Communication-Efficient Network-Distributed Optimization with Differential-Coded Compressors

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

Abstract—**Network-distributed optimization has attracted significant attention in recent years due to its ever-increasing applications. However, the classic decentralized gradient descent (DGD) algorithm is communication-inefficient for large-scale and high-dimensional network-distributed optimization problems. To address this challenge, many compressed DGD-based algorithms have been proposed. However, most of the existing works have high complexity and assume compressors with bounded noise power. To overcome these limitations, in this paper, we propose a new differential-coded compressed DGD (DC-DGD) algorithm. The key features of DC-DGD include: i) DC-DGD works with general SNR-constrained compressors, relaxing the bounded noise power assumption; ii) The differential-coded design entails the same convergence rate as the original DGD algorithm; and iii) DC-DGD has the same low-complexity structure as the original DGD due to a *self-noise-reduction effect*. Moreover, the above features inspire us to develop a hybrid compression scheme that offers a systematic mechanism to minimize the communication cost. Finally, we conduct extensive experiments to verify the efficacy of the proposed DC-DGD and hybrid compressor.**

I. Introduction

Network-distributed optimization, a canonical topic dating back to [1], has received significant interest in recent years thanks to its ever-increasing applications, e.g., distributed learning [2]–[4], multi-agent systems [5], resource allocation [6], localization [7], etc. All these applications involve geographically dispersed datasets that are too big to aggregate due to high communication costs or privacy/security risks, hence necessitating distributed optimization over the network. A notable feature in network-distributed optimization is that there is a *lack of shared memory* due to the absence of a dedicated parameter server – a key component in the hierarchical distributed master/slave architecture. As a result, every node can only exchange and aggregate information with its local neighbors to reach a *consensus* on a global optimal decision.

In the literature, a classic algorithm for solving network-distributed optimization problems is the decentralized gradient descent method (DGD) proposed by Nedic and Ozdaglar [8]. The enduring popularity of DGD lies in its simple *gossip-like* structure, which can be easily implemented in networks. Further, DGD achieves the same convergence rate as the centralized gradient descent method, implying that distributed computation does not sacrifice convergence rate. However, despite the aforementioned salient features, a major limitation of the DGD method is that it requires full information exchanges of the state variables between nodes. Hence, the DGD algorithm is *communication-inefficient* when solving large-size high-dimensional optimization problems in networks with low-speed communication links. For example, consider a distributed image regression problem over a satellite network, where each satellite has images of resolution 2048×2048 [9]. Hence, the parameter dimension is $2048 \times 2048 \approx 4 \times 10^6$ and the communication load per DGD iteration is 134 MB (32-bit floating-point). This is problematic for many satellite networks with low-speed RF (radio frequency) links (typically in the range of hundreds Mbps [10]).

To improve DGD's communication efficiency, recent years have seen a line of research based on exchanging *compressed* information between nodes (see, e.g., [11]–[14]). However, most of the existing works suffer from the following *limitations* (see Section II for in-depth discussions): 1) more complex algorithmic structures compared to DGD due to extra parameter tunings; 2) restricted assumptions on compression noise having *finite* power; and 3) strong i.i.d. (independently identically distributed) distribution assumptions on datasets at different locations, which hardly holds in practice. In addition, most of the existing works simply treat compressors as "blackbox operators" and do not consider how to minimize communication load. The above limitations motivate us to develop new compression-based algorithms for communication-efficient network-distributed optimization.

The major contribution of this paper is that we propose a *differential-coded compression-based* DGD algorithmic framework (DC-DGD) that overcomes the above limitations. Moreover, based on the proposed DC-DCD framework, we propose a hybrid compression scheme that integrates gradient sparsification and ternary operators, which enables dynamic communication load minimization. Our main technical results and their significance are summarized as follows:

- We propose a new *differential-coded* DC-DGD algorithmic framework, where "differential-coded" means that the information exchanged between nodes is the differential between

This work has been supported in part by NSF grants ECCS-1818791, CCF-1758736, CNS-1758757, CNS-1446582; ONR grant N00014-17-1-2417, AFRL grant FA8750-18-1-0107, USDA grant 68-7482-17-009. DISTRIBUTION STATEMENT A: Approved for Public Release; distribution unlimited 88ABW-2019-4486 on 20 September 2019. Other requests shall be referred to AFRL/RIT 525 Brooks Rd Rome, NY 13441. DISCLAIMER: Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of AFRL.

two successive iterations of the variables, rather than the variables themselves. We show that DC-DGD allows us to work with a wide range of general compressors that are only constrained by SNR (signal-to-noise-ratio) and thus could have unbounded noise power. The use of SNR-constrained compressors *relaxes* the commonly adopted assumption on bounded compression noise power in the literature [11]–[13]. More specifically, we show that if a compressor's SNR is greater than $(1-\lambda_N)/(1+\lambda_N)$, where λ_N is the smallest eigenvalue of the consensus matrix used in all DGD-type algorithms, then our DC-DGD algorithm achieves the *same* $O(1/t)$ convergence rate as the original DGD method.

- Not only does the use of SNR-constrained compressors make our DC-DGD framework more general and practical, it also induces a nice "*self-compression-noise-power-reduction effect*" that keeps the algorithmic structure of DC-DGD simple. More specifically, based on a quadratic Lyapunov function of the consensus form of the optimization problem, we show that the accumulated compression noise under DC-DGD shrinks to zero under SNR-constrained compressors and differential-coded information exchange. Hence, there is *no* need to introduce extra mechanisms or parameters to tame the accumulated compression noise for ensuring convergence. As a result, DC-DGD enjoys the *same* low-complexity and efficient convergence rate as the original DGD method.
- The insight on the relationship between DC-DCD and SNR-constrained compressors further inspires us to develop a hybrid compression scheme that integrates gradient sparsification and ternary operators to obtain *controllable* SNR and a high compression ratio simultaneously. The proposed hybrid compression scheme achieves the best of both worlds through a meticulously designed mechanism to minimize the communication load. Specifically, under the hybrid compressor, the communication load minimization can be formulated as an integer programming problem. Based on the special problem structure, we show that the problem can be solved efficiently by a greedy algorithm.

Our results in this paper contribute to the state of the art of theories and algorithm design for communication-efficient network-distributed optimization. The rest of the paper is organized as follows. In Section II, we further review related works on the state of the art of compressed DGD-based optimization algorithms. In Section III, we first present our DC-DGD algorithm and then analyze its convergence gaurantees. In Section IV, we present a family of hybrid operators and a greedy algorithm is proposed to choose the optimal hybrid operator. Numerical results are provided in Section V. We conclude this paper in Section VI.

II. Related Works

As mentioned earlier, compression-based DGD algorithms have received increasing attention in recent years. In this section, we provide a more in-depth survey on the state of the art in this area to put our work into comparative perspectives. Broadly speaking, compression-based DGD algorithms can be categorized as follows (some fall into multiple categories):

1) Uncoded Noise-Power-Constrained Compressed DGD: In the literature, most of the early attempts on compressed DGD were focused on noise-power-constrained compressors, which are easier to analyze. One notable recent work is the QDGD method proposed by Reisizadeh *et al.* [11]. The main idea of QDGD is to introduce an ϵ_t-scaled aggregation of compressed local copies coupled with an ϵ_t-scaled local gradient step, where $\epsilon_t = O(1/\sqrt{t})$ is an extra diminishing parameter introduced in each iteration t to dampen the noise power. However, due to the timid gradient step-size $\epsilon_t\alpha$ (α is the original local gradient step-size in DGD), the convergence rate of QDGD is $O(1/t^{1/4})$, which is much slower than the original DGD. Also, the algorithm is more complex to use than DGD due to the sensitivity in tuning the extra parameter ϵ_t. Moreover, QDGD was focused on strongly convex cases and it is unclear whether its performance results can be straightforwardly extended to non-convex cases.

2) Differential-Coded DGD with Noise-Power-Constrained Compressors: Another more recently emerging line of research is the differential-coded DGD approach. For example, in [12], Tang *et al.* proposed the ECD-PSGD algorithm, where extrapolated information is used in each iteration to reduce compression noise. However, it requires computing an optimized step-size in each iteration, which leads to high per-iteration complexity. Also, the convergence rate of ECD-PSGD is $O(\log(t)/\sqrt{t})$, which is slower than the original DGD and its stochastic variant. Another notable example is the ADC-DGD algorithm proposed by Zhang *et al.* [13], where a t^{γ}-amplified differential-coded information (with $\gamma > \frac{1}{2}$) is used in each iteration t. It is shown in [13] that ADC-DGD achieves the same $O(1/t)$ convergence rate as that of the original DGD. However, ADC-DGD runs the risk of arithmetic overflow due to the asymptotically unbounded t^{γ}-amplification factor. This extra γ-parameter selection of ADC-DGD also makes it complex to use compared to DGD.

3) Differential-Coded DGD with SNR-Constrained Compressors: The most related algorithm to ours is the DCD-PSGD algorithm proposed by Tang *et al.* in [12], which is by far the only differential-coded algorithm that can work with SNR-constrained compressors. Although DCD-PSGD shares the above similarities with us, our DC-DGD algorithm differs from DCD-PSGD in the following key aspects: i) DCD-PSGD is designed for parallel training, where a key assumption is that the data at each node are i.i.d., which guarantees that the local objectives are identical. However, our work *relaxes* this assumption and allows the local objectives to be non-identically distributed. ii) The final output of DCD-PSGD is the *average* of all nodes in the network, which could be difficult to implement in network-distributed settings. In contrast, DC-DGD does not require such an averaging at the final output since each node reaches a global optimal consensus. iii) Although both algorithms work with SNR-constrained compressors, the SNR constraint of DCD-PSGD is lower bounded by $4(1-\lambda_N)^2/(1-|\lambda_N|)^2$, while the SNR lower bound of our DC-DGD is $(1-\lambda_N)/(1+\lambda_N)$, where λ_N is the smallest eigenvalue of the consensus matrix. It can be readily verified that our SNR lower bound is much smaller, which implies that our DC-DGD can work with more aggressive compression schemes. iv) To achieve the best convergence rate, DCD-PSGD requires an optimal step-size determined by a set of complex parameters (cf. step-size "γ" in Theorem 1 and Corollary 2 in [12]) and hard to implement in practice. In contrast, the step-size selection in our DC-DGD uses simple sublinearly diminishing series and is easy to implement.

III. Differential-Coded Decentralized Gradient Descent with SNR-Constrained Compressors

In this section, we first present the problem formulation of network-distributed optimization in Section III-A. Then, we will present our DC-DGD algorithm in Section III-B and its main theoretical results in Section III-C. Lastly, we provide proof sketches for the main theoretical results in Section III-D.

A. Problem Formulation of Network-Distributed Optimization

We use an undirected connected graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$ to represent a network, where $\mathcal{N}$ and $\mathcal{L}$ are the sets of nodes and links, respectively, with $|\mathcal{N}| = N$ and $|\mathcal{L}| = E$. We let $\mathbf{x} \in \mathbb{R}^D$ denote a global decision vector to be optimized. In network-distributed optimization, we want to distributively solve a network-wide optimization problem: $\min_{\mathbf{x}\in\mathbb{R}^D} f(\mathbf{x})$, where $f(\mathbf{x})$ can be decomposed node-wise as follows:

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

where each local objective function $f_i(\mathbf{x})$ is only observable to node i. Problem (1) has many real-world applications. For example, in the satellite network image regression problem in Section I, each satellite i distributively collects image data $\{\mathbf{u}_{ij}, \mathbf{v}_{ij}, \theta_{ij}\}_{j=1}^{N_i}$, where $\mathbf{u}_{ij}$, $\mathbf{v}_{ij}$, and θ_{ij} represent the pixels, geographical information, and ground-truth label of the j-th image at satellite i, respectively, and N_i is the size of the local dataset. Suppose that the regression is based on a linear model with parameters $\mathbf{x} = [\mathbf{x}_1^\top \ \mathbf{x}_2^\top]^\top$. Then, the problem can be written as: $\min_{\mathbf{x}} f(\mathbf{x}) \triangleq \min_{\mathbf{x}} \sum_{i=1}^{N} f_i(\mathbf{x})$, where $f_i(\mathbf{x}) \triangleq \frac{1}{N_i}\sum_{j=1}^{N_i}(\theta_{ij} - \mathbf{u}_{ij}^\top \mathbf{x}_1 - \mathbf{v}_{ij}^\top \mathbf{x}_2)^2$. Note that Problem (1) can be written as the following equivalent *consensus form*:

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=1}^{N} f_i(\mathbf{x}_i) \\ \text{subject to} \quad & \mathbf{x}_i = \mathbf{x}_j, \qquad \forall (i,j) \in \mathcal{L}. \end{aligned} \tag{2}$$

where $\mathbf{x}_i \in \mathbb{R}^D$ is the local copy of $\mathbf{x}$ at node i. The constraints in Problem (2) guarantee that all the local copies are equal to each other, hence the name consensus form.

B. The DC-DGD Algorithm

To facilitate the presentation of our DC-DGD algorithm, we first need to formally define two technical notions. The first one is the *SNR-constrained unbiased stochastic compressors*:

Definition 1 (SNR-Constrained Stochastic Unbiased Compressor)**.** A stochastic compressor $C(\cdot)$ is said to be unbiased and constrained by an SNR threshold η if it satisfies $C(\mathbf{z}) = \mathbf{z} + \boldsymbol{\epsilon}_{\mathbf{z}}$, with $\mathbb{E}[\boldsymbol{\epsilon}_{\mathbf{z}}] = \mathbf{0}$ and $\mathbb{E}[\|\boldsymbol{\epsilon}_{\mathbf{z}}\|^2] \le (1/\eta)\|\mathbf{z}\|^2$, $\forall \mathbf{z} \in \mathbb{R}^d$.

We can see from Definition 1 that, for a given compressor, η is its lowest SNR yielded by its largest compression noise power $\mathbb{E}[\|\boldsymbol{\epsilon}_z\|^2]$. We note that SNR-constrained stochastic unbiased compressors are much *less restricted* than the noise-power-constrained stochastic unbiased compressors previously assumed in the literature (see, e.g., [11]–[13]), which satisfy $\mathbb{E}[\boldsymbol{\epsilon}_{\mathbf{z}}] = \mathbf{0}$ and $\mathbb{E}[\|\boldsymbol{\epsilon}_{\mathbf{z}}\|^2] \le \sigma^2$, $\forall$ $\mathbf{z}$. That is, the compression noise power is *universally upper bounded* by a constant σ^2 regardless of the input signal. In contrast, the noise power under SNR-constrained compressors could be arbitrarily large as long as it satisfies a certain SNR requirement, hence being more general. For example, the following are two typical SNR-constrained stochastic unbiased compressors:

Example 1. *[The Sparsifier Operator [15]] For any vector* $\mathbf{z} \in \mathbb{R}^d$, $C(\mathbf{z})$ *outputs a sparse vector with the i-th element* $[C(\mathbf{z})]_i$ *following the Bernoulli*(p) *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 constant. The operation is unbiased and the SNR is lower bounded by is* $p/(1-p)$*.*

Example 2. *[Ternary Operator [16]] For any* $\mathbf{z} \in \mathbb{R}^d$, $C(\mathbf{z}) = \|\mathbf{z}\|_\infty sign(\mathbf{z}) \circ \mathbf{b}_{\mathbf{z}}$*, where* $\circ$ *is the Hadamard product,* $[sign(\mathbf{z})]_i = sign([\mathbf{z}]_i)$ *and* $\mathbf{b}_{\mathbf{z}}$ *is a random vector with the i-th element* $[\mathbf{b}_{\mathbf{z}}]_i$ *following the Bernoulli distribution:*

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

The operation is unbiased and the noise power $\mathbb{E}[\|\boldsymbol{\epsilon}_{\mathbf{z}}\|^2] = \sum_{i=1}^{d} |z_i|(\|\mathbf{z}\|_\infty - |z_i|)$ *and hence* $\eta = \Theta(d)$*.*

Next, we introduce the notion of *consensus matrix*, which is denoted as $\mathbf{W} \in \mathbb{R}^{N\times N}$ in this paper. As will be seen later, the entries $[\mathbf{W}]_{ij}$ in $\mathbf{W}$ define the weight parameters used by each node to perform local information aggregation. Mathematically, $\mathbf{W}$ satisfies the following properties:

a) *Doubly Stochastic:* $\sum_{i=1}^{N}[\mathbf{W}]_{ij} = \sum_{j=1}^{N}[\mathbf{W}]_{ij} = 1$.
b) *Symmetric:* $[\mathbf{W}]_{ij} = [\mathbf{W}]_{ji}$, $\forall i, j \in \mathcal{N}$.
c) *Network-Defined Sparsity Pattern:* $[\mathbf{W}]_{ij} > 0$ if $(i, j) \in \mathcal{L}$ and $[\mathbf{W}]_{ij} = 0$ otherwise, $\forall i, j \in \mathcal{N}$.

Collectively, properties a) and b) imply that the spectrum of $\mathbf{W}$ (i.e., the set of all eigenvalues) lies in the interval $(-1, 1]$ on the real line, with exactly one eigenvalue being equal to 1. Further, since all eigenvalues are real, they can be sorted as $-1 < \lambda_N(\mathbf{W}) \le \cdots \le \lambda_1(\mathbf{W}) = 1$. For convenience, we define a parameter $\beta \triangleq \max\{|\lambda_2(\mathbf{W})|, |\lambda_N(\mathbf{W})|\} \in (0, 1)$, i.e., the second-largest eigenvalue of $\mathbf{W}$ in magnitude. Simply speaking, the use of the consensus matrix is due to the fact that $(\mathbf{W} \otimes \mathbf{I}_P)\mathbf{x} = \mathbf{x}$ *if and only* if $\mathbf{x}_i = \mathbf{x}_j$, $(i, j) \in \mathcal{L}$,

[8] where $\mathbf{x} = [\mathbf{x}_1^\top, \ldots, \mathbf{x}_N^\top]^\top$ and $\otimes$ represents the Kronecker product. Therefore, Problem (2) can be reformulated as $\min_{\mathbf{x}\in\mathbb{R}^D} \sum_{i=1}^N f_i(\mathbf{x}_i)$, s.t. $(\mathbf{W}\otimes\mathbf{I}_P)\mathbf{x} = \mathbf{x}$, which further leads to the original DGD algorithmic design [8].

With the notions of SNR-constrained unbiased stochastic compressors and consensus matrix, we are now in a position to present our DC-DGD algorithmic framework. To this end, we let $\mathcal{N}_i \triangleq \{j \in \mathcal{N} : (i,j) \in \mathcal{L}\}$ denote the set of local neighbors of node i. Then, our DC-DGD is stated as follows:

Algorithm 1: Differential-Coded Compressed Decentralized Gradient Descent Method (DC-DGD).

```
Initialization:
1. Set the initial state $\mathbf{x}_{i,0} = \mathbf{y}_{i,0} = \mathbf{z}_{i,0} = \mathbf{0}$, $\forall i$.
2. Let $t=1$, $\mathbf{z}_{i,1} = -\alpha_1 \nabla f_i(\mathbf{x}_{i,0})$, and $\mathbf{d}_{i,1} = \mathbf{z}_{i,1} - \mathbf{x}_{i,0}$, $\forall i$.
Main Loop:
3. In the $t$-th iteration, each node sends the differential-coded compressed information $C(\mathbf{d}_{i,t})$ to its neighbors, where $C(\cdot)$ is an SNR-constrained stochastic unbiased compressor. Meanwhile, upon the reception of all neighbors' information, each node performs the following updates:
    a) Local copy inexact update: $\mathbf{x}_{i,t} = \mathbf{x}_{i,t-1} + C(\mathbf{d}_{i,t})$.   (3)
    b) Weighted local aggregation step:
       $\mathbf{y}_{i,t} = \mathbf{y}_{i,t-1} + \sum_{j\in\mathcal{N}_i} [\mathbf{W}]_{ij} C(\mathbf{d}_{j,t})$.   (4)
    c) Local gradient step: $\mathbf{z}_{i,t+1} = \mathbf{y}_{i,t} - \alpha_t \nabla f_i(\mathbf{x}_{i,t})$.   (5)
    d) Local differential update: $\mathbf{d}_{i,t+1} = \mathbf{z}_{i,t+1} - \mathbf{x}_{i,t}$.   (6)
4. Stop if some preferred 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) The combined update structure in Steps 3-b) and 3-c) is the *same* as the original DGD algorithm, which contains a weighted local aggregation step and a local gradient step. Notably, DC-DGD only has one parameter: the step-size α_t (same as DGD). Thus, DC-DGD enjoys the *identical* structural complexity as that of the original DGD method.

2) DC-DGD is *memory-efficient*: In DC-DGD, each node only needs to store three local variables: $\mathbf{x}_{i,t}$, $\mathbf{y}_{i,t}$ and $\mathbf{z}_{i,t}$. This is in stark contrast to some DGD-based algorithms, e.g., ADC-DGD [13] and DCD-PSGD [12], where each node needs to store all values of the previous iteration from its neighbors, which is unscalable for large and dense networks that have high node degrees.

3) Compared to the original DGD algorithm and many of its variants, a notable difference in DC-DGD is that the gradient $\nabla f_i(\mathbf{x}_{i,t})$ in Step 3-c) is calculated based on an *inexact* update from $\mathbf{x}_{i,t-1}$ and the compressed differential $C(\mathbf{d}_{i,t})$ (i.e., Step 3-a)), rather than using an exact update. This is derived from the convergence of a chosen Lyapunov function (to be defined soon). Interestingly, we will show that this modification does not harm the algorithm's convergence speed because the difference between inexact and exact updates is negligible when the Lyapunov function is near convergence.

Before we prove the convergence of DC-DGD, it is insightful to offer some intuitions on why DC-DGD retains most of the simple structural properties of the original DGD and does *not* need extra mechanism/parameter(s) to tame compression noises. First, we define the following Lyapunov function:

$$L_{\alpha_t}(\mathbf{x}) \triangleq \frac{1}{2}\mathbf{x}^\top(\mathbf{I} - \mathbf{W}\otimes\mathbf{I}_d)\mathbf{x} + \alpha_t f(\mathbf{x}). \tag{7}$$

We note that $L_{\alpha_t}(\mathbf{x})$ is also used for proving the convergence of several other DGD-based algorithms (e.g., [17], [18]). To understand our DC-DGD algorithm, we rewrite its updates Steps 3-a) – 3-d) in the following vector form:

$$\begin{cases} \mathbf{x}_t = \mathbf{x}_{t-1} + C(\mathbf{d}_t), \\ \mathbf{y}_t = \mathbf{y}_{t-1} + (\mathbf{W}\otimes\mathbf{I}_d)C(\mathbf{d}_t), \\ \mathbf{z}_{t+1} = \mathbf{y}_t - \alpha_t \nabla f(\mathbf{x}_t), \\ \mathbf{d}_{t+1} = \mathbf{z}_{t+1} - \mathbf{x}_t, \end{cases} \tag{8}$$

where $\mathbf{y} \triangleq [\mathbf{y}_1^\top, \ldots, \mathbf{y}_n^\top]^\top$, $\mathbf{z} \triangleq [\mathbf{z}_1^\top, \ldots, \mathbf{z}_n^\top]^\top$ and $\mathbf{d} \triangleq [\mathbf{d}_1^\top, \ldots, \mathbf{d}_n^\top]^\top$. Note that with $\mathbf{y}_0 = \mathbf{0}$, we have $\mathbf{y}_t = (\mathbf{W}\otimes\mathbf{I}_d)\mathbf{x}_t$ by induction. Hence, we can rewrite the updates as:

$$\begin{cases} \mathbf{x}_t = \mathbf{x}_{t-1} + C(\mathbf{d}_t) = \mathbf{x}_{t-1} + \mathbf{z}_t - \mathbf{x}_{t-1} + \boldsymbol{\epsilon}_t = \mathbf{z}_t + \boldsymbol{\epsilon}_t, \\ \mathbf{z}_{t+1} = (\mathbf{W}\otimes\mathbf{I}_d)\mathbf{x}_t - \alpha_t \nabla f(\mathbf{x}_t) = \mathbf{x}_t - \nabla L_{\alpha_t}(\mathbf{x}_t), \\ \mathbf{d}_{t+1} = \mathbf{z}_{t+1} - \mathbf{x}_t = -\nabla L_{\alpha_t}(\mathbf{x}_t), \end{cases}$$

where $\boldsymbol{\epsilon}_t$ is a compression noise satisfying $\mathbb{E}[\boldsymbol{\epsilon}_t] = \mathbf{0}$ and $\mathbb{E}[\|\boldsymbol{\epsilon}_t\|^2] \le (1/\eta)\|\mathbf{d}_t\|^2 = (1/\eta)\|\nabla L_\alpha(\mathbf{x}_t)\|^2$. That is, the power of the noise $\boldsymbol{\epsilon}_t$ depends on the difference between two successive iterations, which in turn is the gradient of the Lyapunov function $\nabla L_{\alpha_t}(\mathbf{x}_t)$. As the algorithm converges (to be proved soon), $\nabla L_{\alpha_t}(\mathbf{x}_t) \to \mathbf{0}$ implies that $\mathbb{E}[\|\boldsymbol{\epsilon}_t\|^2] \to \mathbf{0}$. Hence, *no* extra effort is required to tame the noise power thanks to this *self-compression-noise-power-reduction effect*.

C. Main Theoretical Results

In this subsection, we will establish the convergence of the proposed DC-DGD algorithm. Our convergence results are proved under the following mild assumptions:

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

- *(Lower boundedness) There exists an optimal $\mathbf{x}^*$ with $\|\mathbf{x}^*\| < \infty$ such that $f(\mathbf{x}) \ge f(\mathbf{x}^*)$, $\forall \mathbf{x}$;*
- *(Lipschitz continuous gradient) there exists a constant $L > 0$ such that $\forall \mathbf{x}_1, \mathbf{x}_2$, $\|\nabla f_i(\mathbf{x}_1) - \nabla f_i(\mathbf{x}_2)\| \le L\|\mathbf{x}_1 - \mathbf{x}_2\|$, $\forall i$;*
- *(Bounded gradient) there exists a constant $D > 0$ such that for all $\mathbf{x}$, $\|\nabla f_i(\mathbf{x})\| \le D$, $\forall i$.*

Note that the first two bullets are standard in convergence analysis: The first one ensures the existence of optimal solution and the second guarantees the smoothness of the local objectives. The third bullet is needed to bound the deviation of local copies to their mean (cf. Theorem 2). It is equivalent to $f_i(\cdot)$ being D-Lipschitz continuous. This mild assumption has been widely adopted in analyzing non-convex optimization algorithms in the literature (see, e.g., [19]–[21]).

To show the convergence of DC-DGD, we will show that the iterates $\{\mathbf{x}_t\}_{t=1}^\infty$ and the gradient $\{\nabla f(\mathbf{x}_t)\}_{t=1}^\infty$ are bounded over all iterations, and the summation of the gradients of the Lyapunov function over the iterations is also bounded.

Theorem 1. *Under Assumption 1, if a constant step-size* $\alpha \leq (\lambda_N(\eta+1)+\eta-1)/L(1+\eta)$ *is used, where* η *is the SNR threshold satisfying* $\eta > (1-\lambda_N)/(1+\lambda_N)$*, then the sum of the gradients of the Lyapunov function* L_α *is bounded:*

$$\sum_{\tau=0}^{t} \mathbb{E}[\|\nabla L_\alpha(\mathbf{x}_\tau)\|^2] \leq \frac{2\alpha\big(f(\mathbf{0})-f(\mathbf{x}^*)\big)}{1+\lambda_N-\alpha L-(1-\lambda_N+\alpha L)/\eta}.$$

Note that Theorem 1 has a key condition on the SNR threshold: $\eta > (1-\lambda_N)/(1+\lambda_N)$. This SNR lower bound is to guarantee the feasible domain for the step-size α. Interestingly, it can be seen that as $\lambda_N \to 1$ (i.e., a sparse consensus matrix $\mathbf{W}$), the lower bound for SNR η shrinks to zero, meaning that as the network gets *sparser*, we could adopt compressors with *larger compression ratios*.

Next, we bound the derivation of each local copy from the mean of all local copies in any iteration t:

Theorem 2. *Under Assumption 1 and with the same step-size and SNR selections as in Theorem 1, in each iteration t, the deviations of local copies from the mean can be bounded as:*

$$\mathbb{E}[\|\mathbf{x}_t-\bar{\mathbf{x}}_t\|^2] \leq \left(\frac{\alpha N D}{1-\beta}\right)^2 + \sum_{\tau=1}^{t} \beta^{2(t-\tau)} \mathbb{E}[\|\nabla L_\alpha(\mathbf{x}_{\tau-1})\|^2]/\eta,$$

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 $\mathbb{E}[\nabla L_\alpha(\mathbf{x}_t)]$ is bounded, which is guaranteed by Theorem 1. Lastly, based on Theorems 1 and 2, we show that DC-DGD converges to an error ball of a stationary point of the global objective function at rate $O(1/t)$:

Theorem 3. *Under Assumption 1, if the step-size satisfies* $\alpha \leq (\lambda_N(\eta+1)+\eta-1)/L(1+\eta)$*, then it holds that*

$$\sum_{\tau=0}^{t} \mathbb{E}[\|\nabla f(\bar{\mathbf{x}}_\tau)\|^2] \leq C_1(\alpha,\beta)[f(\mathbf{0})-f(\mathbf{x}^*)] + \frac{\alpha^2 N^2 D^2 L}{(1-\beta)^2} t,$$

where $C_1(\alpha,\beta) = 4(\frac{\alpha}{(1-\beta^2)}+\frac{L}{2})/[(1+\lambda_N-\alpha L)\eta-(1-\lambda_N+\alpha L)]+\frac{2N}{\alpha}$*. Thus, DC-DGD converges at rate* $O(1/t)$ *to an error ball that depends on parameters* (α, N, D, L, β)*:*

$$\min_{\tau=0,\cdots,t} \mathbb{E}[\|\nabla f(\bar{\mathbf{x}}_\tau)\|^2] \leq \frac{C_1(\alpha,\beta)[f(\mathbf{0})-f(\mathbf{x}^*)]}{t} + \frac{\alpha^2 N^2 D^2 L}{(1-\beta)^2}.$$

Note that in Theorem 3, similar to the original DGD algorithm, the size of the error ball is determined by two terms: The first one is a convergence error with sublinear diminishing rate $O(1/t)$; The second term is the approximation error affected by the step-size and the network structure (characterized by N and β). Therefore, to reach an optimal solution, the step-size α needs to be small so that the second term 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 first term.

The next result shows that with diminishing step-size $\alpha_t = O(1/t^{1/3})$, DC-DGD converges to a first-order stationary point (optimal solution in convex problems) at rate $O(1/t^{2/3})$:

Corollary 1. *Let* $\alpha_t = (C_2/t)^{1/3}$*, where* $C_2 \triangleq (f(\mathbf{0})-f(\mathbf{x}^*))(1-\beta)^2/D^2N^2L$*, and* $\alpha_t \leq (\lambda_N(\eta+1)+1-\eta)/L(1+\eta)$*, then the convergence rate of DC-DGD is:*

$$\min_{\tau\in[0,t]} \mathbb{E}[\|\nabla f(\bar{\mathbf{x}}_\tau)\|^2] \leq \frac{3\big(f(\mathbf{0})-f(\mathbf{x}^*)\big)^{2/3}(D^2N^2L)^{1/3}}{(1-\beta)^{2/3}t^{2/3}} + O\Big(\frac{1}{t}\Big).$$

D. Proofs of the Main Theoretical Results

Due to space limitation, we provide proof sketches of the main theoretical results in this subsection and relegate the full proofs to our online technical report [22].

Proof Sketch of Theorem 1. Let $\mathcal{F}_t \triangleq \sigma(\mathbf{x}_1,\cdots,\mathbf{x}_t)$ denote a filtration. It can be shown that the Lyapunov function $L_\alpha(\mathbf{x})$ has $(1-\lambda_n+\alpha L)$-Lipschitz gradients. It then follows that:

$$L_\alpha(\mathbf{x}_{t+1}) \leq L_\alpha(\mathbf{x}_t) - \langle \nabla L_\alpha(\mathbf{x}_t), \nabla L_\alpha(\mathbf{x}_t) - \boldsymbol{\epsilon}_{t+1}\rangle + \frac{(1-\lambda_N+\alpha L)}{2}[\|\nabla L_\alpha(\mathbf{x}_t)\|^2 + \|\boldsymbol{\epsilon}_{t+1}\|^2 - 2\langle \nabla L_\alpha(\mathbf{x}_t), \boldsymbol{\epsilon}_{t+1}\rangle].$$

Taking conditional expectation and using the properties of SNR-constrained unbiased compressors yield: $\mathbb{E}[L_\alpha(\mathbf{x}_{t+1})|\mathscr{F}_t] \leq L_\alpha(\mathbf{x}_t) + \frac{1}{2}[(\alpha L - \lambda_N - 1) + \frac{(1-\lambda_N+\alpha L)}{\eta}]\|\nabla L_\alpha(\mathbf{x}_t)\|^2$. Since $\eta > (1-\lambda_N)/(1+\lambda_N)$, we have $(\lambda_N(\eta+1)+\eta-1)/L(1+\eta) > 0$. Then, by setting step-size α as stated in the theorem, we have $[\alpha L - \lambda_N - 1 + (1-\lambda_N+\alpha L)/\eta] < 0$. It then follows that $-[\alpha L - \lambda_N - 1 + (1-\lambda_N+\alpha L)/\eta]\|\nabla L_\alpha(\mathbf{x}_t)\|^2 \leq 2(L_\alpha(\mathbf{x}_t) - \mathbb{E}[L_\alpha(\mathbf{x}_{t+1})|\mathscr{F}_t])$. Taking full expectation on both sides and telescoping from 0 to t, we have:

$$-[\alpha L - \lambda_N - 1 + (1-\lambda_N+\alpha L)/\eta] \times \sum_{\tau=1}^{t} \mathbb{E}[\|\nabla L_\alpha(\mathbf{x}_t)\|^2] \leq 2(L_\alpha(\mathbf{x}_0) - \mathbb{E}[L_\alpha(\mathbf{x}_{t+1})]). \quad (9)$$

Since $\mathbb{E}[L_\alpha(\mathbf{x}_{t+1})] \geq \alpha \mathbb{E}[f(\mathbf{x}_{t+1})] \geq \alpha \sum_{i=1}^{n} f_i(\mathbf{x}^*)$, after rearranging terms, we can conclude that:

$$\sum_{\tau=1}^{t} \mathbb{E}[\|\nabla L_\alpha(\mathbf{x}_t)\|^2] \leq \frac{2\alpha(\sum_{i=1}^{N} f_i(\mathbf{0}) - \sum_{i=1}^{N} f_i(\mathbf{x}_*))}{1+\lambda_N-\alpha L-(1-\lambda_N+\alpha L)/\eta},$$

and the proof is complete. □

Proof Sketch of Theorem 2. For notation convenience, We let $\tilde{\mathbf{W}} \triangleq \mathbf{W} \otimes \mathbf{I}_d$. From (8), we can obtain:

$$\begin{cases} \mathbf{x}_1 = \tilde{\mathbf{W}}\mathbf{x}_0 - \alpha_0 \nabla f(\mathbf{x}_0) - \boldsymbol{\epsilon}_1 = -\alpha_0 \nabla f(\mathbf{x}_0) - \boldsymbol{\epsilon}_1, \\ \mathbf{x}_2 = \tilde{\mathbf{W}}\mathbf{x}_1 - \alpha_1 \nabla f(\mathbf{x}_1) - \boldsymbol{\epsilon}_2 \\ \quad = -\tilde{\mathbf{W}}\alpha_0 \nabla f(\mathbf{x}_0) - \alpha_1 \nabla f(\mathbf{x}_1) - \tilde{\mathbf{W}}\boldsymbol{\epsilon}_1 - \tilde{\mathbf{W}}\boldsymbol{\epsilon}_2, \\ \quad \vdots \\ \mathbf{x}_t = -\sum_{\tau=0}^{t-1} \alpha \tilde{\mathbf{W}}^{t-\tau-1} \nabla f(\mathbf{x}_\tau) - \sum_{\tau=1}^{t} \tilde{\mathbf{W}}^{t-\tau} \boldsymbol{\epsilon}_\tau. \end{cases}$$

Using the above equations, we can derive the following inequality for the deviation from the mean $\bar{\mathbf{x}}_t$:

$$\|\mathbf{x}_t - \bar{\mathbf{x}}_t\|^2 = \|\mathbf{x}_t - (1/N)\mathbf{1}\mathbf{1}^\top \mathbf{x}_t\|^2$$
$$\leq 2\Big\|\sum\nolimits_{\tau=0}^{t-1} \alpha(\tilde{\mathbf{W}}^{t-\tau-1} - (1/N)\mathbf{1}\mathbf{1}^\top)\nabla f(\mathbf{x}_\tau)\Big\|^2$$

$$+ 2\sum_{\tau=1}^{t} \|(\tilde{\mathbf{W}}^{t-\tau} - (1/N)\mathbf{1}\mathbf{1}^\top)\boldsymbol{\epsilon}_\tau\|^2$$

$$+ 2\sum_{\tau=1}^{t} \sum_{s=1, s\neq\tau}^{t} \Big\langle \Big(\tilde{\mathbf{W}}^{t-\tau} - \frac{\mathbf{1}\mathbf{1}^\top}{N}\Big)\boldsymbol{\epsilon}_\tau, \Big(\tilde{\mathbf{W}}^{t-s} - \frac{\mathbf{1}\mathbf{1}^\top}{N}\Big)\boldsymbol{\epsilon}_s \Big\rangle.$$

Taking the expectation on both sides, noting $\mathbb{E}[\boldsymbol{\epsilon}_t] = \mathbf{0}$, and after some algebraic manipulations, we arrive at:

$$\mathbb{E}[\|\mathbf{x}_t - \bar{\mathbf{x}}_t\|^2] \leq \Big(\frac{\alpha N D}{1-\beta}\Big)^2 + \sum_{\tau=1}^{t} \beta^{2(t-\tau)} \mathbb{E}[\|\nabla L_\alpha(\mathbf{x}_{\tau-1})\|^2]/\eta,$$

which completes the proof. □

Proof Sketch of Theorem 3. First, we prove a key descending inequality on $\bar{\mathbf{x}}_t = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_{i,t}$. From the update rule $\mathbf{x}_{t+1} = \tilde{\mathbf{W}}\mathbf{x}_t - \alpha\nabla f(\mathbf{x}_t) - \boldsymbol{\epsilon}_{t+1}$, we have $\bar{\mathbf{x}}_{t+1} = \bar{\mathbf{x}}_t - \frac{\alpha}{N}\sum_{i=1}^{N} \nabla f_i(\mathbf{x}_{i,t}) - \bar{\boldsymbol{\epsilon}}_{t+1}$. It then follows that:

$$\bar{f}(\bar{\mathbf{x}}_{t+1}) \leq \bar{f}(\bar{\mathbf{x}}_t) - \langle \nabla \bar{f}(\bar{\mathbf{x}}_t), \frac{\alpha}{N}\sum_{i=1}^{N} \nabla f_i(\mathbf{x}_{i,t}) + \bar{\boldsymbol{\epsilon}}_{t+1} \rangle$$

$$+ \frac{L}{2}\Big[\Big\|\frac{\alpha}{N}\nabla f(\mathbf{x}_{i,t})\Big\|^2 + \|\bar{\boldsymbol{\epsilon}}_{t+1}\|^2 + 2\Big\langle \frac{\alpha}{N}\nabla f(\mathbf{x}_{i,t}), \bar{\boldsymbol{\epsilon}}_{t+1} \Big\rangle\Big].$$

where $\bar{f}(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} f_i(x_i)$. Taking conditional expectation on both sides and after some algebraic manipulations, we can show that

$$\mathbb{E}[\bar{f}(\bar{\mathbf{x}}_{t+1})|\mathscr{F}_t] \leq \bar{f}(\bar{x}_t) - \frac{\alpha}{2}\|\nabla \bar{f}(\bar{x}_t)\|^2 +$$

$$\frac{\alpha}{2}\|\frac{1}{n}\sum_{i=1}^{n} \nabla f_i(x_{i,t}) - \nabla \bar{f}(\bar{x}_t)\|^2 + \frac{L}{2n^2\eta}\|\nabla L_\alpha(\mathbf{x}_t)\|^2.$$

Taking full expectation, telescoping the inequality from $\tau = 0$ to t, and after further algebraic manipulations, we have:

$$\frac{\alpha}{2}\sum_{\tau=0}^{t} \mathbb{E}[\|\nabla f(\bar{\mathbf{x}}_\tau)\|^2] \leq \Big[\frac{1}{N} + \Big(\frac{\alpha L}{(1-\beta^2)N^2\eta} + \frac{L}{2N^2\eta}\Big) \times$$

$$\frac{2\alpha}{1 + \lambda_N - \alpha L - (1 - \lambda_N + \alpha L)/\eta}\Big][f(\mathbf{0}) - f(\mathbf{x}_*)] + \frac{\alpha^3 D^2 L t}{(1-\beta)^2},$$

which, after further rearrangements, yields the result stated in the theorem. This completes the proof. □

IV. A Hybrid Compression Design under DC-DGD for Communication Cost Minimization

Inspired by previous theoretical insights, in this section, our goal is to design a *hybrid* SNR-constrained compression scheme to achieve high communication cost saving, while having a controllable SNR. Recall from Section III-A that the sparsifier can control the compression noise power by adjusting the probability p and 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. Therefore, if the SNR η threshold is large, the communication cost will be close to sending uncompressed copy dc_1. For the ternary operator, its compression noise power is $\sum_{i=1}^{d} |z_i|(\|\mathbf{z}\|_\infty - |z_i|)$, which is *not* directly controllable by any parameter. The communication cost is $c_1 + (d-1)c_0'$, where c_0' is the cost for the ternary values $\{-1, 0, 1\}$.

In general, the communication cost of a ternary-compressed vector is much smaller than that of the sparse-compressed vector: For example, if using 32-bit floating numbers and one bit for zero, the cost for a d-dimensional sparse compressed vector is $[32p + (1-p)]d$. In contrast, for the ternary operator, the cost will be $32 + 2(d-1)$ if using 32-bit floating numbers and two bits for the ternary values. With a larger SNR threshold η (i.e., larger p) and high dimensionality d, the communication cost of the ternary compressor is much smaller. Therefore, to have a *controllable* compression noise power as well as high communication cost savings, a promising solution is to *combine* the sparse and the ternary compressors.

To this end, consider a d-dimensional vector $\mathbf{z} = [z_1, \cdots, z_d]^\top$. We can sort and rearrange the elements of $\mathbf{z}$ in descending order of magnitude to have: $z_{[1]}, \ldots, z_{[d]}$, with $|z_{[i]}| \geq |z_{[i+1]}|$, $i = 1, \ldots, d-1$. For the first largest s_1 elements, we apply the ternary compressor, while for the rest of the elements, we use the sparse compressor, i.e.,

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

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

As a result, the compression noise power levels of the first s_1 largest elements and the rest are $\sum_{i=1}^{s_1} |z_{[i]}|(|z_{[1]}| - |z_{[i]}|)$ and $(1/p - 1)\sum_{i=s_1+1}^{d} z_{[i]}^2$, respectively. In order to ensure the effective SNR of the hybrid scheme satisfies $\eta > C$ for some lower bound C, we have:

$$\text{(ternary)}: \ |z_{[i]}|(|z_{[1]}| - |z_{[i]}|) < (1/C)z_{[i]}^2, \ \forall i \leq s_1 \quad (10)$$

$$\text{(sparsifer)}: \ (1/p - 1)z_{[i]}^2 < (1/C)z_{[i]}^2, \ \forall i > s_1. \quad (11)$$

To satisfy (10) and (11), we have $s_1 = \arg\min_i\{|z_{[i]}| > \frac{1}{1+1/C}|z_{[1]}|\}$ and $p > \frac{1}{1+1/C}$, respectively. Then, on average, the compressed vector has $1 + (d - s_1)p$ floating numbers and $(s_1 - 1) + (d - s_1)(1-p)$ ternary values, which is more efficient compared to that under the sparsifier compressor.

In fact, the hybrid compression idea above can be generalized to achieve further communication cost savings: Instead of just using $z_{[1]}$ for the ternary compression, we can select multiple "*anchor elements*" $\{z_{[q_1]}, \cdots, z_{[q_k]}\}$. There are s_i elements between $z_{[q_i]}$ and $z_{[q_{i+1}]}$. Thus, a d-dimensional vector can be partitioned into $(k+1)$ groups. For the elements with indices in $(q_i, q_i + s_i)$, we apply the ternary compressor based on $z_{[q_i]}$. For the remaining elements, we apply the sparsifier operator. Similar to (10), we have

$$|z_{[j]}|(|z_{[q_i]}| - |z_{[j]}|) < (1/C)z_{[j]}^2, \ \forall j \in (q_i, q_i + s_i). \quad (12)$$

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. Moreover, we need to save the indices of the anchor elements, for which we need $\lceil \log(k+1) \rceil$ bits per element.

Given an SNR threshold η, the communication saving of our hybrid compression scheme is highly dependent on the group number k and the positions of the anchor elements, which can be optimized by solving an integer programming problem. Take 32-bit floating numbers and 2-bit ternary values as an example. To achieve the maximum communication saving, the group number k and the locations of the anchor elements can be determined by solving:

$$\min_{k,\{z_{[q_i]}\}} \left\{ 32 \underbrace{\left[k + \left(d - \sum_{i=1}^{k} s_i\right)p\right]}_{\text{Number of floating values}} + [2 + \underbrace{\lceil \log(k+1)\rceil}_{\substack{\text{cost of storing}\\ \text{anchor indices}}}] \times \underbrace{\left[\left(\sum_{i=1}^{k} s_i - k\right) + \left(d - \sum_{i=1}^{k} s_i\right)(1-p)\right]}_{\text{Number of ternary values}} \right\}. \tag{13}$$

Problem (13) is an integer optimization problem, which can be shown to be equivalent to bin packing problems, thus being NP-hard. However, an efficient greedy heuristic algorithm can be developed by leveraging the special problem structure. Specifically, we note that the objective function is increasing and decreasing with respect to k and $\sum_{i=1}^{k} s_i$, respectively. Therefore, we can find anchor points $\{\mathbf{z}_{[q_i]}\}_{i=1}^{k}$ and their corresponding ternary sets (of size s_i) by checking (12); if the ternary cost of the s_i elements is smaller than the sparsifier cost, we remove these s_i elements from the current vector; otherwise, we use the sparsifier compressor on the current vector. We summarize the greedy algorithm as follows:

Algorithm 2: A greedy algorithm for solving Problem (13).

```
Initialization:
1. Sort and rearrange the elements of vector $\mathbf{z}$ in a descending order of magnitude.
2. Let $i = 1$. Set the ternary set $\mathcal{T}$ as empty.
Main Loop:
3. Inner Loop:
    3.1) For each element $\mathbf{z}_{[j]}$, $j \notin \mathcal{T}$, find the set: $\mathcal{S}_j = \{z_{[k]} : |z_{[k]}|(|z_{[j]}| - |z_{[k]}|) < z_{[k]}^2/C, k \notin \mathcal{T}\}$.
    3.2) Set $q_i = \arg\max |\mathcal{S}_j|$ and $s_i = \max |\mathcal{S}_j|$.
4. Compare the ternary cost $32 + 2(s_i - 1)$ with the sparsifier 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 $\mathcal{T}$, let $i \leftarrow i + 1$ and go to Step 3; otherwise, break the loop.
Final Step:
6. Apply the ternary operator to each group in $\mathcal{T}$ and the sparse operator to $\mathcal{T}^c$.
```

Now, we analyze the running time complexity of the greedy algorithm. First of all, the sorting requires $O(d\log(d))$ time. The worst-case number of iterations in the main loop is $O(d)$; while in each inner loop, it takes $O(d)$ steps to find the ternary set for each element. Hence, the overall time-complexity of Algorithm 2 is $O(d^2 + d\log(d))$.

V. Numerical Results

In this section, we perform extensive numerical experiments to validate the performances of our proposed DC-DGD algorithm and the hybrid compression scheme.

1) Convergence of DC-DGD: In this simulation, we adopt the sparsifier compression in Example 1 and vary the probability parameter p to induce different SNR threshold values. Consider a five-node circle network in Fig. 1(a) with the global objective function: $\min_{\mathbf{x}} f(\mathbf{x}) = f_1(\mathbf{x}) + f_2(\mathbf{x}) + f_3(\mathbf{x}) + f_4(\mathbf{x}) + f_5(\mathbf{x})$, where

$$f_i(\mathbf{x}) = \begin{cases} \log(1 + (\mathbf{a}_i^\top \mathbf{x} + b_i)^2/2), & \text{if } i = 1, 2; \\ (\mathbf{a}_i^\top \mathbf{x} - b_i)^2/2, & \text{if } i = 3, 4, 5. \end{cases} \tag{14}$$

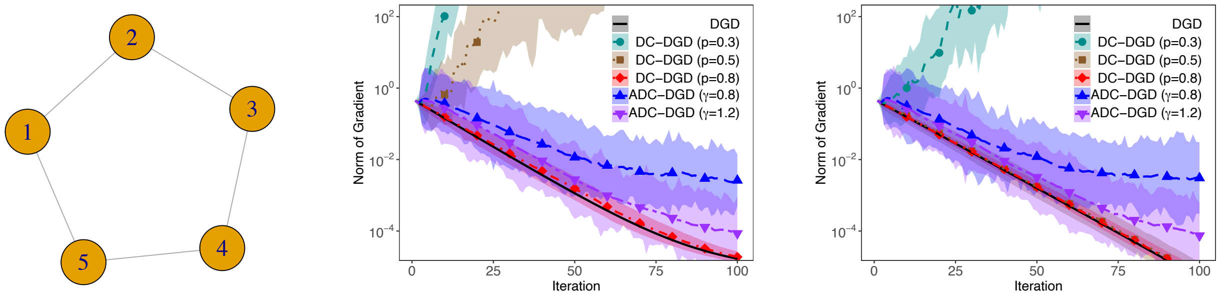
In (14), the coefficients $\{\mathbf{a}_i, b_i\}_{i=1}^5$ are randomly generated from the standard Gaussian distribution. Note that $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ are non-convex and the remaining are convex. In our simulations, we use the following two consensus matrices:

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

Note that $\lambda_N(\mathbf{W}_1) = -0.45$ and $\lambda_N(\mathbf{W}_2) = 0.09$. We compare the original DGD, the ADC-DGD [13], and our DC-DGD algorithms. For DC-DGD, the sparsifier probability parameter p is chosen from $\{0.3, 0.5, 0.8\}$. Note that since ECD-PSGD and DCD-PSGD in [12] are using stochastic gradients and hence results are not directly comparable, they are not included in the simulations. In ADC-DGD, we adopt the low-precision representation (see [11, Example 1]) and choose the amplifying exponent γ from $\{0.8, 1.2\}$. We use fixed step-size 0.1 and repeat 50 independent trials for each setting. The simulation results are presented in Figs. 1(b)–1(c).

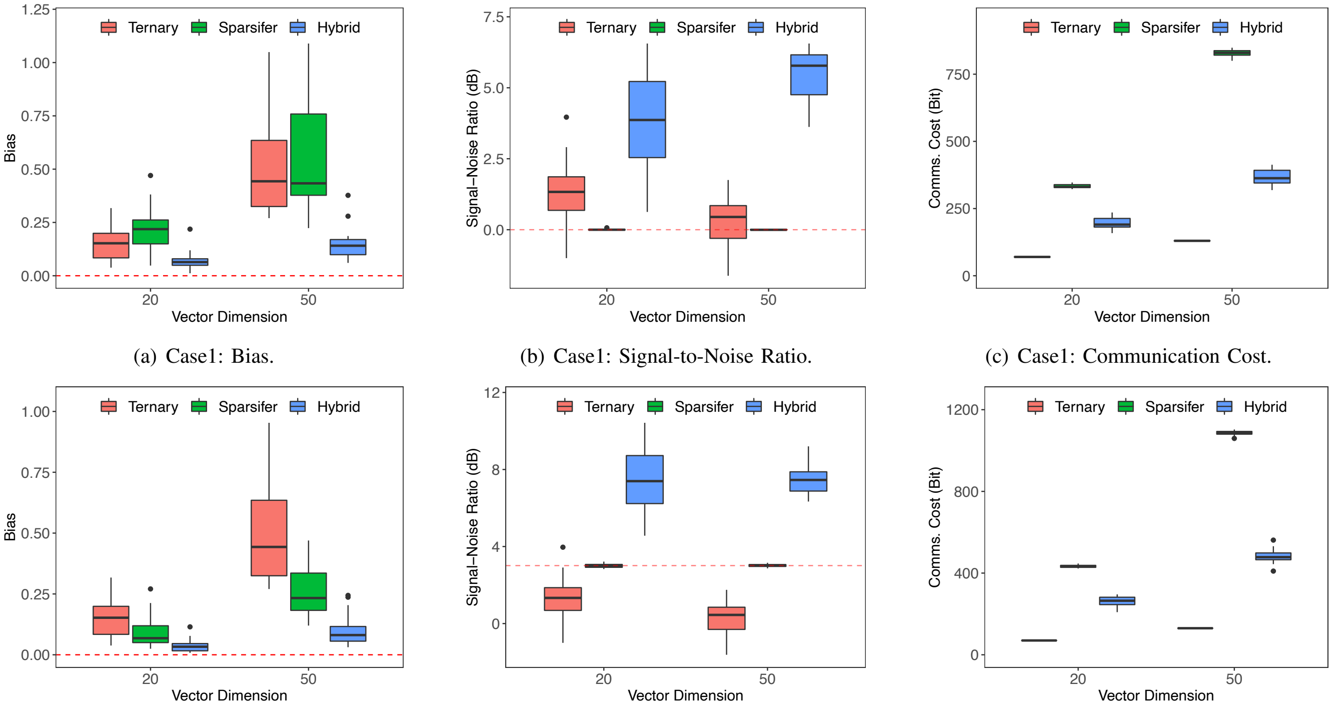
Fig. 1(b) illustrates the convergence of the three algorithms with $\mathbf{W}_1$. We can see that DC-DGD converges with $p = 0.8$ but fails to converge with p chosen from $\{0.3, 0.5\}$. This confirms our Theorem 1: From Example 1 and Theorem 1, the lower bound of p can be derived from $p/(p-1) > (1 - \lambda_N(\mathbf{W}_1))/(1 + \lambda_N(\mathbf{W}_1))$, which is 0.72. Thus, choosing $p \in \{0.3, 0.5\}$ (i.e., $p < 0.72$) violates the convergence condition in Theorem 1. Moreover, we note that, with $p = 0.8$, the convergence speed of the DC-DGD is *almost the same as the original DGD* (the black dashed line). Fig. 1(c) presents the convergence performance of these algorithms with $\mathbf{W}_2$. Following similar derivations, one can show that the lower bound of p is 0.45. In this case, DC-DGD converges for $p = 0.5$ and fails to converge for $p = 0.3$, which confirms Theorem 1 again. In both cases, we can see that DC-DGD converges faster and has smaller variances than ADC-DGD.

2) Compression Operator Comparison: Next, we compare three SNR-constrained compressors: the sparsifier, the ternary compressor, and our proposed hybrid compressor. We generate 20 d-dimensional vectors independently from the



(a) (b) (c)

Fig. 1. (a) The five-node circle network; (b-c) Performance comparsion: Convergence error vs Iteration with the consensus matrices $\mathbf{W}_1$ and $\mathbf{W}_2$, respectively. The black solid curve is the original DGD algorithm. The other curves represent the error averaged over 50 trials and the shaded regions indicate the standard deviations of results over random trials.



(a) Case1: Bias.

(b) Case1: Signal-to-Noise Ratio.

(c) Case1: Communication Cost.

(d) Case2: Bias.

(e) Case2: Signal-to-Noise Ratio.

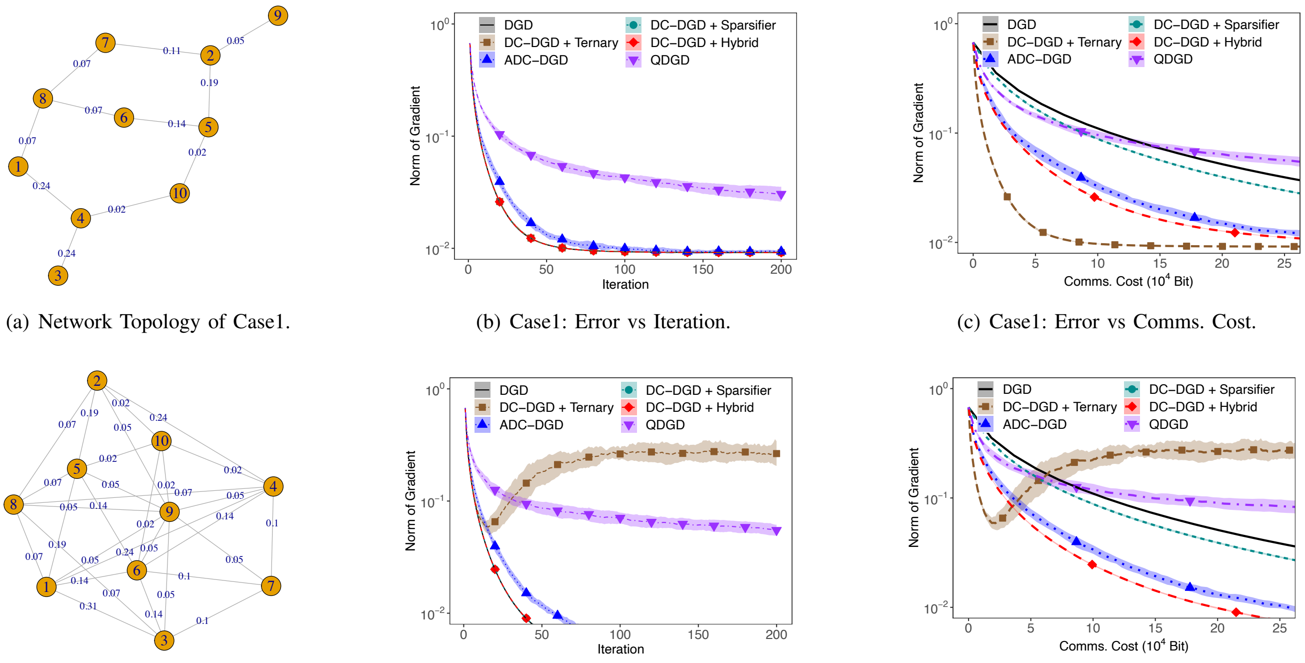
(f) Case2: Communication Cost.

Fig. 2. Comparisons between three compressors: (a)-(c) are the boxplots for the SNR lower bound as 0dB; and (d)-(e) are the boxplots for the SNR lower bound 3 dB. The red dashed lines in (a) and (d) represents 0; the red dashed lines in (b)&(e) present the SNR lower bound 0 dB and 3 dB, respectively.

multivariate Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ with $d \in \{20, 50\}$. We apply three operators on each vector, respectively, and conduct 100 trials. For any $\mathbf{x}$ and the compressed $C(\mathbf{x})$, we evaluate: 1) bias: $\|\mathbb{E}[C(\mathbf{x})] - \mathbf{x}\|$; 2) signal-to-noise ratio (SNR): $\|\mathbf{x}\|^2/\mathrm{Var}[C(\mathbf{x})]$; and 3) communication cost. Here the SNR is corresponding to η in Theorem 1. The smaller bias and the larger the SNR (less noisy), the better the compressor. To calculate the communication cost, we use 32-bit floating numbers and 2-bit ternary numbers. For the sparsifier operator, only one bit is used to represent value 0. Note that SNR is controllable by adjusting p in the sparsifier and our hybrid compressors. To illustrate this advantage, we set the SNR lower bound as 0 dB and 3 dB. In both cases, the parameters are optimized for the largest communication cost savings: For the 3 dB SNR lower bound, we have $p = \frac{2}{3}$ for the sparsifier and $\eta = 2$ for the hybrid compressor; For the 0 dB SNR lower bound, we have $p = \frac{1}{2}$ for the sparsifier and $\eta = 1$ for the hybrid compressor. Boxplots results are illustrated in Fig. 2.

In Fig. 2(a) and 2(d), we can see that our hybrid compressor has the smallest bias, while the bias of the sparsifier increases as p decreases. We can see from Fig. 2(b) and 2(e) that our hybrid compressor can make the SNR larger than the given bound, while the ternary operator cannot. The communication costs are shown in Fig. 2(c) and 2(f). Although the ternary compressor has the lowest cost, it cannot control its SNR. In contrast, our hybrid scheme achieves almost 50% cost savings compared to the sparsifier scheme under all circumstances.

3) Real-World Data Experiments: Lastly, we compare DC-DGD with the original DGD [8], QDGD [11], ADC-DGD [13] with 10-node networks and real-world data.



(a) Network Topology of Case1.

(b) Case1: Error vs Iteration.

(c) Case1: Error vs Comms. Cost.

(d) Network Topology of Case2.

(e) Case2: Error vs Iteration.

(f) Case2: Error vs Comms. Cost.

Fig. 3. (a) an (d) Two ten-node network examples. The consensus weights are shown on the corresponding edges. (b) and (e) Convergence in terms of iterations; (c) and (f) Convergence in terms of communication cost. The curves are averaged over 10 trials and the shaded regions represent the standard deviation of results over random trials.

We consider a classification task on the Spambase dataset from UCI repository [23]. This dataset contains email spam data from 4601 email messages and 57 features. The data are evenly distributed to 10 machines. The local objective $f_i(\mathbf{x})$ is a logistic regression problem with a non-convex regularizer [24]: $-\frac{1}{n_i}\sum_{j=1}^{n_i}[y_{ij}\log(\frac{1}{1+\exp(-\mathbf{x}^\top\zeta_{ij})}) + (1-y_{ij})\log(\frac{\exp(-\mathbf{x}^\top\zeta_{ij})}{1+\exp(-\mathbf{x}^\top\zeta_{ij})})] + \rho\sum_{i=1}^{d}\frac{x_i^2}{1+x_i^2}$, where the label $y_{ij} \in \{0,1\}$, the feature $\zeta_{ij} \in \mathbb{R}^{57}$ and $\rho = 0.1$ in our experiment. For ADC-DGD and QDGD, floating numbers are randomly quantized to integers with the low-precision representation. In our DC-DGD, we test three compressors: the sparsifier, the ternary compressor, and our hybrid compressor. We use 32 bits for the floating numbers, 8 bits for integers (int8), and 2 bits for ternary values. In addition, value 0 is represented by 1 bit in the sparsifier. We use two different network topologies as shown in Figs. 3(a) and 3(d). For the first topology, $\beta = 0.98$ and $\lambda_N = 0.24$; For the second topology, $\beta = 0.88$ and $\lambda_N = -0.37$. The simulation results are shown in Fig. 3.

We can see that DC-DGD with the ternary compressor does not converge under the second topology. This is because the SNR-threshold is *not* controllable under the ternary compressor. Thus, ternary compressor is not a safe choice in DGD-type algorithms. Fig. 3(b) and 3(e) illustrate the convergence rates of the algorithms. We can see that the QDGD has the slowest convergence speed, which is followed by ADC-DGD. Note that DC-DGD, when converged, has *almost the same speed as the original DGD*. Fig. 3(c) and 3(f) compare the communication cost of these algorithms. In Fig. 3(c), we see that the ternary compressor has the lowest communication cost (approximately 10^5 bits to achieve error 10^{-2}). However, ternary compressor does not work in the second network. We can also see that DC-DGD with our hybrid compressor converges in both networks and has the *lowest communication cost* under the second network (approximately 2×10^5 bits to achieve error 10^{-2}). In contrast, ADC-DGD costs 2.5×10^5 bits and other methods cost more than 2.5×10^5 bits. Moreover, we note that our DC-DGD has smallest variance compared to ADC-DGD and QDGD (compare the shaded regions), which suggests that our DC-DGD is more stable.

VI. Conclusion

In this paper, we designed and analyzed a new differential-coded compressed decentralized gradient descent (DC-DGD) algorithm for communication-efficient network-distributed optimization. The key features of our DC-DGD algorithm include: i) DC-DGD works with general compression schemes that are only constrained by SNR (signal-to-noise ratio); ii) By exchanging the differentials between successive iterations (hence the name differential-coded), the DC-DGD algorithm converges at the same $O(1/t)$ rate as the original DGD; iii) DC-DGD enjoys the same low-complexity algorithmic structure as the original DGD algorithm and does not require additional mechanisms to tame compression noise thanks to its *self compression noise reduction effect*. Based on the above theoretical insights, we proposed a new family of hybrid SNR-constrained compressors that integrate sparsifier and ternary operators. We showed that our hybrid compressor has a controllable SNR-threshold and offers a systematic framework to minimize communication costs. Moreover, by leveraging the special problem structure, we developed an efficient greedy algorithm to reduce the communication cost.

References

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

[2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Foundations and Trends in Machine Learning*, vol. 3, no. 11, pp. 1–122, 2011.

[3] J. Konecny, B. McMahan, and D. Ramage, “Federated optimization: Distributed optimization beyond the datacenter,” *arXiv preprint arXiv:1511.03575*, 2015.

[4] A. Nedic, A. Olshevsky, and C. A. Uribe, “Distributed learning for cooperative inference,” *arXiv preprint arXiv:1704.02718*, 2017.

[5] A. Nedic, A. Ozdaglar, and P. A. Parrilo, “Constrained consensus and optimization in multi-agent networks,” *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, 2010.

[6] L. Xiao and S. Boyd, “Optimal scaling of a gradient method for distributed resource allocation,” *Journal of optimization theory and applications*, vol. 129, no. 3, pp. 469–488, 2006.

[7] M. Rabbat and R. Nowak, “Decentralized source localization and tracking wireless sensor networks,” in *Proc. IEEE ICASSP*, vol. 3, 2004, pp. 921–924.

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

[9] A. Garg, S. V. Naidu, H. Yahia, and D. Singh, “Wavelet based resolution enhancement for low resolution satellite images,” in *Proc. 9th IEEE International Conference on Industrial and Information Systems (ICIIS2014)*, Gwalior, India, December Dec 2014.

[10] A. A. Shatnawi and M. N. M. Warip, “Challenges to inter-satellite communication system: A review,” *International Journal of Computer Applications*, vol. 148, no. 6, pp. 22–25, August 2016.

[11] 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.

[12] 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.

[13] X. Zhang, J. Liu, Z. Zhu, and E. S. Bentley, “Compressed distributed gradient descent: Communication-efficient consensus over networks,” in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 2431–2439.

[14] A. Reisizadeh, H. Taheri, A. Mokhtari, H. Hassani, and R. Pedarsani, “Robust and communication-efficient collaborative learning,” in *Advances in Neural Information Processing Systems*, 2019, pp. 8386–8397.

[15] 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.

[16] 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.

[17] 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.

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

[19] 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.

[20] Y. Zhou, Y. Liang, and H. Zhang, “Generalization error bounds with probabilistic guarantee for sgd in nonconvex optimization,” *arXiv preprint arXiv:1802.06903*, 2018.

[21] S. J. Reddi, A. Hefny, S. Sra, B. Poczos, and A. Smola, “Stochastic variance reduction for nonconvex optimization,” in *International conference on machine learning*, 2016, pp. 314–323.

[22] X. Zhang, J. Liu, Z. Zhu, and E. S. Bentley, “Communication-efficient network-distributed optimization with differential-coded compressors,” *arXiv preprint arXiv:1912:03208*, 2019.

[23] D. Dua and C. Graff, “UCI machine learning repository,” 2017. [Online]. Available: http://archive.ics.uci.edu/ml

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