Abstract. One fundamental problem in decentralized multi-agent optimization is the trade-off between gradient/sampling complexity and communication complexity. In this paper we propose new algorithms whose gradient and sampling complexities are graph topology invariant, while their communication complexities remain optimal. Specifically, for convex smooth deterministic problems, we propose a primal dual sliding (PDS) algorithm that is able to compute an $\varepsilon$-solution with $O((L/\varepsilon)^{1/2})$ gradient complexity and $O((L/\varepsilon)^{1/2} + \|A\|/\varepsilon)$ communication complexity, where $L$ is the smoothness parameter of the objective function and $A$ is related to either the graph Laplacian or the transpose of the oriented incidence matrix of the communication network. The complexities can be further improved to $O((L/\mu)^{1/2} \log(1/\varepsilon))$ and $O((L/\mu)^{1/2} \log(1/\varepsilon) + \|A\|/\varepsilon^{1/2})$ respectively with the additional assumption of strong convexity modulus $\mu$. We also propose a stochastic variant, namely the primal dual sliding (SPDS) algorithm for convex smooth problems with stochastic gradients. The SPDS algorithm utilizes the mini-batch technique and enables the agents to perform sampling and communication simultaneously. It computes a stochastic $\varepsilon$-solution with $O((L/\varepsilon)^{1/2} + (\sigma/\varepsilon)^2)$ sampling complexity, which can be further improved to $O((L/\mu)^{1/2} \log(1/\varepsilon) + \sigma^2/\varepsilon)$ in the strong convexity case. Here $\sigma^2$ is the variance of the stochastic gradient. The communication complexities of SPDS remain the same as that of the deterministic case. All the aforementioned gradient and sampling complexities match the lower complexity bounds for centralized convex smooth optimization and are independent of the network structure. To the best of our knowledge, these gradient and sampling complexities have not been obtained before in the literature of decentralized optimization.

Key words. Multi-agent optimization, decentralized optimization, saddle point problems, gradient complexity, sampling complexity, communication complexity, gradient sliding

AMS subject classifications. 90C25, 90C06, 49M37, 93A14, 90C15

1. Introduction. The problem of interest in this paper is the following decentralized multi-agent optimization problem in which the agents will collaboratively minimize an overall objective as the sum of all local objective $f_i$'s:

$$\min_{x \in X} \sum_{i=1}^{m} f_i(x).$$

Here each $f_i : X^{(i)} \to \mathbb{R}$ is a convex smooth function defined over a closed convex set $X^{(i)} \subseteq \mathbb{R}^d$ and $X := \cap_{i=1}^{m} X^{(i)}$. Under decentralized settings, each agent is expected to collect data, perform numerical operations using local data, and pass information to the neighboring agents in a communication network; no agent has full knowledge about other agents’ local objectives or the communication network. This type of decentralized problems has many applications in signal processing, control, robust statistical inference, machine learning among others (see, e.g., [6, 11, 25, 26, 21, 7]).

In this paper, we assume that the communication network between the agents are defined by a connected undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N} = \{1, \ldots, m\}$.
is the set of indices of agents and $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ is the set of communication edges between them. Each agent $i \in \mathcal{N}$ could directly communicate with the agents in its neighborhood $N_i := \{ j \in \mathcal{N} \mid (i, j) \in \mathcal{E} \} \cup \{ i \}$. For convenience we assume that there exists a loop $(i, i)$ for all agents $i \in \mathcal{N}$. In addition, we assume that the local objectives are all large-dimensional functions and the $i$-th agent may only be able to access the information of its local objective $f_i(x)$ through a sampling process of its first-order information, e.g., the agent has objective $f_i(x) = \mathbb{E}_{\xi_i}[F_i(x, \xi_i)]$, where the expectation is taken with respect to the random variable $\xi_i$ and the distribution of $\xi_i$ is not known in advance. In order to solve problem (1.1) collaboratively, the agents have to sample the first-order information of their own local objective $f_i$ and also communicate with the neighbors in the communication network in order to reach consensus. Our goal in this paper is to design an algorithm that is efficient, in terms of both the sampling and communication complexity, on solving the multi-agent decentralized problem (1.1).

1.1. Problem formulation and assumptions. The aforementioned multi-agent problem can also be formulated as the following linearly constrained problem:

\begin{equation}
\min_{x \in \mathcal{X}} f(x) := \sum_{i=1}^{m} f_i(x^{(i)}) \quad \text{s.t.} \quad Ax = 0,
\end{equation}

where the overall objective function $f : \mathcal{X} \to \mathbb{R}$ is defined on the Cartesian product $\mathcal{X} := \mathcal{X}^{(1)} \times \cdots \times \mathcal{X}^{(m)} \subseteq \mathbb{R}^{md}$ and we use the notation $x = (x^{(1)}, \ldots, x^{(m)}) \in \mathcal{X}$ to summarize a collection of local solutions $x^{(i)} \in \mathcal{X}^{(i)}$. The matrix $A$ in the linear constraint $Ax = 0$ is designed to enforce the conditions $x^{(i)} = x^{(j)}$ for all agents $i$ and $j$ that are connected by a communication edge. One possible choice of $A$ is $A = \mathcal{L} \otimes I_d \in \mathbb{R}^{md \times md}$ where $I_d \in \mathbb{R}^{d \times d}$ is an identity matrix and $\mathcal{L} \in \mathbb{R}^{m \times m}$ is the graph Laplacian matrix whose $(i, j)$-entry is $|N_i| - 1$ if $i = j$, $-1$ if $i \neq j$ and $(i, j) \in \mathcal{E}$, and 0 otherwise. Here $|N_i|$ is the degree of node $i$. One other choice of $A$ is $A = \mathcal{B}^T \otimes I_d \in \mathbb{R}^{|\mathcal{E}| \times md}$ where $\mathcal{B} \in \mathbb{R}^{m \times |\mathcal{E}|}$ is an oriented incidence matrix of graph $\mathcal{G}$. Here $|\mathcal{E}|$ is the number of edges in $\mathcal{G}$. For both cases, it can be shown that $Ax = 0$ if and only if $x^{(i)} = x^{(j)}$ for all agents $i$ and $j$ that are connected via a communication edge. Note that problem (1.2) is equivalent to the saddle point problem

\begin{equation}
\min_{x \in \mathcal{X}} \max_{z \in \mathbb{R}^{md}} f(x) + \langle Ax, z \rangle.
\end{equation}

We assume that there exists a saddle point $(x^*, z^*)$ for problem (1.3).

In this paper, our goal is to solve the multi-agent problem (1.1) by obtaining an $\varepsilon$-approximate solution $\bar{x}$ to the linearly constrained problem (1.2) such that $f(\bar{x}) - f(x^*) \leq \varepsilon$ and $\|A\bar{x}\|_2 \leq \varepsilon$. We make the following assumptions on the information and sampling process of the local objective functions. For each $i$, we assume that

\begin{equation}
f_i(x^{(i)}) := \tilde{f}_i(x^{(i)}) + \mu \nu_i(x^{(i)})
\end{equation}

where $\mu \geq 0$, $\tilde{f}_i$ is a convex smooth function, and $\nu_i$ is a strongly convex function with strong convexity modulus 1 with respect to a norm $\| \cdot \|$. Here the gradient $\nabla \tilde{f}_i$ is Lipschitz continuous with constant $\tilde{L}$ with respect to the norm $\| \cdot \|$. The $i$-th agent can access first-order information of $\tilde{f}_i$ through a stochastic oracle which returns an unbiased gradient estimator $G_i(x^{(i)}, \xi^{(i)})$ for any inquiry point $x^{(i)} \in \mathcal{X}^{(i)}$, where $\xi^{(i)}$ is a sample of an underlying random vector with unknown distribution. The unbiased gradient estimator satisfies $\mathbb{E}[G_i(x^{(i)}, \xi^{(i)})] = \nabla f_i(x^{(i)})$ and

\begin{equation}
\mathbb{E}[\|G_i(x^{(i)}, \xi^{(i)}) - \nabla f_i(x^{(i)})\|_2^2] \leq \sigma^2, \quad \forall x^{(i)} \in \mathcal{X}^{(i)},
\end{equation}

This manuscript is for review purposes only.
where \( \| \cdot \|_2 \) is the Euclidean norm. Note that the above assumption also covers the deterministic case, i.e., when \( \sigma = 0 \). For simplicity, in this paper we assume that the strong convexity constant \( \mu \), the Lipschitz constant \( \bar{L} \), and the variance \( \sigma^2 \) are the same among all local objective functions. With the introduction of \( \tilde{f}_i \)'s in (1.4), we can study the cases of both strongly convex and general convex objective functions for the multi-agent optimization problem (1.1). For convenience, we introduce the following notation for describing the overall objective function:

\[
(1.6) \quad f(x) := \tilde{f}(x) + \mu \nu(x) \quad \text{where} \quad \tilde{f}(x) := \sum_{i=1}^{m} \tilde{f}_i(x^{(i)}) \quad \text{and} \quad \nu(x) := \sum_{i=1}^{m} \nu_i(x^{(i)}).
\]

### 1.2. Related works and contributions of this paper

In many real-world applications, the topology of the communication network may constantly change, due to possible connectivity issues especially for agents that are Internet-of-Things devices like cellphones or car sensors. Therefore, it is important to design decentralized algorithms with sampling or computation complexity independent of the graph topology [20]. Ideally, our goal is to develop decentralized methods whose sampling (resp. gradient computation) complexity bounds are graph invariant and in the same order as those of centralized methods for solving stochastic (resp. deterministic) problems. Although there are fruitful research results in which the sampling or computational complexities and communication complexities are separated for decentralized (stochastic) optimization, for instance, (stochastic) (sub)gradient based algorithms [30, 22, 5, 29, 19, 16], dual type or ADMM based decentralized methods [28, 2, 27, 1, 31, 3], communication efficient methods [14, 4, 12, 16], and second-order methods [18, 17], none of the existing decentralized methods can achieve the aforementioned goal. In this paper, we pursue the goal of developing graph topology invariant decentralized optimization algorithms based on the gradient sliding methods [13, 15].

Our contributions in this paper can be summarized as follows.

First, for the general convex deterministic problem (1.1), we propose a novel decentralized algorithm, namely the primal dual sliding (PDS) algorithm, that is able to compute an \( \varepsilon \)-solution with \( \mathcal{O}(1)((\bar{L}/\varepsilon)^{1/2}) \) gradient complexity. This complexity is invariant with respect to the topology of the communication network. To the best of our knowledge, this is the first decentralized algorithm for problem (1.1) that achieves the same order of gradient complexity bound as those for centralized methods. Such gradient complexity can be improved to \( \mathcal{O}(1)((\bar{L}/\mu)^{1/2} \log(1/\varepsilon)) \) for the strongly convex case of problem (1.1).

Second, for the general convex stochastic problem (1.1) in which the gradients of the objective functions can only be estimated through stochastic first-order oracle, we propose a stochastic primal dual sliding (SPDS) algorithm that is able to compute an \( \varepsilon \)-solution with \( \mathcal{O}(1)((\bar{L}/\varepsilon)^{1/2} + (\sigma/\varepsilon)^2) \) sampling complexity. The established complexity is also invariant with respect to the topology of the communication network. Such result can be improved to \( \mathcal{O}(1)((\bar{L}/\mu)^{1/2} \log(1/\varepsilon) + \sigma^2/\varepsilon) \) for the strongly convex case.

Third, as a byproduct, we show that a simple extension of the PDS algorithm can be applied to solve certain convex-concave saddle point problems with bilinear coupling. For general convex smooth case, the number of gradient evaluations of \( \nabla \tilde{f} \) and matrix operations (involving \( A \) and \( A^\top \)) are bounded by \( \mathcal{O}(1)((\bar{L}/\varepsilon)^{1/2}) \) and \( \mathcal{O}(1)((\bar{L}/\varepsilon)^{1/2} + \|A\|/\varepsilon) \) respectively. As a special case, our proposed algorithm is also able to solve convex smooth optimization problems with linear constraints. This is the first time such complexity results are achieved for linearly constrained convex smooth optimization. We also extend our results to strongly convex problems.
1.3. Organization of the paper. This paper is organized as follows. In section 2 we present the PDS algorithm for decentralized optimization. In section 3 we present the SPDS algorithm for problems that require sampling of stochastic gradients. The byproduct results on general bilinearly coupled saddle point problems are presented in Section 4. To facilitate reading, we postpone all the major proofs to Section 5. In Section 6 we present some preliminary numerical experiment results. Finally the concluding remarks are presented in Section 7.

2. The primal dual sliding algorithm for decentralized optimization. In this section, we propose a primal dual sliding (PDS) algorithm for solving the linearly constrained formulation (1.2) for decentralized multi-agent optimization in which each agent has access to the deterministic first-order information of its local objective $f_i$ (i.e., $\sigma = 0$ in the assumption (1.5)). Inspired by the saddle point formulation (1.3) and the decoupling of the strongly convex term in (1.4), we propose to study the following saddle point problem:

$$\min_{x \in \mathcal{X}} \max_{y \in \mathbb{R}^m, z \in \mathbb{R}^m} \mu \nu(x) + \langle x, y + A^T z \rangle - \tilde{f}^*(y).$$

Here $\tilde{f}^*$ is the convex conjugate of the function $\tilde{f}$ defined in (1.6). Specifically, $\tilde{f}^*$ can be described as the sum of the convex conjugates $\tilde{f}^i_*$ of functions $\tilde{f}_i$, i.e.,

$$\tilde{f}^*(y) = \sum_{i=1}^m \tilde{f}^i_*(\hat{y}^{(i)}),$$

where $y := (y^{(1)}, \ldots, y^{(m)}) \in \mathbb{R}^m$ and $y^{(1)}, \ldots, y^{(m)} \in \mathbb{R}^d$.

Our proposed PDS algorithm is a primal-dual algorithm that constantly maintains and updates primal variables $x$ and dual variables $y$ and $z$. Most importantly, its algorithmic scheme allows the skipping of computations of gradients $\nabla \tilde{f}$ from time to time, to which we refer as a “sliding” feature. We describe the proposed PDS algorithm in Algorithms 2.1 and 2.2, where Algorithm 2.1 describes the implementation from agent $i$'s perspective and Algorithm 2.2 focuses on implementation over the whole communication network. Here in Algorithm 2.1 we assume that $A = \mathcal{L} \otimes I_d$ is defined by the graph Laplacian $\mathcal{L}$ of the communication network; similar implementation can be described if $A = B^T \otimes I_d$ is defined by an oriented incidence matrix $B$.

A few remarks are in place for Algorithms 2.1 and 2.2. First, in Algorithm 2.1, $V_i$ and $W_i$ are prox-functions utilized by the $i$-th agent. Specifically, they are defined based on strongly convex functions $\nu$ and $\tilde{f}^i_*$ respectively:

$$V_i(x^{(i)}, \hat{x}^{(i)}) := \nu(x^{(i)}) - \nu(\hat{x}^{(i)}) - \langle \nu'(\hat{x}^{(i)}), x^{(i)} - \hat{x}^{(i)} \rangle, \forall x^{(i)}, \hat{x}^{(i)} \in X^{(i)},$$

$$W_i(\hat{y}^{(i)}, y^{(i)}) := \tilde{f}^i_*(\hat{y}^{(i)}) - \tilde{f}^i_* (\hat{y}^{(i)}) - \langle (\tilde{f}^i_*)'(\hat{y}^{(i)}), y^{(i)} - \hat{y}^{(i)} \rangle, \forall \hat{y}^{(i)}, y^{(i)} \in \mathbb{R}^d,$$

where $(\tilde{f}^i_*)'$ denotes the subgradient of $\tilde{f}^i_*$. We will also use the notation

$$V(\hat{x}, x) := \sum_{i=1}^m V_i(\hat{x}^{(i)}, x^{(i)})$$

in the description of Algorithm 2.2. Second, the major difference between Algorithms 2.1 and 2.2 is the update step for $y_k$. However, it can be shown that the two update steps for $y_k$, i.e., (2.3) and (2.5), are equivalent and hence Algorithms 2.1 and 2.2 are equivalent. Indeed, in view of the definition of $W$ in (2.8) and (2.9) and the optimality condition of (4.10), we have $-\hat{x}_k + (1 + \tau_k)(\tilde{f}^*)'(y_k) - \tau_k(\tilde{f}^*)'(y_{k-1}) = 0$ for certain subgradients $(\tilde{f}^*)'(y_k)$ and $(\tilde{f}^*)'(y_{k-1})$ of $\tilde{f}^*$. Consequently, if $\hat{x}_{k-1} = (\tilde{f}^*)'(y_{k-1})$ or equivalently $y_{k-1} = \nabla \tilde{f}(\hat{x}_{k-1})$, and $\hat{x}_k$ is defined recursively as in (2.2), then we have $y_k = \nabla \tilde{f}(\hat{x}_k)$. Noting in the description of Algorithms 2.1 and 2.2 that
Algorithm 2.1 The PDS algorithm for solving (2.1), from agent $i$’s perspective

Choose $x_0^{(i)} \in X^{(i)}$, and set $\bar{x}_0^{(i)} = x_0^{(i)}$, and $\bar{y}_0 = \nabla f(\bar{x}_0^{(i)})$, and $\tau_0 = 0$. 
for $k = 1, \ldots, N$ do

Compute

$$\bar{x}_k^{(i)} = \bar{x}_{k-1}^{(i)} + \lambda_k (\bar{x}_{k-1}^{(i)} - x_{k-2}^{(i)})$$

(2.2)

$$\bar{y}_k^{(i)} = (\bar{x}_k^{(i)} + \tau_k \bar{z}_{k-1}^{(i)}) / (1 + \tau_k)$$

(2.3)

and set $x_k^{0(i)} = \bar{x}_{k-1}^{(i)}, x_k^{0(i)} = x_{k-1}^{(i)}$, and $x_k^{1(i)} = \bar{x}_{k-1}^{(i)}$ (set $x_1^{1(i)} = x_0^{(i)}$).

for $t = 1, \ldots, T_k$ do

Compute

$$\bar{u}_t^{(i)} = x_{k-1}^{t-1(i)} + \alpha_k^{(i)} (x_k^{t-1(i)} - x_k^{t-2(i)})$$

and

$$\bar{z}_k^{(i)} = \arg \min_{z^{(i)} \in \mathbb{R}^d} \left\{ - \sum_{j \in N_i} L^{(i,j)} \bar{u}_t^{(j)}, z^{(i)} \right\} + \frac{\eta_k}{2} \| z - \bar{z}_{k-1}^{(i)} \|^2$$

(2.4)

$$x_t^{(i)} = \arg \min_{x^{(i)} \in X^{(i)}} \left\{ y_k^{(i)} + \sum_{j \in N_i} L^{(i,j)} \bar{z}_k^{(j)}, x^{(i)} \right\} + \eta_k V_i(x_{k-1}^{t-1(i)}, x^{(i)})$$

end for

Set $x_k^{(i)} = x_k^{T_k(i)}$, $z_k^{(i)} = x_k^{T_k(i)}$, $x_k^{(i)} = \sum_{t=1}^{T_k} x_t^{(i)} / T_k$, and $\bar{z}_k^{(i)} = \sum_{t=1}^{T_k} z_t^{(i)} / T_k$.

end for

Output $\sum_{k=1}^N \beta_k \bar{x}_k^{(i)}$.

Algorithm 2.2 The PDS algorithm for solving (2.1), whole network perspective

Choose $x_0, z_0 \in X^{(i)}$, and set $\bar{x}_0 = x_0 = x_0, y_0 = \nabla f(\bar{x}_0)$, and $z_0 = 0$.
for $k = 1, \ldots, N$ do

Compute

$$\bar{x}_k = \bar{x}_{k-1} + \lambda_k (\bar{x}_{k-1} - x_{k-2})$$

(2.5)

$$y_k = \arg \min_{y \in \mathbb{R}^d} \left\{ - \bar{x}_k, y \right\} + \tau_k W(y_{k-1}, y).$$

Set $x_k^0 = x_{k-1}, x_k^0 = z_{k-1}$, and $x_k^{1} = x_{k-1}^{T_k-1}$ (when $k = 1$, set $x_1^{1} = x_0$).
for $t = 1, \ldots, T_k$ do

Compute

$$\bar{u}_t^k = x_{k-1}^{t-1} + \alpha_k (x_k^{t-1} - x_k^{t-2})$$

and

$$\bar{z}_k^t = \arg \min_{z \in \mathbb{R}^d} \left\{ - A \bar{u}_t^k, z \right\} + \frac{\eta_k}{2} \| z - \bar{z}_{k-1} \|^2$$

(2.7)

$$x_k^t = \arg \min_{x \in \mathbb{R}^d} \left\{ y_k + A^T \bar{z}_k^t, x \right\} + \eta_k V(x_k^{t-1}, x) + p_k V(x_{k-1}, x)$$

end for

Set $x_k = x_k^{T_k}$, $z_k = \bar{z}_k^T$, $\bar{x}_k = \sum_{t=1}^{T_k} x_t^T / T_k$, and $\bar{z}_k = \sum_{t=1}^{T_k} z_t^T / T_k$.

end for

Output $\sum_{k=1}^N \beta_k \bar{x}_k$. 

This manuscript is for review purposes only.
Then we have
\begin{align}
\frac{\partial}{\partial t} f(\sum_{k=1}^{N} \beta_k) \leq \left( \sum_{k=1}^{N} \beta_k \right)^{-1} \beta_1 \left[ \frac{q_1}{T^1} + p_1 \right] V(x_0, x^*) \quad \text{and} \\
\|A x_N\|_2 \leq \left( \sum_{k=1}^{N} \beta_k \right)^{-1} \beta_1 \left[ \frac{q_1}{T^1} (\|z^*\|_2 + 1)^2 + \left( \frac{q_1}{T^1} + p_1 \right) V(x_0, x^*) \right].
\end{align}

**Theorem 2.2.** Denote \( \tau := \sqrt{2L/\mu} \) and \( \Delta := [2\tau + 1] \) if \( \mu > 0 \), and \( \Delta := +\infty \) if \( \mu = 0 \). Suppose that the parameters in Algorithm 2.1 are set to the following: for all \( k \leq \Delta \),
\begin{align}
\tau_k = \frac{k-1}{2}, \quad \lambda_k = \frac{k-1}{k}, \quad \beta_k = k, \quad p_k = \frac{2L}{k}, \quad T_k = \left[ \frac{kR\|A\|}{L} \right].
\end{align}
for all $k > \Delta$, \( \tau_k = \tau, \lambda_k = \lambda := \frac{\tau}{1+\tau}, \beta_k = \Delta \lambda^{-(k-\Delta)}, \)
\( p_k = \frac{\tilde{T}}{1+\tau}, \quad T_k = \left\lfloor \frac{2(1+\tau)N\|\mathcal{A}\|}{L\lambda^{1/(1+\tau)}} \right\rfloor, \)
and for all $k$ and $t$,
\[
\eta_k = (p_k + \mu)(t-1) + p_k T_k, q_{k}^{1} = \begin{cases} \frac{\beta_{k-1}T_k}{\beta_{k}\tau_{k-1}}, & \text{if } k \geq 2 \text{ and } t = 1; \\ 1, & \text{otherwise.} \end{cases}
\]

Applying Algorithm 2.1 to solve problem (1.2), we have
\[
f(\pi_N) - f(x^*) \leq \min \left\{ \frac{2}{N}, \lambda^{N-\Delta} \right\} \cdot 4\tilde{L}V(x_0, x^*)
\]
\[
\|\mathcal{A}T_N\|_2 \leq \min \left\{ \frac{2}{N_2}, \lambda^{N-\Delta} \right\} \left[ \frac{4\tilde{L}(\|z^*\|_2 + 1)^2 + 4\tilde{L}V(x_0, x^*)}{\epsilon} \right],
\]
Specially, if we set
\[
R = \frac{\|z^*\|_2+1}{4\sqrt{V(x_0, x^*)}},
\]
then we have the following gradient and communication complexity results for Algorithm 2.1 (here $O(1)$ is a constant that is independent of $N$, $\|\mathcal{A}\|$ and $\epsilon$):

a) If $\mu = 0$, i.e., problem (1.2) is smooth and convex, then we can obtain an $\epsilon$-solution after at most $N := \left\lfloor \frac{4\sqrt{\tilde{L}}V(x_0, x^*)}{\epsilon} \right\rfloor$ gradient evaluations and $2N + 96\|\mathcal{A}\|/\epsilon$ communications.

b) If $\mu > 0$, i.e., problem (1.2) is smooth and strongly convex, then we can obtain an $\epsilon$-solution after at most $N := O(1) \left( 1 + \sqrt{\frac{\tilde{L}}{\mu} \log(\tilde{L}V(x_0, x^*)/\epsilon)} \right)$ gradient evaluations and $2N + O(1)\|\mathcal{A}\| \left( 1 + 1/\sqrt{\epsilon} \right)$ communications.

Proof. In view of the selection of $\eta_k$ and $q_k^1$, we can observe that
\[
\eta_k^{t-1}q_k^1 \geq \frac{\eta_k T_k}{2\beta_k} \left( \frac{T_k}{R} \right)^2 = \begin{cases} \|\mathcal{A}\|^2 \left( \frac{\tilde{T}_k}{k\|\mathcal{A}\|} \right)^2, & k \leq \Delta; \\ \frac{\tilde{L}k^{-\Delta}T_k}{(2+2\|\mathcal{A}\|/\Delta)} \left( \frac{\tilde{L}k^{-\Delta}T_k}{2(1+\tau)N\|\mathcal{A}\|} \right)^2, & k > \Delta, \end{cases}
\]
and together with the definitions of $\Delta$ and $T_k$ we have $\eta_k^{t-1}q_k^1 \geq \|\mathcal{A}\|^2$ for all $k \geq 1$ and $t \geq 2$. Using this observation, it is easy to verify that (2.11) holds. Noting that the above observation also implies that $\eta_k^{T_k}q_k^{T_k} \geq \|\mathcal{A}\|^2$, and also noting that $\tau_1 = 0$ and that $p_N(\tau_N + 1) = \tilde{L}(N+1)/N$ when $N \leq \Delta$ and $p_N(\tau_N + 1) = \tilde{L}$ when $N > \Delta$, we have (2.12) holds. It remains to verify (2.10). Note that for all $k$ we have
\[
\beta_k T_k - \alpha_k^1 = \beta_k - T_k \quad \text{and} \quad \beta_k T_k - q_k^{1} = \tilde{L}T_k T_k - (2R^2) = \beta_k T_k q_k^{T_k} \quad \text{We consider}
\]
three cases: $2 \leq k \leq \Delta$, $k = \Delta + 1$, and $k > \Delta + 1$. When $2 \leq k \leq \Delta$, we have
\[
\beta_{k-1}(\tau_{k-1} + 1) = (k-1)k/2 = \beta_{k}\tau_{k}, \\
\beta_{k-1} - 1 = \beta_k \lambda_k, \\
\tilde{L}\lambda_k = \tilde{L}(k-1)/k < \tilde{L} = p_{k-1}\tau_{k}, \\
\eta_k^{T_k}q_k^1 \geq \eta_k^{T_k}q_k^1 \geq \left( \frac{T_k}{(k-1)N\|\mathcal{A}\|} \right)^2 \geq 1,
\]
\[
\beta_k T_k (\eta_k + p_k T_k) - \beta_{k-1} T_k (\mu + \eta_k^{T_k} + p_k) = -(k-1)\mu T_k T_k \leq 0.
\]
When \( k = \Delta + 1 \), from the definition of \( \Delta \) we have \( 2\tau + 1 \leq \Delta \leq 2\tau + 2 \). Therefore
\[
\beta_k \tau_k - \beta_{k-1} (\tau_{k-1} + 1) = \Delta \lambda^{-1} \tau - \Delta (\Delta + 1)/2 = (\Delta/2)(2(1 + \tau) - (\Delta + 1)) \leq 0,
\]
\[
\beta_{k-1} - \beta_k \lambda_k = \Delta - \Delta \lambda^{-1} \lambda = 0,
\]
\[
\hat{L}\lambda_k - p_{k-1} \tau_k = \hat{L}\lambda - (2\hat{L}/\Delta) \tau \leq \hat{L}\lambda - (\hat{L}/(1 + \tau)) \tau = 0,
\]
\[
\frac{\eta_{k-1}^{\tau_k}}{\alpha_k^{\tau_k} \|A\|^2} \geq \frac{\eta_{k-1}^{\tau_k}}{\alpha_k^{\tau_k} \|A\|^2} \geq \left( \frac{\hat{L}\lambda}{\Delta R_k \|A\|} \right)^2 \geq 1,
\]
\[
\beta_k \tau_{k-1} (\eta_{k}^{\tau_k} + p_k T_k) - \beta_{k-1} T_k (\mu + \eta_{k-1}^{\tau_{k-1}} + p_{k-1}) = \mu \left[ \Delta (\tau - 1) + 2(1/\tau) \right] T_k T_{k-1} \leq 0.
\]
When \( k > \Delta + 1 \), we have
\[
\beta_{k-1} (\tau_{k-1} + 1) = \Delta \lambda^{-(k-1-\Delta)} (\tau + 1) = \Delta \lambda^{-k-\Delta \tau} = \beta_k \tau_k,
\]
\[
\beta_{k-1} = \Delta \lambda^{-k-\Delta} = \beta_k \lambda_k,
\]
\[
\hat{L}\lambda_k = \hat{L}\tau/(1 + \tau) = p_{k-1} \tau_k.
\]
\[
\frac{\eta_{k-1}^{\tau_k}}{\alpha_k^{\tau_k} \|A\|^2} \geq \frac{\eta_{k-1}^{\tau_k}}{\alpha_k^{\tau_k} \|A\|^2} \geq 2^{\tau_k + 2}/\Delta, \left( \frac{\hat{L}\lambda^{k-1-\Delta}}{2(1 + \tau) \|A\|^2} \right)^2 \geq 1,
\]
\[
\frac{\beta_k \tau_{k-1} (\eta_{k}^{\tau_k} + p_k T_k)}{\beta_{k-1} T_k (\mu + \eta_{k-1}^{\tau_{k-1}} + p_{k-1})} = \frac{\tau_k^{(1+\tau)}}{\tau_k^{(1+\tau)} + 1} \leq 1.
\]
We now can conclude that (2.10) holds. Note that the summation \( \sum_{k=1}^N \beta_k \) can be lower bounded as follows: when \( N \leq \Delta \), we have \( \sum_{k=1}^N \beta_k = N(N + 1)/2 > N^2/2 \); when \( N > \Delta \), we have \( \sum_{k=1}^N \beta_k > \beta_N \geq \lambda^{-(N-\Delta)} \).

Applying Proposition 2.1 and using the above note on the summation \( \sum_{k=1}^N \beta_k \) and the parameter selection in (2.13), we obtain results (2.15) and (2.16). Moreover, setting \( R \) to (2.17) we conclude that
\[
\max \left\{ f(\pi_N) - f(x^*), \|A\pi_N\|_2 \right\} \leq \min \left\{ \frac{2}{\lambda^{N-\Delta}}, \lambda^{N-\Delta} \right\} 8\hat{L}V(x_0, x^*).
\]
We conclude immediately that \( \pi_N \) will be an \( \epsilon \)-solution if \( N = \left[ 4\sqrt{\hat{L}V(x_0, x^*)}/\epsilon \right] \) or
\[
N = \left[ \Delta + \log \lambda^{-1} (8\hat{L}V(x_0, x^*)/\epsilon) \right]. \quad \text{As a consequence, when the problem is smooth and convex, i.e.,} \mu \leq 0 \text{by setting} \quad N = \left[ 4\sqrt{\hat{L}V(x_0, x^*)}/\epsilon \right] \quad \text{we can obtain an \( \epsilon \)-solution} \quad \pi_N. \quad \text{Using the definitions of} \quad T_k \quad \text{and} \quad R \quad \text{in} \quad (2.13) \quad \text{and} \quad (2.17) \quad \text{respectively, the total number of communication rounds required by the PDS algorithm can be bounded by}
\]
\[
2\sum_{k=1}^N T_k \leq 2N + N(N + 1) R \|A\|/\|L\| \leq 2N + 96 \|A\|/\|L\|. \quad \text{When the problem is smooth and strongly convex, i.e.,} \mu > 0 \text{we may set} \quad N := \left[ \Delta + \log \lambda^{-1} (8\hat{L}V(x_0, x^*)/\epsilon) \right] \quad \text{to obtain an \( \epsilon \)-solution} \quad \pi_N. \quad \text{In view of the fact that} \quad - \log \lambda = \log(1 + 1/\tau) \geq 1/(1 + \tau), \quad \text{the total number of gradient computations is bounded by}
\]
\[
N \leq 1 + \Delta + \frac{1}{-\log \lambda} \log \left( \frac{8\hat{L}V(x_0, x^*)}{\epsilon} \right) \leq 1 + \Delta + (\tau + 1) \left( \log \left( \frac{8\hat{L}V(x_0, x^*)}{\epsilon} \right) \right) \quad = \mathcal{O}(1) \left( 1 + \sqrt{\hat{L}/\lambda} \log(LV(x_0, x^*)/\epsilon) \right).
\]
Also, observing that \( \lambda^{-N} \leq 8\lambda^{-\Delta-1} \hat{L}V(x_0, x^*)/\epsilon \), using the definitions of \( \lambda \), \( T_k \) and \( R \) in (2.13), (2.14) and (2.17), the total number of communication rounds required by
the PDS algorithm is bounded by

\[
2\sum_{k=1}^{N} T_k \leq 2\sum_{k=1}^{\Delta} \left( \frac{kR\|A\|}{L} + 1 \right) + 2\sum_{k=\Delta+1}^{N} \left( \frac{2(1+\tau)\|A\|R}{L\lambda} + 1 \right)
\]

\[
= 2N + \frac{\Delta(\Delta+1)R\|A\|}{L} + \frac{4(1+\tau)R\lambda^2}{L(1-\lambda^2)} \cdot \|A\| \left( \lambda^{-\frac{\xi}{2}} - \lambda^{-\frac{\xi}{2}} \right)
\]

\[
\leq 2N + O(1)\|A\| + O(1)\|A\|/\sqrt{\xi}.
\]

In view of the above theorem, we can conclude that the number of gradient evaluations required by the PDS algorithm to obtain an \(\varepsilon\)-solution to the decentralized optimization problem (1.2) does not depend on the communication network topology; it is only dependent on the Lipschitz constant \(\tilde{L}\), strong convexity parameter \(\mu\), and proximity to an optimal solution \(V(x_0, x^*)\). Moreover, the number of gradient evaluations required by the PDS algorithm is in the same order of magnitude as those optimal centralized algorithms (see, e.g., [24]). To the best of our knowledge, this is the first time such complexity bounds for gradient evaluations are established for decentralized algorithms. Note that the communications complexity for computing an \(\varepsilon\)-solution still depends on the graph topology of the communication network.

3. The stochastic primal-dual sliding algorithm for stochastic decentralized optimization. In this section, we follow the assumption of the unbiased gradient estimators stated in (1.5) and propose a stochastic primal-dual sliding (SPDS) algorithm for solving the decentralized multi-agent optimization problem (1.2) under the stochastic first-order oracle. Our proposed SPDS algorithm is a stochastic primal-dual algorithm that possesses a similar “sliding” scheme as the PDS algorithm. In particular, SPDS integrates the mini-batch technique and uses increasing batch size, while skipping stochastic gradient computation from time to time. We describe the SPDS algorithm in Algorithms 3.1 and 3.2.

**Algorithm 3.1** The SPDS algorithm for solving (2.1), from agent \(i\)'s perspective.

Modify (2.3) and (2.4) in Algorithm 2.1 to

\[
(3.1) \quad v_k^{(i)} = \frac{1}{c_k} \sum_{j=1}^{c_k} G_i(z_k^{(i)}, x^{(j)}_k) \text{ and }
\]

\[
x_k^{1,(i)} = \arg \min_{x^{(i)} \in \mathcal{X}(i)} \mu \nu \left( x + \left( v_k^{(i)} + \sum_{j \in N_i} \mathcal{L}^{(i,j)} x^{(j)}_k, x^{(i)} \right) \right) + \frac{\eta_k}{2} V_i(x_k^{1,(i)}, x^{(i)}) + \frac{\tau}{2} ||x_k^{1,(i)} - x^{(i)}||^2_2.
\]

**Algorithm 3.2** The SPDS algorithm for solving (2.1), whole network perspective.

Modify (2.7) in Algorithm 2.2 to

\[
s_k = \arg \min_{x \in \mathcal{X}} \mu \nu (x) + \langle v_k + A^T z_k, x \rangle + \eta_k^f V(x_k^{1,(i)}, x) + \frac{\tau}{2} ||x_{k-1} - x||^2_2,
\]

in which \(v_k\) satisfies \(E[v_k] = y_k\) and

\[
E[||v_k - y_k||^2_2] \leq \sigma_k^2, \text{ where } \sigma_k := \frac{\eta_k}{\sqrt{c_k}}.
\]

The difference between the SPDS and PDS algorithms is that the SPDS algorithm uses a mini-batch of \(c_k\) stochastic samples of gradients while PDS requires the access of exact gradients, i.e., \(y_k = \nabla f_i(x_k)\). As a result, the update of \(y_k\) is replaced by a mini-batch update step and equations (2.4) and (2.7) of the PDS algorithm are replaced
by their stochastic version accordingly. In order to use the mini-batch technique, we need to replace the prox-function $V(\cdot, \cdot)$ in the PDS algorithm in Algorithm 2.2 by Euclidean distances. Recalling the assumption on the variance of stochastic samples in (1.5) and noting that the batch size at the $k$-th outer iteration is $c_k$, it is easy to obtain that the variance of the gradient estimator $v_k$ is $\sigma_k^2 := \sigma^2/c_k$, as stated in (2.10). It should also be noted that in many applications the $i$-th agent may be able to collect samples $\{\xi_{k,j}^{(i)}\}_{j=1}^{c_k}$ of the underlying random variable in (3.1) for computing the gradient estimator $v_k^{(i)}$. For example, in online supervised machine learning the $i$-th agent can receive samples $\xi_{k,j}^{(i)}$ from online data streams of the training dataset. In such case, the SPDS algorithm allows all agents to collect stochastic samples and perform communications simultaneously. This is because that as soon as the mini-batch computation of the gradient estimator $v_k^{(i)}$ in (3.1) is finished in the $k$-th outer iteration, the $i$-th agent can start collecting future stochastic samples $\{\xi_{k+1,j}^{(i)}\}_{j=1}^{c_k}$ for the next outer iteration. Consequently, the collection of future samples and the communication with neighboring agents in the current $k$-th outer iteration can be performed simultaneously.

The convergence of the SPDS algorithm is described in the following proposition, whose proof is presented in Section 5. Followed by the proposition we also describe example parameter choices in Theorem 3.2.

**Proposition 3.1.** Suppose that the parameters of Algorithm 3.1 satisfy conditions described in (2.10)–(2.12), among which two conditions are modified to

$$2\tilde{L}\lambda_k \leq p_{k-1}\tau_k, \ \forall k \geq 2, \text{ and } p_N(\tau_N + 1) \geq 2\tilde{L},$$

in which the Lipschitz constant $\tilde{L}$ is with respect to the Euclidean norm. We have

$$\mathbb{E}[f(\mathbf{x}_N) - f(x^*)] \leq \left(\sum_{k=1}^{N}\beta_k\right)^{-1} \left[\frac{\bar{d}_p}{2} \left(\frac{n_1}{p_1} + p_1\right) \|x_0 - x^*\|_2^2 + \sum_{k=1}^{N}\frac{\beta_p\sigma^2}{p_k c_k}\right]$$

and

$$\mathbb{E}[\|A\mathbf{x}_N\|_2] \leq \left(\sum_{k=1}^{N}\beta_k\right)^{-1} \left[\frac{\bar{d}_q}{2\tilde{L}} \left(\|z^*\|_2 + 1\right)^2 + \frac{\bar{d}_p}{2} \left(\frac{n_1}{p_1} + p_1\right) \|x_0 - x^*\|_2^2 + \sum_{k=1}^{N}\frac{\beta_p\sigma^2}{p_k c_k}\right].$$

**Theorem 3.2.** Denote $\tau := 2\sqrt{\tilde{L}/\mu}$ and $\Delta := [2\tau + 1]$ if $\mu > 0$, and $\Delta := +\infty$ if $\mu = 0$. Suppose that the maximum number of outer iterations $N$ is specified, and that the parameters in Algorithm 3.1 are set to the following: for all $k \leq \Delta$,

$$\tau_k = \frac{k-1}{2}, \ \lambda_k = \frac{k-1}{k}, \ \beta_k = k, \ \ p_k = \frac{4\tilde{L}}{k}, \ T_k = \left[kR\|A\|\right]/L, \ c_k = \left[\frac{\min\{N, \Delta\}\beta_k c_k}{p_k L}\right],$$

for all $k \geq \Delta + 1$,

$$\tau_k = \tau, \ \lambda_k = \lambda := \frac{\tau}{1+\tau}, \ \beta_k = \Delta \lambda^{-(k-\Delta)}, \ \ p_k = \frac{2\tilde{L}}{1+\tau},$$

$$T_k = \left[\frac{2(1+\tau)R\|A\|}{k-\Delta \tilde{L}\lambda}\right], \ c_k = \left[\frac{(1+\tau)^2 \Delta c_{k-1}}{\lambda^2 \sum_{i=1}^{N}\Delta c_{i}}\right],$$

and for all $k$ and $t$,

$$\eta_k = (p_k + \mu)(t-1) + p_k T_k, \ q_k = \frac{\eta_k}{\alpha_k T_k}, \ \alpha_k = \begin{cases} 1 & k \geq 2 \text{ and } t = 1 \\ \frac{p_k T_{k-1}}{p_k T_{k-1}} & \text{otherwise.} \end{cases}$$

This manuscript is for review purposes only.
Applying Algorithm 3.1 to solve problem (1.2) we have

\[ f(x_N) - f(x^*) \leq \min \left\{ \frac{3}{N^2}, \lambda^{N-\Delta} \right\} \left[ \frac{4\bar{L}}{c} \|x_0 - x^*\|^2 + \frac{\bar{L}c^2}{2} \right] \]  

and

\[ \|Ax_N\|_2 \leq \min \left\{ \frac{3}{N^2}, \lambda^{N-\Delta} \right\} \left[ \frac{\bar{L}}{c^2} (\|x^*\|_2^2 + 1)^2 + 4\bar{L} \|x_0 - x^*\|^2 + \frac{\bar{L}c^2}{2} \right]. \]

Specially, if we set

\[ R = \frac{\|x^*\|_2 + 1}{4\|x_0 - x^*\|_2^2} \quad \text{and} \quad c = \frac{2\sigma^2}{\|x_0 - x^*\|_2^2}, \]

then we have the following sampling and communication complexity results for Algorithm 3.1 (here \( O(1) \) is a constant that is independent of \( \sigma, N, \|A\| \) and \( c \):

a) If \( \mu = 0 \), by setting \( N = \left\lceil \sqrt{10\bar{L}} \|x_0 - x^*\|_2^2 / \varepsilon \right\rceil \) we can obtain a stochastic \( \varepsilon \)-solution with \( N + 800\sigma^2\|x_0 - x^*\|_2^2 / \varepsilon \) gradient samples and \( N + 600\|A\|_2 / \varepsilon \) rounds of communication.

b) If \( \mu > 0 \), by setting \( N = \left\lceil \Delta + \log_{\lambda^{-1}}(5\bar{L}) \|x_0 - x^*\|_2^2 / \varepsilon \right\rceil \) we can obtain a stochastic \( \varepsilon \)-solution with \( O(1) \left( 1 + \sqrt{\frac{\bar{L}}{\mu}} \log(\bar{L}) \|x_0 - x^*\|^2 / \varepsilon \right) + \sigma^2(1 + 1/\varepsilon) \) samples and \( O(1) \left( 1 + \sqrt{\frac{\bar{L}}{\mu}} \log(\bar{L}) \|x_0 - x^*\|^2 / \varepsilon \right) + \|A\|_2(1 + 1/\varepsilon) \) rounds of communication.

Proof. Following a similar argument to the proof of Theorem 2.2, it is easy to verify that the parameter selections in (3.4) - (3.6) satisfy the conditions stated in Proposition 3.1. The results in (3.7) and (3.8) follows immediately from the proposition and our parameter selections. We now consider the case when \( \mu = 0 \) (hence \( \Delta = \infty \)). By the choice of \( N \), the selections of \( \beta_k, p_k, R \), and \( c \) in (3.4) and (3.9), and noting that \( \sum_{k=1}^{N} k^2 \leq N^3 \) we can bound the total number of gradient samples by

\[
\sum_{k=1}^{N} c_k \leq \sum_{k=1}^{N} \left( 1 + \frac{cN^2}{4\bar{L}^2} \right) = N + \frac{\sigma^2 N^4}{2\bar{L}^2 \|x_0 - x^*\|^2} \sum_{k=1}^{N} k^2 \\
\leq N + \frac{\sigma^2 N^4}{2\bar{L}^2 \|x_0 - x^*\|^2} = N + \frac{800\sigma^2 \|x_0 - x^*\|^2}{\varepsilon^2}.
\]

In the case when \( \mu > 0 \), similar to the proof of Theorem 2.2 we can observe two bounds \( N \leq O(1) \left( 1 + \sqrt{\frac{\bar{L}}{\mu}} \log(\bar{L}) \|x_0 - x^*\|^2 / \varepsilon \right) \) and \( \lambda^{-N} \leq (1/\varepsilon)O(1) \). In view of these bounds, the descriptions of \( c_k \) and \( c \) in (3.4), (3.5), and (3.9), we can bound the total number of gradient samples by

\[
\sum_{k=1}^{N} c_k \leq \sum_{k=1}^{\Delta} \left( 1 + \frac{\Delta^2 c^2}{2\bar{L}^2 \|x_0 - x^*\|^2} \right) + \sum_{k=\Delta+1}^{N} \left( 1 + \frac{(1+\tau)^2 \Delta c}{2\bar{L}^2 \lambda^{-\Delta}} \right) \\
= N + O(1) \sigma^2 + O(1) \frac{1 - \lambda^{\frac{N-\Delta}{2}}}{1 - \lambda^2} - \sigma^2 \lambda^{-(N-\Delta)} \\
\leq O(1) \left( 1 + \sqrt{\frac{\bar{L}}{\mu}} \log(\bar{L}) \|x_0 - x^*\|^2 / \varepsilon \right) + O(1) \sigma^2 + O(1) \sigma^2 / \varepsilon.
\]

In view of Theorem 3.2, we can conclude that the sampling complexity achieved by the SPDS algorithm does not depend on the communication network topology;
it is only dependent on the Lipschitz constant $\tilde{L}$, strong convexity parameter $\mu$, the variance of the gradient estimators $\sigma^2$, and proximity to an optimal solution $\|x_0 - x^*\|_2$.

Moreover, the achieved sampling complexity is in the same order of magnitude as those optimal centralized algorithms, such as accelerated stochastic approximation method [8, 9]. To the best of our knowledge, this is the first time such sampling complexity bounds are established for stochastic decentralized algorithms. Indeed, the total number of communications required to compute an $\varepsilon$-solution still depends on the graph topology. It should also be noted that the constant factors in part a) of the above theorem (800 and 600) can be further reduced, e.g., by assuming that $N \geq 5$. However, our focus is on the development of order-optimal complexity results and hence we skip these refinements.

4. A byproduct on solving bilinearly coupled saddle point problems. In this section, we present a byproduct of our analysis and show that the PDS algorithm described in Algorithm 2.2 is in fact an efficient algorithm for solving certain convex-concave saddle point problem with linear coupling. Specifically, in this section we consider the following saddle point problem:

\[
\begin{align*}
\min_{x \in \mathcal{X}} & \max_{z \in \mathcal{Z}} f(x) + \langle Ax, z \rangle - b(z) \\
\mbox{s.t.} & \mathcal{A}x = b.
\end{align*}
\]

where $\mathcal{X}$ and $\mathcal{Z}$ are closed convex sets, $\mathcal{A}$ is a linear operator, and $f$ and $h$ are convex functions. Here, we assume that $h(z)$ is relatively simple, and that $f(x) = f(x) + \mu \nu(x)$, where $\mu \geq 0$, $\nu : \mathcal{X} \to \mathbb{R}$ is a strongly convex function with strong convexity modulus 1 with respect to a norm $\| \cdot \|$, and $\tilde{f} : X \to \mathbb{R}$ is a convex smooth function whose gradient is Lipschitz with constant $\tilde{L}$ with respect to the norm $\| \cdot \|$.

Note that problem (4.1) is equivalent to

\[
\begin{align*}
\min_{x \in \mathcal{X}} & \max_{y \in \mathcal{Y}, z \in \mathcal{Z}} \mu(x) + \langle x, y + A^\top z \rangle - \tilde{f}^*(y) - h(z) \\
\mbox{s.t.} & \mathcal{Y} := \text{dom} \tilde{f}^*. Also, observe that when $\mathcal{Z}$ is a vector space and $h(z) = \langle b, z \rangle$, the saddle point problem (4.1) becomes

\[
\begin{align*}
\min_{x \in \mathcal{X}} f(x) \mbox{ s.t. } \mathcal{A}x = b.
\end{align*}
\]

If in addition $b = 0$, $\mathcal{Z} = \mathbb{R}^m$, and $\mathcal{X}$, $f$, and $\mathcal{A}$ are defined in (1.2), then we obtain the linearly constrained reformulation of the multi-agent optimization problem (1.1).

We describe an extension of the PDS algorithm in Algorithm 4.1. In the algorithm description, $U$, $V$, and $W$ are prox-functions defined by

\[
\begin{align*}
(4.4) & \quad U(\hat{z}, z) := \zeta(z) - \zeta(\hat{z}) - \langle \zeta'(\hat{z}), z - \hat{z} \rangle, \\
(4.5) & \quad V(\hat{x}, x) := \nu(x) - \nu(\hat{x}) - \langle \nu'(\hat{x}), x - \hat{x} \rangle, \quad \text{and} \\
(4.6) & \quad W(v, y) := \tilde{f}^*(y) - \tilde{f}^*(v) - \langle (\tilde{f}^*)'(v), y - v \rangle.
\end{align*}
\]

respectively, where $\nu$ and $\tilde{f}^*$ are defined in (4.2) and $\zeta : \mathcal{Z} \to \mathbb{R}$ is a strongly convex function with strong convexity modulus 1 with respect to a norm $\| \cdot \|$. Noting from the strong convexity of $\zeta$ and $\nu$ we have for any $\hat{z}, z \in \mathcal{Z}$ and $\hat{x}, x \in \mathcal{X}$ that

\[
\begin{align*}
(4.7) & \quad U(\hat{z}, z) \geq \frac{1}{2} \| \hat{z} - z \|^2 \quad \text{and} \quad V(\hat{x}, x) \geq \frac{1}{2} \| \hat{x} - x \|^2.
\end{align*}
\]

Moreover, since $\tilde{f}$ is convex and $\nabla \tilde{f}$ is Lipschitz with constant $\tilde{L}$ with respect to norm $\| \cdot \|$, $\tilde{f}^*$ should be strongly convex with strong convexity parameter $1/\tilde{L}$ with respect
to the dual norm $\| \cdot \|_*$. Consequently, noting the definition of $W$ in (4.6) we have

$$W(v, y) \geq \frac{1}{2L} \|v - y\|_*^2, \quad \forall v, y \in \mathcal{Y}.$$  

(4.8)

Algorithm 4.1  An equivalent alternative description of the PDS algorithm

Choose $x_0 \in \mathcal{X}$, $y_0 \in \mathcal{Y}$, and $z_0 \in \mathcal{Z}$. Set $\hat{x}_0 = x_0$, $\hat{y}_0 = y_0$, $\hat{z}_0 = z_0$.

for $k = 1, \ldots, N$ do

Compute

$$\hat{x}_k = x_{k-1} + \lambda_k (\hat{x}_{k-1} - x_{k-2}), \quad y_k := \arg\min_{y \in \mathcal{Y}} \left\{-\hat{x}_k, y\right\} + \hat{f}^*(y) + \tau_k W(y_{k-1}, y).$$

Set $x^0_k = x_{k-1}$, $z^0_k = z_{k-1}$, and $x^{-1}_k = x^{T_k-1}_{k-1}$ (when $k = 1$, set $x^{-1}_1 = x_0$).

for $t = 1, \ldots, T_k$ do

$$u^t_k = x^{t-1}_k + \alpha^t_k (x^{t-1}_k - x^{t-2}_k),$$

$$z^t_k = \arg\min_{z \in \mathcal{Z}} h(z) + \langle -A u^t_k, z \rangle + q^t_k U(z^{t-1}_k, z),$$

$$x^t_k = \arg\min_{x \in \mathcal{X}} \mu(x) + \langle y_k + A^\top z^t_k, x \rangle + \eta^t_k V(x^{t-1}_k, x) + p_k V(x_{k-1}, x)$$

end for

Set $x_k = x^{T_k}_k$, $z_k = z^{T_k}_k$, $\hat{x}_k = \sum_{t=1}^{T_k} x^t_k / T_k$, and $\hat{z}_k = \sum_{t=1}^{T_k} z^t_k / T_k$.

end for

Output $\pi_N := (\pi_N, \pi_N, \pi_N) := \left(\sum_{k=1}^{N} \beta_k\right)^{-1} \left(\sum_{k=1}^{N} \beta_k (\hat{x}_k, y_k, \hat{z}_k)\right)$. 

Extending the analysis of Algorithm 2.2, it is straightforward to observe that the equivalent description of the PDS algorithm described in Algorithm 4.1 can be applied to solve the more general saddle point problem (4.2). We will evaluate the accuracy through the gap function defined by

$$Q(\tilde{w}_k, w) := \left[\mu w(\hat{x}_k) + \langle \hat{x}_k, y + A^\top z \rangle - \hat{f}^*(y) - h(z)\right]$$

$$- \left[\mu w(x + y_k + A^\top \hat{z}_k) - \hat{f}^*(y_k) - h(\hat{z}_k)\right],$$

(4.11)

in which $\tilde{w}_k := (\hat{x}_k, y_k, \hat{z}_k)$ and $w := (x, y, z)$. We have the following proposition describing the convergence result, whose proof will be presented in Section 5.

PROPOSITION 4.1. Suppose that the conditions (2.10)–(2.12) (where $\|A\|$ is the norm induced by $\| \cdot \|$ and $\| \cdot \|$) described in Proposition 2.1 hold for Algorithm 4.1.

For all $w \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$ we have the following convergence result when Applying Algorithm 4.1 to solve problem (4.2):

$$Q(\pi_N, w) \leq \left(\sum_{k=1}^{N} \beta_k\right)^{-1} \beta_1 \left[\frac{q^1}{T_1} U(z_0, z) + \left(\frac{q^1}{T_1} + p_1\right) V(x_0, x)\right].$$

(4.12)

Moreover, in the special case of problem (4.3) (i.e., if $\mathcal{Z}$ is a vector space with Euclidean norm $\| \cdot \| = \| \cdot \|_2$, the prox-function $U(\cdot, \cdot) = \| \cdot - \cdot \|_2^2 / 2$, and $h(z) = \langle b, z \rangle$), then
we can set the initial value $z_0 = 0$ in Algorithm 4.1 and obtain

\begin{equation}
(4.13) \quad f(\pi_N) - f(x^*) \leq \left( \sum_{k=1}^N \beta_k \right)^{-1} \beta_1 \left( \frac{q_1^2}{p_1} + p_1 \right) V(x_0, x^*) \quad \text{and}
\end{equation}

\begin{equation}
(4.14) \quad \|\mathbf{A} \pi_N - b\|_2 \leq \left( \sum_{k=1}^N \beta_k \right)^{-1} \beta_1 \left[ \frac{q_1^2}{p_1}(\|z^*\|_2 + 1)^2 + \left( \frac{q_1^2}{p_1} + p_1 \right) V(x_0, x^*) \right].
\end{equation}

We provide a set of example parameters for Algorithm 4.1 in the theorem below.

Theorem 4.2. Suppose that the parameters of Algorithm 4.1 are set to the ones described in Theorem 2.2. Applying Algorithm 4.1 to solve problem (4.1) we have

\[ Q(w_N, w) \leq \min \left\{ \frac{L}{N}, \lambda^{N-2} \right\} \left[ \frac{L}{2\sqrt{2}} U(z_0, z^*) + 4L V(x_0, x^*) \right]. \]

Specially, if $R$ is set to (2.17), then we have the following complexity results for Algorithm 4.1 to obtain an $\epsilon$-solution of problem (4.1) (here $O(1)$ is a constant that is independent of $N$, $\|A\|$ and $\epsilon$):

a) If $\mu = 0$, then the PDS algorithm can obtain an $\epsilon$-solution of problem (4.1) after at most $O(1) \left( \sqrt{LV(x_0, x^*)/\epsilon} \right)$ number of gradient evaluations and

\[ O(1) \left( \sqrt{LV(x_0, x^*)/\epsilon} + \|A\|/\epsilon \right) \] number of operator computations (involving $A$ and $A^T$).

b) If $\mu > 0$, then the PDS algorithm can obtain an $\epsilon$-solution of problem (4.1) after at most $O(1) \left( 1 + \sqrt{2L/\mu \log(LV(x_0, x^*)/\epsilon)} \right)$ number of gradient evaluations and $O(1) \left( 1 + \sqrt{2L/\mu \log(LV(x_0, x^*)/\epsilon)} + \|A\| (1 + 1/\sqrt{2}) \right)$ number of operator computations (involving $A$ and $A^T$).

The proof of the theorem below is similar to that of Theorem 2.2 and thus is skipped.

Theorem 5.1. Suppose that the parameters of Algorithm 4.1 are set to the ones described in Theorem 2.2. Applying Algorithm 4.1 to solve problem (4.1) we have

\[ Q(w_N, w) \leq \min \left\{ \frac{L}{N}, \lambda^{N-2} \right\} \left[ \frac{L}{2\sqrt{2}} U(z_0, z^*) + 4L V(x_0, x^*) \right]. \]

This manuscript is for review purposes only.
respectively. Letting \( \hat{w}_k := (\hat{x}_k, y_k, \hat{z}_k) \) we have

\[
\sum_{k=1}^{N} \beta_k Q(\hat{w}_k, w) + A + B \leq C + D, \quad \forall w := (x, y, z) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}
\]

where

\[
A := \sum_{k=1}^{N} \beta_k \left[ -\langle \hat{x}_k, y \rangle + \langle x, y_k \rangle + \langle v_k, \hat{x}_k - x \rangle - \hat{f}^*(v_k) + \hat{f}^*(y) \right]
\]

\[
B := \sum_{k=1}^{N} \frac{\beta_k}{T_k} \sum_{t=1}^{T_k} \left[ (A^T z_k^t - A^T z, x_k^t - \hat{u}_k^t) \right.
\]

\[
+ q_k^t U(z_k^{t-1}, z_k^t) + \eta_k^t V(x_k^{t-1}, x_k^t) \left.ight],
\]

\[
C := \sum_{k=1}^{N} \frac{\beta_k}{T_k} \sum_{t=1}^{T_k} \left[ q_k^t U(z_k^{t-1}, z_k^t) - q_k^t U(\hat{z}_k^t, z_k^t) \right],
\]

\[
D := \sum_{k=1}^{N} \frac{\beta_k}{T_k} \sum_{t=1}^{T_k} \left[ \eta_k^t V(x_k^{t-1}, x_k^t) - (\mu + \eta_k^t + p_k)V(x_k^t, x) + p_k V(x_k, x_k) \right].
\]

Proof. By the optimality conditions of (5.1) and (5.2) and definitions of \( U \) and \( V \) in (4.4) and (4.5) respectively, we have

\[
\langle h'(z_k^t) - A\hat{u}_k^t + q_k^t \xi' (z_k^t) - q_k^t \xi' (z_k^{t-1}), z_k^t - z \rangle \leq 0, \quad \forall z \in \mathcal{Z}, \text{ and}
\]

\[
\langle (\mu + \eta_k^t + p_k)\nu'(x_k^t) + v_k + A^T z_k^t - \eta_k^t \nu'(z_k^{t-1}) - p_k \nu'(x_k-1), x_k^t - x \rangle \leq 0, \quad \forall x \in \mathcal{X}.
\]

In view of the above results, the convexity of \( h \) and \( \nu \), and the definitions of \( U \) and \( V \) we obtain the following two relations: for any \( z \in \mathcal{Z} \) and \( x \in \mathcal{X} \),

\[
h(z_k^t) - h(z) + \langle -A\hat{u}_k^t, z_k^t - z \rangle + q_k^t U(z_k^{t-1}, z_k^t) + q_k^t U(z_k^t, z) \leq q_k^t U(z_k^{t-1}, z), \text{ and}
\]

\[
\langle v_k + A^T z_k^t, x_k^t - x \rangle + \mu \nu(x_k^t) - \mu \nu(x)
\]

\[
+ \eta_k^t V(x_k^{t-1}, x_k^t) + (\mu + \eta_k^t + p_k)V(x_k^t, x) + p_k V(x_k, x_k)
\]

\[
\leq \eta_k^t V(x_k^{t-1}, x) + p_k V(x_k, x_k).
\]

Summing up the two relations above, while noting that

\[
\langle -A\hat{u}_k^t, z_k^t - z \rangle + \langle v_k + A^T z_k^t, x_k^t - x \rangle
\]

\[
= \langle A^T z_k^t - A^T z, x_k^t - \hat{u}_k^t \rangle + \langle A^T z, x_k^t \rangle - \langle v_k + A^T z_k^t, x \rangle + \langle v_k, x_k^t \rangle,
\]

we have

\[
\langle A^T z_k^t - A^T z, x_k^t - \hat{u}_k^t \rangle + \langle A^T z, x_k^t \rangle - \langle v_k + A^T z_k^t, x \rangle + \langle v_k, x_k^t \rangle
\]

\[
+ h(z_k^t) - h(z) + q_k^t U(z_k^{t-1}, z_k^t) + q_k^t U(z_k^t, z)
\]

\[
+ \mu \nu(x_k^t) - \mu \nu(x) + \eta_k^t V(x_k^{t-1}, x_k^t) + (\mu + \eta_k^t + p_k)V(x_k^t, x) + p_k V(x_k, x_k)
\]

\[
\leq q_k^t U(z_k^{t-1}, z) + \eta_k^t V(x_k^{t-1}, x) + p_k V(x_k, x_k).
\]

Summing from \( t = 1, \ldots, T_k \) and noting the definitions of \( \hat{x}_k \) and \( \hat{z}_k \) and the convexity
of functions $h$ and $\nu$ we have

$$
T_k \left[ \langle A^T z, \hat{x}_k \rangle - \langle v_k + A^T \hat{z}_k, x \rangle + \langle v_k, \hat{x}_k \rangle + h(\hat{z}_k) - h(z) + \mu \nu(\hat{x}_k) - \mu \nu(x) \right]
$$

$$
+ \sum_{t=1}^{T_k} p_k \left[ V(x_{k-1}, x_k^t) + \langle A^T z_k^t - A^T x_{k-1}^t - u_k^t, + \nu_k(z_k^t, x_k^t) + \eta_k^t V(x_{k-1}^t, x_k^t) \right]
$$

$$
\leq \sum_{t=1}^{T_k} \left[ q_k^t U(z_{k-1}^t, z) - q_k^t U(z_k^t, z) \right].
$$

We conclude (5.3) by multiplying the above relation by $\beta_k / T_k$, noting the definition of the gap function $Q(\hat{w}_k, w)$ in (4.11) and summing the resulting relation from $k = 1, \ldots, N$.

**Lemma 5.2.** Suppose that $v_k = y_k$ where $y_k$ is defined by (4.9) and (4.10). If

$$
(5.7) \quad \tau_k = 0, \beta_k \tau_k \leq \beta_{k-1}(\tau_{k-1} + 1), \beta_{k-1} = \beta_k \lambda_k, \text{ and } L\lambda_k \leq p_k - 1 \tau_k, \forall k \geq 2,
$$

then we have

$$
A \geq -\beta_{N}(x_{N-1} - \hat{x}_N, y_N - y) + \frac{\beta_{N}p_{N}}{2} \|x_{N-1} - \hat{x}_N\|^2 + \beta_{N}(\tau_N + 1)W(y_N, y).
$$

**Proof.** By the optimality condition of (4.10) and the definition of $W$ in (4.6)

$$
\langle -\hat{x}_k + (1 + \tau_k)(\hat{f}^*)(y_k) - \tau_k(\hat{f}^*)(y_{k-1}), y_k - y \rangle \leq 0, \forall y \in \mathcal{Y},
$$

which together with the convexity of $\hat{f}^*$ and the definition of $W$ in (4.6) implies

$$
(5.8) \quad \langle -\hat{x}_k, y_k - y \rangle + \hat{f}^*(y_k) - \hat{f}^*(y_k) + \tau_k W(y_{k-1}, y_k) + (\tau_k + 1)W(y_k, y)
$$

$$
\leq \tau_k W(y_{k-1}, y), \forall y \in \mathcal{Y}.
$$

Combining the above relation with $A$ in (5.4) and noting that $v_k = y_k$ we have

$$
A \geq \sum_{k=1}^{N_k} \beta_k \left[ (\tau_k + 1)W(y_k, y) - \tau_k W(y_{k-1}, y) + \tau_k W(y_{k-1}, y_k)
$$

$$
- \langle \hat{x}_k, y \rangle + \langle y_k, \hat{x}_k \rangle + \langle -\hat{x}_k, y_k - y \rangle + \frac{p_k}{\tau_k} \sum_{t=1}^{T_k} V(x_{k-1}, x_k^t) \right]
$$

From the definition of $\hat{x}_k$ we can observe that

$$
- \langle \hat{x}_k, y \rangle + \langle y_k, \hat{x}_k \rangle + \langle -\hat{x}_k, y_k - y \rangle = -\langle x_{k-1} + \lambda_k(\hat{x}_{k-1} - x_{k-2}) - \hat{x}_k, y_k - y \rangle
$$

$$
= \lambda_k(x_{k-2} - \hat{x}_{k-1}, y_{k-1} - y) - \langle x_{k-1} - \hat{x}_k, y_k - y \rangle + \lambda_k(x_{k-2} - \hat{x}_{k-1}, y_k - y_{k-1}).
$$

Also, by the definition of $\hat{x}_k$ and the fact that $V$ is lower bounded in (4.7) we have

$$
\frac{1}{\tau_k} \sum_{t=1}^{T_k} p_k V(x_{k-1}, x_k^t) \geq p_k V(x_{k-1}, \hat{x}_k) \geq \frac{p_k}{\tau_k} \|x_{k-1} - \hat{x}_k\|^2.
$$
Applying the above two observations, the bound of $W$ in (4.8), the parameter assumption (5.7), and recalling that $x_{-1} = \tilde{x}_0$ in Algorithm 4.1 we have

\begin{align*}
A \geq & \sum_{k=1}^N [\beta_k (\tau_k + 1) W(y_k, y) - \beta_k \tau_k W(y_{k-1}, y) \\
& + \beta_k \lambda_k \langle x_{k-2} - \tilde{x}_{k-1}, y_{k-1} - y \rangle - \beta_k \langle x_{k-1} - \tilde{x}_k, y_k - y \rangle \\
& + \beta_k \lambda_k \langle x_{k-2} - \tilde{x}_{k-1}, y_k - y_{k-1} \rangle + \frac{\beta_k \alpha_k}{2 \lambda_k} \| x_{k-1} - \tilde{x}_k \|^2 + \frac{\beta_k \tau_k}{2L} \| y_{k-1} - y_k \|^2 ] \\
& \geq \beta_N (\tau_N + 1) W(y_N, y) - \beta_N \langle x_{N-1} - \tilde{x}_N, y_N - y \rangle + \frac{\beta_N \alpha_N}{2 \lambda_N} \| x_{N-1} - \tilde{x}_N \|^2 \\
& + \sum_{k=2}^N \beta_k \lambda_k \| x_{k-2} - \tilde{x}_{k-1} \| \| y_k - y_{k-1} \| \ast \\
& + \frac{\beta_k \alpha_k}{2 \lambda_k} \| x_{k-2} - \tilde{x}_{k-1} \|^2 + \frac{\beta_k \tau_k}{2L} \| y_{k-1} - y_k \|^2 ] \\
& \geq \beta_N (\tau_N + 1) W(y_N, y) - \beta_N \langle x_{N-1} - \tilde{x}_N, y_N - y \rangle + \frac{\beta_N \alpha_N}{2 \lambda_N} \| x_{N-1} - \tilde{x}_N \|^2,
\end{align*}

where the last inequality follows from the simple relation $a^2 + b^2 \geq 2ab$ and

\[(\beta_k \lambda_k)^2 - \beta_{k-1} \eta_k - (\beta_k \tau_k / \bar{L}) = (\beta_k^2 \lambda_k / \bar{L})(\bar{L} \sigma_k - p_{k-1} \tau_k) \leq 0.\]

There are cases when $v_k$ is not equal to $y_k$, for example, in Algorithm 3.2. The following lemma provide a tight bound for $A$ under this case.

**Lemma 5.3.** Suppose that $y_k$ is defined by (4.9) and (4.10) and $v_k = v_k(y_k, \xi_k)$ is an unbiased estimator of $y_k$ with respect to random variable $\xi_k$ such that $E \| v_k - y_k \|_s = 0$ and $E \| v_k - y_k \|_s^2 \leq \sigma_k^2$, where $\| \cdot \|_s$ is the dual norm of $\| \cdot \|$. If the parameters satisfy

\[
\tau_1 = 0, \beta_k \tau_k \leq \beta_k (\tau_k + 1), \beta_k - 1 = \beta_k \lambda_k, \text{ and } 2 \bar{L} \lambda_k \leq p_{k-1} \tau_k, \forall k \geq 2,
\]

then we have

\[
E[A] \geq E \left[ -\beta_N \langle x_{N-1} - \tilde{x}_N, y_N - y \rangle + \frac{\beta_N \alpha_N}{2 \lambda_N} \| x_{N-1} - \tilde{x}_N \|^2 - \sum_{k=1}^N \frac{\beta_k \sigma_k^2}{p_k} \right. \\
+ \beta_N (\tau_N + 1) W(y_N, y)
\]

**Proof.** In view of the optimality condition described previously in (5.8) and the definition of $A$ in (5.4) we have $A \geq A_1 + A_2$ where

\[
A_1 \defined \sum_{k=1}^N \beta_k \left[ -\beta_k \eta_k + \langle y_k, \tilde{x}_k \rangle + \langle -\tilde{x}_k, y_k - y \rangle + \frac{\beta_k \alpha_k}{2 \lambda_k} \sum_{t=1}^{T_k} V(x_k, x_k^t) \right. \\
+ \tau_k W(y_k - 1, y_k) + (\tau_k + 1) W(y_k, y) - \tau_k W(y_k, y) \right] \text{ and}
\]

\[
A_2 \defined \sum_{k=1}^N \beta_k \frac{1}{p_k} \sum_{t=1}^{T_k} \left[ \frac{p_k}{4} V(x_k, x_k^t) + \langle v_k - y_k, x_k^t - x_{k-1} \rangle + \langle v_k - y_k, x_k - x \rangle \right].
\]

Applying the same argument as in Lemma 5.2 to $A_1$, we have

\[
A_1 \geq -\beta_N \langle x_{N-1} - \tilde{x}_N, y_N - y \rangle + \frac{\beta_N \alpha_N}{4} \| x_{N-1} - \tilde{x}_N \|^2 + \beta_N (\tau_N + 1) W(y_N, y).
\]

To finish the proof it suffices to show that $E[A_2] \geq - \sum_{k=1}^N \beta_k \sigma_k^2 / p_k$. Noting from the definition of $v_k$ that $E_\xi_1, \ldots, \xi_{k-1} [\langle v_k - y_k, x_k - x \rangle] = 0$ and applying the bound of $V$ in (4.7) and Cauchy-Schwarz inequality to the description of $A_2$ we have

\[
E[A_2] \geq E \left[ \sum_{k=1}^N \beta_k \frac{1}{p_k} \sum_{t=1}^{T_k} \left( \frac{p_k}{4} \| x_{k-1} - x^t_k \| + \| v_k - y_k \| \| x^t_k - x_{k-1} \| \right) \right. \\
\geq \sum_{k=1}^N \beta_k \frac{1}{p_k} \left[ v_k - y_k \| \right] = - \sum_{k=1}^N \frac{\beta_k \sigma_k^2}{p_k}.
\]

This manuscript is for review purposes only.
We continue with the estimates of $B$, $C$, and $D$ in the following two lemmas.

**Lemma 5.4.** Suppose that

\begin{align}
(5.9) & \alpha_k^t = 1, \|A\|^2 \leq \eta_k^{-1}q_k, \forall t \geq 2, k \geq 1, \text{ and} \\
(5.10) & \beta_k T_k-1 \alpha_k^t = \beta_k A_k \|A\|^2 \leq \eta_k^{-1}q_k, \forall k \geq 2,
\end{align}

where $\|A\|$ is the norm induced by norms $\| \cdot \|$ and $| \cdot |$ introduced in (4.7). Then

$B \geq \frac{\beta_N}{T_k} \|A\| z_N - z \| x_{N} - x_{N-1}^{T_N} \| + \frac{\beta_N \eta_k^N}{T_k} \| x_{N}^{T_N} - x_{N}^{T_N} \|^2$.

**Proof.** By the definition of $a_{ik}^t$ in (2.6) and bounds of $U$ and $V$ in (4.7) we have

$\sum_{t=1}^{T_k} \left[ (A^T z_k^t - A^T z, x_k^t - u_k^t) + q_k U(z_k^{t-1}, z_k^t) + \eta_k V(x_k^{t-1}, x_k^t) \right]$

$\geq \sum_{t=1}^{T_k} \left[ -\alpha_k^t (A^T z_k^{t-1} - A^T z, x_k^{t-1} - x_k^{t-2}) + (A^T z_k^t - A^T z, x_k^t - x_k^{t-1}) + \frac{q_k}{2} \| z_k^t - z_k^{t-1} \|^2 + \frac{\eta_k}{2} \| x_k^t - x_k^{t-1} \|^2 - \alpha_k^t (A^T z_k^t - A^T z_k^{t-1}, x_k^t - x_k^{t-2}) \right]$

Also observe that $\alpha_k^t = 1$ for all $t \geq 2$ we have

$\sum_{t=1}^{T_k} \left[ -\alpha_k^t (A^T z_k^{t-1} - A^T z, x_k^{t-1} - x_k^{t-2}) + (A^T z_k^t - A^T z, x_k^t - x_k^{t-1}) \right]$

$= -\alpha_k^t (A^T z_k^t - A^T z, x_k^t - x_k^{t-1}) + (A^T z_k^t - A^T z, x_k^t - x_k^{t-1})$

In addition, in view of the assumptions (5.9) and (5.10) we have

$\sum_{t=1}^{T_k} \left[ -\alpha_k^t (A^T z_k^t - A^T z_k^{t-1}, x_k^t - x_k^{t-2}) \right]$

$= \frac{q_k}{2} \| z_k^t - z_k^{t-1} \|^2 + \frac{\eta_k}{2} \| x_k^t - x_k^{t-1} \|^2 - \alpha_k^t (A^T z_k^t - A^T z_k^{t-1}, x_k^t - x_k^{t-2})$

Applying the above three relations to the definition of $B$ in (5.5), recalling the definitions of $x_k^0, z_k^0, x_k^{-1}$ in Algorithm 4.1, and using the assumption (5.10) we have

$B \geq \sum_{k=1}^{N} \frac{\beta_k}{T_k} \left[ -\alpha_k^t (A^T z_k^0 - A^T z, x_k^0 - x_k^{t-1}) + (A^T z_k^0 - A^T z, x_k^0 + x_k^{t-1}) \right]$

$+ \frac{q_k}{2} \| z_k^0 - z_k^{t-1} \|^2 + \frac{\eta_k}{2} \| x_k^0 - x_k^{t-1} \|^2 - \alpha_k^t (A^T z_k^0 - A^T z_k^{t-1})$

$=: \frac{\beta_N}{T_N} (A^T z_N - A^T z, x_N - x_N^{T_N})$

$+ \frac{\beta_N q_k}{T_N} \| z_k^0 - z_k^{t-1} \|^2 + \frac{\beta_N \eta_k^N}{T_N} \| x_k^0 - x_k^{t-1} \|^2$

$+ \sum_{k=2}^{N} \left[ \beta_k^4 \frac{q_k}{T_k} \| z_k^0 - z_k^{t-1} \|^2 + \frac{\beta_k \eta_k^N}{T_N} \| x_k^0 - x_k^{t-1} \|^2 - \frac{\beta_k^2 \eta_k^N}{T_N} \| A \| \| z_k^0 - z_k^{t-1} \|^2 \right]$

$\geq \frac{\beta_N}{T_N} \| A \| z_N - z \| x_N - x_N^{T_N} \| + \frac{\beta_N \eta_k^N}{T_N} \| x_N^{T_N} - x_N^{T_N} \|^2$.
Here in the last inequality we use the following result from the assumption (5.10):
\[
(\beta_k \alpha_k^1 \|A\|/T_k)^2 - (\beta_k \eta_k^1 / T_k)(\beta_{k-1} \eta_{k-1}^1 / T_{k-1})
= (\beta_k / T_k)^2 \alpha_k^1 (\alpha_k^1 \|A\|^2 - \eta_k^1 \eta_{k-1}^1) \geq 0.
\]

**Lemma 5.5.** If
\[(5.11) \quad q_k^t \leq q_k^{t-1}, \forall t \geq 2, k \geq 1 \text{ and } \beta_k T_{k-1} q_k^1 \leq \beta_{k-1} T_k q_{k-1}^1, \forall k \geq 2,
\]
then \( C \) in (5.6) can be bounded by
\[
C \leq \frac{\beta_q}{T_k^q} U(z_0, z) - \frac{\beta_N T_N}{T^q_N} U(z_N, z).
\]

Also, if
\[(5.12) \quad \eta_k^1 \leq \mu + \eta_k^{t-1} + p_k, \forall t \geq 2, k \geq 1 \text{ and }
\beta_k T_{k-1} (\eta_k^1 + p_k T_k) \leq \beta_{k-1} T_k (\mu + \eta_{k-1}^1 + p_{k-1}), \forall k \geq 2,
\]
then \( D \) in (5.6) can be bounded by
\[
D \leq \frac{\beta_1}{T_1} (\eta_1^1 + p_1 T_1) V(x_0, x) - \frac{\beta_N T_N}{T^N} (\mu + \eta_N^N + p_N) V(x_N, x).
\]

**Proof.** In view of (5.11) and the fact that \( z_k^0 = z_{k-1} \) and \( z_k = z_{k-1}^T \), we have:
\[
C \leq \sum_{k=1}^N \left[ \frac{\beta_q}{T_k^q} U(z_{k-1}, z) - \frac{\beta_q}{T_k^q} U(z_k, z) \right] \leq \frac{\beta_q}{T_k^q} U(z_0, z) - \frac{\beta_N T_N}{T^N} U(z_N, z).
\]
Similarly, by (5.12) and noting that \( x_k^0 = x_{k-1} \) and \( x_k = x_{k-1}^T \) we have
\[
D \leq \sum_{k=1}^N \left[ \frac{\beta}{T_k} (\eta_k^1 + p_k T_k) V(x_{k-1}, x) - \frac{\beta}{T_k} (\mu + \eta_k^1 + p_k) V(x_k, x) \right]
= \sum_{k=1}^N \left[ \frac{\beta}{T_k} (\eta_k^1 + p_k T_k) V(x_{k-1}, x) - \frac{\beta}{T_k} (\mu + \eta_{k-1}^1 + p_{k-1}) V(x_k, x) \right]
\leq \frac{\beta_1}{T_1} (\eta_1^1 + p_1 T_1) V(x_0, x) - \frac{\beta_N T_N}{T^N} (\mu + \eta_N^N + p_N) V(x_N, x).
\]

With Lemmas 5.2, 5.4, and 5.5 and Proposition 5.1, we are ready to prove Proposition 4.1.

**Proof of Proposition 4.1.** Note that the conditions (2.10) and (2.11) for parameters in Algorithm 4.1 are exactly the parameter assumptions (5.7), (5.9) – (5.12), in Lemmas 5.2, 5.4, and 5.5. In view of the definition of \( \bar{w}_N \), the convexity of \( Q \), Proposition 5.1 and Lemmas 5.2, 5.4, and 5.5, we have
\[
\left( \sum_{k=1}^N \beta_k \right) Q(\bar{w}_N, w) \leq \sum_{k=1}^N \beta_k Q(\bar{w}_k, w)
\leq \beta_N (x_{N-1} - \bar{x}_N, y_{N-1} - \bar{y}_N + \frac{\beta_N}{T_N} \|x_{N-1} - \bar{x}_N\|^2 - \beta_N (\tau_N + 1) W(y_N, y)
- \frac{\beta_N}{T_N} \|A\| \|z_N - z\| ||x_N - x_{N-1}^T||^2 - \frac{\beta_N T_N}{2 T_N} \|x_N^T - x_{N-1}^T\|^2
+ \frac{\beta_q}{T_k^q} U(z_0, z) - \frac{\beta_N T_N}{T^N} U(z_N, z)
+ \frac{\beta_1}{T_1} (\eta_1^1 + p_1 T_1) V(x_0, x) - \frac{\beta_N T_N}{T^N} (\mu + \eta_N^N + p_N) V(x_N, x).
\]
We conclude (4.12) from the above inequality by noting from (2.12), (4.7), and (4.8)
that

\[ \beta_N(x_{n-1} - \hat{x}_N, y_{n-1} - y) - \frac{\beta_N}{2} \| x_{n-1} - \hat{x}_N \|^2 - \beta_N(\tau_N + 1)W(y_N, y) \leq 2 \beta_N \| y_N - y \|^2 - \frac{\beta_N(\tau_N + 1)}{2L} \| y_N - y \|^2 \leq 0, \]

and

\[ - \frac{\beta_N}{T_N} \| A \| \| z_N - z \| \| x_N - x_N^{T_N-1} \| - \frac{\beta_N}{T_N} \| x_N^{T_N-1} - x_N \| \| z_N - z \| \leq 0. \]

In the special case when \( Z \) is a vector space with Euclidean norm \( \| \cdot \| = \| \cdot \|_2 \), the
prox-function \( U(\cdot, \cdot) = \| \cdot - \cdot \|^2/2 \), and \( h(z) = (b, z) \), let us choose any saddle point
\( w^* = (x^*, y^*, z^*) \) that solves the SPP (4.2) and denote

\[ \overline{w}_N := (x^*, y^*, z^*_N) \text{ where } z^*_N := \frac{\| z^* \|^2 + 1}{\| A \| - b_2} (A \overline{w}_N - b). \]

Note that in such special case the SPP (4.2) reduces to the linear constrained opti-
mization (4.3), in which \( x^* \) is an optimal solution, \( y^* = \nabla \hat{f}(x^*) \), and \( z^* \) is the
optimal Lagrange multiplier associated with the equality constraints. Choosing \( w = (x^*, \nabla \hat{f}(\overline{w}_N), 0) \) and noting that \( Ax^* = b, \hat{f}(\overline{w}_N) = (\overline{w}_N, \nabla \hat{f}(\overline{w}_N)) = \hat{f}(\nabla \hat{f}(\overline{w}_N)) \), and
\[ f(x^*) \geq (x^*, \overline{y}_N) - \hat{f}(\overline{y}_N) \]
Combining the above result with (4.12) we obtain (4.13) immediately. To prove (4.14)
let us study the gap functions involving \( \overline{w}_N, w^* \), and \( \overline{w}_N \).

Observe that

\[ Q(\overline{w}_N, w^*) = [\mu(x_N) + (x_N, y_N + A^Tz_N) - \hat{f}(y_N) - (b, z_N)] \]

\[ - [\mu(x^*) + (x^*, \overline{y}_N + A^T \overline{z}_N) - \hat{f}(\overline{y}_N) - (b, \overline{z}_N)] \]

\[ \leq [\mu(x_N) + (x_N, y_N) - \hat{f}(y_N) - \mu(x^*) - (x^*, \overline{y}_N) + \hat{f}(\overline{y}_N)] \]

\[ + [\| A \| - b_2 \| z^* \|_2. \]

Here also observe that \( Q(\overline{w}_N, w^*) \geq 0 \) since \( w^* \) is a saddle point. By these observations
and the definition of \( z^*_N \) in (5.13) we have

\[ Q(\overline{w}_N, \overline{w}_N) = [\mu(x_N) + (x_N, y_N + A^Tz_N) - \hat{f}(y_N) - (b, \overline{z}_N)] \]

\[ - [\mu(x^*) + (x^*, \overline{y}_N + A^T \overline{z}_N) - \hat{f}(\overline{y}_N) - (b, \overline{z}_N)] \]

\[ = [\mu(x_N) + (x_N, y_N) - \hat{f}(y_N) - \mu(x^*) - (x^*, \overline{y}_N) + \hat{f}(\overline{y}_N)] \]

\[ + [\| z^* \|^2 + 1] \| A \| - b_2 \| z^* \|_2 \geq \| A \| - b_2 \| z^* \|_2. \]

The above result, together with the bound of \( Q \) in (4.12), and the facts that \( y^* = \nabla \hat{f}(x^*) \), \( z_0 = 0 \), and \( \| z_0 - \overline{z}_N \|^2 = \| \overline{z}_N \|^2 = (\| z^* \|^2 + 1)^2 \), yield the estimate (4.14).

Note that Proposition 2.1 is in fact a special case of Proposition 4.1. Indeed, since
Algorithm 2.1 is the agent view of Algorithm 2.2, which is a special case of Algorithm
4.1 (when \( h(z) \equiv 0 \), \( Z = \mathbb{R}^{md} \), \( z_0 = 0 \), \( \mathcal{A} \), \( f \), and \( \mathcal{A} \) are defined in (1.2), and all norms are Euclidean). Therefore, the results in Proposition 2.1 are immediately implied from Proposition 4.1.

Applying the results in Lemmas 5.3, 5.4, and 5.5 to Proposition 5.1 we can also prove Proposition 3.1. The proof is similar to that of Proposition 4.1 above.

**Proof of Proposition 3.1.** Denoting \( \overline{\omega}_N : = \left( \sum_{k=1}^N \beta_k \right)^{-1} \left( \sum_{k=1}^N \beta_k \bar{w}_k \right) \), let us study the gap function \( Q \) defined in (4.11) in which \( h(\cdot) \equiv 0 \). Noting the convexity of \( Q \), applying Proposition 5.1 (in which \( U(\cdot, \cdot) = \| \cdot - \cdot \|_2^2 / 2 \), \( V(\cdot, \cdot) = \| \cdot - \cdot \|_2^2 / 2 \), and all norms are Euclidean) and the bounds of \( A \) through \( D \) in Lemmas 5.3 (with \( \sigma_k = \sigma / \sqrt{c_k} \) as specified in Algorithm 3.2), 5.4, and 5.5 (in which \( z_0 = 0 \) from the description of Algorithm 3.1), we have

\[
\sum_{k=1}^N \beta_k \mathbb{E}[Q(\overline{\omega}_N, w)] \leq \mathbb{E} \left[ \beta_N (x_{N-1} - \hat{x}_N, y_N - y) - \frac{\beta_N \sigma^2}{4} \| x_{N-1} - \hat{x}_N \|_2^2 + \sum_{k=1}^N \frac{\beta_k \sigma^2}{p_k c_k} \right]
\]
\[
- \beta_N (\tau_N + 1) W(y_N, y)
\]
\[
- \frac{\beta_N}{T_N} \| A \| \| z_N - z \|_2 \| x_N - x \|_2^2 - T_N^{\tau_N} \| x_N^T - x_N \|_2^2
\]
\[
+ \frac{\beta_1 T_1 \| z \|_2^2 - \frac{\beta_N \eta T_N}{2 T_N} \| x_N - x \|_2^2}
\]
\[
+ \frac{\beta_1 T_1 (\eta_1 + p_1 T_1)}{2 T_1} \| x_0 - x \|_2^2 - \frac{\beta_N}{2 T_N} (\mu + \eta T_N + p N) \| x_N - x \|_2^2
\]
\[
\sum_{k=1}^N \beta_k \mathbb{E}[Q(\overline{\omega}_N, w)] \leq \frac{\beta_N \sigma^2}{4} \| z \|_2^2 + \frac{\beta_1 T_1 (\eta_1 + p_1 T_1)}{2 T_1} \| x_0 - x \|_2^2 + \sum_{k=1}^N \frac{\beta_k \sigma^2}{p_k c_k}.
\]

Here noting from (2.12), (3.3), and (4.8) we can simplify the above to

\[
\sum_{k=1}^N \beta_k \mathbb{E}[Q(\overline{\omega}_N, w)] \leq \frac{\beta_N \sigma^2}{4} \| z \|_2^2 + \frac{\beta_1 T_1 (\eta_1 + p_1 T_1)}{2 T_1} \| x_0 - x \|_2^2 + \sum_{k=1}^N \frac{\beta_k \sigma^2}{p_k c_k}.
\]

The remainder of the proof is similar to that of Proposition 4.1 and hence is skipped. □

6. **Numerical experiments.** In this section, we demonstrate the advantages of our proposed PDS method through some preliminary numerical experiments and compare it with the state-of-the-art communication-efficient decentralized method, namely the decentralized communication sliding (DCS) method proposed in [14]. We consider a decentralized convex smooth optimization problem of unregularized logistic regression model over a dataset that is not linearly separable. In order to guarantee a fair comparison, all the implementation details described below are the same as suggested in [14]. In the linear constrained problem formulation (1.2) of the decentralized problem we set \( \mathcal{A} = \mathcal{L} \times I_d \) from the graph Laplacian. For the underlying communication network, we use the Erhos-Renyi algorithm to generate three connected graphs with \( m = 100 \) nodes as shown in Figure 1. Note that nodes with different degrees are drawn in different colors, in particular, \( G_1 \) has a maximum degree of \( d_{\text{max}} = 4 \), \( G_2 \) has \( d_{\text{max}} = 9 \) and \( G_3 \) has \( d_{\text{max}} = 20 \). We also use the same dataset as in [14], a real dataset “ijcnn1” from LIBSVM\(^1\) and choose 20,000 samples from this dataset as our problem instance data to train the decentralized logistic regression model. Since we have \( m = 100 \) nodes (or agents) in the decentralized network, we evenly split these 20,000 samples over 100 nodes, and hence each network node has 200 samples.

\(^1\)This real dataset can be downloaded from https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.
All agents in the network start with the same initial point $x_0 = 0$ and $y_0 = z_0 = 0$.

We then compare the performances of the proposed PDS/SPDS algorithms with DCS (SDCS for stochastic case) proposed in [14] for solving (1.2) with $f_i$ being the logistic loss function. We show the required number of communication rounds and (stochastic) gradient evaluations for the algorithms in comparison to obtain the same target loss. In all problem instances, we use $\| \cdot \|_2$ norm in both primal and dual spaces. All the experiments are programmed in Matlab 2020a and run on Clemson University’s Palmetto high-performance computing clusters (with 6 Intel Xeon Gold CLX 6248R CPUs for a total of 168 cores).

In the deterministic case, for the DCS algorithm we use the parameter setting as suggested in [14] including using a dynamic inner iteration limit as $\min\{10^k, 5,000\}$ for possible performance improvement. For the PDS algorithm we use the parameter setting as suggested in Theorem 2.2 for solving smooth and convex problems ($\mu = 0$). In particular, when setting the inner iteration limit $T_k$ we choose $R = 1/(2\sqrt{2})$ in (2.13). Note that we also try tuning the estimation of Lipschitz constant $\tilde{L}$ for the best performance of the PDS algorithm.

### Table 1
Comparison of the DCS and PDS algorithms in terms of reaching the same target loss

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Graph</th>
<th>Target Loss</th>
<th>Achieved $|Ax|$</th>
<th>Com. rounds</th>
<th>Gradient evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCS</td>
<td>$G_1(d_{max} = 4)$</td>
<td>70</td>
<td>$4.94e-04$</td>
<td>3,110</td>
<td>6,527,500</td>
</tr>
<tr>
<td>PDS</td>
<td>$G_1(d_{max} = 4)$</td>
<td>70</td>
<td>$8.95e-02$</td>
<td>154</td>
<td>24</td>
</tr>
<tr>
<td>DCS</td>
<td>$G_2(d_{max} = 9)$</td>
<td>70</td>
<td>$3.22e-14$</td>
<td>5,624</td>
<td>12,812,500</td>
</tr>
<tr>
<td>PDS</td>
<td>$G_2(d_{max} = 9)$</td>
<td>70</td>
<td>$2.08e-01$</td>
<td>274</td>
<td>25</td>
</tr>
<tr>
<td>DCS</td>
<td>$G_3(d_{max} = 20)$</td>
<td>70</td>
<td>NA $^3$</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>PDS</td>
<td>$G_3(d_{max} = 20)$</td>
<td>70</td>
<td>$1.84e-02$</td>
<td>468</td>
<td>24</td>
</tr>
<tr>
<td>DCS</td>
<td>$G_1(d_{max} = 4)$</td>
<td>60</td>
<td>NA $^3$</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>PDS</td>
<td>$G_1(d_{max} = 4)$</td>
<td>60</td>
<td>$3.60e-01$</td>
<td>236</td>
<td>60</td>
</tr>
<tr>
<td>DCS</td>
<td>$G_2(d_{max} = 9)$</td>
<td>60</td>
<td>NA $^3$</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>PDS</td>
<td>$G_2(d_{max} = 9)$</td>
<td>60</td>
<td>$9.15e-01$</td>
<td>340</td>
<td>58</td>
</tr>
<tr>
<td>DCS</td>
<td>$G_3(d_{max} = 20)$</td>
<td>60</td>
<td>NA $^3$</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>PDS</td>
<td>$G_3(d_{max} = 20)$</td>
<td>60</td>
<td>$2.09e-01$</td>
<td>564</td>
<td>54</td>
</tr>
</tbody>
</table>

$^3$We use “NA” for DCS experiments running more than 8,000 communication rounds but not achieving the target losses.
Table 1 shows the results we obtained from the experiments of solving deterministic logistic regression problems. From the table, to reach the same target loss our proposing PDS algorithm requires less rounds of communication and much less gradient evaluations than the DCS algorithm in [14]. In particular, the number of gradient evaluations barely changes as the maximum degree of the graph increases, which matches our theoretic results on graph topology invariant gradient complexity bounds. Moreover, we can see that both algorithms achieve reasonable feasibility residue, which is measured by $\|Ax\|_2$. It needs to be pointed out that the DCS algorithm achieves a smaller feasibility residue but a much higher loss, while the PDS algorithm spends more efforts in reducing the loss value and maintaining an acceptable feasibility residue. Therefore, we can conclude that the PDS algorithm maintains a better trade-off between loss and feasibility residue than the DCS algorithm.

We also consider the stochastic problem in which the algorithms can only access the unbiased stochastic samples of the gradients. For the SDCS algorithm, we use the parameter setting as suggested in [14] (see Theorem 4 within) except using a dynamic inner iteration limit as $\min\{10k, 5, 000\}$ for possible performance improvement. For the proposed SPDS algorithm, we use the parameter setting as suggested in Theorem 3.2 for solving smooth and convex stochastic problems with $R = 1$ and $c = 1/4$ in (3.4). We set $N$ to be the smallest iteration limit where $N \mod 10 = 0$, and hence $N = 30$ for the cases when we set target loss as 70 and $N = 100$ when target loss is 60.

Table 2 shows the results we obtained from the experiments of solving logistic regression problems using stochastic samples of gradients. Observe from the table that the number of required stochastic gradient samples for the SPDS algorithm do not change as the maximum degree of the graph increases, which matches our theoretic result that the sampling complexity of the SPDS algorithm is graph topology invariant for stochastic problems. The SPDS algorithm can achieve target losses within reasonable amount of CPU time and algorithm iterations, while the SDCS algorithm can not achieve any of the target loss within the required communication rounds. Therefore,

$$\text{Table 2}$$

Comparison of the SDCS and SPDS algorithms in terms of reaching the same target loss

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Graph</th>
<th>Target Loss</th>
<th>Achieved $|Ax|$</th>
<th>Com. rounds</th>
<th>Stochastic grads.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDCS</td>
<td>$G_1(d_{max} = 4)$</td>
<td>70</td>
<td>NA $^3$</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>SPDS</td>
<td>$G_1(d_{max} = 4)$</td>
<td>70</td>
<td>$6.10e-04$</td>
<td>428</td>
<td>431</td>
</tr>
<tr>
<td>SDCS</td>
<td>$G_2(d_{max} = 9)$</td>
<td>70</td>
<td>NA $^3$</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>SPDS</td>
<td>$G_2(d_{max} = 9)$</td>
<td>70</td>
<td>$3.77e-04$</td>
<td>740</td>
<td>431</td>
</tr>
<tr>
<td>SDCS</td>
<td>$G_3(d_{max} = 20)$</td>
<td>70</td>
<td>NA $^3$</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>SPDS</td>
<td>$G_3(d_{max} = 20)$</td>
<td>70</td>
<td>$6.12e-05$</td>
<td>1,418</td>
<td>431</td>
</tr>
<tr>
<td>SDCS</td>
<td>$G_1(d_{max} = 4)$</td>
<td>60</td>
<td>NA $^3$</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>SPDS</td>
<td>$G_1(d_{max} = 4)$</td>
<td>60</td>
<td>$9.68e-04$</td>
<td>506</td>
<td>1,434</td>
</tr>
<tr>
<td>SDCS</td>
<td>$G_2(d_{max} = 9)$</td>
<td>60</td>
<td>NA $^3$</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>SPDS</td>
<td>$G_2(d_{max} = 9)$</td>
<td>60</td>
<td>$8.87e-04$</td>
<td>816</td>
<td>1,433</td>
</tr>
<tr>
<td>SDCS</td>
<td>$G_3(d_{max} = 20)$</td>
<td>60</td>
<td>NA $^3$</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>SPDS</td>
<td>$G_3(d_{max} = 20)$</td>
<td>60</td>
<td>$1.18e-04$</td>
<td>1,496</td>
<td>1,433</td>
</tr>
</tbody>
</table>

Table 2 shows the results we obtained from the experiments of solving logistic regression problems using stochastic samples of gradients. Observe from the table that the number of required stochastic gradient samples for the SPDS algorithm do not change as the maximum degree of the graph increases, which matches our theoretic result that the sampling complexity of the SPDS algorithm is graph topology invariant for stochastic problems. The SPDS algorithm can achieve target losses within reasonable amount of CPU time and algorithm iterations, while the SDCS algorithm can not achieve any of the target loss within the required communication rounds. Therefore,

$^3$We use “NA” for SDCS experiments running more than 4,000 communication rounds but not achieving the target losses.
we can conclude that the proposed SPDS algorithm outperforms the SDCS algorithm in [14] in terms of both sampling and communication complexities.

7. Concluding remarks. In this paper, we present a new class of algorithms for solving a class of decentralized multi-agent optimization problem. Our proposed primal dual sliding (PDS) algorithm is able to compute an approximate solution to the general convex smooth deterministic problem with \( O\left(\sqrt{L/\varepsilon} + \|A\|/\varepsilon\right) \) communication rounds, which matches the current state-of-the-art. However, the number of gradient evaluations required by the PDS algorithm is improved to \( O\left(\sqrt{L/\varepsilon}\right) \) and is invariant with respect to the graph topology. To the best of our knowledge, this is the only decentralized algorithm whose gradient complexity is graph topology invariant. We also propose a stochastic primal dual sliding (SPDS) algorithm that is able to compute an approximate solution to the general convex smooth stochastic problem with \( O\left(\sqrt{L/\varepsilon + \sigma^2/\varepsilon^2}\right) \) sampling complexity, which is also the only algorithm in the literature that has graph topology invariant sampling complexity. Similar convergence results of the PDS and SPDS algorithms are also developed for strongly convex smooth problems.

REFERENCES


This manuscript is for review purposes only.