\mathbf{xx}} \tilde{L}_\alpha \left(\mathbf{x}^*(\boldsymbol{\lambda}^t), \boldsymbol{\lambda}^t \right) \right]^{-1} (\mathbf{I} - \mathbf{W})^{1/2}. \tag{15}$$

Substituting (15) into (14) yields

$$\begin{aligned} (\mathbf{I} - \mathbf{W})^{1/2} \mathbf{x}^*(\boldsymbol{\lambda}^t) - (\mathbf{I} - \mathbf{W})^{1/2} &\left[\nabla^2_{\mathbf{xx}} \tilde{L}_\alpha \left(\mathbf{x}^*(\boldsymbol{\lambda}^t), \boldsymbol{\lambda}^t \right) \right]^{-1} \\ &(\mathbf{I} - \mathbf{W})^{1/2} \left(\boldsymbol{\lambda}^{t+1} - \boldsymbol{\lambda}^t \right) = \mathbf{0}. \end{aligned}$$

By letting $\mathbf{x}^{t+1} = \mathbf{x}^*(\boldsymbol{\lambda}^t)$, we have

$$\boldsymbol{\lambda}^{t+1} = \boldsymbol{\lambda}^t + \left(\mathbf{G}^t \right)^\dagger (\mathbf{I} - \mathbf{W})^{1/2} \mathbf{x}^{t+1}, \tag{16}$$

where

$$\mathbf{G}^t = (\mathbf{I} - \mathbf{W})^{1/2} \left[\nabla^2_{\mathbf{xx}} \tilde{L}_\alpha(\mathbf{x}^{t+1}, \boldsymbol{\lambda}^t) \right]^{-1} (\mathbf{I} - \mathbf{W})^{1/2}.$$

Solving the primal problem (11) exactly is not only numerically expensive but also theoretically unnecessary. One practical approach is to simply apply one Newton's iteration to the problem (11) and let

$$\begin{aligned} \mathbf{x}^{t+1} = &\mathbf{x}^t - \left[\nabla^2_{\mathbf{xx}} \tilde{L}_\alpha(\mathbf{x}^t, \boldsymbol{\lambda}^t) \right]^{-1} \left[\nabla f \left(\mathbf{x}^t \right) \right. \\ &\left. + (\mathbf{I} - \mathbf{W})^{1/2} \boldsymbol{\lambda}^t + \alpha(\mathbf{I} - \mathbf{W}) \mathbf{x}^t \right]. \end{aligned} \tag{17}$$

Unfortunately, the iterative schemes (16) and (17) can not be directly applied in the decentralized setting. Note that $\nabla^2 f(\mathbf{x})$ is a block diagonal matrix whose $i-$th diagonal block is $\nabla^2 f_i(\mathbf{x}_i)$ and $\mathbf{W}$ is a block sparse matrix related to the network structure. Thus, the Hessian $\nabla^2_{\mathbf{xx}} \tilde{L}_\alpha(\mathbf{x}, \boldsymbol{\lambda})$ is neighbor

related in the calculation. However, the Hessian inverse as well as $(\mathbf{I}-\mathbf{W})^{1/2}$ will destroy the neighbor relation. In addition, it is often very expensive to calculate the inverse of primal and dual Hessians in many practical computations. In the following, we would like to modify the iterative schemes (16) and (17) into a decentralized setting that can be also computed in an efficient way.

a) Primal update: Note that (17) is equivalent to

$$\mathbf{x}^{t+1}=\mathbf{x}^t-\tilde{\mathbf{d}}^t,$$

where $\tilde{\mathbf{d}}^t$ satisfies

$$\nabla^2_{\mathbf{xx}}\tilde{L}_\alpha(\mathbf{x}^t,\boldsymbol{\lambda}^t)\tilde{\mathbf{d}}^t=\nabla_{\mathbf{x}}\tilde{L}_\alpha(\mathbf{x}^t,\boldsymbol{\lambda}^t). \tag{18}$$

We would solve the linear equations (18) inexactly by applying the Jacobi relaxation technique [38], which is an effective technique for the iterative method. By splitting the Hessian $\nabla^2_{\mathbf{xx}}\tilde{L}_\alpha(\mathbf{x}^t,\boldsymbol{\lambda}^t)$ into two parts, $\nabla^2 f(\mathbf{x}^t)$ and $\alpha(\mathbf{I}-\mathbf{W})$, the k-th Jacobi relaxation iteration can be conducted by

$$\begin{aligned}\hat{\mathbf{d}}^{t,k+1}=&\theta\big(-\alpha\nabla^2 f(\mathbf{x}^t)^{-1}(\mathbf{I}-\mathbf{W})\hat{\mathbf{d}}^{t,k}\\&+\nabla^2 f(\mathbf{x}^t)^{-1}\nabla_x\tilde{L}_\alpha(\mathbf{x}^t,\boldsymbol{\lambda}^t)\big)+(1-\theta)\hat{\mathbf{d}}^{t,k},\end{aligned}$$

where

$$\hat{\mathbf{d}}^{t,0}=\nabla^2 f(\mathbf{x}^t)^{-1}\nabla_{\mathbf{x}}\tilde{L}_\alpha(\mathbf{x}^t,\boldsymbol{\lambda}^t),$$

$\theta\in[0,1]$ is the relaxation parameter, introduced to accelerate the original Jacobi iteration. For both computation and communication efficiency, our algorithm would only employ one Jacobi iteration as the following:

$$\begin{aligned}\hat{\mathbf{d}}^{t,1}=&\theta\big(-\alpha\nabla^2 f(\mathbf{x}^t)^{-1}(\mathbf{I}-\mathbf{W})\mathbf{d}^{t,0}\\&+\nabla^2 f(\mathbf{x}^t)^{-1}\nabla_{\mathbf{x}}\tilde{L}_\alpha(\mathbf{x}^t,\boldsymbol{\lambda}^t)\big)+(1-\theta)\hat{\mathbf{d}}^{t,0},\end{aligned}$$

and let $\hat{\mathbf{d}}^t=\hat{\mathbf{d}}^{t,1}$ as the approximation to $\tilde{\mathbf{d}}^t$. Combining the above steps, we obtain

$$\hat{\mathbf{d}}^t=\big(\mathbf{I}-\theta\alpha\nabla^2 f(\mathbf{x}^t)^{-1}(\mathbf{I}-\mathbf{W})\big)\nabla^2 f(\mathbf{x}^t)^{-1}\nabla_{\mathbf{x}}\tilde{L}_\alpha(\mathbf{x}^t,\boldsymbol{\lambda}^t). \tag{19}$$

Note that if $\theta=0$, (19) reduces to

$$\hat{\mathbf{d}}^t=\nabla^2 f(\mathbf{x}^t)^{-1}\nabla_{\mathbf{x}}\tilde{L}_\alpha(\mathbf{x}^t,\boldsymbol{\lambda}^t), \tag{20}$$

which is exactly primal update direction in NT [29] with the parameter $\epsilon=0$. To avoid computing $\nabla^2 f(\mathbf{x}^t)$ and its inverse, we would approximate $\nabla^2 f(\mathbf{x}^t)$ by a block positive definite matrix $\mathbf{B}^t$ using BFGS quasi-Newton techniques. Then, by introducing a stepsize $\beta>0$ for global convergence, we have the following primal update,

$$\begin{aligned}\mathbf{x}^{t+1}=&\mathbf{x}^t-\beta\left[\mathbf{I}-\theta\alpha(\mathbf{B}^t)^{-1}(\mathbf{I}-\mathbf{W})\right]\\&\times(\mathbf{B}^t)^{-1}\nabla_{\mathbf{x}}\tilde{L}_\alpha(\mathbf{x}^t,\boldsymbol{\lambda}^t).\end{aligned} \tag{21}$$

More specifically, we update $\mathbf{B}^t$ by the following BFGS formula

$$\mathbf{B}^{t+1}=\begin{bmatrix}\mathbf{B}_1^{t+1}&&\\&\cdots&\\&&\mathbf{B}_n^{t+1}\end{bmatrix}, \tag{22}$$

where we set $\mathbf{B}_i^0=\mathbf{I}_p$,

$$\mathbf{B}_i^{t+1}=\mathbf{B}_i^t-\frac{\mathbf{B}_i^t\mathbf{s}_i^t(\mathbf{s}_i^t)^\mathsf{T}\mathbf{B}_i^t}{(\mathbf{s}_i^t)^\mathsf{T}\mathbf{B}_i^t\mathbf{s}_i^t}+\frac{\mathbf{y}_i^t(\mathbf{y}_i^t)^\mathsf{T}}{(\mathbf{s}_i^t)^\mathsf{T}\mathbf{y}_i^t},\quad t\geq 0, \tag{23}$$

$\mathbf{s}_i^t=\mathbf{x}_i^{t+1}-\mathbf{x}_i^t$ and $\mathbf{y}_i^t=\nabla f_i(\mathbf{x}_i^{t+1})-\nabla f_i(\mathbf{x}_i^t)$.

We have some comments about $\mathbf{B}^t$. First, since f_i is strongly convex, we have $(\mathbf{s}_i^t)^\mathsf{T}\mathbf{y}_i^t>0$, which together with the positive definite initialization of $\mathbf{B}_i^0$ ensures $\mathbf{B}^t\succ\mathbf{0}$ for all $t\geq 0$. Second, as usual, in practical implementation we always directly update $\mathbf{H}^t:=(\mathbf{B}^t)^{-1}$. In particular, using the BFGS inversion formula, we would have $\mathbf{H}_i^0=(\mathbf{B}_i^0)^{-1}$ and

$$\begin{aligned}\mathbf{H}_i^{t+1}=&\mathbf{H}_i^t-\frac{\mathbf{H}_i^t\mathbf{y}_i^t(\mathbf{s}_i^t)^\mathsf{T}+\mathbf{s}_i^t(\mathbf{y}_i^t)^\mathsf{T}\mathbf{H}_i^t}{(\mathbf{s}_i^t)^\mathsf{T}\mathbf{y}_i^t}\\&+\left(1+\frac{(\mathbf{y}_i^t)^\mathsf{T}\mathbf{H}_i^t\mathbf{y}_i^t}{(\mathbf{s}_i^t)^\mathsf{T}\mathbf{y}_i^t}\right)\frac{\mathbf{s}_i^t(\mathbf{s}_i^t)^\mathsf{T}}{(\mathbf{s}_i^t)^\mathsf{T}\mathbf{y}_i^t},\quad t\geq 0.\end{aligned} \tag{24}$$

Note that $\mathbf{H}^t$ is applied in algorithm implementation while $\mathbf{B}^t$ is only used in the paper for convenient theoretical analysis.

Finally, we want to emphasize that it is improper to use BFGS techniques to directly approximate $\nabla^2_{\mathbf{xx}}\tilde{L}_\alpha(\mathbf{x}^t,\boldsymbol{\lambda}^t)=\nabla^2 f(\mathbf{x}^t)+\alpha(\mathbf{I}-\mathbf{W})$ in decentralized setting. Although $\mathbf{x}^\mathsf{T}(\mathbf{I}-\mathbf{W})\mathbf{x}\geq 0$ for all $\mathbf{x}$, its restriction on each node, i.e., $(\mathbf{x}_i)^\mathsf{T}\left(\mathbf{x}_i-\sum_{j\in\mathcal{N}_i}\tilde{W}_{ij}\mathbf{x}_j\right)$ may not be positive. This may destroy the positive definite property of BFGS matrices.

b) Dual update: We first give the following lemma, which can be considered as an extension of **Lemma 1** in [39].

Lemma II.1. *Let $\mathbf{A}\in\mathbb{R}^{m\times n}$ and $\operatorname{rank}(\mathbf{A})=r$ where $r\leq m\leq n$. Let $\mathbf{M}\in\mathbb{R}^{n\times n}$ be symmetric positive definite. Then*

$$\left[\mathbf{A}(\mathbf{M}+\mathbf{A}^\mathsf{T}\mathbf{A})^{-1}\mathbf{A}^\mathsf{T}\right]^\dagger\mathbf{A}=\left[\mathbf{I}+(\mathbf{A}\mathbf{M}^{-1}\mathbf{A}^\mathsf{T})^\dagger\right]\mathbf{A}, \tag{25}$$

which further implies

$$\left[\mathbf{A}(\mathbf{M}+\mathbf{A}^\mathsf{T}\mathbf{N}\mathbf{A})^{-1}\mathbf{A}^\mathsf{T}\right]^\dagger\mathbf{A}=\left[\mathbf{N}+(\mathbf{A}\mathbf{M}^{-1}\mathbf{A}^\mathsf{T})^\dagger\right]\mathbf{A}, \tag{26}$$

where $\mathbf{N}\in\mathbb{R}^{m\times m}$ is any symmetric positive definite matrix.

Proof: See Section VI of the supplementary material. ■

Lemma II.2. *Let $\mathbf{A}\in\mathbb{R}^{m\times n}$ and $\operatorname{rank}(\mathbf{A})=r$ where $r\leq m\leq n$. Let $\mathbf{M}\in\mathbb{R}^{n\times n}$ and $\mathbf{N}\in\mathbb{R}^{m\times m}$ be any symmetric positive definite matrices. Then*

$$\lim_{\epsilon\to 0}\mathbf{A}(\epsilon\mathbf{M}+\mathbf{A}^\mathsf{T}\mathbf{N}\mathbf{A})^{-1}\mathbf{A}^\mathsf{T}=\mathbf{A}(\mathbf{A}^\mathsf{T}\mathbf{N}\mathbf{A})^\dagger\mathbf{A}^\mathsf{T}. \tag{27}$$

Proof: See Section VII of the supplementary material. ■

Based on **Lemma II.1**, the iteration

$$\boldsymbol{\lambda}^{t+1}=\boldsymbol{\lambda}^t+\left(\mathbf{G}^t\right)^\dagger(\mathbf{I}-\mathbf{W})^{1/2}\mathbf{x}^{t+1}$$

can be equivalently written as

$$\boldsymbol{\lambda}^{t+1}=\boldsymbol{\lambda}^t+\alpha(\mathbf{I}-\mathbf{W})^{1/2}\mathbf{x}^{t+1}+\left(\tilde{\mathbf{G}}^t\right)^\dagger(\mathbf{I}-\mathbf{W})^{1/2}\mathbf{x}^{t+1}, \tag{28}$$

where

$$\tilde{\mathbf{G}}^t=(\mathbf{I}-\mathbf{W})^{1/2}(\nabla^2 f(\mathbf{x}^{t+1}))^{-1}(\mathbf{I}-\mathbf{W})^{1/2}.$$

Without the last term, (28) is just the dual ascent step of the standard Augmented Lagrange Method (ALM) for solving (10). Hence, the last term of (28) can be simply viewed as a second-order correction term for the dual update. Meanwhile, we also notice that the computation of $(\mathbf{I}-\mathbf{W})^{1/2}$ is not only expensive but also destroys the network structure. To overcome

this undesirable computation, multiplying (28) by $(\mathbf{I}-\mathbf{W})^{1/2}$ and denoting $\mathbf{v}=(\mathbf{I}-\mathbf{W})^{1/2}\boldsymbol{\lambda}$, we obtain

$$\begin{aligned}\mathbf{v}^{t+1}=&\mathbf{v}^t+\alpha(\mathbf{I}-\mathbf{W})\mathbf{x}^{t+1}\\&+(\mathbf{I}-\mathbf{W})^{1/2}\left(\tilde{\mathbf{G}}^t\right)^{\dagger}(\mathbf{I}-\mathbf{W})^{1/2}\mathbf{x}^{t+1}\\=&\mathbf{v}^t+\alpha(\mathbf{I}-\mathbf{W})\mathbf{x}^{t+1}+(\mathbf{I}-\mathbf{W})\mathbf{D}^t(\mathbf{I}-\mathbf{W})\mathbf{x}^{t+1},\end{aligned}\tag{29}$$

where

$$\mathbf{D}^t=\left((\mathbf{I}-\mathbf{W})^{1/2}\right)^{\dagger}\left(\tilde{\mathbf{G}}^t\right)^{\dagger}\left((\mathbf{I}-\mathbf{W})^{1/2}\right)^{\dagger}.$$

However, in practice, for computation efficiency and numerical stability, we would approximate $\mathbf{D}^t$ by $\tilde{\mathbf{P}}^t\tilde{\mathbf{D}}$, where $\tilde{\mathbf{P}}^t$ is a block diagonal approximation of $(\tilde{\mathbf{G}}^t+r^t\mathbf{I})^{-1}$ and $\tilde{\mathbf{D}}$ is a block diagonal matrix whose i-th diagonal block $\tilde{\mathbf{D}}_i=\frac{1}{1-\tilde{W}_{ii}}\mathbf{I}_p$. Here $r^t>0$ is a scalar. **Lemma II.2** implies that as r^t goes to zero, $(\mathbf{I}-\mathbf{W})^{1/2}(\tilde{\mathbf{G}}^t+r^t\mathbf{I})^{-1}(\mathbf{I}-\mathbf{W})^{1/2}$ approaches $(\mathbf{I}-\mathbf{W})^{1/2}(\tilde{\mathbf{G}}^t)^{\dagger}(\mathbf{I}-\mathbf{W})^{1/2}$. Finally, by introducing a dual stepsize $\gamma>0$ for ensuring global convergence, based on (29), we propose to update $\{\boldsymbol{\lambda}^t\}$ and $\{\mathbf{v}^t\}$ in our new algorithm as

$$\mathbf{v}^{t+1}=\mathbf{v}^t+\gamma(\mathbf{I}-\mathbf{W})\boldsymbol{\nu}^t \text{ and } \boldsymbol{\lambda}^{t+1}=\boldsymbol{\lambda}^t+\gamma(\mathbf{I}-\mathbf{W})^{1/2}\boldsymbol{\nu}^t,\tag{30}$$

where $\boldsymbol{\nu}^t=\alpha\mathbf{x}^{t+1}+\tilde{\mathbf{P}}^t\tilde{\mathbf{D}}(\mathbf{I}-\mathbf{W})\mathbf{x}^{t+1}$.

We now explain our construction of $\tilde{\mathbf{P}}^t$ by using BB [40] and dynamic average consensus techniques to capture certain curvature information of $(\tilde{\mathbf{G}}^t+r^t\mathbf{I})^{-1}$. First, by using again $\mathbf{B}^{t+1}$ to approximate $\nabla^2 f(\mathbf{x}^{t+1})$, noticing $\mathbf{v}^t-\mathbf{v}^{t-1}=(\mathbf{I}-\mathbf{W})^{1/2}(\boldsymbol{\lambda}^t-\boldsymbol{\lambda}^{t-1})$ and (30), let us define

$$\begin{aligned}\boldsymbol{\zeta}^{t-1}:=&(\mathbf{I}-\mathbf{W})^{1/2}(\mathbf{B}^t)^{-1}(\mathbf{I}-\mathbf{W})^{1/2}(\boldsymbol{\lambda}^t-\boldsymbol{\lambda}^{t-1})\\=&(\mathbf{I}-\mathbf{W})^{1/2}(\mathbf{B}^t)^{-1}(\mathbf{v}^t-\mathbf{v}^{t-1})\\=&(\mathbf{I}-\mathbf{W})^{1/2}\boldsymbol{\mu}^{t-1},\end{aligned}$$

where $\boldsymbol{\mu}^{t-1}=(\mathbf{B}^t)^{-1}(\mathbf{v}^t-\mathbf{v}^{t-1})$, and define

$$\boldsymbol{\xi}^{t-1}:=\boldsymbol{\lambda}^t-\boldsymbol{\lambda}^{t-1}=\gamma(\mathbf{I}-\mathbf{W})^{1/2}\boldsymbol{\nu}^{t-1}.$$

Then, the standard centralized BB technique suggests to approximate $(\tilde{\mathbf{G}}^t+r^t\mathbf{I})^{-1}$ by a scalar matrix $p^t\mathbf{I}$, where

$$\begin{aligned}p^t&=\left(\frac{(\boldsymbol{\zeta}^{t-1})^{\mathsf{T}}\boldsymbol{\xi}^{t-1}}{(\boldsymbol{\xi}^{t-1})^{\mathsf{T}}\boldsymbol{\xi}^{t-1}}+r^t\right)^{-1}\\&=\left(\frac{(\mathbf{v}^t-\mathbf{v}^{t-1})^{\mathsf{T}}\boldsymbol{\mu}^{t-1}}{\gamma(\mathbf{v}^t-\mathbf{v}^{t-1})^{\mathsf{T}}\boldsymbol{\nu}^{t-1}}+r^t\right)^{-1}\\&=\left(\frac{\sum_{i=1}^n\tilde{b}_i^t}{\sum_{i=1}^n\tilde{a}_i^t}+r^t\right)^{-1}\end{aligned}\tag{31}$$

with $\tilde{b}_i^t=(\mathbf{v}_i^t-\mathbf{v}_i^{t-1})^{\mathsf{T}}\boldsymbol{\mu}_i^{t-1}$ and $\tilde{a}_i^t=\gamma(\mathbf{v}_i^t-\mathbf{v}_i^{t-1})^{\mathsf{T}}\boldsymbol{\nu}_i^{t-1}=\gamma(\mathbf{v}_i^t-\mathbf{v}_i^{t-1})^{\mathsf{T}}\left(\alpha\mathbf{x}_i^t+\tilde{\mathbf{P}}_i^{t-1}\tilde{\mathbf{D}}_i(\mathbf{x}_i^t-\sum_{j\in\mathcal{N}_i}\tilde{W}_{ij}\mathbf{x}_j^t)\right)$. Obviously, the calculation of p^t in (31) needs global information from all nodes, which can not be realized in the decentralized setting. On the other hand, we also notice that the scalar $\sum_{i=1}^n\tilde{b}_i^t/\sum_{i=1}^n\tilde{a}_i^t$ in (31) is in fact the ratio of the average values of $\tilde{a}_i^t$ and $\tilde{b}_i^t$ over n nodes. Motivated by the idea of dynamic average consensus [41], we set $\tilde{\mathbf{P}}^t$ to have the following block scalar matrix format

$$\tilde{\mathbf{P}}^t=\begin{bmatrix}\tilde{p}_1^t\mathbf{I}_p & & \\ & \ldots & \\ & & \tilde{p}_n^t\mathbf{I}_p\end{bmatrix},\tag{32}$$

where

$$\tilde{p}_i^t=\begin{cases}\left(\mathrm{Proj}_{[\underline{\omega},\overline{\omega}]}\left(\frac{b_i^t}{a_i^t}\right)+r^t\right)^{-1}, & \text{if } a_i^t\neq 0;\\ (\overline{\omega}+r^t)^{-1}, & \text{if } a_i^t=0 \text{ and } b_i^t>0;\\ (\underline{\omega}+r^t)^{-1}, & \text{if } a_i^t=0 \text{ and } b_i^t\leq 0,\end{cases}\tag{33}$$

with $\underline{\omega}>\overline{\omega}>0$ and the ratio $\frac{0}{0}$ being defined as zero. Here, a_i^t and b_i^t are certain estimations of $\frac{1}{n}\sum_{i=1}^n\tilde{a}_i^t$ and $\frac{1}{n}\sum_{i=1}^n\tilde{b}_i^t$, respectively, and can be calculated only by the local information from neighboring nodes. More specifically, in our algorithm a_i^t and b_i^t are calculated as follows:

$$a_i^t=\sum_{j\in\mathcal{N}_i}\tilde{W}_{ij}a_j^{t-1}+\tilde{a}_i^t-\tilde{a}_i^{t-1},\ a_i^0=\tilde{a}_i^0=1,\tag{34}$$

$$b_i^t=\sum_{j\in\mathcal{N}_i}\tilde{W}_{ij}b_j^{t-1}+\tilde{b}_i^t-\tilde{b}_i^{t-1},\ b_i^0=\tilde{b}_i^0=1.\tag{35}$$

With the above calculation, we have the following lemma.

Lemma II.3. $\frac{1}{n}\sum_{i=1}^n a_i^t=\frac{1}{n}\sum_{i=1}^n\tilde{a}_i^t$, $\frac{1}{n}\sum_{i=1}^n b_i^t=\frac{1}{n}\sum_{i=1}^n\tilde{b}_i^t$.

Proof: See Section VIII of the supplementary material. ■

Noticing $\mathbf{v}^t=(\mathbf{I}-\mathbf{W})^{1/2}\boldsymbol{\lambda}^t$, the primal updates in (21) can be also written as

$$\begin{aligned}\mathbf{x}^{t+1}=&\mathbf{x}^t-\beta\left[\mathbf{I}-\theta\alpha(\mathbf{B}^t)^{-1}(\mathbf{I}-\mathbf{W})\right]\\&\times(\mathbf{B}^t)^{-1}\nabla_{\mathbf{x}}L_\alpha(\mathbf{x}^t,\mathbf{v}^t),\end{aligned}\tag{36}$$

where

$$L_\alpha(\mathbf{x},\mathbf{v})=f(\mathbf{x})+\langle\mathbf{v},\mathbf{x}\rangle+\frac{\alpha}{2}\mathbf{x}^{\mathsf{T}}(\mathbf{I}-\mathbf{W})\mathbf{x}.\tag{37}$$

Then, summarizing the above discussion, our decentralized primal-dual method (DPDM) is given in **Algorithm 1**.

Remark 1.

- ***Algorithm 1*** *can be run parallelly on each node, which only uses the local information from neighboring nodes. The updates of* $\mathbf{B}^t$ *and* $\tilde{\mathbf{P}}^t$ *exploit second-order information and only involve matrix-vector products, which would be computationally efficient.*
- *At most* $2+I_{R\setminus\{0\}}(\theta)$ *rounds of vector communications are taken in both the primal and dual updates of DPDM, which is comparable to or mildly higher than some other primal-dual quasi-Newton methods (see Table I for details). However, our numerical experiments show the reduction in the total number of iterations and the CPU time by applying quasi-Newton techniques in DPDM would offset its slightly higher communication cost per iteration.*
- ***Algorithm 1*** *has the following relationships with some existing well-known decentralized algorithms. Set the parameters* $\beta=1$, $\gamma=1$, *and* $\theta=0$ *in* ***Algorithm 1****. If*

Algorithm 1 DPDM

Input: $\mathbf{x}^0$, MaxIter, α, β, γ, θ, $\underline{\omega}$, $\overline{\omega}$, $\mathbf{W}$, $\{r^t\}_{t\geq 0}$.

1: Set $t=0$, $T=\text{MaxIter}$, $\mathbf{v}^0=\mathbf{0}$, $\tilde{\mathbf{P}}^0=\frac{1}{1+r^0}\mathbf{I}$, $\mathbf{H}^0=\mathbf{I}$, $a_i^0=\tilde{a}_i^0=1$, $b_i^0=\tilde{b}_i^0=1$, $i=1,\ldots,n$.

2: If $t\geq T$, stop.

3: $\mathbf{x}^{t+1}=\mathbf{x}^t-\beta\left[\mathbf{I}-\theta\alpha\mathbf{H}^t(\mathbf{I}-\mathbf{W})\right]\mathbf{H}^t\nabla_{\mathbf{x}}L_\alpha(\mathbf{x}^t,\mathbf{v}^t)$.

4: $\mathbf{H}^{t+1}=\begin{bmatrix}\mathbf{H}_1^{t+1} & & \\ & \ldots & \\ & & \mathbf{H}_n^{t+1}\end{bmatrix}$, where $\mathbf{H}_i^{t+1}=\mathbf{H}_i^t-\frac{\mathbf{H}_i^t\mathbf{y}_i^t(\mathbf{s}_i^t)^\top+\mathbf{s}_i^t(\mathbf{y}_i^t)^\top\mathbf{H}_i^t}{(\mathbf{s}_i^t)^\top\mathbf{y}_i^t}+\left(1+\frac{(\mathbf{y}_i^t)^\top\mathbf{H}_i^t\mathbf{y}_i^t}{(\mathbf{s}_i^t)^\top\mathbf{y}_i^t}\right)\frac{\mathbf{s}_i^t(\mathbf{s}_i^t)^\top}{(\mathbf{s}_i^t)^\top\mathbf{y}_i^t}$,

$$\mathbf{s}_i^t=\mathbf{x}_i^{t+1}-\mathbf{x}_i^t \text{ and } \mathbf{y}_i^t=\nabla f_i(\mathbf{x}_i^{t+1})-\nabla f_i(\mathbf{x}_i^t).$$

5: If $t\geq 1$, $\tilde{\mathbf{P}}^t=\begin{bmatrix}\tilde{p}_1^t\mathbf{I}_p & & \\ & \ldots & \\ & & \tilde{p}_n^t\mathbf{I}_p\end{bmatrix}$, where $\tilde{p}_i^t=\begin{cases}\left(\text{Proj}_{[\underline{\omega},\overline{\omega}]}\left(\frac{b_i^t}{a_i^t}\right)+r^t\right)^{-1}, & \text{if } a_i^t\neq 0;\\ \left(\overline{\omega}+r^t\right)^{-1}, & \text{if } a_i^t=0 \text{ and } b_i^t>0;\\ \left(\underline{\omega}+r^t\right)^{-1}, & \text{if } a_i^t=0 \text{ and } b_i^t\leq 0,\end{cases}$

$$a_i^t=\sum_{j\in\mathcal{N}_i}\tilde{W}_{ij}a_j^{t-1}+\tilde{a}_i^t-\tilde{a}_i^{t-1},\quad b_i^t=\sum_{j\in\mathcal{N}_i}\tilde{W}_{ij}b_j^{t-1}+\tilde{b}_i^t-\tilde{b}_i^{t-1},$$

$$\tilde{a}_i^t=\gamma(\mathbf{v}_i^t-\mathbf{v}_i^{t-1})^\top\left(\alpha\mathbf{x}_i^t+\tilde{\mathbf{P}}_i^{t-1}\tilde{\mathbf{D}}_i\mathbf{u}_i^t\right),\quad \tilde{b}_i^t=(\mathbf{v}_i^t-\mathbf{v}_i^{t-1})^\top\mathbf{H}_i^t(\mathbf{v}_i^t-\mathbf{v}_i^{t-1}).$$

6: $\mathbf{v}^{t+1}=\mathbf{v}^t+\gamma(\mathbf{I}-\mathbf{W})\left(\alpha\mathbf{x}^{t+1}+\tilde{\mathbf{P}}^t\tilde{\mathbf{D}}\mathbf{u}^{t+1}\right)$, where $\mathbf{u}^{t+1}=(\mathbf{I}-\mathbf{W})\mathbf{x}^{t+1}$.

7: Set $t=t+1$ and go to Step 2.

Output: $\mathbf{x}^T$.

$\mathbf{B}^t=\nabla^2 f(\mathbf{x}^t)+\epsilon\mathbf{I}$, $\tilde{\mathbf{P}}^t=\mathbf{0}$, *we obtain the algorithm NT [29], while the algorithm ESOM [28] takes* $(\mathbf{B}^t)^{-1}$ *as truncated Taylor's series expansion of the Hessian inverse with* $\tilde{\mathbf{P}}^t=\mathbf{0}$. *On the other hand, if* $\mathbf{B}^t=2\alpha\mathbf{I}$ *and* $\tilde{\mathbf{P}}^t=\mathbf{0}$, ***Algorithm 1*** *can be reduced as*

$$\begin{aligned}\mathbf{x}^{t+1}=&(\mathbf{I}+\mathbf{W})\mathbf{x}^t-\frac{\mathbf{I}+\mathbf{W}}{2}\mathbf{x}^{t-1}-\frac{1}{2\alpha}\nabla f(\mathbf{x}^t)\\&+\frac{1}{2\alpha}\nabla f(\mathbf{x}^{t-1})\end{aligned}$$

which is equivalent to EXTRA [10] with the stepsize $1/(2\alpha)$.

- *A single quasi-Newton primal steps is performed in* ***Algorithm 1***. *Allowing multiple primal updates per iteration will yield a more accurate approximate solution* $\mathbf{x}^{t+1}$ *of minimizing* $L_\alpha(\mathbf{x},\mathbf{v}^t)$, *which would also benefit the dual update. By this consideration, a generalized decentralized primal-dual method (GDPDM) is given as* ***Algorithm 2***, *where* $S\geq 1$ *is the number of primal steps per iteration.*

c) Quasi-Newton tracking: As Newton Tracking [29], we show that DPDM also maintains a quasi-Newton tracking property, implying that the updating direction of each $\mathbf{x}_i^t$ is a local quasi-Newton direction and will be a global quasi-Newton direction when $\{\mathbf{x}_i^t\}_{i=1}^n$ are getting consensus.

First, by (36), we have

$$\begin{aligned}&\mathbf{B}^t\mathbf{x}^{t+1}-\mathbf{B}^t\mathbf{x}^t\\=&-\beta\left[\nabla f\left(\mathbf{x}^t\right)+\mathbf{v}^t+\alpha(\mathbf{I}-\mathbf{W})\mathbf{x}^t\right]\\&+\beta\theta\alpha(\mathbf{I}-\mathbf{W})(\mathbf{B}^t)^{-1}\left[\nabla f\left(\mathbf{x}^t\right)+\mathbf{v}^t+\alpha(\mathbf{I}-\mathbf{W})\mathbf{x}^t\right]\\=&-\beta\left[\nabla f\left(\mathbf{x}^t\right)+\mathbf{v}^t+\alpha(\mathbf{I}-\mathbf{W})\mathbf{x}^t\right]+\beta(\mathbf{I}-\mathbf{W})\mathbf{h}^t,\end{aligned}$$

where

$$\mathbf{h}^t=\theta\alpha(\mathbf{B}^t)^{-1}\left[\nabla f\left(\mathbf{x}^t\right)+\mathbf{v}^t+\alpha(\mathbf{I}-\mathbf{W})\mathbf{x}^t\right].$$

Hence, we have

$$\mathbf{x}^{t+1}=\mathbf{x}^t-\beta\mathbf{d}^t, \tag{38}$$

where $\mathbf{d}^t=(\mathbf{B}^t)^{-1}\mathbf{g}^t$ and

$$\mathbf{g}^t=\nabla f\left(\mathbf{x}^t\right)+\mathbf{v}^t+(\mathbf{I}-\mathbf{W})(\alpha\mathbf{x}^t+\mathbf{h}^t). \tag{39}$$

We now show the quasi-Newton tracking by (38), (39), and (22). Firstly, $\mathbf{B}_i^{t+1}$ given by (23) is actually the solution of the following optimization problem

$$\begin{aligned}\mathbf{B}_i^{t+1}=\arg\min_{\mathbf{Z}}\ &\text{tr}\left[\left(\mathbf{B}_i^t\right)^{-1}\mathbf{Z}\right]-\log\det\left[\left(\mathbf{B}_i^t\right)^{-1}\mathbf{Z}\right],\\ s.t.\quad &\mathbf{Z}\mathbf{s}_i^t=\mathbf{y}_i^t,\quad \mathbf{Z}\succeq 0,\end{aligned}$$

where $\mathbf{s}_i^t=\mathbf{x}_i^{t+1}-\mathbf{x}_i^t$, $\mathbf{y}_i^t=\nabla f_i(\mathbf{x}_i^{t+1})-\nabla f_i(\mathbf{x}_i^t)$. Hence, we have

$$\mathbf{B}_i^{t+1}\mathbf{s}_i^t=\mathbf{y}_i^t. \tag{40}$$

Summing (40) over $i=1,\ldots,n$, we obtain

$$\sum_{i=1}^n\mathbf{B}_i^{t+1}(\mathbf{x}_i^{t+1}-\mathbf{x}_i^t)=\sum_{i=1}^n\left(\nabla f_i(\mathbf{x}_i^{t+1})-\nabla f_i(\mathbf{x}_i^t)\right). \tag{41}$$

When $\{\mathbf{x}_i^t\}_{i=1}^n$ are consensus for large t, i.e., $\mathbf{x}_i^t=\bar{\mathbf{x}}^t$ for all $i=1,\ldots,n$ and large t, (41) implies $\sum_{i=1}^n\mathbf{B}_i^{t+1}$ will satisfy the global quasi-Newton secant equation of the original problem (1), i.e.,

$$\sum_{i=1}^n\mathbf{B}_i^{t+1}(\bar{\mathbf{x}}^{t+1}-\bar{\mathbf{x}}^t)=\nabla F(\bar{\mathbf{x}}^{t+1})-\nabla F(\bar{\mathbf{x}}^t),$$

Algorithm 2 GDPDM

Input: $\mathbf{x}^0$, MaxIter1, MaxIter2, α, β, γ, θ, $\underline{\omega}$, $\overline{\omega}$, $\mathbf{W}$, $\{r^t\}_{t\geq 0}$.

1: Set $t=0$, $T=$MaxIter1, $S=$MaxIter2, $\mathbf{v}^0=\mathbf{0}$, $\tilde{\mathbf{P}}^0=\frac{1}{1+r^0}\mathbf{I}$, $\mathbf{H}^0=\mathbf{I}$, $a_i^0=\tilde{a}_i^0=1$, $b_i^0=\tilde{b}_i^0=1$, $i=1,\ldots,n$.

2: If $t\geq T$, stop.

3: $\mathbf{x}^{t,0}=\mathbf{x}^t$, $\mathbf{H}^{t,0}=\mathbf{H}^t$.

4: For $s=0,1,\ldots,S-1$

5: $\mathbf{x}^{t,s+1}=\mathbf{x}^{t,s}-\beta\left[\mathbf{I}-\theta\alpha\mathbf{H}^{t,s}(\mathbf{I}-\mathbf{W})\right]\mathbf{H}^{t,s}\nabla_{\mathbf{x}}L_\alpha(\mathbf{x}^{t,s},\mathbf{v}^t)$.

6: $\mathbf{H}^{t,s+1}=\begin{bmatrix}\mathbf{H}_1^{t,s+1} & & \\ & \ldots & \\ & & \mathbf{H}_n^{t,s+1}\end{bmatrix}$, where

$$\mathbf{H}_i^{t+1,s}=\mathbf{H}_i^{t,s}-\frac{\mathbf{H}_i^{t,s}\mathbf{y}_i^{t,s}(\mathbf{s}_i^{t,s})^\mathsf{T}+\mathbf{s}_i^{t,s}(\mathbf{y}_i^{t,s})^\mathsf{T}\mathbf{H}_i^{t,s}}{(\mathbf{s}_i^{t,s})^\mathsf{T}\mathbf{y}_i^{t,s}}+\left(1+\frac{(\mathbf{y}_i^{t,s})^\mathsf{T}\mathbf{H}_i^{t,s}\mathbf{y}_i^{t,s}}{(\mathbf{s}_i^{t,s})^\mathsf{T}\mathbf{y}_i^{t,s}}\right)\frac{\mathbf{s}_i^{t,s}(\mathbf{s}_i^{t,s})^\mathsf{T}}{(\mathbf{s}_i^{t,s})^\mathsf{T}\mathbf{y}_i^{t,s}},$$
$$\mathbf{s}_i^{t,s}=\mathbf{x}_i^{t,s+1}-\mathbf{x}_i^{t,s}\text{ and }\mathbf{y}_i^{t,s}=\nabla f_i(\mathbf{x}_i^{t,s+1})-\nabla f_i(\mathbf{x}_i^{t,s}).$$

7: End for.

8: $\mathbf{x}^{t+1}=\mathbf{x}^{t,S}$, $\mathbf{H}^{t+1}=\mathbf{H}^{t,S}$.

9: If $t\geq 1$, $\tilde{\mathbf{P}}^t=\begin{bmatrix}\tilde{p}_1^t\mathbf{I}_p & & \\ & \ldots & \\ & & \tilde{p}_n^t\mathbf{I}_p\end{bmatrix}$, where $\tilde{p}_i^t=\begin{cases}\left(\mathrm{Proj}_{[\underline{\omega},\overline{\omega}]}\left(\frac{b_i^t}{a_i^t}\right)+r^t\right)^{-1}, & \text{if } a_i^t\neq 0;\\ (\overline{\omega}+r^t)^{-1}, & \text{if } a_i^t=0 \text{ and } b_i^t>0;\\ (\underline{\omega}+r^t)^{-1}, & \text{if } a_i^t=0 \text{ and } b_i^t\leq 0,\end{cases}$

$$a_i^t=\sum_{j\in\mathcal{N}_i}\tilde{W}_{ij}a_j^{t-1}+\tilde{a}_i^t-\tilde{a}_i^{t-1},\quad b_i^t=\sum_{j\in\mathcal{N}_i}\tilde{W}_{ij}b_j^{t-1}+\tilde{b}_i^t-\tilde{b}_i^{t-1},$$
$$\tilde{a}_i^t=\gamma(\mathbf{v}_i^t-\mathbf{v}_i^{t-1})^\mathsf{T}\left(\alpha\mathbf{x}_i^t+\tilde{\mathbf{P}}_i^{t-1}\tilde{\mathbf{D}}_i\mathbf{u}_i^t\right),\quad \tilde{b}_i^t=(\mathbf{v}_i^t-\mathbf{v}_i^{t-1})^\mathsf{T}\mathbf{H}_i^t(\mathbf{v}_i^t-\mathbf{v}_i^{t-1}).$$

10: $\mathbf{v}^{t+1}=\mathbf{v}^t+\gamma(\mathbf{I}-\mathbf{W})\left(\alpha\mathbf{x}^{t+1}+\tilde{\mathbf{P}}^t\tilde{\mathbf{D}}\mathbf{u}^{t+1}\right)$, where $\mathbf{u}^{t+1}=(\mathbf{I}-\mathbf{W})\mathbf{x}^{t+1}$.

11: Set $t=t+1$ and go to Step 2.

Output: $\mathbf{x}^T$.

where $F(\mathbf{z})=\sum_{i=1}^n f_i(\mathbf{z})$. On the other hand, by (39) and (30), we have

$$\begin{aligned}&\mathbf{B}^{t+1}\mathbf{d}^{t+1}-\mathbf{B}^t\mathbf{d}^t=\mathbf{g}^{t+1}-\mathbf{g}^t \\ =&\nabla f\left(\mathbf{x}^{t+1}\right)-\nabla f\left(\mathbf{x}^t\right)+\mathbf{v}^{t+1}-\mathbf{v}^t \\ &+(\mathbf{I}-\mathbf{W})\left[\alpha(\mathbf{x}^{t+1}-\mathbf{x}^t)+\mathbf{h}^{t+1}-\mathbf{h}^t\right] \\ =&\nabla f\left(\mathbf{x}^{t+1}\right)-\nabla f\left(\mathbf{x}^t\right)+(\mathbf{I}-\mathbf{W})\tilde{\mathbf{g}}^t,\end{aligned}\tag{42}$$

where $\tilde{\mathbf{g}}^t=\alpha(\mathbf{x}^{t+1}-\mathbf{x}^t)+\gamma\boldsymbol{\nu}^t+\mathbf{h}^{t+1}-\mathbf{h}^t$. The i-th block of (42) can be written as

$$\begin{aligned}&\mathbf{B}_i^{t+1}\mathbf{d}_i^{t+1}-\mathbf{B}_i^t\mathbf{d}_i^t \\ =&\nabla f_i\left(\mathbf{x}_i^{t+1}\right)-\nabla f_i\left(\mathbf{x}_i^t\right)+\left(\tilde{\mathbf{g}}_i^t-\sum_{j\in\mathcal{N}_i}\tilde{W}_{ij}\tilde{\mathbf{g}}_j^t\right).\end{aligned}\tag{43}$$

Summing (43) over $i=1,\ldots,n$ and using $\tilde{\mathbf{W}}^\mathsf{T}\mathbf{1}=\mathbf{1}$ yields

$$\sum_{i=1}^n\mathbf{B}_i^{t+1}\mathbf{d}_i^{t+1}=\sum_{i=1}^n\mathbf{B}_i^t\mathbf{d}_i^t+\sum_{i=1}^n\left(\nabla f_i(\mathbf{x}_i^{t+1})-\nabla f_i(\mathbf{x}_i^t)\right).\tag{44}$$

Since $\mathbf{v}^0=\mathbf{0}$ in DPDM, we have from (39) that

$$\sum_{i=1}^n\mathbf{B}_i^0\mathbf{d}_i^0=\sum_{i=1}^n\mathbf{g}_i^0=\sum_{i=1}^n\nabla f_i\left(\mathbf{x}_i^0\right).$$

So, summing (44) over t, we have

$$\sum_{i=1}^n\mathbf{B}_i^t\mathbf{d}_i^t=\sum_{i=1}^n\nabla f_i\left(\mathbf{x}_i^t\right).\tag{45}$$

Then, when $\{\mathbf{x}_i^t\}_{i=1}^n$ are getting consensus for large t, we have from $\mathbf{d}_i^t=(\mathbf{x}_i^t-\mathbf{x}_i^{t+1})/\beta$ that $\{\mathbf{d}_i^t\}_{i=1}^n$ are also getting consensus for large t, which implies $\frac{1}{n}\sum_{i=1}^n\mathbf{d}_i^t=:\bar{\mathbf{d}}^t=\mathbf{d}_i^t$ for all $i=1,\ldots,n$ and large t. Hence, by (45), we have

$$\sum_{i=1}^n\mathbf{B}_i^t\bar{\mathbf{d}}^t=\sum_{i=1}^n\nabla f_i\left(\bar{\mathbf{x}}^t\right),\tag{46}$$

which together with (41) shows $\bar{\mathbf{d}}^t$ is a a global quasi-Newton direction of original problem (1).

III. CONVERGENCE ANALYSIS

In this section, we will analyze the global convergence of **Algorithm 2**. Note that **Algorithm 1** is a special case of **Algorithm 2** with $S=1$. For convenience, let us denote $\mathbf{Z}=\mathbf{I}-\mathbf{W}$, $\rho=\boldsymbol{\lambda}_{\max}(\mathbf{Z})$, $\sigma=\boldsymbol{\lambda}_2(\mathbf{Z})$ and

$$\tilde{\mathbf{B}}^{t,s}=\left[\mathbf{I}-\theta\alpha(\mathbf{B}^{t,s})^{-1}\mathbf{Z}\right]\left(\mathbf{B}^{t,s}\right)^{-1}.$$

Then, **Algorithm 2** gives

$$\mathbf{x}^{t,s+1}=\mathbf{x}^{t,s}-\beta\tilde{\mathbf{B}}^{t,s}\nabla_{\mathbf{x}}L_\alpha(\mathbf{x}^{t,s},\mathbf{v}^t).\tag{47}$$

By **Assumption 1**, **2**, we can easily get

$$\mu\mathbf{I} \preceq \nabla_{\mathbf{xx}} L_\alpha(\mathbf{x},\mathbf{v}) = \nabla_{\mathbf{xx}} \tilde{L}_\alpha(\mathbf{x},\boldsymbol{\lambda}) \preceq \mathcal{L}\mathbf{I},\ \forall \mathbf{x}\in\mathbb{R}^{np}, \quad (48)$$

where $\mathcal{L} = L + \rho\alpha$. We also make the following assumption.

Assumption 3. *The approximate matrices $\{\mathbf{B}^{t,s}\}$ satisfy*

$$\psi\mathbf{I} \preceq \left(\mathbf{B}^{t,s}\right)^{-1} \preceq \bar{\psi}\mathbf{I},$$

for any t and s, where $\bar{\psi} > \psi > 0$. And the parameter θ is chosen such that $0 \le \theta < \min\{1, \psi/(\alpha\rho)\}$.

From **Assumption 3**, we can easily derive

$$\psi\mathbf{I} \preceq \tilde{\mathbf{B}}^{t,s} \preceq \Psi\mathbf{I}, \text{ where } \Psi = \frac{\bar{\psi}\psi}{\psi - \theta\alpha\rho}.$$

Let $\mathbf{P}^t = \alpha\mathbf{I} + (\mathbf{I}-\mathbf{W})^{1/2}\tilde{\mathbf{P}}^t\tilde{\mathbf{D}}(\mathbf{I}-\mathbf{W})^{1/2}$. From (33), we can also derive

$$\alpha\mathbf{I} \preceq \mathbf{P}^t \preceq \bar{\alpha}\mathbf{I}, \text{ where } \bar{\alpha} = \alpha + \frac{\rho}{\underline{\omega}(1-\max_i\{\tilde{W}_{ii}\})}.$$

In addition, by (30), we have

$$\mathbf{v}^{t+1} = \mathbf{v}^t + \gamma(\mathbf{I}-\mathbf{W})^{1/2}\mathbf{P}^t(\mathbf{I}-\mathbf{W})^{1/2}\mathbf{x}^{t+1}, \quad (49)$$

and

$$\boldsymbol{\lambda}^{t+1} = \boldsymbol{\lambda}^t + \gamma\mathbf{P}^t(\mathbf{I}-\mathbf{W})^{1/2}\mathbf{x}^{t+1}. \quad (50)$$

In the following, we define β_1 and β_2 be lower and upper bounds of eigenvalues of $\beta\tilde{\mathbf{B}}^{t,s}$:

$$\beta_1 = \beta\psi \quad \text{and} \quad \beta_2 = \beta\Psi, \quad (51)$$

and let γ_1 and γ_2 the lower and upper bounds to eigenvalues of $\gamma\mathbf{P}^t$:

$$\gamma_1 = \gamma\alpha \quad \text{and} \quad \gamma_2 = \gamma\bar{\alpha}. \quad (52)$$

By **Assumption 2** and Slater's condition, which holds due to linear constraints, strong duality holds for the problem (10). Hence, a dual equivalent problem to the problem (10) as well as the problem (3) is

$$\max_{\boldsymbol{\lambda}\in\mathbb{R}^{np}} g(\boldsymbol{\lambda}), \quad (53)$$

where $g(\boldsymbol{\lambda}) = \min_{\mathbf{x}\in\mathbb{R}^{np}} \tilde{L}_\alpha(\mathbf{x},\boldsymbol{\lambda})$ is the dual function. In addition, denoting $\tilde{f}(\cdot) := f(\cdot) + (\alpha/2)\|\cdot\|_{\mathbf{Z}}$, we have

$$\begin{aligned} g(\boldsymbol{\lambda}) =& \min_{\mathbf{x}}\{f(\mathbf{x}) + \boldsymbol{\lambda}^{\mathsf{T}}\mathbf{Z}^{1/2}\mathbf{x} + (\alpha/2)\mathbf{x}^{\mathsf{T}}\mathbf{Z}\mathbf{x}\} \quad (54)\\ =& -\tilde{f}^*(-\mathbf{Z}^{1/2}\boldsymbol{\lambda}), \end{aligned}$$

where $\tilde{f}^*$ is the conjugate function of $\tilde{f}$. Note that $\tilde{f}^*$ is $1/\mathcal{L}$-strongly convex. Let $-\mathbf{v}^*$ be the unique minimizer of $\tilde{f}^*$ and Λ^* be the dual optimal solution set of (53). Then, it follows from (54) that

$$\Lambda^* = \{\boldsymbol{\lambda}^* : \mathbf{Z}^{1/2}\boldsymbol{\lambda} = \mathbf{v}^*\}$$

and for any $\boldsymbol{\lambda}^* \in \Lambda^*$, we have

$$g(\boldsymbol{\lambda}^*) = -\tilde{f}^*(-\mathbf{Z}^{1/2}\boldsymbol{\lambda}^*) = -\tilde{f}^*(-\mathbf{v}^*).$$

Furthermore, by (53) and the definition of $\mathbf{x}^*(\boldsymbol{\lambda})$ in (11), we have $g(\boldsymbol{\lambda}) = \tilde{L}_\alpha(\mathbf{x}^*(\boldsymbol{\lambda}),\boldsymbol{\lambda})$,

$$\nabla g(\boldsymbol{\lambda}) = \mathbf{Z}^{1/2}\mathbf{x}^*(\boldsymbol{\lambda}), \quad (55)$$

and $\mathbf{x}^*(\boldsymbol{\lambda}^*) = \mathbf{x}^*$ by the strong duality. The following lemma reveals some important properties of the function $g(\cdot)$.

Lemma III.1. *Under **Assumptions 1**, **2** and **3**, the dual function $g(\cdot)$ is L_g-Lipschitz smooth and for any $\boldsymbol{\lambda}^* \in \Lambda^*$, the PL inequality holds*

$$g(\boldsymbol{\lambda}^*) - g(\boldsymbol{\lambda}^t) \le \frac{1}{2\mu_g}\|\nabla g(\boldsymbol{\lambda}^t)\|^2$$

for all $t \ge 0$, where $\mu_g = \sigma/\mathcal{L}$ and $L_g = \rho/\mu$.

Proof: See Section IX of the supplementary material. ■

To establish a key recursive relation in our convergence analysis, we define the following quantities:

$$\begin{aligned} &\Delta^t_{\boldsymbol{\lambda}} = g(\boldsymbol{\lambda}^*) - g\left(\boldsymbol{\lambda}^t\right),\\ &\Delta^t_{\mathbf{x}} = \tilde{L}_\alpha\left(\mathbf{x}^t,\boldsymbol{\lambda}^t\right) - \tilde{L}_\alpha\left(\mathbf{x}^*\left(\boldsymbol{\lambda}^t\right),\boldsymbol{\lambda}^t\right) \text{ and}\\ &\Delta^{t,s}_{\mathbf{x}} = \tilde{L}_\alpha\left(\mathbf{x}^{t,s},\boldsymbol{\lambda}^t\right) - \tilde{L}_\alpha\left(\mathbf{x}^*(\boldsymbol{\lambda}^t),\boldsymbol{\lambda}^t\right),\\ &\text{for } s = 0,1,\ldots,S-1, \end{aligned}$$

where $\boldsymbol{\lambda}^*$ can be an optimal dual solution in Λ^* and $\Delta^t_{\boldsymbol{\lambda}}$ is the dual optimality gap. We define a potential function by combing the performance metrics $\Delta^t_{\boldsymbol{\lambda}}$ and $\Delta^t_{\mathbf{x}}$ as

$$\Delta^t = 7\Delta^t_{\boldsymbol{\lambda}} + \Delta^t_{\mathbf{x}}. \quad (56)$$

The following theorem provides the linear decay rate of Δ^t under proper choice of primal and dual step sizes.

Theorem III.2. *Under **Assumptions 1**, **2**, and **3**, if the primal and dual step sizes, namely β and γ, satisfy $\beta \le \frac{1}{\Psi\mathcal{L}}$ and*

$$\gamma < \min\left\{\frac{1}{6\bar{\alpha}L_g}, \left(\frac{1}{(1-\beta\psi\mu)^S} - 1\right)\frac{\mu^2}{11\bar{\alpha}\mathcal{L}\rho}\right\},$$

there exists a constant $\kappa \in (0,1)$ such that

$$\Delta^{t+1} \le \kappa\Delta^t,$$

where

$$\begin{aligned} \kappa = \max\Bigg\{& 1 - \frac{2}{7}\left(1 - 6L_g\gamma\bar{\alpha}\right)\mu_g\gamma\alpha, \quad (57)\\ & \left(1 + \frac{(12L_g\gamma\bar{\alpha}+10)\mathcal{L}\rho\gamma\bar{\alpha}}{\mu^2}\right)(1-\beta\psi\mu)^S\Bigg\}. \end{aligned}$$

Especially, when $S = 1$, if $\beta \le \frac{1}{2\Psi\mathcal{L}}$ and $\gamma \le \min\left\{\frac{1}{12\bar{\alpha}L_g}, \frac{4\mu^2}{99\bar{\alpha}\mathcal{L}\rho}\right\}$, there exists a constant $q \in (0,1)$ such that

$$\Delta^{t+1} \le (1-q)\Delta^t,$$

where $q = \min\left\{\frac{\gamma_1\mu_g}{7}, \frac{\beta\psi\mu}{2}\right\}$.

Proof: See Section X of the supplementary material. ■

By **Theorem III.2**, we can directly establish that $\mathbf{x}^t$ converges to the unique minimizer $\mathbf{x}^* = \mathbf{x}^*(\boldsymbol{\lambda}^*)$ of the original problem (10) at a linear rate.

Corollary III.3. *Under **Assumptions 1**, **2**, and **3**, if the primal and dual step sizes, namely β and γ, satisfy $\beta \le \frac{1}{\Psi\mathcal{L}}$ and*

$$\gamma < \min\left\{\frac{1}{6\bar{\alpha}L_g}, \left(\frac{1}{(1-\beta\psi\mu)^S} - 1\right)\frac{\mu^2}{11\bar{\alpha}\mathcal{L}\rho}\right\},$$

the iterates $\{\mathbf{x}^t\}$ *generated by* ***Algorithm 2*** *converge to* $\mathbf{x}^*(\boldsymbol{\lambda}^*)$ *linearly, more specifically, there exists a constant* $\kappa \in (0,1)$ *such that*

$$\left\|\mathbf{x}^t - \mathbf{x}^*(\boldsymbol{\lambda}^*)\right\|^2 \leq c\kappa^t,$$

where $c = 4\Delta^0/\left(\mu \min\left\{\frac{7\mu}{\mathcal{L}}, 1\right\}\right)$ *and* $0 < \kappa < 1$ *is defined in* (57). *Especially, when* $S = 1$, *if* $\beta \leq \frac{1}{2\Psi\mathcal{L}}$ *and* $\gamma \leq \min\left\{\frac{1}{12\bar{\alpha}L_g}, \frac{4\mu^2}{99\bar{\alpha}\mathcal{L}\rho}\right\}$, *we have* $\|\mathbf{x}^t - \mathbf{x}^*(\boldsymbol{\lambda}^*)\|^2 \leq c(1-q)^t$, *where* $q = \min\left\{\frac{\gamma_1\mu_g}{7}, \frac{\beta\psi\mu}{2}\right\}$.

Proof: See Section XI of the supplementary material. ■

Remark 2. *The condition numbers of the objective function and augmented Lagrangian function are defined as*

$$\kappa_f := \frac{L}{\mu} \quad \textit{and} \quad \kappa_l := \frac{\mathcal{L}}{\mu}.$$

The condition number of the network can be defined as

$$\kappa_g := \frac{\lambda_{\max}(\mathbf{I} - \tilde{\mathbf{W}})}{\lambda_2(\mathbf{I} - \tilde{\mathbf{W}})} = \frac{\rho}{\sigma},$$

which measures the network topology and is an important factor affecting the performance of decentralized methods. In general, a smaller condition number means greater connectivity of the network. From ***Corollary III.3****, we can derive the iteration complexity of our DPDM as*

$$\mathcal{O}\left(\kappa_l^2\kappa_g log\left(\frac{1}{\epsilon}\right)\right).$$

When S *is sufficiently large, the iteration complexity of GDPDM can reduce to* $\mathcal{O}\left(\kappa_l\kappa_g log\left(\frac{1}{\epsilon}\right)\right)$.

Our methods do not have a faster-than-linear rate since the dual Hessian is approximated by the BB technique. It is well-known that the BB method only has a linear convergence rate for minimizing strongly convex problems, which is the same as the steepest descent (SD) method. However, this does not limit the practical much superior performance of the BB method over the SD method. In our experiments, our methods are directly compared with the primal-dual methods using other dual ascent step sizes, such as NT and ESOM.

Note that existing decentralized second-order and quasi-Newton methods exhibit different performance across several aspects, including primal-dual updates, convergence rates, computational costs, etc.. Here we compare our method with other major decentralized second-order and quasi-Newton methods. The comparisons are summarized in Table I.

IV. NUMERICAL EXPERIMENTS

In this section, we would like to test and compare our developed algorithms with some well-developed first-order and second-order algorithms on solving the linear regression problems and the logistic regression problems over a connected undirected network with edge density $d \in (0,1]$. An additional experiment about the effect of network topology is presented in Section XII of the supplementary material. For the generated network, we choose the Metropolis constant edge weight matrix [36] as the mixing matrix, that is

$$\tilde{W}_{ij} = \begin{cases} \frac{1}{\max\{\deg(i),\deg(j)\}+1}, & \text{if } (i,j) \in \mathcal{E}; \\ 0, & \text{if } (i,j) \notin \mathcal{E} \text{ and } i \neq j; \\ 1 - \sum_{k \in \mathcal{N}_i \setminus \{i\}} \tilde{W}_{ik}, & \text{if } i = j, \end{cases}$$

where $(i,j) \in \mathcal{E}$ indicates there is an edge between node i and node j, and $\deg(i)$ means the degree of node i.

Based on our analysis, we also propose an algorithm GDPDM$^+$, which applies an adaptive stop criteria for the inner loop as stated in **Algorithm 3**, while the remaining steps are the same as those of GDPDM. In fact, it is not difficult to see the global convergence of GDPDM$^+$ can be also obtained within the convergence analysis framework of GDPDM. Our comparison algorithms in this section include GT [15], EXTRA [10], ESOM-1 [27] (called ESOM below), NT [29], DBFGS [26], Damped regularized limited-memory DFP [33] (called DR-LM-DFP below), Damped limited-memory BFGS [33] (called D-LM-BFGS below), and DQN [34].

Algorithm 3 Inner loop of GDPDM$^+$ with respect to node i

$\mathbf{x}_i^{t,0} = \mathbf{x}_i^t$, $\mathbf{H}_i^{t,0} = \mathbf{H}_i^t$, block$(i) = 1$.
For $s = 0, \ldots, S-1$
  If block$(i) == 1$
    Update $\mathbf{x}_i^{t,s+1}$ by (36).
    Update $\mathbf{H}_i^{t,s+1}$ by (24).
  Else
    $\mathbf{x}_i^{t,s+1} = \mathbf{x}_i^{t,s}$, and $\mathbf{H}_i^{t,s+1} = \mathbf{H}_i^{t,s}$.
  End if.
  If $\|\mathbf{x}_i^{t,s+1} - \mathbf{x}_i^t\| \leq c\|\mathbf{v}_i^t - \mathbf{v}_i^{t-1}\| (0 < c < 1)$
    block$(i) = 0$.
  End if.
End for.

We use GDPDM(K) and GDPDM(K)$^+$to denote GDPDM and GDPDM$^+$ with $S = K$, respectively. In addition, we update r^t in (33) as $r^t = c_r(\eta_r)^t$, where $c_r > 0$ and $0 < \eta_r < 1$, and set $\overline{\omega} >> \underline{\omega} > 0$ to be a large and a small positive constant, respectively. Recall that $\tilde{p}_i^t$ is an approximation of p^t, which can be rewritten as

$$p^t = \left(\frac{\|\boldsymbol{\lambda}^t - \boldsymbol{\lambda}^{t-1}\|^2_{\hat{\mathbf{B}}^{t+1}}}{\|\boldsymbol{\lambda}^t - \boldsymbol{\lambda}^{t-1}\|^2} + r^t\right)^{-1},$$

where $\hat{\mathbf{B}}^{t+1} = (\mathbf{I} - \mathbf{W})^{1/2}(\mathbf{B}^{t+1})^{-1}(\mathbf{I} - \mathbf{W})^{1/2}$. By **Assumptions 3**, we have $\frac{1}{\rho\bar{\psi}+r^t} \leq p^t \leq \frac{1}{\sigma\psi+r^t}$. If $\tilde{p}_i^t$ tends to p^t, we would have $\tilde{p}_i^t \in [\frac{1}{\rho\bar{\psi}+r^t}, \frac{1}{\sigma\psi+r^t}] \subset [\frac{1}{\overline{\omega}+r^t}, \frac{1}{\underline{\omega}+r^t}]$. The success of each algorithm is measured by

$$\text{Relative error} := \frac{1}{n}\sum_{i=1}^n \frac{\|\mathbf{x}_i^t - \mathbf{z}^*\|}{\|\mathbf{z}^*\| + 1},$$

where the true solution $\mathbf{z}^*$ is explicitly obtained for the linear regression problem and is pre-computed by a centralized algorithm for the logistic regression problem. In our experiments, we would analyze the impact of networks with different condition numbers. So we further introduce the communication

Table I
COMPARISONS OF DECENTRALIZED SECOND-ORDER AND QUASI-NEWTON METHODS

	Primal order	Dual order	Convergence rate	Iteration complexity	Assumption[3]	Calculate Hessian or not	Communication overhead[4]
NT[5][29]	Second	First	Linear	$O(\max\{\kappa_f\sqrt{\kappa_g}+\kappa_f^2, \kappa_g^{3/2}/\kappa_f+\kappa_f\sqrt{\kappa_g}\}log(\frac{1}{\epsilon}))$	SC+LG+LH	Yes	p
ESOM-K [27]	Second	First	Linear	$O\left(\kappa_f^2/\sigma log\left(\frac{1}{\epsilon}\right)\right)$	SC+LG+LH	Yes	$(1+K)p$
DBFGS [26]	Quasi second[1]	\	Inexactly linear[2]	\	SC+LG	No	$3p$
DQN [34]	Quasi second	\	asymptotic	\	C+LG	No	$3p$
DR-LM-DFP [33]	Quasi second	\	Linear	$O\left(\kappa_f^2\kappa_g^2 log\left(\frac{1}{\epsilon}\right)\right)$	SC+LG	No	$2p$
D-LM-BFGS [33]	Quasi second	\	Linear	$O\left(\kappa_f^2\kappa_g^2 log\left(\frac{1}{\epsilon}\right)\right)$	SC+LG	No	$2p$
PD-QN-K [28]	Quasi second	Quasi second	\	\	SC+LG	No	$(4+K)p$
DPDM	Quasi second	Quasi second	Linear	$O\left(\kappa_l^2\kappa_g log\left(\frac{1}{\epsilon}\right)\right)$	SC+LG	No	$(2+I_{R\setminus\{0\}}(\theta))p+2^\star$

[1] "Quasi second" means the second-order information is captured by Hessian approximations using gradient information.
[2] "Inexactly linear" means that the method only converges linearly to a small neighborhood of the solution.
[3] "C", "SC", "LG", "LH" respectively mean "Convexity", "Strong convexity", "Lipschitz continuous gradient", "Lipschitz continuous Hessian".
[4] For the fixed underlying network, communication overhead can be defined as rounds of communication per iteration $\times$ dimensions of transmitted scalars, vectors, or matrices.
[5] The iteration complexity of NT is computed when $\mathbf{x}^t$ is close to $\mathbf{x}^*$. So the global iteration complexity is larger than the value given above.
* Here, the addition of 2 is because of the additional 2 scalar communications of a_j^t and b_j^t in each iteration of DPDM.

volume which can be calculated as follows:

$$\begin{aligned}&\text{Communication volume}\\&=\text{number of iterations}\\&\quad\times\text{number of communication rounds per iteration}\\&\quad\times\text{number of edges, i.e., } dn(n-1)/2\\&\quad\times\text{dimension of transmitted vectors on each edge.}\end{aligned}$$

In all experiments, we set the number of nodes $n=10$. For all comparison algorithms, we initialize $\mathbf{x}^0=\mathbf{0}$ and set $\mathbf{v}^0=\mathbf{0}$ for our algorithms. All experiments are coded in MATLAB R2017b and run on a laptop with Intel Core i5-9300H CPU, 16GB RAM, and Windows 10 operating system.

A. Linear Regression Problem

In this subsection, we investigate the impacts of the condition number of the objective function, denoted as κ_f, by comparing our algorithms with first-order algorithms, EXTRA [10] and GT [15]. We consider the following optimization problem

$$\min_{\mathbf{z}\in\mathbb{R}^p}\sum_{i=1}^{n}\frac{1}{2}\mathbf{z}^\mathsf{T}\mathbf{A}_i\mathbf{z}+\mathbf{b}_i^\mathsf{T}\mathbf{z}, \tag{58}$$

where $\mathbf{A}_i\in\mathbb{R}^{p\times p}$ and $\mathbf{b}_i\in\mathbb{R}^p$ are private data available to node i. To control the condition number of problem (58), we construct $\mathbf{A}_i=\mathbf{Q}^\mathsf{T}\,\mathrm{diag}\{a_1,...,a_p\}\mathbf{Q}$, where $\mathbf{Q}$ is a random orthogonal matrix. We set $a_1=1$ and a_p as an arbitrarily large number, and generate $a_j\sim\mathrm{U}(1,2)$ for $j=2,\ldots,p-1$, where $\mathrm{U}(1,2)$ represents the uniform distribution from 1 to 2. So $\kappa_f=a_p/a_1=a_p$.

We set $p=1000$ and edge density $d=0.36$ for the network, where $\kappa_g=9.8$. We set a_p = 10, 10^2, 10^3, 10^4. All the algorithm parameters are set for their better performance and are listed in Table II where parameter notations follow the source papers.

From figure 1, We find that for such simple quadratic problems, GDPDM(1), namely DPDM is most efficient since the outer iteration number has no significant change but the communication cost increases as the number of inner iterations increases. In this case, one BFGS iteration is sufficient to minimize the quadratic primal problem.

Table II
PARAMETER SETTINGS FOR LINEAR REGRESSION

	GT	EXTRA	GDPDM(1) (GDPDM(2); GDPDM(3); GDPDM(4); GDPDM(4)$^+$)
$\kappa_f=10$	\	\	$\beta=0.49(0.31;0.22;0.17;0.57)$, $\theta=0.32$, $\alpha=2.8$, $\gamma=1$, $r^t=0.95^t$, $c=0.4$
$\kappa_f=10^2$	$\eta=5\times10^{-3}$	$\alpha=10^{-2}$	
$\kappa_f=10^3$	$\eta=5\times10^{-4}$	$\alpha=10^{-3}$	
$\kappa_f=10^4$	$\eta=5\times10^{-5}$	$\alpha=10^{-4}$	

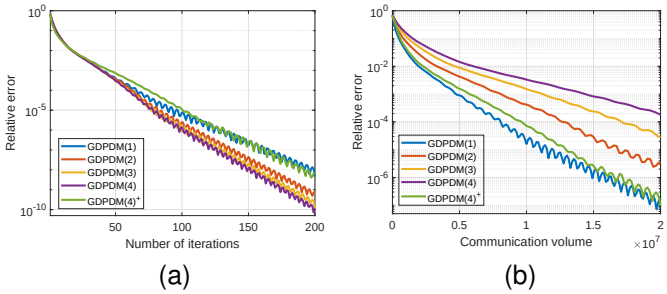

Figure 1. Comparisons among GDPDMs in terms of the iteration number and communication volume for $\kappa_f=10$.

Figure 2 shows that our algorithm is more robust to the problem condition number than EXTRA and GT. The convergence rate of first-order algorithms becomes obviously slow when the condition number increases.

B. Logistic Regression Problem

In this subsection, we firstly compare our algorithms with GT [15] and two second-order algorithms: ESOM [27] and NT [29], which use Hessian in the primal domain with dual ascent in the dual domain. We consider the logistic regression

$$\min_{\mathbf{z}\in\mathbb{R}^p}\sum_{i=1}^{n}\sum_{j=1}^{n_i}\log\left(1+\exp(-b_{ij}\mathbf{a}_{ij}^\mathsf{T}\mathbf{z})\right)+\frac{\hat{\lambda}}{2}\|\mathbf{z}\|^2, \tag{59}$$

where $\mathbf{a}_{ij}\in\mathbb{R}^p$ are the feature vectors and $b_{ij}\in\{-1,+1\}$ are the labels. The experiments are conducted on four datasets

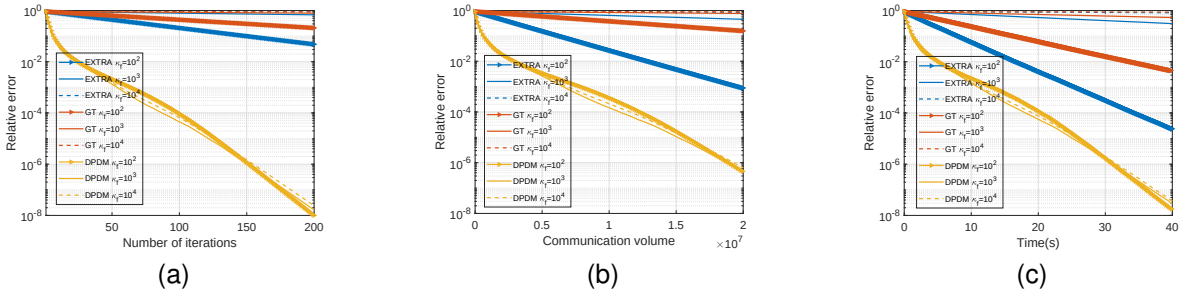

Figure 2. Comparisons with decentralized first-order algorithms in terms of the iteration number, communication volume, and CPU time (in seconds).

from the LIBSVM library: **mushroom**, **ijcnn1**, **w8a**, and **a9a**. The edge density $d = 0.36$ and the regularization parameter $\hat{\lambda} = 1$. All the algorithm parameters are set for their better performance and are listed in Table III where parameter notations follow the source papers.

Figures 3, 4, 5, and 6 show that our algorithms converge significantly faster than GT, ESOM, and NT in terms of both the iteration number and CPU time. As more primal updates are allowed in each inner iteration of GDPDMs, the (outer) iteration number is reduced while the CPU time is often increased. However, it can be seen that GDPDM(4)$^+$, which adaptively controls the number of inner iterations for solving the more complex and nonlinear logistic regression problem, performs generally best and keeps a good balance between the iteration number and CPU time. We can see that the performance of GDPDM(4)$^+$ is close to GDPDM(4) in terms of the number of iterations and has about the same low communication and time cost as GDPDM(1).

On the other hand, since multiple primal updates in each iteration would lead to more communication cost, GDPDMs do not outperform the second-order method NT in terms of communication volume. For saving communication in NT, the topology-dependent term $\alpha(\mathbf{I}-\mathbf{W})$ is removed when involving matrix inverse calculation as shown in (20). This renders NT needing a big regularization parameter ϵ to compensate for the loss of network topology information. And ϵ is required to be bounded below for convergence of NT, which could also affect the fast local convergence. Most GDPDMs are more communication-efficient than the second-order method ESOM, while all GDPDMs are superior to the first-order method GT. Moreover, ESOM and NT obviously perform significantly worse in terms of CPU time, since exact Hessian and matrix inversions are calculated in their iterations. Figures 3d, 4d, 5d and 6d show the needed time and communication volume for reaching a 10^{-10}-accuracy solution. We think it may not be an ideal way to trade much time efficiency for a slight improvement in communication. Our GDPDMs, such as GDPDM(1), GDPDM(4)$^+$ are located near the bottom left corner, implying they are quite efficient in both communication cost and CPU time.

We now compare our algorithms with several well-developed decentralized quasi-Newton algorithms: DBFGS [26], DR-LM-DFP [33], D-LM-BFGS [33], and DQN [34]. All the algorithm parameters are set for their better performance and are listed in Table IV where parameter notations follow the source papers.

From figures 7 and 8, we see that our algorithm is more efficient than other algorithms. The reason why DR-LM-DFP and D-LM-BFGS are very slow is that their quasi-Newton matrices are constructed using some significant regularization or damping techniques which could badly affect the approximation to the Hessian for capturing the second-order information. DQN has a similar recursion to DR-LM-DFP, yet it introduces an additional communication step updating the search direction and allows for uncoordinated (different) step sizes among the nodes. While it accelerates convergence in terms of iteration count, it also increases the communication burden. Furthermore, DBFGS is an inexact method which only converges to a small neighborhood of the solution.

To sum up, numerical experiments show our new GDPDMs perform better than currently well-developed decentralized methods due to the applications of quasi-Newton techniques to reduce the computational cost and the explorations of second-order information in both primal and dual updates to accelerate the convergence.

V. CONCLUSIONS

This paper considers a decentralized consensus optimization problem and proposes a decentralized primal-dual method (DPDM) and its generalization (GDPDM) with multiple primal updates per iteration. Both primal and dual updates effectively explore second-order information, where Hessian approximations are constructed with at most matrix-vector products. The single Jacobi loop with block-wise BFGS approximation is conducted in the primal domain. Based on a new approximate dual gradient variation, the dual ascent step with a novel second-order correction is implemented in the dual domain. We show the updating direction of each node asymptotically tends towards the global quasi-Newton direction. The relationship between GDPDM and some first or second-order methods is also established. Under proper assumptions, GDPDMs have globally linear convergence. The numerical results indicate that GDPDMs are not only very robust on the condition number of the problem, but also perform significantly better in terms of iteration number and CPU time than all the comparison decentralized methods, including EXTRA, GT, ESOM, NT, DBFGS, DR-LM-DFP, D-LM-BFGS, and DQN.

This article has been accepted for publication in IEEE Transactions on Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSP.2025.3547787

Table III
PARAMETER SETTINGS FOR LOGISTIC REGRESSION WITH DECENTRALIZED FIRST-ORDER AND SECOND-ORDER ALGORITHMS AND OUR GDPDMS

	GT	ESOM	NT	GDPDM(1) (GDPDM(2); GDPDM(4); GDPDM(4)$^+$)
mushroom # of samples ($\sum_{i=1}^n n_i = 8120$) # of features ($p = 112$)	$\eta = 0.08$	$\epsilon = 0.3$, $\alpha = 3.9$	$\epsilon = 3.1$, $\alpha = 3.9$	$\beta = 0.48(0.3; 0.17; 0.51)$, $\theta = 0.18$, $\gamma = 1$, $\alpha = 3.6$, $r^t = 0.95^t$, $c = 0.6$
ijcnn1 # of samples ($\sum_{i=1}^n n_i = 49990$) # of features ($p = 22$)	$\eta = 0.06$	$\epsilon = 0.1$, $\alpha = 3.9$	$\epsilon = 3.5$, $\alpha = 3.9$	$\beta = 0.41(0.33; 0.22; 0.44)$, $\theta = 0.15$, $\gamma = 1$, $\alpha = 4$, $r^t = 0.95^t$, $c = 0.3$
w8a # of samples ($\sum_{i=1}^n n_i = 49740$) # of features ($p = 300$)	$\eta = 0.06$	$\epsilon = 0.2$, $\alpha = 4.1$	$\epsilon = 3.4$, $\alpha = 4.0$	$\beta = 0.47(0.38; 0.22; 0.45)$, $\gamma = 1$, $\alpha = 3.6$, $\theta = 0.17(0.17; 0.17; 0.16)$, $r^t = 0.95^t$, $c = 0.6$
a9a # of samples ($\sum_{i=1}^n n_i = 32560$) # of features ($p = 123$)	$\eta = 0.06$	$\epsilon = 0.1$, $\alpha = 3.9$	$\epsilon = 3.3$, $\alpha = 3.9$	$\beta = 0.38(0.34; 0.19; 0.42)$, $\theta = 0.15$, $\gamma = 1$, $\alpha = 4.0$, $r^t = 0.95^t$, $c = 0.45$

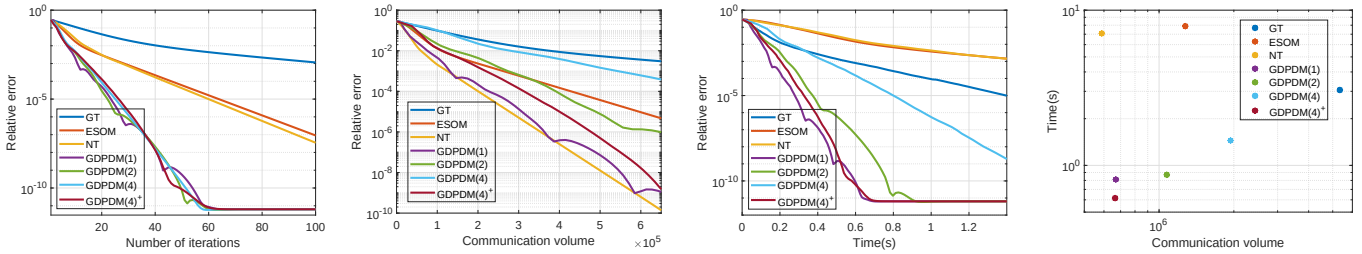

Figure 3. (a-c) Comparisons with decentralized second-order algorithms in terms of the iteration number, communication volume, and CPU time (in seconds) using **mushroom** dataset. (d) Balance between time and communication volume.

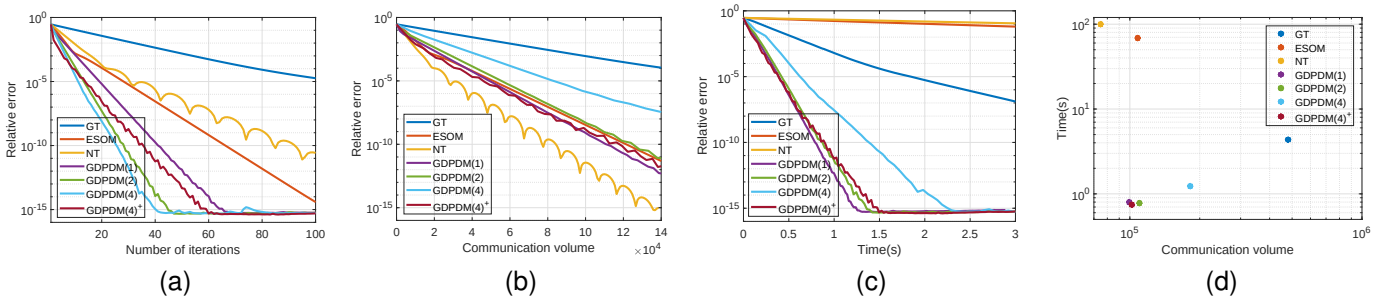

Figure 4. (a-c) Comparisons with decentralized second-order algorithms in terms of the iteration number, communication volume, and CPU time (in seconds) using **ijcnn1** dataset. (d) Balance between time and communication volume.

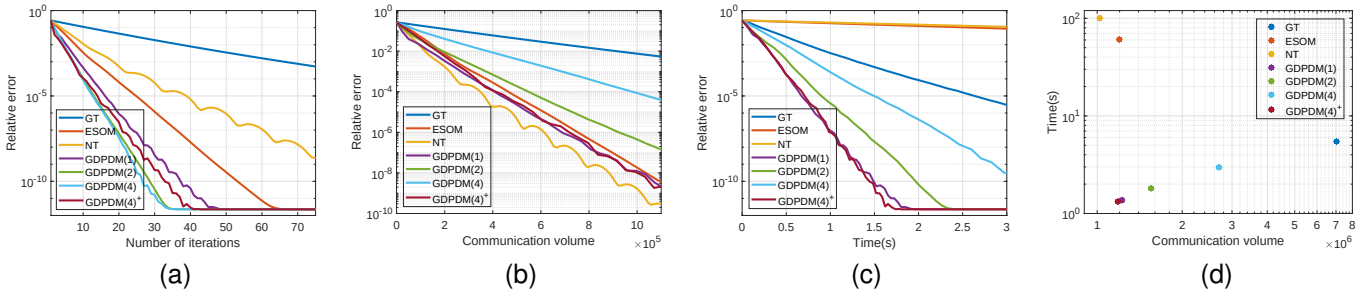

Figure 5. (a-c) Comparisons with decentralized second-order algorithms in terms of the iteration number, communication volume, and CPU time (in seconds) using **w8a** dataset. (d) Balance between time and communication volume.

REFERENCES

[1] G. Fusco and M. Russo, "A decentralized approach for voltage control by multiple distributed energy resources," *IEEE Transactions on Smart Grid*, vol. 12, no. 4, pp. 3115–3127, 2021.

This article has been accepted for publication in IEEE Transactions on Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSP.2025.3547787

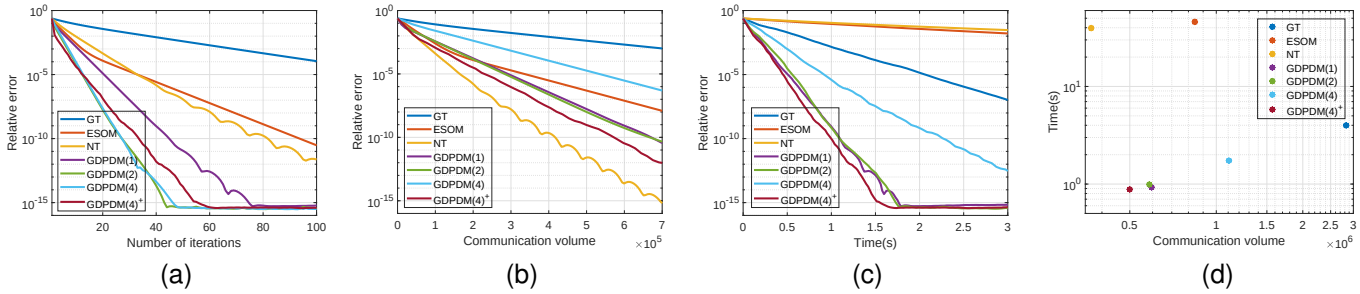

Figure 6. (a-c) Comparisons with decentralized second-order algorithms in terms of the iteration number, communication volume, and CPU time (in seconds) using **a9a** dataset. (d) Balance between time and communication volume.

Table IV
PARAMETER SETTINGS FOR LOGISTIC REGRESSION WITH DECENTRALIZED QUASI-NEWTON ALGORITHMS

	DBFGS	DR-LM-DFP	D-LM-BFGS	DQN
ijcnn1	$\alpha = 0.01$, $\epsilon = 0.02$, $\gamma = 0.1$, $\Gamma = 0.1$	$\alpha = 0.07$, $\rho = 0.04$, $\epsilon = 10^{-3}$, $\beta = 1$, $\mathcal{B} = 10^4$, $\tilde{L} = 1$, $M = 10$	$\alpha = 0.48$, $\epsilon = 10^{-3}$, $\beta = 10^{-3}$, $\mathcal{B} = 10^4$, $\tilde{L} = 20$, $M = 8$	$\alpha_i^t = 0.11\zeta_i^{t*}$, $\gamma = 1$
a9a	$\alpha = 0.01$, $\epsilon = 0.03$, $\gamma = 0.25$, $\Gamma = 0.05$	$\alpha = 0.07$, $\rho = 0.03$, $\epsilon = 5 \times 10^{-3}$, $\beta = 10^{-2}$, $\mathcal{B} = 10^4$, $\tilde{L} = 1$, $M = 8$	$\alpha = 0.45$, $\epsilon = 10^{-3}$, $\beta = 10^{-3}$, $\mathcal{B} = 10^4$, $\tilde{L} = 10$, $M = 3$	$\alpha_i^t = 0.11\zeta_i^t$, $\gamma = 3$

* ζ_i^t is the random variable generated over node i at iteration t and satisfies the uniform distribution over interval $(0.5, 1.5)$.

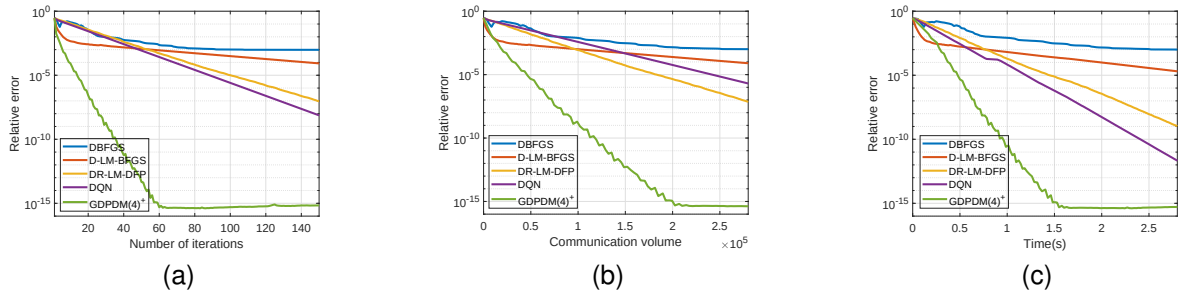

Figure 7. Comparisons with decentralized quasi-Newton algorithms in terms of the iteration number, communication volume, and CPU time (in seconds) using **ijcnn1** dataset.

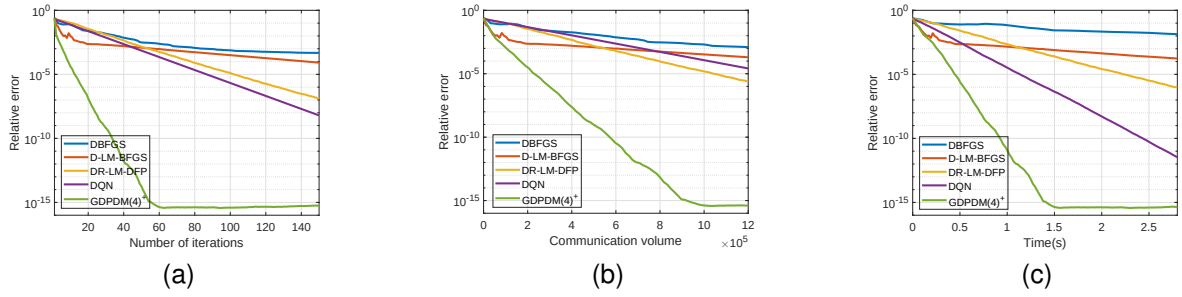

Figure 8. Comparisons with decentralized quasi-Newton algorithms in terms of the iteration number, communication volume, and CPU time (in seconds) using **a9a** dataset.

[2] E. Jeong, M. Zecchin, and M. Kountouris, "Asynchronous decentralized learning over unreliable wireless networks," in *2022 International Conference on Communications (ICC)*, 2022, pp. 607–612.

[3] X. Zhang, C. Hu, B. He, and Z. Han, "Distributed reptile algorithm for meta-learning over multi-agent systems," *IEEE Transactions on Signal Processing*, vol. 70, pp. 5443–5456, 2022.

[4] Z. Chen, Z. Li, C. Guo, J. Wang, and Y. Ding, "Fully distributed robust reserve scheduling for coupled transmission and distribution systems," *IEEE Transactions on Power Systems*, vol. 36, no. 1, pp. 169–182, 2020.

[5] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1142–1154, 2022.

[6] A. Nedić and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[7] K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," *SIAM Journal on Optimization*, vol. 26, no. 3, pp. 1835–1854, 2016.

[8] J. Zeng and W. Yin, "On nonconvex decentralized gradient descent," *IEEE Transactions on Signal Processing*, vol. 66, no. 11, pp. 2834–

2848, 2018.

[9] A. S. Berahas, R. Bollapragada, N. S. Keskar, and E. Wei, "Balancing communication and computation in distributed optimization," *IEEE Transactions on Automatic Control*, vol. 64, no. 8, pp. 3141–3155, 2018.

[10] W. Shi, Q. Ling, G. Wu, and W. Yin, "Extra: An exact first-order algorithm for decentralized consensus optimization," *SIAM Journal on Optimization*, vol. 25, no. 2, pp. 944–966, 2015.

[11] H. Li and Z. Lin, "Revisiting extra for smooth distributed optimization," *SIAM Journal on Optimization*, vol. 30, no. 3, pp. 1795–1821, 2020.

[12] Z. Li, W. Shi, and M. Yan, "A decentralized proximal-gradient method with network independent step-sizes and separated convergence rates," *IEEE Transactions on Signal Processing*, vol. 67, no. 17, pp. 4494–4506, 2019.

[13] K. Yuan, B. Ying, X. Zhao, and A. H. Sayed, "Exact diffusion for distributed optimization and learning—part i: Algorithm development," *IEEE Transactions on Signal Processing*, vol. 67, no. 3, pp. 708–723, 2018.

[14] J. Xu, S. Zhu, Y. C. Soh, and L. Xie, "Augmented distributed gradient methods for multi-agent optimization under uncoordinated constant stepsizes," in *2015 IEEE Conference on Decision and Control (CDC)*, 2015, pp. 2055–2060.

[15] G. Qu and N. Li, "Harnessing smoothness to accelerate distributed optimization," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 3, pp. 1245–1260, 2017.

[16] A. Nedić, A. Olshevsky, and W. Shi, "Achieving geometric convergence for distributed optimization over time-varying graphs," *SIAM Journal on Optimization*, vol. 27, no. 4, pp. 2597–2633, 2017.

[17] J. Zhang, K. You, and K. Cai, "Distributed dual gradient tracking for resource allocation in unbalanced networks," *IEEE Transactions on Signal Processing*, vol. 68, pp. 2186–2198, 2020.

[18] Z. Song, L. Shi, S. Pu, and M. Yan, "Optimal gradient tracking for decentralized optimization," *Mathematical Programming*, pp. 1–53, 2023.

[19] D. Jakovetić, "A unification and generalization of exact distributed first-order methods," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 5, no. 1, pp. 31–46, 2018.

[20] F. Mansoori and E. Wei, "Flexpd: A flexible framework of first-order primal-dual algorithms for distributed optimization," *IEEE Transactions on Signal Processing*, vol. 69, pp. 3500–3512, 2021.

[21] S. Zhu, M. Hong, and B. Chen, "Quantized consensus admm for multi-agent distributed optimization," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4134–4138.

[22] C. Zhang, M. Ahmad, and Y. Wang, "Admm based privacy-preserving decentralized optimization," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 565–580, 2018.

[23] G. Mancino-Ball, Y. Xu, and J. Chen, "A decentralized primal-dual framework for non-convex smooth consensus optimization," *IEEE Transactions on Signal Processing*, vol. 71, pp. 525–538, 2023.

[24] A. Mokhtari, Q. Ling, and A. Ribeiro, "Network newton distributed optimization methods," *IEEE Transactions on Signal Processing*, vol. 65, no. 1, pp. 146–161, 2016.

[25] D. Bajovic, D. Jakovetic, N. Krejic, and N. K. Jerinkic, "Newton-like method with diagonal correction for distributed optimization," *SIAM Journal on Optimization*, vol. 27, no. 2, pp. 1171–1203, 2017.

[26] M. Eisen, A. Mokhtari, and A. Ribeiro, "Decentralized quasi-newton methods," *IEEE Transactions on Signal Processing*, vol. 65, no. 10, pp. 2613–2628, 2017.

[27] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro, "A decentralized second-order method with exact linear convergence rate for consensus optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 4, pp. 507–522, 2016.

[28] M. Eisen, A. Mokhtari, and A. Ribeiro, "A primal-dual quasi-newton method for exact consensus optimization," *IEEE Transactions on Signal Processing*, vol. 67, no. 23, pp. 5983–5997, 2019.

[29] J. Zhang, Q. Ling, and A. M.-C. So, "A newton tracking algorithm with exact linear convergence for decentralized consensus optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 7, pp. 346–358, 2021.

[30] Y. Li, P. G. Voulgaris, and N. M. Freris, "A communication efficient quasi-newton method for large-scale distributed multi-agent optimization," in *2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 4268–4272.

[31] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro, "Dqm: Decentralized quadratically approximated alternating direction method of multipliers," *IEEE Transactions on Signal Processing*, vol. 64, no. 19, pp. 5158–5173, 2016.

[32] Z. Zhang, S. Yang, and W. Xu, "Decentralized admm with compressed and event-triggered communication," *Neural Networks*, 2023.

[33] J. Zhang, H. Liu, A. M.-C. So, and Q. Ling, "Variance-reduced stochastic quasi-newton methods for decentralized learning," *IEEE Transactions on Signal Processing*, vol. 71, pp. 311–326, 2023.

[34] O. Shorinwa and M. Schwager, "Distributed quasi-newton method for multi-agent optimization," *IEEE Transactions on Signal Processing*, vol. 72, pp. 3535–3546, 2024.

[35] A. H. Sayed, "Diffusion adaptation over networks," in *Academic Press Library in Signal Processing*, 2014, vol. 3, pp. 323–453.

[36] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *Journal of parallel and distributed computing*, vol. 67, no. 1, pp. 33–46, 2007.

[37] S. U. Pillai, T. Suel, and S. Cha, "The perron-frobenius theorem: some of its applications," *IEEE Signal Processing Magazine*, vol. 22, no. 2, pp. 62–75, 2005.

[38] D. M. Young, *Iterative solution of large linear systems*. Elsevier, 2014.

[39] Y. Yuan, "Analysis on a superlinearly convergent augmented lagrangian method," *Acta Mathematica Sinica, English Series*, vol. 30, no. 1, pp. 1–10, 2014.

[40] J. Barzilai and J. M. Borwein, "Two-point step size gradient methods," *IMA journal of numerical analysis*, vol. 8, no. 1, pp. 141–148, 1988.

[41] M. Zhu and S. Martínez, "Discrete-time dynamic average consensus," *Automatica*, vol. 46, no. 2, pp. 322–329, 2010.