Communication-Efficient Decentralized Gradient Descent with State-Dependent Compression

Xin Zhang[†] Jia Liu[‡] Zhengyuan Zhu[†] Elizabeth S. Bentley[*]
[†]Department of Statistics, Iowa State University
[‡]Department of Computer Science, Iowa State University
[*]Air Force Research Laboratory, Information Directorate

***Abstract*—Decentralized optimization over networks has attracted a flurry of research in recent years. To address the decentralized optimization problems, one of the most well-known approaches is the distributed gradient descent method (DGD) ([1]). However, as many distributed algorithm, DGD also meets the communication bottleneck. To overcome this limitation, in this work, we propose the Differential Compression Distributed Gradient Descent (DC-DGD) algorithm based on the state-dependent compression, which is a variance reduction method. Moreover, a novel family of the hybrid operators is proposed for maximally reducing the communication load and we provide a greedy algorithm to automatically determine the optimal hybrid operator. Lastly, we use the throughout experiments to validate our theoretical results. It is shown that our method is the most communication-efficient, compared with several state of the art communication-efficient DGD-based algorithm.**

I. Introduction

In this paper, we focus on designing and analyzing the decentralized algorithm to optimize the global objective function in a connected network:

$$\min_{x\in\mathbb{R}^d} f(x) = \min_{x\in\mathbb{R}^d} \sum_{i=1}^{n} f_i(x), \tag{1}$$

where $f_i(x)$ is the local objective function in the node i, n is the number of the nodes in the network and $x \in \mathbb{R}^d$ is the d-dimentional global parameter. The above decentralized optimization problem is first originated in the control community ([2]–[4]), but also attracts a flurry of research research in the field of statistical learning and deep machine learning ([5]–[8]): For example, multiple agents from a connected network are involved to learn a common/consensual policy ([9]). However, each agent can only access to its local objective function based on the locally observed data. To reach the optimal policy, it is necessary to design a decentralized algorithm for the agents to collaboratively solve the global objective function.

The decentralized gradient descent (DGD) algotihm [1], [10], [11] is one of the most effective methods to address the above decentralized optimization problem. In DGD, the problem (1) can be relaxed as:

$$\min \sum_{i=1}^{n} f_i(x_i) \quad s.t.\ (\mathbf{W}\otimes \mathbf{I}_d)\mathbf{x} = \mathbf{x}, \tag{2}$$

where $\mathbf{x} \triangleq [x_1^\top, \ldots, x_n^\top]^\top$ with $x_i \in \mathbb{R}^d$ as the local copy of x at node i, $\mathbf{W}$ is the concensus matrix based on the network topoloty, $\mathbf{I}_d$ denotes the d-dimensional identity matrix, and the operator $\otimes$ denotes the Kronecker product. The concensus matrix $\mathbf{W} \in \mathbb{R}^{n\times n}$ in (2) is commonly assumed to be symmetric and doubly stochastic with $[\mathbf{W}]_{ij} > 0$ if the node i and j are connected in the network and $[\mathbf{W}]_{ij} = 0$ otherwise. It's commonly assumed that the eigenvalues of $\mathbf{W}$ satisfy $1 = \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n > -1$. At each iteration of DGD, the node i updates the local copy according to:

$$x_{i,k+1} = \underbrace{\sum_{j\in\mathcal{N}_i} [\mathbf{W}]_{ij} x_{j,k}}_{\text{Consensus step}} - \underbrace{\alpha_k \nabla f_i(x_{i,k})}_{\text{Gradient step}}. \tag{3}$$

The update (3) consists of a consensus step and a local gradient step, which can be easily implemented in a network with one round communication.

Though its simple implementation and fast convergence, it is recently noticed that the DGD algorithm also meets the communication bottleneck as many other distributed algorithms ([12]–[16]): The communication of the local copies between neighboring nodes is costly or sometimes even prohibitive. For instance, consider the image regression problem ([17]) on the typical anatomical magnetic resonance image (MRI) with size 256-by-256-by-256 ([18]). The dimention of the regression parameters (i.e. the length of the local copy) is $256^3 = 16,777,216$ and the corresponding cost at one iteration in DGD is over 60MB (under 32 bit float). This huge amount of message will be problematic in a wireless sensor network or a mobile phone network. To reduce the communication burden, we consider to improve the DGD algorithm with the unbiased compression method [19]–[21]. Instead of the full message, the compressed low precise message is communicated, and hence the cost will be dramatically reduced without undermining the algorithm covergence.

Unfortunately, designing an effcient DGD-based compression algorithm is non-trivial as it involves two inter-dependent key components: 1) an optimal unbiased compression strategy with the balance of the noise variance and the representation cost; 2) a proper algorithm structure to control the corresponding compression noise. In the existing literatures, serveral unbiased compression strategies have been proposed, such as the quantizer ([21]), the ternary operator ([19]) and the sparsifier ([20]). Nevertheless, most of these works focused on the distributed algorithms under the master/slave

architecture, which are essentially different from the decentralized algorithms in the network. Additionaly, to guarantee the algorithm convergence, many work simply adopted a bounded variance assumption, without studying properities of these compression strategies in-depth. On the other hand, there exist a few works in which DGD-based compression algorithms were designed and analyzed ([12]–[14]). However, the algorithm structures in these work are not the optimal: many of them introduce additional parameters to control the compression noise, which in turn cause the slow convergence e.g. QDGD ([12]) or memory overflow risk e.g. ADC-DGD ([14]) and ECD-PSGD ([13]). In light of these limitations, we consider the following fundamental question: Could we develope a DGD-based compression algorithm without extra parameters but taking the lowest communication cost? In this paper, we answer the above question affirmatively. In specific, we propose a DGD-based compression algorithm for solving the network optimizsation problem (1), named Differential Compression Decentralized Gradient Descent, and a family of hybrid sparsification operators to gain maximal communication saving. Collectively, the main contributions of this paper are highlighted as follows:

- We develope the Differential Compression Decentralized Gradient Descent (DC-DGD) based on the state-dependent compression operator. Instead of the local copies, our algritghm has the nodes to transmit the compressed differences between two sucessive iterates. Thanks to the state-dependent compression operator, the variace of the compression noise will shrink to zero so that no extra parameters are needed for the variance control. We show that our DC-DGD enjoys the same covergence performence as the DGD.
- We propose a family of hybrid sparsification operators, which combine the structures of the sparsifer and the ternary operator. Contrast with the two existing operators, our compression operator has the controllable variance and less communication cost. Also we propose a greedy algorithm to find the best hybrid operator in each iteration to gain the maximal communication saving.

The rest of the paper is organized as follows. In Section II, we firtst review related works on the compression strategies and DGD-based compression algorithm. In Section III, we first our DC-DGD algorithm and then analyze its convergence gaurantees. In Section IV, we developed a family of hybrid operators and a greedy algorithm is proposed to choose the optimal hybrid operator. Numerical results are provided in Section V and in Section VI we provide some concluding remarks and future work.

II. Related Works

In this section, we first provide a short discussion on the state of the art unbiased stochastic compression method. We then focus on the recent advances of communication-efficient DGD-based algorithms, in which the compression idea is utilized.

A. Unbiased Stochastic Compression Method

Among various compression methods ([16], [19]–[22]) for overcomiong the communication hurdle in the distributed algorithms, one of the most effective methods is the unbiased stochastic compression method, with which the compressed output is a random variable/vector with the expectation as the input. Based on the output variance, the unbiased stochastic compression method can be divided into two main classes:

Definition 1 (Stochastic Unbiased Compression Method)**.** *A compression operator $C(\cdot)$ is stochastic unbiased if it satisfies $C(z) = z + \epsilon_z$, with $\mathbb{E}[\epsilon_z] = 0$ where ϵ_z is the compression noise. Furthermore,*

- *if $\mathbb{E}[\epsilon_z^2] \leq \sigma^2$, $\forall$ z and a constant σ, the operator is State-Independent Compression Operator;*
- *if $\mathbb{E}[\epsilon_z^2] \leq \eta^2\|z\|^2$, $\forall$ z and a constant η, it is State-Dependent Compression Operator.*

Note that a difference between the two classes is that the variance of the compression noise is bounded proportionally to the inputs magnitude in state-dependent compression operator while the upper bound for the noise variance is fixed for state-independent compression operator.

In the following, we give a few examples of the stochastic unbiased compression operators:

Example 1 (The Randomly Rounding Operater [21])**.** *For $\mathbf{z} = (z_1, \cdots, z_d)^\top \in \mathbb{R}^d$, the k-th element of $[C(\mathbf{z})]_k$ is:*

$$[C(\mathbf{z})]_i = \begin{cases} \lfloor z_i \rfloor + 1, \textit{with probability } (1 - p_i), \\ \lfloor z_i \rfloor, \textit{with probability } p_i. \end{cases}$$

where $\lfloor z \rfloor$ presents the largest integer smaller than z and the probability $p_i = z_i - \lfloor z_i \rfloor$. It's state-independent and the compression noise variance is bounded by $d/4$.

Example 2 (The Sparsifier [20])**.** *For $\mathbf{z} = (z_1, \cdots, z_d)^\top \in \mathbb{R}^d$, $C(\mathbf{z})$ is a sparse vector with the i-th element $[C(\mathbf{z})]_i$ following the Bernoulli distribution:*

$$\begin{cases} \Pr([C(\mathbf{z})]_i = \frac{\mathbf{z}_i}{p}) = p, \\ \Pr([C(\mathbf{z})]_i = 0) = 1 - p, \end{cases}$$

where $p \in (0, 1]$ is a probability. It's state-dependent and the compression noise variance is $(1/p - 1)\|\mathbf{z}\|^2$.

Example 3 (The Ternary Operator [19])**.** *For $\mathbf{z} = (z_1, \cdots, z_d)^\top \in \mathbb{R}^d$, $C(\mathbf{z}) = \|\mathbf{z}\|_\infty sign(\mathbf{z}) \circ b_\mathbf{z}$, where $\circ$ is the Hadamard product and $b_\mathbf{z}$ is a random vector with the i-th element $[b_\mathbf{z}]_i$ following the Bernoulli distribution:*

$$\begin{cases} \Pr([b_\mathbf{z}]_i = 1) = |z_i|/\|\mathbf{z}\|_\infty, \\ \Pr([b_\mathbf{z}]_i = 0) = 1 - |z_i|/\|\mathbf{z}\|_\infty. \end{cases}$$

It's state-dependent and the noise variance is $\sum_{i=1}^d |z_i|(\|\mathbf{z}\|_\infty - |z_i|)$, which is strictly bounded by $d\|\mathbf{z}\|^2$

In this work, we focus on the state-dependent compression operator and show that in our well-designed algorithm structure it can significantly reduce the communication load without

undermining the convergence. Additionally, we propose a novel family of state-dependent compression operators, which is variance adjustable and more communication-efficient.

B. Existing DGD-based Compression Algorithms

Recently, it also attracts an increasing attention to design communication-efficient DGD-based algorithm with the compression methods. [12] proposed the QDGD algorithm for the strongly convex objectives. In the QDGD, at t-th iteration, the nodes communicates the compressed local copies $z_{i,t} = C(x_{i,t})$, and then each node update the local copy according to

$$x_{i,t+1} = (1-\epsilon_t+\epsilon_t \mathbf{W}_{ii})x_{i,t}+\epsilon_t \sum_{j\in\mathcal{N}_i} \mathbf{W}_{ij} z_{j,t} - \alpha\epsilon_t \nabla f_i(x_{i,t}). \tag{4}$$

To gaurantee the convergence of QDGD, the parameter ϵ_t needs to be diminishing $O(1/T^{1/2})$ to control the variance caused by the compression noise. However, the convergence rare of QDGD is slow, only $\mathbb{E}[\|x_{i,T} - x_*\|^2] \leq O(1/T^{1/4})$. Another DGD-based compression algorithm named ADC-DGD is proposed in [14] for nonconvex objectives. The ADC-DGD is based on the state-independent compression operator. In ADC-DGD, the communicated information at t-th iteration is the compressed amplified-differential $d_{i,t} = C(t^\gamma y_{i,t})$, where γ is the amplifying parameter to control the noise variance and $y_{i,t}$ is the local differential between two successive iterates. The updating is

$$x_{i,t+1} = \sum_{j\in\mathcal{N}_i} \mathbf{W}_{i,j}\tilde{x}_{j,t} - \alpha\nabla f_i(x_{i,t}), \tag{5}$$

where $x_{i,t}$ is the exact local copy and $\tilde{x}_{i,t}$ is the imprecise copy based on the compressed differential: $\tilde{x}_{i,t} = \tilde{x}_{i,t-1} + d_{i,t}/t^\gamma$. It has been shown that with $\gamma > 1/2$, the ADC-DGD can convergence and the fastest rate to the exact solution is $o(1/T^{1/2})$. Nevertheless, the ADC-DGD has the risk that the amplified-differential might overflow with larger γ. In [13], the ECD-PSGD algorithm is proposed for the decentralized deep learning model traning. The state-independent compression operator is also adopted in ECD-PSGD. The extrapolated information are communicated between the nodes, $z_{i,t} = C[(1-t/2)x_{i,t-1} + (t/2)x_{i,t}]$. Then, the nodes update the local copy according to

$$x_{i,t+1} = \sum_{j\in\mathcal{N}_i} \mathbf{W}_{i,j}\tilde{x}_{j,t} - \alpha\nabla g_i(x_{i,t}), \tag{6}$$

where $\nabla g_i(x_{i,t})$ is a stochastic gradient with $\mathbb{E}[\nabla g_i(x_{i,t})] = \nabla f_i(x_{i,t})$ and the bounded variance, $\tilde{x}_{i,t}$ is the imprecise copy recovered with $z_{i,t} : \tilde{x}_{i,t} = (1-2/t)\tilde{x}_{i,t-1} + (2/t)z_{i,t}$. The ECD-PSGD adopts the extrapolation method to reduce the compression noise. However, the convergence rate of the ECD-PSGD is $O(\log(T)/\sqrt{T})$ and the theoretically optimal step-size is hard to find in practice. To summarize, almost the existing DGD-based compression algorithm with the unbiased stochastic compression operator need some extra strategies to control the compression noise. The drawbacks of these strategies could be the slower convergence(for diminishing step-size strategy) or the overflow risk(for the amplified strategy and the extrapolation strategy). In our work, we aim to design a DGD-based compression algorithm which has a simple structure and can avoid the above risk.

The most related algorithm to ours is the DCD-PSGD algorithm in [13]. Though the similar alogrithm structure is shared by the two algorithms, our DC-DGD algorithm differs from DCD-PSGD in the following key aspects: 1) DCD-PSGD is designed for parallelly training deep learning models. A key assumption in the DCD-PSGD is that the data used by each node are i.i.d., which gaurantees that the local objectives are identical. However, our work relaxes this assumption and allows the local objectives to be non-identical. 2) Both of the two algorithms depend on the state-dependent operator and the constant η defined in Definition 1 plays an important role in their convergence results. In DCD-PSGD, the upper bound of η is $\sqrt{(1-|\lambda_n|)^2/4(1-\lambda_n)^2}$, while the corresponding upper bound in our result is $\sqrt{(1+\lambda_n)/(1-\lambda_n)}$, where λ_n is the smallest eigenvalue of the concensus matrix $\mathbf{W}$. With simply calculation, it can be shown that our upper bound is lower than theirs. 3) To reach the best convergence rate, only one optimal step-size is available for the DCD-PSGD, which is associated with a group of complex parameters. In contrast, the step-size selection in our DC-DGD follows the standard sublinearly diminishing strategy and is more easily implemented in practice. In a word, our work provides a better understanding on the success of the DGD-based algorithm with differential compression technique.

III. Differential Compression Decentralized Gradient Descent

In this Section, we will first introduce our DC-DGD algorithm in Section III-A. Then, we will present the main theoretical results in Section III-B

A. Algorithm

Our differential compression decentralized gradient descent algorithm is stated in the following:

Algorithm: Differential Compression Decentralized Gradient Descent (DC-DGD).

Initialization:

1. Set the initial state $x_{i,0} = y_{i,0} = z_{i,0} = 0$, $\forall i$.
2. Let $t = 1$, and update $z_{i,1} = -\alpha\nabla f_i(x_{i,0})$, and $d_{i,1} = z_{i,1} - x_{i,0}$, $\forall i$.

Main Loop:

3. In the t-th iteration, each node sends the compressed differential $C(d_{i,t-1})$ to its neighbors, where $C(\cdot)$ is a state-dependent compression operation. Also, upon collecting all

neighbors' information, each node updates the following local values:

$$\text{(Local Inexact Copy Update):}$$
$$x_{i,t} = x_{i,t-1} + C(d_{i,t}) \tag{7}$$
$$\text{(Network Consensus Update):}$$
$$y_{i,t} = y_{i,t-1} + \sum\nolimits_{j\in\mathcal{N}_i}[\mathbf{W}]_{ij}C(d_{j,t}) \tag{8}$$
$$\text{(Local Gradient Update):}$$
$$z_{i,t+1} = y_{i,t} - \alpha\nabla f_i(x_{i,t}) \tag{9}$$
$$\text{(Local Differential Update):}$$
$$d_{i,t+1} = z_{i,t+1} - x_{i,t} \tag{10}$$

4. Stop if some convergence criterion is met; otherwise, let $t \leftarrow t+1$ and go to Step 3.

Several important remarks on the DC-DGD algorithm are in order: 1) In the DC-DGD, each node needs to store three local variables: $x_{i,t}$, $y_{i,t}$ and $z_{i,t}$. This memory cost is much less than that in some other DGD-based compression algorithms, such as ADC-DGD and DCD-PSGD, in which additional memory is required store the values of its neighbors in the previous iteration. 2) At the communication step, each node sends out a compressed local differential $C(d_{i,t})$ and the corresponding variance depends on the magnitude of $d_{i,t}$ with the state-dependent compression operator. As shown in Section III-B, $d_{i,t}$ will converge to zero as iterating. Hence, no extra effort is required while the communication load is reduced. 3) In the original DGD algorithm and its variants, the local gradient is calculated based on the exact local copy, which is obtained with the consensus step and gradient updating step (corresponding to $z_{i,t}$ under our notation). However, the gradient used in our DC-DGD is calculated based on $x_{i,t}$, which is estimated from the previous local copy and the compressed differential. We will show that this modification does no harm to the algorithm convergence. 4) The overall updating structure of our DC-DGD is similar to the original DGD algorithm, involving a consensus step and a gradient updating step, and only one parameter, the step-size α_t, is introduced. Thus, the complexity of DC-DGD is almost identical to the DGD algorithm.

B. Convergence Gaurantees

In this part, we provide the convergence analysis on DC-DGD. For notational convenience, we introduce the following quantities

$$\mathbf{F}(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i) \tag{11}$$
$$\mathbf{L}_{\alpha_t}(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top(\mathbf{I} - \mathbf{W}\otimes\mathbf{I}_d)\mathbf{x} + \alpha_t\mathbf{F}(\mathbf{x}). \tag{12}$$

Note that: 1) $\mathbf{F}(\cdot) : \mathbb{R}^{nd} \to \mathbb{R}$, is a function depends on the local copies. 2) $\mathbf{L}_{\alpha_t}(\mathbf{x}) : \mathbb{R}^{nd} \to \mathbb{R}$ is the commonly used Lyapunov function for proving the convergence of DGD-based algorithm ([10], [11]). To better understand our DC-DGD algorithm, we summarize its updating as the followings vector form:

$$\begin{cases} \mathbf{x}_t = \mathbf{x}_{t-1} + C(\mathbf{d}_t) \\ \mathbf{y}_t = \mathbf{y}_{t-1} + (\mathbf{W}\otimes\mathbf{I}_d)C(\mathbf{d}_t) \\ \mathbf{z}_{t+1} = \mathbf{y}_{t-1} - \alpha\nabla\mathbf{F}(\mathbf{x}_t) \\ \mathbf{d}_{t+1} = \mathbf{z}_{t+1} - \mathbf{x}_t \end{cases} \tag{13}$$

where $\mathbf{y} \triangleq [y_1^\top, \ldots, y_n^\top]^\top$, $\mathbf{z} \triangleq [z_1^\top, \ldots, z_n^\top]^\top$ and $\mathbf{d}_t \triangleq [d_1^\top, \ldots, d_n^\top]^\top$. Note that with $\mathbf{y}_0 = (\mathbf{W}\otimes\mathbf{I}_d)\mathbf{x}_0 = \mathbf{0}$, we have $\mathbf{y}_t = (\mathbf{W}\otimes\mathbf{I}_d)\mathbf{x}_t$ by induction. Hence, we can rewrite the updating as:

$$\begin{cases} \mathbf{x}_t = \mathbf{x}_{t-1} + C(\mathbf{d}_t) = \mathbf{x}_{t-1} + \mathbf{z}_t - \mathbf{x}_{t-1} + \boldsymbol{\epsilon}_t = \mathbf{z}_t + \boldsymbol{\epsilon}_t \\ \mathbf{z}_{t+1} = (\mathbf{W}\otimes\mathbf{I}_d)\mathbf{x}_t - \alpha_t\nabla\mathbf{F}(\mathbf{x}_t) = \mathbf{x}_t - \nabla\mathbf{L}_\alpha(\mathbf{x}_t) \\ \mathbf{d}_{t+1} = \mathbf{z}_{t+1} - \mathbf{x}_t = -\nabla\mathbf{L}_\alpha(\mathbf{x}_t) \end{cases} \tag{14}$$

where $\boldsymbol{\epsilon}_t$ is the compression error with $\mathbb{E}[\boldsymbol{\epsilon}_t] = \mathbf{0}$ and $\mathbb{E}[\|\boldsymbol{\epsilon}_t\|^2] \le \eta^2\|\mathbf{d}_t\|^2 = \eta^2\|\nabla\mathbf{L}_\alpha(\mathbf{x}_t)\|^2$. It can be seen that in the DC-DGD, the variance of $\boldsymbol{\epsilon}_t$ depends on the difference between two successive iterates, which is the gradient of the Lyapunov function $\nabla\mathbf{L}_\alpha(\mathbf{x}_t)$. As the algorithm converges, $\nabla\mathbf{L}_\alpha(\mathbf{x}_t)$ will reduce to zero as well as the variance of $\boldsymbol{\epsilon}_t$. Thus, there is no need to adopt the variance reduction stategies, such as the diminishing step-size or the amplifying procedure.

Our convergence results are established under the following assumptions:

Assumption 1. *The local objective functions $f_i(\cdot)$ satisfies:*

- *(Lower boundedness) There exists an optimal x_* with $\|x_*\| < \infty$ such that $\forall x$, $f(x) \ge f(x_*)$;*
- *(Lipschitz continuous gradient) there exists a constant $L > 0$ such that $\forall x, y$, $\|\nabla f_i(x) - \nabla f_i(y)\| \le L\|x - y\|$, $\forall i$;*
- *(Growth rate at infinity). If the domain for x is unbounded, then there exists constant $M > 0$ such that $\lim_{\|\mathbf{x}\|\to\infty}\frac{\|\mathbf{x}\|}{\mathbf{F}(\mathbf{x})} \le M$.*

Note that the first two bullets are standard in convergence analysis of gradient descent type algorithms: The first one ensures the existence of optimal solution and the second guarantees the smoothness of the local objectives. The third bullet is proposed in [14] to guarantee the iterates $\mathbf{x}$ will be bounded. Actually, it is a stronger version of the coercivity assumption in [11].

First, we show that the iterates $\mathbf{x}_t$ and the gradient $\nabla\mathbf{F}(\mathbf{x}_t)$ is bounded during the DC-DGD's updating, and the summation of the gradients of the Lyapunov function over the iteration is also bounded.

Theorem 1. *Under Assumption 1, if the step-size $\alpha \le \frac{\lambda_n(\eta^2+1)+1-\eta^2}{L(1+\eta^2)}$, where η is the variance-amplifying constant in the compression with $\eta^2 < \frac{1+\lambda_n}{1-\lambda_n}$, then*

- *there exists a constant $B > 0$ such that $\mathbb{E}[\|\mathbf{x}_t\|] \le B$ and $\mathbb{E}[\|\nabla\mathbf{F}(\mathbf{x}_t)\|] \le L(B + \sqrt{n}\|x_*\|)$.*

- *the gradients of the Lyapunov function* $\mathbf{L}_\alpha$ *is bounded, i.e.*

$$\sum_{t=0}^{T}\mathbb{E}[\|\nabla \mathbf{L}_\alpha(\mathbf{x}_t)\|^2] \leq \frac{2\alpha\big(f(0)-f(x_*)\big)}{1+\lambda_n-\alpha L-(1-\lambda_n+\alpha L)\eta^2}. \tag{15}$$

Note that Theorem 1 has a key condition on the compression: $\eta^2 < (1+\lambda_n)(1-\lambda_n)$. This upper bounded is derived to gaurantee the feasible domain for the step-size α. It can be seen that with $\lambda_n \to 1$ (i.e. a sparse concensus matrix $\mathbf{W}$), the upper bound for η^2 goes to infinity. Then we bound the derivation from the local copies to the mean of the copies.

Theorem 2. *Under Assumption 1, if* $\mathbb{E}[\nabla \mathbf{F}(\mathbf{x}_t)]$ *is bounded by a constant* D*, then the deviation from mean can be bounded by the followings:*

$$\mathbb{E}[\|\mathbf{x}_T-\bar{\mathbf{x}}_T\|^2] \leq (\frac{\alpha D}{1-\beta})^2 + \eta^2\sum_{t=1}^{T}\beta^{2(T-t)}\mathbb{E}[\|\nabla \mathbf{L}_\alpha(\mathbf{x}_{t-1})\|^2], \tag{16}$$

where $\bar{\mathbf{x}}_T = \mathbf{1}\mathbf{1}^\top\mathbf{x}_T/n$ *and* $\beta = \max\{|\lambda_2|, |\lambda_n|\}$.

Theorem 2 requires that the expectation of gradient $\mathbb{E}[\nabla \mathbf{F}(\mathbf{x}_t)]$ is bounded. This is gauranteed by Theorem 1. In the end, we show that DC-DGD algorithm will reach into an error ball of the global objective's stationary point.

Theorem 3. *Under Assumption 1, if the step-size satisfying* $\alpha \leq \frac{\lambda_n(\eta^2+1)+1-\eta^2}{L(1+\eta^2)}$*, then it holds that*

$$\sum_{t=0}^{T}\mathbb{E}[\|\nabla \mathbf{F}(\bar{\mathbf{x}}_t)\|^2] \leq C_1(\alpha,\beta)[f(0)-f(x_*)] + \frac{\alpha^2 C_2}{(1-\beta)^2}T, \tag{17}$$

where $C_1(\alpha,\beta) = [2\eta^2(\frac{\alpha}{(1-\beta^2)}+\frac{L}{n})/(1+\lambda_n-\alpha L-(1-\lambda_n+\alpha L)\eta^2)+\frac{2}{\alpha}]$, $C_2 = [L(B+\sqrt{n}\|x_*\|)]^2$. *Thus, the ergodic convergence rate is*

$$\min_{t=0,\cdots,T}\mathbb{E}[\|\nabla \mathbf{F}(\bar{\mathbf{x}}_t)\|^2] \leq \frac{C_1(\alpha,\beta)[f(0)-f(x_*)]}{T} + \frac{C_2\alpha^2}{(1-\beta)^2}, \tag{18}$$

where $\nabla \mathbf{F}(\bar{\mathbf{x}}_t)\|^2 = \sum_{i=1}^{n} f_i(\frac{1}{n}\sum_{i=1}^{n} x_{i,t})$.

Note that in Theorem 3, similar with the original DGD algorithm, the error is bounded by two terms: the first one is convergence error with sublinear rate; the second term is the approximation error affected by the step-size and the network structure. In order to reach the exact solution, the step-size α needs to be small so that the approximation error is close to zero. However, as $\alpha \to 0$, the coefficient for the convergence error $C_1(\alpha,\beta) \approx 2/\alpha \to \infty$, which in turn requires more iterations for shrinking the error. In the following corollary, we show that with diminishing step-size α at rate $O(1/T^{1/3})$, the optimal convergence rate can reach $O(1/T^{2/3})$.

Corollary 1. *Set the step-size satisfying* $\alpha = (C_3/T)^{1/3}$ *with* $C_3 = (f(0)-f(x_*))(1-\beta)^2/C_2$*, and* $\alpha \leq \frac{\lambda_n(\eta^2+1)+1-\eta^2}{L(1+\eta^2)}$*, then the convergence rate for the DC-DGD is*

$$\sum_{t=0}^{T}\mathbb{E}[\|\nabla \mathbf{F}(\bar{\mathbf{x}}_t)\|^2] \leq \frac{3\big(f(0)-f(x_*)\big)^{2/3}C_2^{1/3}}{(1-\beta)^{2/3}T^{2/3}} + O(\frac{1}{T}). \tag{19}$$

IV. Compression Extension: the Hybrid Operators

In this section, we will provide a family of State-Dependent Compression Operators, which is the hybrid of the sparsifier and the ternary operator. Recall the two operators in Section II-A. The sparsifier can control η^2 by adjusting the probability p while the expected communication cost for a d-dimensional vector is $d[c_1 p + c_0(1-p)]$, where c_1 is the cost for sending a floating number and c_0 is the cost for value 0. Thus, restricted by a smaller η^2, the communication cost will be closer to the cost for sending the uncompressed copy dc_1. For the ternary operator, the variance is $\sum_{i=1}^{d}|z_i|(\|\mathbf{z}\|_\infty - |z_i|)$, which cannot be directly controlled by any parameter; while the communication cost is $c_1 + (d-1)c_0'$, where c_0' is the cost for the ternary values $\{-1,0,1\}$. Usually, the communication cost of the ternary compressed vector is much smaller than the sparse compressed vector: for example, if using 32-bit for floating number and 1-bit for zero value, the cost for a d-dimensional sparse compressed vector is $[32p+(1-p)]d$; however, for the ternary operator, the cost will be $32+2(d-1)$ if using 32-bit for floating number and 2-bit for the tenary values. With lower variance-amplifying constant η^2(i.e. smaller p) and high dimension d, the communcation saving will be more with the ternary compressed vector. Therefore, to have the compression variance controllable as well as saving more communicatoin cost, a good solution is to merge the sparse operator and the ternary operator.

Consider a d-dimensional vector $\mathbf{z} = (z_1, \cdots, z_d)^T$. It can be descendingly rearranged according to the magnitude of the z_i's absoluted value:

$$\underbrace{z_{[1]},\ z_{[2]},\ \cdots,\ z_{[s_1-1]},\ z_{[s_1]}}_{\text{ternary compression}},\ \underbrace{z_{[s_1+1]},\ \cdots,\ z_{[d-1]},\ z_{[d]}}_{\text{sparse compression}}$$

$$\Rightarrow\ \underbrace{z_{[1]},\quad 0,\quad \cdots,\quad -1,\quad 1}_{\text{ternary compressed}},\ \underbrace{\frac{z_{[s_1+1]}}{p},\ \cdots,\quad 0,\quad \frac{z_{[d]}}{p}}_{\text{sparse compressed}}$$

with $|z_{[i]}| \geq |z_{[i+1]}|$, $\forall i$. For the first largest s_1 elements, we have the ternary operator on them and thus the variance will be $\sum_{i=1}^{s_1}|\mathbf{z}_i|(|\mathbf{z}_{[1]}| - |z_{[i]}|)$; for the rest of elements, we put the sparse operator on them and it will have the corresponding variance $(1/p-1)\sum_{i=s_1+1}^{d} z_{[i]}^2$. To bound the variance-amplifying coefficient $\eta^2 < C$, it only needs to have

$$\text{(ternary)}:\ |\mathbf{z}_{[i]}|(|\mathbf{z}_{[1]}| - |z_{[i]}|) < Cz_{[i]}^2,\ \forall i \leq s_1 \tag{20}$$

$$\text{(sparse)}:\ (1/p-1)z_{[i]}^2 < Cz_{[i]}^2,\ \forall i > s_1. \tag{21}$$

To satisfy (20), we have $s_1 = \arg\min_i\{|z_{[i]}| > \frac{1}{1+C}|z_{[1]}|\}$. And for (21), it requires $p > \frac{1}{1+C}$. Then under the expectation, the compressed vector will have $1+(d-s_1)p$ floating numbers and $(s_1-1)+(d-s_1)(1-p)$ ternary values, which can save more communication cost compared with the sparse compressed vector.

The saving of the above hybrid compression operator depends on the number of the ternary compressed elements s_1. To get a further saving, we consider to improve the above hybrid operator with the idea of the quantization . Instead of just using $z_{[1]}$ for the ternary compression, several

reference elements $\{z_{[q_1]}, \cdots, z_{[q_k]}\}$ are selected. Similar as the condition in (20), s_i elements can be found in the ternary of $z_{[q_i]}$ with

$$|\mathbf{z}_{[j]}|(|\mathbf{z}_{[q_i]}| - |z_{[j]}|) < Cz_{[j]}^2, \ \forall j \in (q_i, q_i + s_i). \tag{22}$$

Thus, the d elemets of the vector can be partitioned into $(k+1)$ group: for the elements in $(q_i, q_i + s_i)$, we have the ternary compression on them with $z_{[q_i]}$; for the elements out of all $(q_i, q_i + s_i)$, the sparse operator will be applied. Then the compressed vector has $k + (d - \sum_{i=1}^k s_i)p$ floating and $(\sum_{i=1}^k s_i - k) + (d - \sum_{i=1}^k s_i)(1-p)$ ternary values. In addition, we need to mark the group identity for the ternary values. However, this cost is very small, usually $\log(k+1)$ for each element.

In the above, we give a family of the hybrid compression operators, which include the sparse operator and the ternary operator as the special cases. Given a variance constrain $\eta^2 < C$, the communication saving is highly related with the group number k and the reference elements. Take 32-bit floating number and 2-bit ternary value as an exmaple. To reach the maximal communication saving, the group number and the reference element can be selected based on the following objective function:

$$\min_{k,\{z_{[q_i]}\}} 32 * \underbrace{[k + (d - \sum_{i=1}^k s_i)p]}_{\#\text{floating number}} + [2 + \underbrace{\log(k+1)}_{\text{cost of identity}}] * \underbrace{[(\sum_{i=1}^k s_i - k) + (d - \sum_{i=1}^k s_i)(1-p)]}_{\#\text{ternary number}}. \tag{23}$$

It will be hard to give a exact solution to (23). However, we can find an approximate solution heuristically. Note that the objective is increasing with larger k and decreasing with larger $\sum_{i=1}^k s_i$. Therefore, the following greedy algorithm can be adopted: with the current vector, we search s_i for each element and find the element with the largest s_i; if the ternary cost on the s_i elements is smaller than the sparse cost, we remove these s_i elements from the current vector and update the vector; otherwise, we have the sparse operator on the elements in the current vector. We summarize the greedy algorithm in the following:

Algorithm: The greedy algorithm for an approximate solution of (23).

Initialization:

1. Descendingly rearrange the gradient vector $\mathbf{z}$ based on the elements' magnitudes.
2. Set the ternary set T as empty.

Main Loop:

3. Search s_i for each element in the current vector and find the element with the largest s_i;
4. Compare the ternary cost $32 + 2(s_i - 1)$ with the sparse cost $[32p + 2(1-p)]s_i$;
5. If the ternary cost is smaller, then remove the corresponding elements from the current vector and add them to T and go to Step 3; otherwise, break the loop.

Ending:

5. Apply the ternary operator to each group in T and the sparse operator to T^c.

V. Numerical Study

In this section, we will present throughout numerical experiments to validate the performance of our proposed DC-DGD algorithm and examine the efficiveness of our hybrid compression operator.

A. Convergence of DC-DGD

In this part, we first focus on the sparse operator (see Example 2) and study the effect of the non-zero probability p. Consider a five-node circle network (see Figure 1(a)) with the global objective function:

$$\min_x f(x) = f_1(x) + f_2(x) + f_3(x) + f_4(x) + f_5(x),$$
$$\text{with } f_i(x) = \begin{cases} \log(1 + (a_i x + b_i)^2/2), \text{if } i = 1, 2; \\ (a_i x - b_i)^2/2, \text{if } i = 3, 4, 5, \end{cases} \tag{24}$$

where the coefficients $\{a_i, b_i\}$ are generated from standard normal distribution. Note that the first two ojective funcions are non-convex and the rest three are convex.

In our simulation, two different concensus matrices are used,

$$\mathbf{W}_1 = \begin{bmatrix} 1/5 & 2/5 & 0 & 0 & 2/5 \\ 2/5 & 1/5 & 0 & 0 & 2/5 \\ 0 & 2/5 & 1/5 & 2/5 & 0 \\ 0 & 0 & 2/5 & 1/5 & 2/5 \\ 2/5 & 0 & 0 & 2/5 & 1/5 \end{bmatrix},$$
$$\mathbf{W}_2 = \begin{bmatrix} 1/2 & 1/4 & 0 & 0 & 1/4 \\ 1/4 & 1/2 & 1/4 & 0 & 0 \\ 0 & 1/4 & 1/2 & 1/4 & 0 \\ 0 & 0 & 1/4 & 1/2 & 1/4 \\ 1/4 & 0 & 0 & 1/4 & 1/2 \end{bmatrix}. \tag{25}$$

Note that $\lambda_n(\mathbf{W}_1) = -0.45$ and $\lambda(\mathbf{W}_2) = 0.09$. We compare three methods: the conventional DGD algorithm, our DC-DGD algorithm and the ADC-DGD algorithm. For the DC-DGD, we adopt the sparsifer with the non-zero probability p from $\{0.3, 0.5, 0.8\}$. In the ADC-DGD, we consider the low-precision representation (see Example 1 in [12]) and choose the amplifying exponent γ from $\{0.8, 1.2\}$. The step-size is fixed as 0.1 and we repeat 50 independent trials for each setting.

The experimental results are summarized in Figure 1(b)-1(c). In specific, Figure 1(b) shows the convergence performance of the three algorithms under the concensus matrix $\mathbf{W}_1$. It can be seen that the DC-DGD with p from $\{0.3, 0.5\}$ doesn't converge while it converges with $p = 0.8$. This is because from Example 2 and Theorem 1 the corresponding lower bound of the non-zero probability p can be derived from $1/p - 1 < (1 + \lambda_n(\mathbf{W}_1))/(1 - \lambda_n(\mathbf{W}_1))$, which is

0.72. Thus, with $p < 0.72$ (i.e. the cases with 0.3 and 0.5), Theorem 1 doesn't hold. However, with $p = 0.8$, the convergence speed of the DC-DGD is almost the same as the conventional DGD algorithm (the black dashed line), which is faster than the ADC-DGD. In Figure 1(c), the convergence performance of these algorithms under the concensus matrix $\mathbf{W}_2$. With the similar calculation, the lower bound of the non-zero probability p is 0.45. In this case, the DC-DGD algorithm with $p = 0.5$ converges while it with $p = 0.3$ still diverges, as shown in Figure 1(c). In the both case, we can see that the DC-DGD with the proper nonzero probability has faster convergence speed and smaller standard deviation, compared with the ADC-DGD.

B. Compression Operator Comparsion

In this part, we have a comparsion between the three state-dependent operators: the sparse operator, the ternary operator and our proposed hybrid operator. 20 d-dimensional vectors are independently generated from the multivaraite normal distribution $N(\mathbf{0}, \mathbf{I}_d)$, with d from $\{20, 50\}$. We apply the three operators on each vector, repectively, and 100 times trials are conducted. For any vector $\mathbf{x}$ and the compressed vector $C(\mathbf{x})$,we focus on the following metrics: 1) Bias: $\|\mathbb{E}[C(\mathbf{x})] - \mathbf{x}\|$; 2) Variance-Norm ratio: $\text{Var}[C(\mathbf{x})]/\|\mathbf{x}\|^2$; 3) Communication cost. Here the Variance-Norm ratio is corresponding to η^2 in Theorem 1. The smaller bias and ratio, the better the operator is. To calculate the communication cost, we use 32 bit for the floating number and 2 bit for the ternary number. In addtional, for the sparse operator, only one bit is used to present the value 0. Note that the Variance-Norm ratio can be controlled by adjusting the parameters in the sparse operator and our hybrid operator. To illustrate this advantage, we set the upper bound for the raio as $1/2$ and 1. For both of the cases, the parameters in the two operators are set to save the most cost: with the bound as $1/2$, $p = 2/3$ in the sparse operator and $\eta^2 = 1/2$ in the hybrid operator; with the bound as 1, $p = 1/2$ in the sparse operator and $\eta^2 = 1$ in the hybrid operator. The results are summarized in Figure 2.

From Figure 2(a)-2(d), it can be seen that our hybrid operator has the smallest bias, while for the sparse operator the bias will increase as the non-zero probability p decreases. Figure 2(b)-2(e) compares the Variance-Norm ratio. It can be seen that our hybrid operator can strictly control the ratio below the given bound, while the tenary operator has the ratio over the bound. The communication costs are shown in Figure 2(c)-2(f). Though the tenary operator has the smallest cost, it cannot gaurantee the Variance-Norm ratio to be controlled. Compared with the sparse operator, our hybrid operator can save almost 50% cost under all the circumstances.

C. Real Data Study

Lastly, we compare our DC-DGD algorithm with a few other competitive algorithms. We consider a classification task on the spambase data set from UCI repository. This data set contains email spam data for $n = 4601$ email messages and $d = 57$ features. The spam data are evenly distributed into 10 machines. For each machine, the local objective $f_i(\mathbf{x})$ is a logistic regression problem with the nonconvex regularizer ([23], [24]):

$$-\frac{1}{n_i}\sum_{j=1}^{n_i}[y_{ij}\log(\frac{1}{1+\exp(-\mathbf{x}^\top\zeta_{ij})})+ (1-y_{ij})\log(\frac{\exp\mathbf{x}^\top\zeta_{ij}}{1+\exp-\mathbf{x}^\top\zeta_{ij}})]+\rho\sum_{i=1}^{d}\frac{x_i^2}{1+x_i^2}, \tag{26}$$

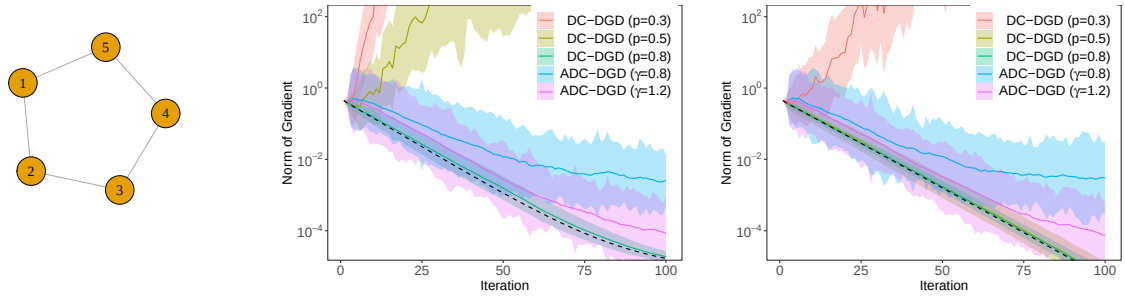
where the label $y_{ij} \in \{0, 1\}$, the feature $\mathbf{x} \in \mathbb{R}^{57}$ and $\rho = 0.1$ in our experiment.

The following four algorithms are compared: the conventional DGD algorithm ([1]), the ADC-DGD algorithm ([14]), the QDGD algoithm ([12]) and our proposed DC-DGD algorithm. For the ADC-DGD and QDGD, the low-precision representation is used, in which the floating number will be randomly quantized to integer number. In our DC-DGD algorithm, we consider the three compression operator: the sparsifer, the tenary operator and our hybrid operator. We use the following way to calculate the communication cost: 32 bits for the floating number, 8 bits for the integer number (int8) and 2 bit for tenary number. In additional, the value 0 are presented with 1 bit in the DC-DGD with the sparsifer. Also, we consider two different network structures, which are shown in Figure 3(a)-3(d). Note that for the first structure, $\beta = 0.98$ and $\lambda_n = 0.24$; while for the second structure, $\beta = 0.88$ and $\lambda_n = -0.37$. The simulation results are shown in Figure 3.

The first row in Figure 3 shows the results of the first network structure while the second row shows those of the second structure. It can be find that the DC-DGD algorithm with the tenary operator doesn't converge under the second structure. This is because the Variance-Norm ratio cannot be controlled. Thus, the tenary operator is not a safe compression way within the scheme of the DC-DGD algorithm. Figure 3(b)-3(b) shows the convergence rates of these algorithms. We can see that the QDGD has the slowest convergence rate and follows by the ADC-DGD, while the survived DC-DGD has almost the same rate as the conventional DGD algorithm. Figure 3(c)-3(f) compares the communication cost of these algorithm. In Figure 3(c), we can find that the tenary operator has the smallest communication cost, around 10^5 bits to the error 10^{-2}. However, this operator doesn't work in the second network. It can be also seen that the DC-DGD algorithm with our hybrid operator can converge under the two networks and use the smallest bit under the second network. It costs almost 2×10^5 bits to the error 10^{-2}, while the ADC-DGD costs 2.5×10^5 bits and other methods cost more than 2.5×10^5 bits. Additionally, note that our DC-DGD algorthms have thinner standard deviation shade region, compared with ADC-DGD and QDGD, which means our algorithm is more stable.

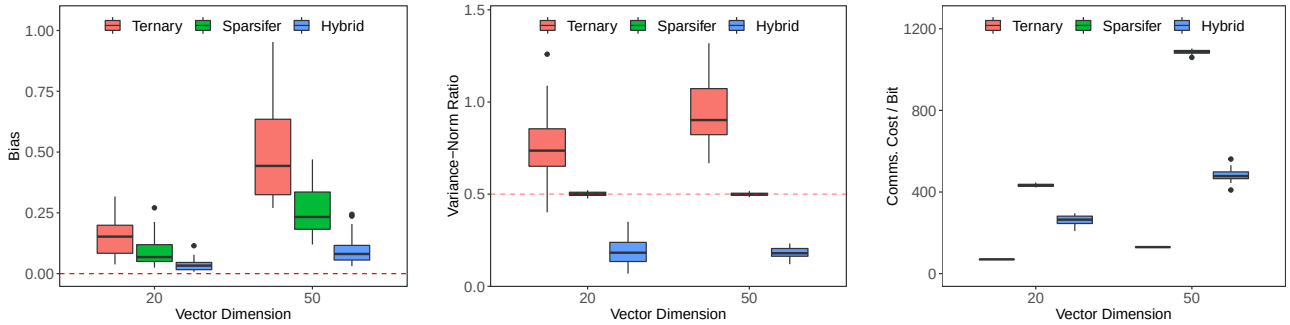
VI. Conclusion

In this work, we designed and analyzed a novel communication-efficient DGD-based algorithm, named differential compression decentralized gradient descent (DC-DGD). Instead of the local copy, our DC-DGD has the

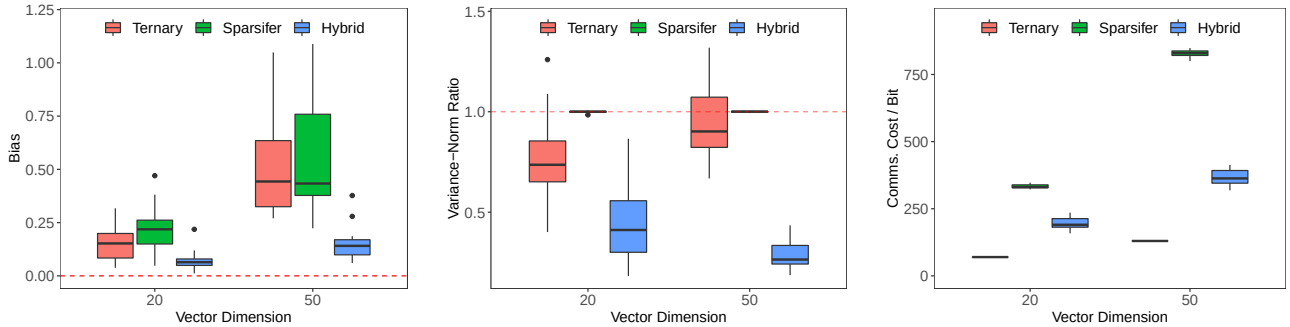


(a) (b) (c)

Fig. 1. (a) The five-node circle network; (b-c) Performance comparsion: Convergence error vs Iteration with the concensus matrices $\mathbf{W}_1$ and $\mathbf{W}_2$, respectively. The black dashed is the conventional DGD algorithm. The solid lines presents the error averaged over 50 trials and the shaded region indicates the standard deviation of results over random trials.



(a) Case1: Bias (b) Case1: Var-Norm Ratio (c) Case1: Comms. Cost

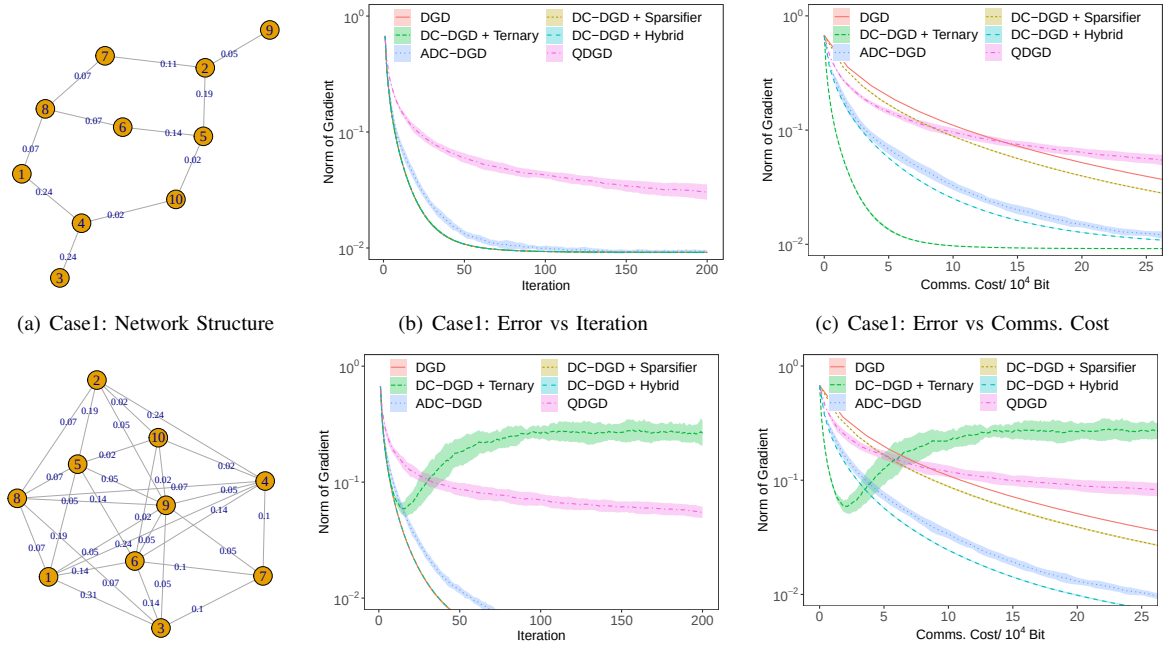


(d) Case2: Bias (e) Case2: Var-Norm Ratio (f) Case2: Comms. Cost

Fig. 2. Comparsion between three operators: (a)-(c) are the boxplots for the ratio upper bound as 0.5 and (d)-(e) are the boxplots for the ratio upper bound as 1. The red dashed lines in (a)&(d) present value 0; the red dashed lines in (b)&(e) present the upper bounds,0.5 and 1, respectively.

nodes communicating the differential between two successive iterates, which is the gradient of the Lyapunov funciton, and the state-dependent compression strategy is adopted to compress the information for communication saving. On the theoretical side, we show that the compression noise under our algorithm structure is self-reduced as iteration and the optimal convergence rate of DC-DGD is proved as $O(T^{-2/3})$ under some mild conditions. On the practical side, we develope a novel family of state-dependent compression operator, named Hybrid Operator. It is shown that our hybrid operator is variance controllable and costs lower communication. Also, a greedy algorithm is proposed for determining the optimal hybrid operator dynamically. In the end, we use throughout experiments to validate the effectiveness of our DC-DGD algorithm and hybrid operator. It is shown that our method is more communication-efficient.

There are serveral limitations in our work. First, we give an upper bound for determining the compression operator. However, our bound is sufficient. It will be interesting to explore the sufficient and necessary upper bound, with which the communication cost can be further reduced. Second, our algorithm can only get into an error ball of the stationary point with a fixed step-size and $O(1/T)$ rate. There exist serveral correction procedures ([25]–[27]), which can help DGD-based algorithms exactly reach the stationary point with a fixed step-size. These correction procedures can also be adopted for



(a) Case1: Network Structure (b) Case1: Error vs Iteration (c) Case1: Error vs Comms. Cost

(d) Case2: Network Structure (e) Case2: Error vs Iteration (f) Case2: Error vs Comms. Cost

Fig. 3. (a)&(d) Two ten-node networks. The concensus weights are on the corresponding edges. (b)&(e) Convergence error vs Iteration; (c)&(f) Convergence error vs Communication Cost. The lines presents the error averaged over 10 trials and the shaded region indicates the standard deviation of results over random trials.

improve our communication-efficient DGD-based algorithm. Third, though our optimal hybrid operator can reduce more communication cost, it introduces extra computation with the greedy algorithm and without any approximation analysis. It will be of the great help to design and analyze a selection algorithm with low complexity and exact solution.

REFERENCES

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

[2] J. N. Tsitsiklis, "Problems in decentralized decision making and computation." Massachusetts Inst of Tech Cambridge Lab for Information and Decision Systems, Tech. Rep., 1984.

[3] X. Mao, Y. Gu, and W. Yin, "Walk proximal gradient: An energy efficient algorithm for consensus optimization," *IEEE Internet of Things Journal*, 2018.

[4] Z. Jiang, K. Mukherjee, and S. Sarkar, "On consensus-disagreement tradeoff in distributed optimization," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 571–576.

[5] F. Facchinei, G. Scutari, and S. Sagratella, "Parallel selective algorithms for nonconvex big data optimization," *IEEE Transactions on Signal Processing*, vol. 63, no. 7, pp. 1874–1889, 2015.

[6] Z. Jiang, A. Balu, C. Hegde, and S. Sarkar, "Collaborative deep learning in fixed topology networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 5904–5914.

[7] Y. Liu, J. Liu, and T. Basar, "Differentially private gossip gradient descent," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 2777–2782.

[8] W. Wang, J. Wang, M. Kolar, and N. Srebro, "Distributed stochastic multi-task learning with graph regularization," *arXiv preprint arXiv:1802.03830*, 2018.

[9] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2681–2690.

[10] 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.

[11] J. Zeng and W. Yin, "On nonconvex decentralized gradient descent," *IEEE Transactions on signal processing*, vol. 66, no. 11, pp. 2834–2848, 2018.

[12] A. Reisizadeh, A. Mokhtari, H. Hassani, and R. Pedarsani, "Quantized decentralized consensus optimization," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 5838–5843.

[13] H. Tang, S. Gan, C. Zhang, T. Zhang, and J. Liu, "Communication compression for decentralized training," in *Advances in Neural Information Processing Systems*, 2018, pp. 7652–7662.

[14] X. Zhang, J. Liu, Z. Zhu, and E. S. Bentley, "Compressed distributed gradient descent: Communication-efficient consensus over networks," *arXiv preprint arXiv:1812.04048*, 2018.

[15] A. Berahas, R. Bollapragada, N. S. Keskar, and E. Wei, "Balancing communication and computation in distributed optimization," *IEEE Transactions on Automatic Control*, 2018.

[16] A. S. Berahas, C. Iakovidou, and E. Wei, "Nested distributed gradient methods with adaptive quantized communication," *arXiv preprint arXiv:1903.08149*, 2019.

[17] J. Goldsmith, L. Huang, and C. M. Crainiceanu, "Smooth scalar-on-image regression via spatial bayesian variable selection," *Journal of Computational and Graphical Statistics*, vol. 23, no. 1, pp. 46–64, 2014.

[18] H. Zhou, L. Li, and H. Zhu, "Tensor regression with applications in neuroimaging data analysis," *Journal of the American Statistical Association*, vol. 108, no. 502, pp. 540–552, 2013.

[19] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," in *Advances in neural information processing systems*, 2017, pp. 1509–1519.

[20] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," in *Advances in Neural Information Processing Systems*, 2018, pp. 1299–1309.

[21] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Communication-efficient sgd via gradient quantization and encoding," in *Advances in Neural Information Processing Systems*, 2017, pp. 1709–1720.

[22] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar,

"Signsgd: Compressed optimisation for non-convex problems," in *International Conference on Machine Learning*, 2018, pp. 559–568.

[23] A. Antoniadis, I. Gijbels, and M. Nikolova, "Penalized likelihood regression for generalized linear models with non-quadratic penalties," *Annals of the Institute of Statistical Mathematics*, vol. 63, no. 3, pp. 585–615, 2011.

[24] Z. Wang, Y. Zhou, Y. Liang, and G. Lan, "Stochastic variance-reduced cubic regularization for nonconvex optimization," *arXiv preprint arXiv:1802.07372*, 2018.

[25] J. Xu, S. Zhu, Y. C. Soh, and L. Xie, "Convergence of asynchronous distributed gradient methods over stochastic networks," *IEEE Transactions on Automatic Control*, vol. 63, no. 2, pp. 434–448, 2017.

[26] 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.

[27] Z. Li, W. Shi, and M. Yan, "A decentralized proximal-gradient method with network independent step-sizes and separated convergence rates," *arXiv preprint arXiv:1704.07807*, 2017.