Journal of Machine Learning Research 23 (2022) 1-41 Submitted 10/20; Revised 2/22; Published 8/22

# Variance Reduced EXTRA and DIGing and Their Optimal Acceleration for Strongly Convex Decentralized Optimization

**Huan Li** LIHUANSS@NANKAI.EDU.CN
*Institute of Robotics and Automatic Information Systems*
*College of Artificial Intelligence*
*Nankai University*
*Tianjin 300071, China*

**Zhouchen Lin** * ZLIN@PKU.EDU.CN
*Key Laboratory of Machine Perception*
*School of Artificial Intelligence*
*Peking University*
*Beijing 100871, China*

**Yongchun Fang** FANGYC@NANKAI.EDU.CN
*Institute of Robotics and Automatic Information Systems*
*College of Artificial Intelligence*
*Nankai University*
*Tianjin 300071, China*

**Editor:** Suvrit Sra

## Abstract

We study stochastic decentralized optimization for the problem of training machine learning models with large-scale distributed data. We extend the widely used EXTRA and DIGing methods with variance reduction (VR), and propose two methods: VR-EXTRA and VR-DIGing. The proposed VR-EXTRA requires the time of $\mathcal{O}((\kappa_s+n)\log\frac{1}{\epsilon})$ stochastic gradient evaluations and $\mathcal{O}((\kappa_b+\kappa_c)\log\frac{1}{\epsilon})$ communication rounds to reach precision $\epsilon$, which are the best complexities among the non-accelerated gradient-type methods, where $\kappa_s$ and $\kappa_b$ are the stochastic condition number and batch condition number for strongly convex and smooth problems, respectively, $\kappa_c$ is the condition number of the communication network, and $n$ is the sample size on each distributed node. The proposed VR-DIGing has a little higher communication cost of $\mathcal{O}((\kappa_b+\kappa_c^2)\log\frac{1}{\epsilon})$. Our stochastic gradient computation complexities are the same as the ones of single-machine VR methods, such as SAG, SAGA, and SVRG, and our communication complexities keep the same as those of EXTRA and DIGing, respectively. To further speed up the convergence, we also propose the accelerated VR-EXTRA and VR-DIGing with both the optimal $\mathcal{O}((\sqrt{n\kappa_s}+n)\log\frac{1}{\epsilon})$ stochastic gradient computation complexity and $\mathcal{O}(\sqrt{\kappa_b\kappa_c}\log\frac{1}{\epsilon})$ communication complexity. Our stochastic gradient computation complexity is also the same as the ones of single-machine accelerated VR methods, such as Katyusha, and our communication complexity keeps the same as those of accelerated full batch decentralized methods, such as MSDA. To the best of our knowledge, our accelerated methods are the first to achieve both the optimal stochastic

∗. Z. Lin is also with Institute for Artificial Intelligence, Peking University, Beijing, China, and Peng Cheng Laboratory, Shenzhen, China. Z. Lin is the corresponding author.

gradient computation complexity and communication complexity in the class of gradient-type methods.

**Keywords:** Stochastic decentralized optimization, EXTRA, DIGing, variance reduction, acceleration

## 1. Introduction

Emerging machine learning applications involve huge amounts of data samples, and the data are often distributed across multiple machines for storage and computational reasons. In this paper, we consider the following distributed convex optimization problem with $m$ nodes, and each node has $n$ local training samples:

$$\min_{x\in\mathbb{R}^p} \sum_{i=1}^{m} f_{(i)}(x), \quad \text{where} \quad f_{(i)}(x) = \frac{1}{n}\sum_{j=1}^{n} f_{(i),j}(x), \tag{1}$$

where the local component function $f_{(i),j}$ represents the $j$th sample of node $i$, and it is not accessible by any other node in the communication network. The network is abstracted as a connected and undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, ..., m\}$ is the set of nodes, and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is the set of edges. Nodes $i$ and $j$ can send information to each other if and only if $(i, j) \in \mathcal{E}$. The goal of the networked nodes is to cooperatively solve problem (1) via local computation and communication, that is, each node $i$ makes its decision only based on the local computations on $f_{(i)}$, for example, the gradient, and the local information received from its neighbors in the network.

When the local data size $n$ is large, the cost of computing the full batch gradient $\nabla f_{(i)}$ at each iteration is expensive. To address the issue of large-scale distributed data, stochastic decentralized algorithms are often used to solve problem (1), where each node only randomly samples one component gradient at each iteration (extendable to the mini-batch settings with more than one randomly selected component). Most decentralized algorithms alternate between computations and communications. Thus to compare the performance of such methods, two measures are used: the number of communication rounds and the number of stochastic gradient evaluations, where one communication round allows each node to send information to their neighbors, for example, $\mathcal{O}(1)$ vectors of size $p$, and one stochastic gradient evaluation refers to computing the randomly sampled $\nabla f_{(i),j}$ for all $i \in \mathcal{V}$ in parallel (Kovalev et al., 2020b).

Although stochastic decentralized optimization has been a hot topic in recent years, and several algorithms have been proposed, to the best of our knowledge, in the class of algorithms not relying on the expensive dual gradient evaluations, there is no algorithm optimal in both the number of communication rounds and the number of stochastic gradient evaluations (Kovalev et al., 2020b), where "optimal" means matching the corresponding lower bounds. In this paper, we extend two widely used decentralized algorithms of EXTRA (Shi et al., 2015) and DIGing (Nedić et al., 2017; Qu and Li, 2018), which have sparked a lot of interest in the distributed optimization community, to stochastic decentralized optimization by combining them with the powerful variance reduction technique. Furthermore, we propose two accelerated stochastic decentralized algorithms, which are optimal in the above two measures of communications and stochastic gradient computations.

### 1.1 Notations and Assumptions

Denote $x_{(i)} \in \mathbb{R}^p$ to be the local variable for node $i$. To simplify the algorithm description in a compact form, we introduce the aggregate objective function $f(\mathbf{x})$ with its aggregate variable $\mathbf{x}$ and aggregate gradient $\nabla f(\mathbf{x})$ as

$$\mathbf{x} = \begin{pmatrix} x_{(1)}^T \\ \vdots \\ x_{(m)}^T \end{pmatrix}, \quad f(\mathbf{x}) = \sum_{i=1}^m f_{(i)}(x_{(i)}), \quad \nabla f(\mathbf{x}) = \begin{pmatrix} \nabla f_{(1)}(x_{(1)})^T \\ \vdots \\ \nabla f_{(m)}(x_{(m)})^T \end{pmatrix}. \tag{2}$$

Denote $x^*$ to be the optimal solution of problem (1), and let $\mathbf{x}^* = \mathbf{1}(x^*)^T$, where $\mathbf{1}$ is the column vector of $m$ ones. Denote $I$ as the identity matrix, and $\mathcal{N}_{(i)}$ as the neighborhood of node $i$. Denote $\text{Ker}(U) = \{x \in \mathbb{R}^m | Ux = 0\}$ as the kernel space of matrix $U \in \mathbb{R}^{m\times m}$, and $\text{Span}(U) = \{y \in \mathbb{R}^m | y = Ux, \forall x \in \mathbb{R}^m\}$ as the linear span of all the columns of $U$. For matrices, we denote $\|\cdot\|$ as the Frobenius norm for simplicity without ambiguity, since it is the only matrix norm we use in this paper. The notation $A \succeq B$ means $A - B$ is positive semidefinite.

We make the following assumptions for the functions in (1).

**Assumption 1** *Each $f_{(i)}(x)$ is $L_{(i)}$-smooth and $\mu$-strongly convex. Each $f_{(i),j}(x)$ is $L_{(i),j}$-smooth and convex.*

We say a function $g(x)$ is $L$-smooth if its gradient satisfies $\|\nabla g(\mathbf{y}) - \nabla g(\mathbf{x})\| \le L\|\mathbf{y} - \mathbf{x}\|$. Motivated by (Hendrikx et al., 2021, 2020), we define several notations as follows:

$$L_f = \max_i L_{(i)}, \quad \overline{L}_{(i)} = \frac{1}{n}\sum_{j=1}^n L_{(i),j}, \quad \overline{L}_f = \max_i \overline{L}_{(i)}, \quad \kappa_s = \frac{\overline{L}_f}{\mu}, \quad \kappa_b = \frac{L_f}{\mu}. \tag{3}$$

Then $f(\mathbf{x})$ is also $\mu$-strongly convex and $L_f$-smooth. It always holds that $L_{(i)} \le \overline{L}_{(i)} \le nL_{(i)}$[1], which further gives

$$L_f \le \overline{L}_f \le nL_f \quad \text{and} \quad \kappa_b \le \kappa_s \le n\kappa_b. \tag{4}$$

We follow (Hendrikx et al., 2021) to call $\kappa_b$ the batch condition number, and $\kappa_s$ the stochastic condition number, which are classical quantities in the analysis of batch optimization methods and finite-sum optimization methods, respectively. Generally, we have $\kappa_s \ll n\kappa_b$, see (Allen-Zhu, 2018) for the example and analysis.

In decentralized optimization, communication is often represented as a matrix multiplication with a weight matrix $W \in \mathbb{R}^{m\times m}$. We make the following assumptions for this weight matrix associated to the network[2].

**Assumption 2**

1. *$W_{i,j} \neq 0$ if and only if agents $i$ and $j$ are neighbors or $i = j$. Otherwise, $W_{i,j} = 0$.*
2. *$W = W^T$, $I \succeq W \succeq \omega I$, and $W\mathbf{1} = \mathbf{1}$.*

1. See footnote 14 in (Allen-Zhu, 2018) for the analysis.
2. The weights can be assigned heuristically or optimized given the fixed graph structure (Boyd et al., 2004).

We let $\omega = 0$ for EXTRA, and $\omega = \frac{\sqrt{2}}{2}$ for DIGing. We can relax $\omega$ to be any small positive constant for DIGing[3], and fix it to $\frac{\sqrt{2}}{2}$ to simplify the analysis. For EXTRA, we can also relax the condition to $I \succeq W \succeq (-1+\delta)I$ for any small positive constant $\delta$[4]. Part 2 of Assumption 2 implies that the eigenvalues of $W$ lie in $[\omega, 1]$, and its largest one $\sigma_1(W)$ equals 1. Moreover, if the network is connected, we have $\sigma_2(W) < 1$, where $\sigma_2(W)$ means the second largest eigenvalue. We often use

$$\kappa_c = \frac{1}{1-\sigma_2(W)} \tag{5}$$

as the condition number of the communication network, which upper bounds the ratio between the largest eigenvalue and the smallest non-zero eigenvalue of $(I-W)$, which is a gossip matrix (Scaman et al., 2017).

As will be introduced in the next section, we often use $\kappa_b$ and $\kappa_c$ to describe the number of communication rounds, and $\kappa_s$ for the number of stochastic gradient evaluations in stochastic decentralized optimization.

### 1.2 Literature Review

In this section, we give a brief review for the decentralized and stochastic methods, as well as their combination. Table 1 sums up the complexities of the representative ones.

#### 1.2.1 Full Batch Decentralized Algorithms

Distributed optimization has gained significant attention for a long time (Bertsekas, 1983; Tsitsiklis et al., 1986). The modern distributed gradient descent (DGD) was proposed in (Nedić and Ozdaglar, 2009) for the general network topology, and was further extended in (Nedić, 2011; Ram et al., 2010; Yuan et al., 2016). These algorithms are usually slow due to the diminishing step-size, and suffer from the sublinear convergence even for strongly convex and smooth objectives. To avoid the diminishing step-size and speed up the convergence, several methods relying on tracking the differences of gradients have been proposed. Typical examples include EXTRA (Shi et al., 2015), DIGing (Nedić et al., 2017; Qu and Li, 2018), NIDS (Li et al., 2019), and other similar algorithms (Xu et al., 2015; Xin et al., 2018). Especially, EXTRA (Li and Lin, 2020) and NIDS (Li et al., 2019) have the $\mathcal{O}((\kappa_b+\kappa_c)\log\frac{1}{\epsilon})$ complexity both in communications and full batch gradient evaluations to solve problem (1) to reach precision $\epsilon$, which is the best among the non-accelerated algorithms. DIGing has a slight higher complexity of $\mathcal{O}((\kappa_b+\kappa_c^2)\log\frac{1}{\epsilon})$ (Alghunaim et al., 2021). Another typical class of distributed algorithms is based on the Lagrangian function, and they work with the Fenchel dual. Examples include the dual ascent (Terelius et al., 2011; Scaman et al., 2017; Uribe et al., 2020), ADMM (Iutzeler et al., 2016; Makhdoumi and Ozdaglar, 2017; Aybat et al., 2018), and the primal-dual method (Lan et al., 2020; Scaman et al., 2018; Hong et al., 2017; Jakovetić, 2019). However, the dual-based methods often need to compute the gradient of the Fenchel conjugate of the local functions, called dual gradient in the sequel, which is expensive.

3. In this case, condition (15) is relaxed to $\|V\mathbf{x}\|^2 \le (1-\omega^2)\|\mathbf{x}\|^2$. By the similar proofs of Theorem 2, we can obtain the $\mathcal{O}(\frac{1}{\omega^2}(\kappa_b+\kappa_c^2)\log\frac{1}{\epsilon})$ complexity for DIGing.
4. In this case, the complexity of EXTRA becomes $\mathcal{O}(\frac{1}{\delta}(\kappa_b+\kappa_c)\log\frac{1}{\epsilon})$.

Nesterov's acceleration technique is an efficient approach to speed up the convergence of first-order methods, and it has also been successfully applied to decentralized optimization. Typical examples include the distributed Nesterov gradient with consensus (Jakovetić et al., 2014), the distributed Nesterov gradient descent (Qu and Li, 2020), the multi-step dual accelerated method (MSDA) (Scaman et al., 2017, 2019), accelerated penalty method (Li et al., 2020b), accelerated EXTRA (Li and Lin, 2020), and the accelerated proximal alternating predictor-corrector method (APAPC) (Kovalev et al., 2020b). Some of these methods have suboptimal computation complexity, and Chebyshev acceleration (CA) (Arioli and Scott, 2014) is a powerful technique to further reduce the computation cost. Scaman et al. (2017, 2019) proved the $\Omega(\sqrt{\kappa_b\kappa_c}\log\frac{1}{\epsilon})$ lower bound on the number of communication rounds and the $\Omega(\sqrt{\kappa_b}\log\frac{1}{\epsilon})$ lower bound on the number of full batch gradient evaluations, which means that any first-order full batch decentralized methods cannot be faster than these bounds. The MSDA and APAPC methods with CA achieve these lower bounds.

### 1.2.2 Stochastic Algorithms on a Single Machine

Stochastic gradient descent (SGD) has been the workhorse in machine learning. However, since the variance of the noisy gradient will not go to zero, SGD often suffers from the slow sublinear convergence. Variance reduction (VR) was designed to reduce the negative effect of the noise, which can improve the stochastic gradient computation complexity to $\mathcal{O}((\kappa_s+n)\log\frac{1}{\epsilon})$. On the other hand, full batch methods, such as gradient descent, require $\mathcal{O}(\kappa_b\log\frac{1}{\epsilon})$ iterations, and thus $\mathcal{O}(n\kappa_b\log\frac{1}{\epsilon})$ individual gradient evaluations for finite-sum problems with $n$ samples, which may be much larger than $\mathcal{O}((n+\kappa_s)\log\frac{1}{\epsilon})$ when $\kappa_s \ll n\kappa_b$. Representative examples of VR methods include SAG (Schmidt et al., 2017), SAGA (Defazio et al., 2014), and SVRG (Johnson and Zhang, 2013; Xiao and Zhang, 2014). We can further accelerate the VR methods to the $\mathcal{O}((\sqrt{n\kappa_s}+n)\log\frac{1}{\epsilon})$ stochastic gradient computation complexity by Nesterov's acceleration technique. Examples include Katyusha (Allen-Zhu, 2018) and its extensions in (Zhou et al., 2019; Kovalev et al., 2020a). Other accelerated stochastic algorithms can be found in (Lan and Zhou, 2018; Lin et al., 2018; Fercoq and Richtárik, 2015; Lin et al., 2015). Lan and Zhou (2018) proved the $\Omega((\sqrt{n\kappa_s}+n)\log\frac{1}{\epsilon})$ lower bound for strongly convex and smooth stochastic optimization, and Katyusha achieves this lower bound.

### 1.2.3 Stochastic Decentralized Algorithms

To address the issue of large-scale distributed data, Chen and Sayed (2012) and Ram et al. (2010) extended the DGD method to the distributed stochastic gradient descent (DSGD). To further improve the convergence of stochastic decentralized algorithms, Pu and Nedić (2021) combined DSGD with gradient tracking, Mokhtari and Ribeiro (2016) combined EXTRA with SAGA, and proposed the decentralized double stochastic averaging gradient algorithm, Xin et al. (2020b) combined gradient tracking with the VR technique, and two algorithms are proposed, namely, GT-SAGA and GT-SVRG. Li et al. (2020a) generalized the approximate Newton-type method called DANE with gradient tracking and variance reduction. See (Xin et al., 2020a) for a detailed review for the non-accelerated stochastic decentralized algorithms. Hendrikx et al. (2021) proposed an accelerated decentralized stochastic algorithm called ADFS for problems with finite-sum structures, which achieves the optimal $\mathcal{O}(\sqrt{\kappa_b\kappa_c}\log\frac{1}{\epsilon})$

communication complexity. However, ADFS is a dual-based method, and it needs to compute the dual gradient at each iteration, which is expensive. Recently, Hendrikx et al. (2020) further proposed a dual-free decentralized method with variance reduction, called DVR, which achieves the $\mathcal{O}(\kappa_b\sqrt{\kappa_c}\log\frac{1}{\epsilon})$ communication complexity and the $\mathcal{O}((\kappa_s+n)\log\frac{1}{\epsilon})$ stochastic gradient computation complexity. These complexities can be further improved to $\widetilde{\mathcal{O}}(\sqrt{\kappa_b\kappa_c}\sqrt{\frac{n\kappa_b}{\kappa_s}}\log\frac{1}{\epsilon})$ and $\widetilde{\mathcal{O}}((\sqrt{n\kappa_s}+n)\log\frac{1}{\epsilon})$ by the Catalyst acceleration (Lin et al., 2018), respectively, where $\widetilde{\mathcal{O}}$ hides the poly-logarithmic factor, which is at least $\mathcal{O}(\log\kappa_b)$[5]. We see that DVR-Catalyst achieves the optimal stochastic gradient computation complexity up to log factor. However, its communication cost is increased by a factor $\mathcal{O}(\sqrt{\frac{n\kappa_b}{\kappa_s}})$ compared with ADFS, which is always much larger than 1 in machine learning applications[6], and it is of the $\mathcal{O}(\sqrt{n})$ order in the worst case. Hendrikx et al. (2021) proved the $\Omega((\sqrt{n\kappa_s}+n)\log\frac{1}{\epsilon})$ stochastic gradient computation and the $\Omega(\sqrt{\kappa_b\kappa_c}\log\frac{1}{\epsilon})$ communication lower bounds. The study on acceleration for the general stochastic problems without finite-sum structures can be found in (Dvinskikh and Gasnikov, 2021), (Gorbunov et al., 2019), and (Fallah et al., 2019). See the recent review (Gorbunov et al., 2022) for the accelerated stochastic decentralized algorithms.

### 1.3 Contributions

Although both the decentralized methods and stochastic methods have been well studied, their combination still has much work to do. For example, as far as we know, there is no gradient-type stochastic decentralized method achieving both the state-of-the-art communication and stochastic gradient computation complexities (either accelerated or non-accelerated) of the decentralized methods and stochastic methods simultaneously. In this paper we aim to address this issue. Our contributions include:

1. We extend the widely used EXTRA and DIGing methods to deal with large-scale distributed data by combining them with the powerful VR technique. We prove the $\mathcal{O}((\kappa_s+n)\log\frac{1}{\epsilon})$ stochastic gradient computation complexity and the $\mathcal{O}((\kappa_b+\kappa_c)\log\frac{1}{\epsilon})$ communication complexity for VR-EXTRA, which are the best complexities among the non-accelerated stochastic decentralized methods as far as we know. The stochastic gradient computation complexity is the same as the single-machine VR methods, while the communication complexity is the same as the full batch EXTRA. For VR-DIGing, we establish the $\mathcal{O}((\kappa_s+n)\log\frac{1}{\epsilon})$ stochastic gradient computation complexity and the $\mathcal{O}((\kappa_b+\kappa_c^2)\log\frac{1}{\epsilon})$ communication complexity. The latter one is a little worse than that of VR-EXTRA on the dependence of $\kappa_c$. Due to the parallelism across $m$ nodes, running VR-EXTRA and VR-DIGing with $mn$ samples is as fast as running the single-machine VR methods with $n$ samples.

2. To further speed up the convergence, we combine EXTRA and DIGing with the accelerated VR technique. The proposed Acc-VR-EXTRA achieves the optimal $\mathcal{O}((\sqrt{n\kappa_s}+n)\log\frac{1}{\epsilon})$ stochastic gradient computation complexity and the optimal

5. See Proposition 17 in (Lin et al., 2018) and Corollary 7 in (Li and Lin, 2020).
6. As discussed in Section 1.2.2 for the comparison between the VR methods and gradient descent, stochastic methods have no advantage when $\kappa_s\approx n\kappa_b$. We often assume $\kappa_s\ll n\kappa_b$.

Table 1: Comparisons of various state-of-the-art decentralized and stochastic algorithms. See (3) and (5) for the definitions of $\kappa_b$, $\kappa_s$, and $\kappa_c$. $\widetilde{\mathcal{O}}$ hides the poly-logarithmic factors. The complexities of Acc-VR-EXTRA and Acc-VR-DIGing hold under some conditions to restrict the size of $\kappa_c$. See part 1 of Remarks 9 and 12. Acc-VR-EXTRA-CA and Acc-VR-DIGing-CA remove these restrictions.

| Methods | stochastic gradient computation complexity | communication complexity | dual gradient based ? |
|---|---|---|---|
| Full batch decentralized algorithms | | | |
| EXTRA (Shi et al., 2015) (Li and Lin, 2020) | $\mathcal{O}\left(n\left(\kappa_b+\kappa_c\right)\log\frac{1}{\epsilon}\right)$ | $\mathcal{O}\left(\left(\kappa_b+\kappa_c\right)\log\frac{1}{\epsilon}\right)$ | no |
| DIGing (Nedić et al., 2017) (Alghunaim et al., 2021) | $\mathcal{O}\left(n\left(\kappa_b+\kappa_c^2\right)\log\frac{1}{\epsilon}\right)$ | $\mathcal{O}\left(\left(\kappa_b+\kappa_c^2\right)\log\frac{1}{\epsilon}\right)$ | no |
| MSDA+CA (Scaman et al., 2017) | $\mathcal{O}\left(n\sqrt{\kappa_b}\log\frac{1}{\epsilon}\right)$ | $\mathcal{O}\left(\sqrt{\kappa_b\kappa_c}\log\frac{1}{\epsilon}\right)$ | yes |
| APAPC+CA (Kovalev et al., 2020b) | $\mathcal{O}\left(n\sqrt{\kappa_b}\log\frac{1}{\epsilon}\right)$ | $\mathcal{O}\left(\sqrt{\kappa_b\kappa_c}\log\frac{1}{\epsilon}\right)$ | no |
| Stochastic algorithms on a single machine | | | |
| VR methods (Schmidt et al., 2017) (Defazio et al., 2014) (Johnson and Zhang, 2013) | $\mathcal{O}\left(\left(\kappa_s+n\right)\log\frac{1}{\epsilon}\right)$ | \ | no |
| Katyusha (Allen-Zhu, 2018) | $\mathcal{O}\left(\left(\sqrt{n\kappa_s}+n\right)\log\frac{1}{\epsilon}\right)$ | \ | no |
| Stochastic decentralized algorithms | | | |
| GT-SAGA (Xin et al., 2020b) | $\mathcal{O}\left(\left(\kappa_s^2\kappa_c^2+n\right)\log\frac{1}{\epsilon}\right)$ | $\mathcal{O}\left(\left(\kappa_s^2\kappa_c^2+n\right)\log\frac{1}{\epsilon}\right)$ | no |
| GT-SVRG (Xin et al., 2020b) | $\mathcal{O}\left(\left(\kappa_s^2\kappa_c^2\log\kappa_s+n\right)\log\frac{1}{\epsilon}\right)$ | $\mathcal{O}\left(\left(\kappa_s^2\kappa_c^2\log\kappa_s+n\right)\log\frac{1}{\epsilon}\right)$ | no |
| ADFS (Hendrikx et al., 2021) | $\mathcal{O}\left(\left(\sqrt{n\kappa_s}+n\right)\log\frac{1}{\epsilon}\right)$ | $\mathcal{O}\left(\sqrt{\kappa_b\kappa_c}\log\frac{1}{\epsilon}\right)$ | yes |
| DVR+CA (Hendrikx et al., 2020) | $\mathcal{O}\left(\left(\kappa_s+n\right)\log\frac{1}{\epsilon}\right)$ | $\mathcal{O}\left(\kappa_b\sqrt{\kappa_c}\log\frac{1}{\epsilon}\right)$ | no |
| DVR+Catalyst (Hendrikx et al., 2020) | $\widetilde{\mathcal{O}}\left(\left(\sqrt{n\kappa_s}+n\right)\log\frac{1}{\epsilon}\right)$ | $\widetilde{\mathcal{O}}\left(\sqrt{\kappa_b\kappa_c}\sqrt{\frac{n\kappa_b}{\kappa_s}}\log\frac{1}{\epsilon}\right)$ | no |
| Lower bounds (Hendrikx et al., 2021) | $\Omega\left(\left(\sqrt{n\kappa_s}+n\right)\log\frac{1}{\epsilon}\right)$ | $\Omega\left(\sqrt{\kappa_b\kappa_c}\log\frac{1}{\epsilon}\right)$ | \ |
| Our results for stochastic decentralized optimization | | | |
| VR-EXTRA | $\mathcal{O}\left(\left(\kappa_s+n\right)\log\frac{1}{\epsilon}\right)$ | $\mathcal{O}\left(\left(\kappa_b+\kappa_c\right)\log\frac{1}{\epsilon}\right)$ | no |
| VR-DIGing | $\mathcal{O}\left(\left(\kappa_s+n\right)\log\frac{1}{\epsilon}\right)$ | $\mathcal{O}\left(\left(\kappa_b+\kappa_c^2\right)\log\frac{1}{\epsilon}\right)$ | no |
| Acc-VR-EXTRA | $\mathcal{O}\left(\left(\sqrt{n\kappa_s}+n\right)\log\frac{1}{\epsilon}\right)$ | $\mathcal{O}\left(\sqrt{\kappa_b\kappa_c}\log\frac{1}{\epsilon}\right)$ | no |
| Acc-VR-DIGing | $\mathcal{O}\left(\left(\sqrt{n\kappa_s}+n\right)\log\frac{1}{\epsilon}\right)$ | $\mathcal{O}\left(\kappa_c\sqrt{\kappa_b}\log\frac{1}{\epsilon}\right)$ | no |
| Acc-VR-EXTRA+CA | $\mathcal{O}\left(\left(\sqrt{n\kappa_s}+n\right)\log\frac{1}{\epsilon}\right)$ | $\mathcal{O}\left(\sqrt{\kappa_b\kappa_c}\log\frac{1}{\epsilon}\right)$ | no |
| Acc-VR-DIGing+CA | $\mathcal{O}\left(\left(\sqrt{n\kappa_s}+n\right)\log\frac{1}{\epsilon}\right)$ | $\mathcal{O}\left(\sqrt{\kappa_b\kappa_c}\log\frac{1}{\epsilon}\right)$ | no |

$\mathcal{O}(\sqrt{\kappa_b\kappa_c}\log\frac{1}{\epsilon})$ communication complexity under some mild conditions to restrict the size of $\kappa_c$. The proposed Acc-VR-DIGing has the optimal $\mathcal{O}((\sqrt{n\kappa_s}+n)\log\frac{1}{\epsilon})$ stochastic gradient computation complexity and the $\mathcal{O}(\kappa_c\sqrt{\kappa_b}\log\frac{1}{\epsilon})$ communication complexity with a little worse dependence on $\kappa_c$. The two methods are implemented in a single loop, and thus they are practical. We further combine Acc-VR-EXTRA and Acc-VR-DIGing with the Chebyshev acceleration to remove the restrictions on the size of $\kappa_c$, and improve the communication complexity of Acc-VR-DIGing to be optimal. Our complexities do not hide any poly-logarithmic factor. To the best of our knowledge, our methods are the first to exactly achieve both the optimal stochastic gradient computation complexity and the communication complexity in the class of gradient-type methods.

Table 1 summarizes the complexity comparisons to the state-of-the-art stochastic decentralized methods. Our VR-EXTRA has the same stochastic gradient computation complexity as DVR-CA, but our communication cost is lower than theirs when $\kappa_c \leq \mathcal{O}(\kappa_b^2)$. On the other hand, by combining with Chebyshev acceleration, our VR-EXTRA and VR-DIGing can also obtain the $\mathcal{O}(\kappa_b\sqrt{\kappa_c}\log\frac{1}{\epsilon})$ communication complexity. For the accelerated methods, our Acc-VR-EXTRA-CA and Acc-VR-DIGing-CA outperform DVR-Catalyst on the stochastic gradient computation complexity at least by the poly-logarithmic factor $\mathcal{O}(\log\kappa_b)$, and our communication cost is also lower than that of DVR-Catalyst by the factor $\mathcal{O}\left(\sqrt{\frac{n\kappa_b}{\kappa_s}}\right)$. On the other hand, DVR and its Catalyst acceleration require $\mathcal{O}(np)$ memory at each node, while our methods only need $\mathcal{O}(p)$ memory[7]. Although ADFS has the same complexities as our Acc-VR-EXTRA-CA and Acc-VR-DIGing-CA, our methods are gradient-type methods, while theirs requires to compute the dual gradient at each iteration, which is much more expensive.

## 2. Non-accelerated Variance Reduced EXTRA and DIGing

We first review the classical EXTRA and DIGing methods in Section 2.1. Then we develop the variance reduced EXTRA and DIGing in Sections 2.2 and 2.3. At last, we discuss the complexities of the proposed methods in Section 2.4.

### 2.1 Review of EXTRA and DIGing

A traditional way to analyze the decentralized optimization model is to write problem (1) in the following equivalent manner:

$$\min_{x_{(1)},\cdots,x_{(m)}} \sum_{i=1}^{m} f_{(i)}(x_{(i)}), \quad \text{s.t.} \quad x_{(1)} = x_{(2)} = \cdots = x_{(m)}.$$

Following (Alghunaim et al., 2021) and using the notations in (2), we further reformulate the above problem as the following linearly constrained problem:

$$\min_{\mathbf{x}} f(\mathbf{x}) + \frac{1}{2\alpha}\|V\mathbf{x}\|^2, \quad \text{s.t.} \quad U\mathbf{x} = 0, \tag{6}$$

7. This is similar to the memory cost comparison between SAG/SAGA and SVRG.

where the symmetric matrices $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{m\times m}$ satisfy

$$U\mathbf{x} = 0 \Leftrightarrow x_{(1)} = \cdots = x_{(m)} \quad \text{and} \quad V\mathbf{x} = 0 \Leftrightarrow x_{(1)} = \cdots = x_{(m)}. \tag{7}$$

where $\frac{1}{2\alpha}\|V\mathbf{x}\|^2$ can be regarded as the augmented term in the augmented Lagrange method (Bertsekas, 1982), which may speed up the convergence than the methods based on the pure Lagrangian function. Introducing the following augmented Lagrangian function

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \frac{1}{2\alpha}\|V\mathbf{x}\|^2 + \frac{1}{\alpha}\langle U\mathbf{x}, \lambda\rangle,$$

we can apply the basic gradient method with a step-size $\alpha$ in the Gauss−Seidel-like order to compute the saddle point of problem (6), which leads to the following iterations (Alghunaim et al., 2021; Nedić et al., 2017):

$$\begin{aligned} \mathbf{x}^{k+1} &= \mathbf{x}^k - \left(\alpha\nabla f(\mathbf{x}^k) + U\lambda^k + V^2\mathbf{x}^k\right), \\ \lambda^{k+1} &= \lambda^k + U\mathbf{x}^{k+1}. \end{aligned} \tag{8}$$

Iteration (8) is a unified algorithmic framework, and different choices of $U$ and $V$ give different methods (Alghunaim et al., 2021). Specifically, when we choose $U = \sqrt{\frac{I-W}{2}}$ and $V = \sqrt{\frac{I-W}{2}}$, (8) reduces to the famous EXTRA algorithm (Shi et al., 2015), which consists of the following iterations:

$$\mathbf{x}^{k+1} = (I+W)\mathbf{x}^k - \frac{I+W}{2}\mathbf{x}^{k-1} - \alpha\left(\nabla f(\mathbf{x}^k) - \nabla f(\mathbf{x}^{k-1})\right).$$

When we choose $U = I - W$ and $V = \sqrt{I-W^2}$, (8) reduces to the DIGing (Nedić et al., 2017) method with the following iterations:

$$\begin{aligned} \mathbf{s}^{k+1} &= W\mathbf{s}^k + \nabla f(\mathbf{x}^k) - \nabla f(\mathbf{x}^{k-1}), \\ \mathbf{x}^{k+1} &= W\mathbf{x}^k - \alpha\mathbf{s}^{k+1}. \end{aligned}$$

Both EXTRA and DIGing rely on tracking the differences of gradients at each iteration.

### 2.2 Development of VR-EXTRA and VR-DIGing

Now, we come to extend EXTRA and DIGing with the variance reduction technique proposed in SVRG (Johnson and Zhang, 2013). Specifically, SVRG maintains a snapshot vector $w_{(i)}^k$ after several SGD iterations, and keeps an iterative estimator $\widetilde{\nabla} f_{(i)}(x_{(i)}^k) = \nabla f_{(i),j}(x_{(i)}^k) - \nabla f_{(i),j}(w_{(i)}^k) + \nabla f_{(i)}(w_{(i)}^k)$ of the full batch gradient for some randomly selected $j$. When extending EXTRA and DIGing to stochastic decentralized optimization, a straightforward idea is to replace the local gradient $\nabla f_{(i)}(x_{(i)}^k)$ in (8) by its VR estimator $\widetilde{\nabla} f_{(i)}(x_{(i)}^k)$. However, in this way the resultant algorithm needs the same number of stochastic gradient evaluations and communication rounds to precision $\epsilon$. As summarized in Table 1, our goal is to provide computation and communication complexities matching those of SVRG and EXTRA/DIGing, respectively, which are not equal. To address this issue, we use the mini-batch VR technique,

that is, select $b$ independent samples with replacement as a mini-batch $\mathbb{S}_{(i)}$, and use this mini-batch to update the VR estimator. By carefully choosing the mini-batch size $b$, we can balance the communication and stochastic gradient computation costs. Moreover, to simplify the algorithm development and analysis, we adopt the loopless SVRG proposed in (Kovalev et al., 2020a). Combining the above ideas, we have the following VR variant of (8) described in a distributed way:

$$\nabla_{(i)}^k = \frac{1}{b}\sum_{j\in\mathbb{S}_{(i)}^k}\frac{1}{np_{(i),j}}\left(\nabla f_{(i),j}(x_{(i)}^k) - \nabla f_{(i),j}(w_{(i)}^k)\right) + \nabla f_{(i)}(w_{(i)}^k), \quad \forall i, \tag{9a}$$

$$x_{(i)}^{k+1} = x_{(i)}^k - \left(\alpha\nabla_{(i)}^k + \sum_{j\in\mathcal{N}_{(i)}} U_{ij}\lambda_{(j)}^k + \sum_{j\in\mathcal{N}_{(i)}}(V^2)_{ij}x_{(j)}^k\right), \quad \forall i, \tag{9b}$$

$$\lambda_{(i)}^{k+1} = \lambda_{(i)}^k + \sum_{j\in\mathcal{N}_{(i)}} U_{ij}x_{(j)}^{k+1}, \quad \forall i, \tag{9c}$$

$$w_{(i)}^{k+1} = \begin{cases} x_{(i)}^k \text{ with probability } \frac{b}{n}, \\ w_{(i)}^k \text{ with probability } 1-\frac{b}{n}, \end{cases} \quad \forall i, \tag{9d}$$

where the mini-batch VR estimator update rule (9a) is motivated by (Allen-Zhu, 2018), in which each sample $j$ on node $i$ is selected with probability $p_{(i),j} = \frac{L_{(i),j}}{\sum_{j=1}^n L_{(i),j}}$. The probabilistic update of the snapshot vector in (9d) is motivated by (Kovalev et al., 2020a), in which we update the full batch gradient $\nabla f_{(i)}(w_{(i)}^{k+1})$ if $w_{(i)}^{k+1} = x_{(i)}^k$; otherwise, we use the old one. Steps (9b) and (9c) come from (8), but replacing the local gradients by their VR estimators. In steps (9a) and (9d), each node selects $\mathbb{S}_{(i)}^k$ and computes $w_{(i)}^{k+1}$ independent of the other nodes.

At last, we write (9a)-(9d) in the EXTRA/DIGing style. Similar to (2), we denote

$$\nabla^k = \begin{pmatrix} (\nabla_{(1)}^k)^T \\ \vdots \\ (\nabla_{(m)}^k)^T \end{pmatrix} \tag{10}$$

to simplify the algorithm description. From steps (9b) and (9c), we have

$$\mathbf{x}^{k+1} = (2I - U^2 - V^2)\mathbf{x}^k - (I - V^2)\mathbf{x}^{k-1} - \alpha\left(\nabla^k - \nabla^{k-1}\right) \tag{11}$$

in the compact form. Plugging $U = \sqrt{\frac{I-W}{2}}$ and $V = \sqrt{\frac{I-W}{2}}$ into (11), we have

$$\mathbf{x}^{k+1} = (I+W)\mathbf{x}^k - \frac{I+W}{2}\mathbf{x}^{k-1} - \alpha\left(\nabla^k - \nabla^{k-1}\right),$$

which is the VR variant of EXTRA, called VR-EXTRA. Plugging $U = I - W$ and $V = \sqrt{I-W^2}$ into (11), we have

$$\mathbf{x}^{k+1} = 2W\mathbf{x}^k - W^2\mathbf{x}^{k-1} - \alpha\left(\nabla^k - \nabla^{k-1}\right),$$

**Algorithm 1** VR-EXTRA and VR-DIGing

Initialize: $x_{(i)}^0 = w_{(i)}^0 = x_{int}$, $\lambda_{(i)}^0 = 0$, compute $x_{(i)}^1$ and $w_{(i)}^1$ for all $i$ by (9b) and (9d), respectively. Let $\alpha = \mathcal{O}(\frac{1}{\max\{L_f, \kappa\mu\}})$ and $b = \frac{\max\{\bar{L}_f, n\mu\}}{\max\{L_f, \kappa\mu\}}$, where $\kappa = 2\kappa_c$ for EXTRA, and $\kappa = \kappa_c^2$ for DIGing. Let $s_{(i)}^1 = \nabla f_{(i)}(w_{(i)}^0)$ for DIGing.

Let distribution $\mathcal{D}_{(i)}$ be to output $j \in [1, n]$ with probability $p_{(i),j} = \frac{L_{(i),j}}{n\bar{L}_{(i)}}$.

**for** $k = 1, 2, ...$ **do**

Step 1: $\mathbb{S}_{(i)}^k \leftarrow b$ independent samples from $\mathcal{D}_{(i)}$ with replacement, $\forall i$,

Step 2: Compute $\nabla_{(i)}^k$ by (9a), $\forall i$,

Step 3: For EXTRA, compute $x_{(i)}^{k+1}$ by

$$x_{(i)}^{k+1} = \left(x_{(i)}^k + \sum_{j\in\mathcal{N}_{(i)}} W_{ij}x_{(j)}^k\right) - \frac{1}{2}\left(x_{(i)}^{k-1} + \sum_{j\in\mathcal{N}_{(i)}} W_{ij}x_{(j)}^{k-1}\right) - \alpha\left(\nabla_{(i)}^k - \nabla_{(i)}^{k-1}\right), \forall i,$$

For DIGing, compute $x_{(i)}^{k+1}$ by

$$s_{(i)}^{k+1} = \sum_{j\in\mathcal{N}_{(i)}} W_{ij}s_{(j)}^k + \nabla_{(i)}^k - \nabla_{(i)}^{k-1}, \qquad x_{(i)}^{k+1} = \sum_{j\in\mathcal{N}_{(i)}} W_{ij}x_{(j)}^k - \alpha s_{(i)}^{k+1}, \quad \forall i,$$

Step 4: Compute $w_{(i)}^{k+1}$ by (9d), $\forall i$.

**end for**

which is further equivalent to the following method, called VR-DIGing,

$$\mathbf{s}^{k+1} = W\mathbf{s}^k + \nabla^k - \nabla^{k-1},$$
$$\mathbf{x}^{k+1} = W\mathbf{x}^k - \alpha\mathbf{s}^{k+1}.$$

We see that VR-EXTRA and VR-DIGing are quite similar to the original EXTRA and DIGing. The only difference is that we replace the local gradients by their VR estimators. Thus the implementation is as simple as that of the original EXTRA and DIGing. We give the specific descriptions of VR-EXTRA and VR-DIGing in Algorithm 1 in a distributed way, including the parameter settings. To discuss EXTRA and DIGing in a unified framework, we denote

$$\kappa = 2\kappa_c \text{ for EXTRA} \qquad \text{and} \qquad \kappa = \kappa_c^2 \text{ for DIGing.} \tag{12}$$

See Lemma 1 for the reason. We will use $\kappa$ frequently in this paper when we do not distinguish EXTRA and DIGing, and the readers can use (12) to get the specific properties of EXTRA and DIGing, respectively.

### 2.3 Extension to Large $\kappa$

The particular choice of the mini-batch size $b$ in Algorithm 1 may be smaller than 1 when $\kappa$ is large, which makes the algorithm meaningless. We discuss EXTRA and DIGing in a

unified way in this section, so we use $\kappa$ in this section, which is defined by (12). In fact, $b \geq 1$ if and only if $\kappa \leq \max\{\kappa_s, n\}$, see the proof of Theorem 2 in Section 4. In this section we consider the case of $\kappa > \max\{\kappa_s, n\}$.

Intuitively speaking, when $\kappa$ is very large such that $\kappa_b + \kappa \geq \kappa_s + n$, to reach the desired $\mathcal{O}((\kappa_b + \kappa)\log\frac{1}{\epsilon})$ communication complexity and the $\mathcal{O}((\kappa_s + n)\log\frac{1}{\epsilon})$ stochastic gradient computation complexity, as summarized in Table 1, we should perform less than 1 stochastic gradient evaluation in average at each iteration. This observation motivates us to introduce some zero samples, that is to say, let $f_{(i),n+1} = \cdots = f_{(i),n'} = 0$ for all $i$, and consider problem

$$\min_{x\in\mathbb{R}^p} \sum_{i=1}^{m} f'_{(i)}(x), \quad \text{where} \quad f'_{(i)}(x) = \frac{1}{n'}\sum_{j=1}^{n'} f_{(i),j}(x). \tag{13}$$

The zero samples do not spend time to compute the stochastic gradient. We see that problems (13) and (1) are equivalent. To use Algorithm 1 to solve problem (13), we denote

$$L_{(i),j} = \frac{n\mu n' - n\overline{L}_{(i)}}{n'-n} \quad \text{for all} \quad n < j \leq n',$$

and let each sample be selected with probability $\frac{L_{(i),j}}{\sum_{j=1}^{n'} L_{(i),j}}$. Then we select the samples in $[1, n]$ with probability $\frac{\overline{L}_{(i)}}{\mu n'}$, and select the zero samples with probability $1 - \frac{\overline{L}_{(i)}}{\mu n'}$. It can be seen that $f'_{(i)}(x)$ is $\frac{nL_{(i)}}{n'}$-smooth and $\frac{n\mu}{n'}$-strongly convex. Define the following notations:

$$n' = \kappa, \quad \mu' = \frac{n\mu}{n'}, \quad L'_f = \max_i \frac{nL_{(i)}}{n'} = \frac{nL_f}{n'}, \quad \overline{L}'_{(i)} = \frac{\sum_{j=1}^{n'} L_{(i),j}}{n'}, \quad \overline{L}'_f = \max_i \overline{L}'_{(i)}. \tag{14}$$

We can easily check $\alpha = \mathcal{O}(\frac{1}{\max\{L'_f, \kappa\mu'\}}) = \mathcal{O}(\frac{1}{n\mu})$, and $b = \frac{\max\{\overline{L}'_f, n'\mu'\}}{\max\{L'_f, \kappa\mu'\}} = 1$. See the proof of Theorem 4 in Section 4. Then we can use Algorithm 1 to solve problem (13).

### 2.4 Complexities

We prove the convergence of VR-EXTRA and VR-DIGing in a unified framework. From Assumption 2, we have the following easy-to-identify lemma, where the third inequality in (15) can be proved similarly to Lemma 4 in (Li et al., 2020b).

**Lemma 1** *Suppose that Assumption 2 holds with $\omega = 0$ for EXTRA. Let $U = V = \sqrt{\frac{I-W}{2}}$. Then we have*

$$\|U\mathbf{x}\|^2 \leq \|V\mathbf{x}\|^2, \quad \|V\mathbf{x}\|^2 \leq \frac{1}{2}\|\mathbf{x}\|^2, \quad \textit{and} \quad \|U\lambda\|^2 \geq \frac{1}{\kappa}\|\lambda\|^2, \forall \lambda \in \textit{Span}(U), \tag{15}$$

*where $\kappa = \frac{2}{1-\sigma_2(W)} = 2\kappa_c$. Suppose that Assumption 2 holds with $\omega = \frac{\sqrt{2}}{2}$ for DIGing. Let $U = I - W$ and $V = \sqrt{I - W^2}$. Then (15) also holds with $\kappa = \frac{1}{(1-\sigma_2(W))^2} = \kappa_c^2$.*

Denote the following set of random variables:

$$\mathbb{S}^k = \cup_{i=1}^m \mathbb{S}_{(i)}^k, \qquad \xi^k = \{\mathbb{S}^0, \mathbf{w}^1, \mathbb{S}^1, \mathbf{w}^2 \cdots, \mathbb{S}^{k-1}, \mathbf{w}^k\}.$$

The next theorem gives the communication complexity and stochastic gradient computation complexity of algorithm (9a)-(9d) in a unified way.

**Theorem 2** *Suppose that Assumption 1 holds, and $U$ and $V$ satisfy (7) and (15). Let $\alpha = \frac{1}{28\max\{L_f,\kappa\mu\}}$ and $\lambda^0 = 0$.*

1. *If $\kappa \leq \max\{\kappa_s, n\}$, let $b = \frac{\max\{\overline{L}_f, n\mu\}}{\max\{L_f,\kappa\mu\}}$. Then algorithm (9a)-(9d) requires the time of $\mathcal{O}((\kappa_b+\kappa)\log\frac{1}{\epsilon})$ communication rounds and $\mathcal{O}((\kappa_s+n)\log\frac{1}{\epsilon})$ stochastic gradient evaluations to find $\mathbf{x}^k$ such that $\mathbb{E}_{\xi^k}[\|\mathbf{x}^k - \mathbf{x}^*\|^2] \leq \epsilon$.*

2. *If $\kappa \geq \max\{\kappa_s, n\}$, let $b = 1$. Then algorithm (9a)-(9d) requires the time of $\mathcal{O}((\kappa_b+\kappa)\log\frac{1}{\epsilon})$ communication rounds and $\mathcal{O}((\kappa_b+\kappa)\log\frac{1}{\epsilon})$ stochastic gradient evaluations to find $\mathbf{x}^k$ such that $\mathbb{E}_{\xi^k}\left[\|\mathbf{x}^k - \mathbf{x}^*\|^2\right] \leq \epsilon$.*

**Remark 3** *Let's explain the time of one communication rounds and one stochastic gradient evaluation. At each iteration, algorithm (9a)-(9d) performs one round of communication, that is, each node $i$ receives information $x_{(j)}^k$ and $\lambda_{(j)}^k$ from its neighbors for all $j \in \mathcal{N}_{(i)}$. Then each node $i$ selects $\mathbb{S}_{(i)}^k$ randomly and computes $\nabla_{(i)}^k$ with $b$ stochastic gradient evaluations. $\nabla f_{(i)}(w_{(i)}^{k+1})$ is updated with probability $b/n$, and each time with $n$ stochastic gradient evaluations. So each node computes $b$ stochastic gradients in average at each iteration. Since the computation is performed in parallel across all the nodes, we say that each iteration requires the time of one communication round and $b$ stochastic gradient evaluations in average.*

From Theorem 2, we see that when $\kappa \geq \max\{\kappa_s, n\}$, the stochastic gradient computation cost increases to $\mathcal{O}((\kappa_b+\kappa)\log\frac{1}{\epsilon})$. We can use the zero-sample strategy described in Section 2.3 to reduce the computation cost to $\mathcal{O}((\kappa_s+n)\log\frac{1}{\epsilon})$, as described in the following theorem.

**Theorem 4** *Suppose that Assumption 1 and conditions (7) and (15) hold. Assume $\kappa > \max\{\kappa_s, n\}$. Applying Algorithm 1 to solve problem (13), it requires the time of $\mathcal{O}((\kappa_b+\kappa)\log\frac{1}{\epsilon})$ communication rounds and $\mathcal{O}((\kappa_s+n)\log\frac{1}{\epsilon})$ stochastic gradient evaluations to find an $\epsilon$-precision solution of problem (1) such that $\mathbb{E}_{\xi^k}[\|\mathbf{x}^k - \mathbf{x}^*\|^2] \leq \epsilon$.*

We see that by introducing the zero samples with carefully designed $n'$ and $L_{(i),j}$ for $n < j \leq n'$, the complexities in Theorem 4 keep the same as those in part one of Theorem 2.

For the particular VR-EXTRA and VR-DIGing methods, we have the following complexities accordingly, where we replace $\kappa$ in Theorems 2 and 4 by $2\kappa_c$ and $\kappa_c^2$, respectively.

**Corollary 5** *Suppose that Assumptions 1 and 2 hold with $\omega = 0$. Use the zero-sample strategy if $2\kappa_c \geq \max\{\kappa_s, n\}$. Then the VR-EXTRA method in Algorithm 1 requires the time of $\mathcal{O}((\kappa_b+\kappa_c)\log\frac{1}{\epsilon})$ communication rounds and $\mathcal{O}((\kappa_s+n)\log\frac{1}{\epsilon})$ stochastic gradient evaluations to find $\mathbf{x}^k$ such that $\mathbb{E}_{\xi^k}[\|\mathbf{x}^k - \mathbf{x}^*\|^2] \leq \epsilon$.*

**Corollary 6** *Suppose that Assumptions 1 and 2 hold with* $\omega = \frac{\sqrt{2}}{2}$. *Use the zero-sample strategy if* $\kappa_c^2 \geq \max\{\kappa_s, n\}$. *Then the VR-DIGing method in Algorithm 1 requires the time of* $\mathcal{O}((\kappa_b + \kappa_c^2)\log\frac{1}{\epsilon})$ *communication rounds and* $\mathcal{O}((\kappa_s + n)\log\frac{1}{\epsilon})$ *stochastic gradient evaluations to find* $\mathbf{x}^k$ *such that* $\mathbb{E}_{\xi^k}[\|\mathbf{x}^k - \mathbf{x}^*\|^2] \leq \epsilon$.

**Remark 7**

1. *The communication complexity of VR-DIGing has a worse dependence on* $\kappa_c$ *than that of VR-EXTRA. This is because EXTRA uses* $U = \sqrt{\frac{I-W}{2}}$ *in problem (6), while DIGing uses* $U = I - W$. *From Lemma 1, we see that different choice of* $U$ *gives different order of* $\kappa_c$.

2. *From Table 1, we see that EXTRA and VR-EXTRA have the same communication complexity, and DIGing and VR-DIGing also have the same communication complexity. Thus extending EXTRA and DIGing to stochastic decentralized optimization does not need to pay a price of more communication cost theoretically.*

3. *When* $\kappa \leq \max\{\kappa_s, n\}$, *running VR-EXTRA and VR-DIGing with* $mn$ *samples needs the time of* $\mathcal{O}((\kappa_s + n)\log\frac{1}{\epsilon})$ *stochastic gradient evaluations by parallelism, which is the same as that of running the single-machine VR methods with* $n$ *samples when we ignore the communication time. On the other hand, when we run the single-machine VR methods with* $mn$ *samples, the required time increases to* $\mathcal{O}((\kappa_s + mn)\log\frac{1}{\epsilon})$. *Thus the linear speedup is achieved when* $n$ *is larger than* $\kappa_s$. *The situation of* $\kappa > \max\{\kappa_s, n\}$ *is more complicated because at each iteration, some machines would be computing gradients, while others would be idle if the zero-sample is chosen. Parallelism is destroyed and the actual running time would be larger than the time of* $\mathcal{O}((\kappa_s + n)\log\frac{1}{\epsilon})$ *stochastic gradient evaluations.*

4. *Both in theory and in practice, we can choose a larger mini-batch size* $b$ *than the particular choice given in Algorithm 1, at the expense of a higher stochastic gradient computation complexity than* $\mathcal{O}((\kappa_s + n)\log\frac{1}{\epsilon})$. *However, the communication complexity remains unchanged. See the proof of Theorem 2. Denote* $\tau$ *to be the ratio between the practical running time of performing one communication round and one stochastic gradient computation. If* $\kappa_s + n \leq \tau(\kappa_b + \kappa)$, *that is, communications dominate the total running time, we can increase the mini-batch size to* $\frac{\max\{\overline{L}_f, n\mu\}}{\max\{L_f, \kappa\mu\}}\frac{\tau(\kappa_b+\kappa)}{\kappa_s+n} = \tau$, *which does not increase the total running time of* $\mathcal{O}(\tau(\kappa_b + \kappa)\log\frac{1}{\epsilon})$.

## 3. Accelerated Variance Reduced EXTRA and DIGing

In this section, we develop the accelerated VR-EXTRA and VR-DIGing methods. In algorithm (9a)-(9d), we combine (8) with the loopless SVRG to get the non-accelerated methods. To develop the accelerated methods, a straightforward idea is to combine (8) with the loopless Katyusha proposed in (Kovalev et al., 2020a), which leads to the following algorithm (16a)-(16f). We give the parameter settings in Algorithm 2. We will not write (16a)-(16f) in the EXTRA/DIGing style since the resultant methods are complex, and they

**Algorithm 2** Acc-VR-EXTRA and Acc-VR-DIGing

Initialize: $x_{(i)}^0 = z_{(i)}^0 = w_{(i)}^0$, $\lambda_{(i)}^0 = 0$ for all $i$, $\alpha = \mathcal{O}(\frac{1}{L_f})$, $b = \max\{\frac{\max\{\sqrt{n\overline{L}_f/\mu}, n\}}{\max\{\sqrt{\kappa L_f/\mu}, \kappa\}}, \frac{\overline{L}_f}{L_f}\}$, $\theta_1 = \min\{\frac{1}{2}\sqrt{\frac{\kappa\mu}{L_f}}, \frac{1}{2}\}$, $\theta_2 = \frac{\overline{L}_f}{2L_f b}$, where $\kappa = 2\kappa_c$ for EXTRA, and $\kappa = \kappa_c^2$ for DIGing.

Let $U = V = \sqrt{\frac{I-W}{2}}$ for EXTRA, and $U = I - W$ and $V = \sqrt{I - W^2}$ for DIGing.

Let distribution $\mathcal{D}_{(i)}$ be to output $j \in [1, n]$ with probability $p_{(i),j} = \frac{L_{(i),j}}{n\overline{L}_{(i)}}$.

**for** $k = 0, 1, 2, \ldots$ **do**

$\mathbb{S}_{(i)}^k \leftarrow b$ independent samples from $\mathcal{D}_{(i)}$ with replacement, $\forall i$,

Perform steps (16a)-(16f), $\forall i$,

**end for**

are not very similar to the original EXTRA and DIGing besides the feature of tracking the differences of gradients.

$$y_{(i)}^k = \theta_1 z_{(i)}^k + \theta_2 w_{(i)}^k + (1 - \theta_1 - \theta_2) x_{(i)}^k, \quad \forall i, \tag{16a}$$

$$\nabla_{(i)}^k = \frac{1}{b} \sum_{j \in \mathbb{S}_{(i)}^k} \frac{1}{np_{(i),j}} \left( \nabla f_{(i),j}(y_{(i)}^k) - \nabla f_{(i),j}(w_{(i)}^k) \right) + \nabla f_{(i)}(w_{(i)}^k), \quad \forall i, \tag{16b}$$

$$z_{(i)}^{k+1} = \frac{1}{1 + \frac{\mu\alpha}{\theta_1}} \left( \frac{\mu\alpha}{\theta_1} y_{(i)}^k + z_{(i)}^k - \frac{1}{\theta_1} \left( \alpha \nabla_{(i)}^k + \sum_{j \in \mathcal{N}_{(i)}} U_{ij} \lambda_{(j)}^k + \theta_1 \sum_{j \in \mathcal{N}_{(i)}} (V^2)_{ij} z_{(j)}^k \right) \right), \forall i, \tag{16c}$$

$$\lambda_{(i)}^{k+1} = \lambda_{(i)}^k + \theta_1 \sum_{j \in \mathcal{N}_{(i)}} U_{ij} z_{(j)}^{k+1}, \quad \forall i, \tag{16d}$$

$$x_{(i)}^{k+1} = y_{(i)}^k + \theta_1 \left( z_{(i)}^{k+1} - z_{(i)}^k \right), \quad \forall i, \tag{16e}$$

$$w_{(i)}^{k+1} = \begin{cases} x_{(i)}^k \text{ with probability } \frac{b}{n}, \\ w_{(i)}^k \text{ with probability } 1 - \frac{b}{n}, \end{cases} \quad \forall i. \tag{16f}$$

In the above algorithm, steps (16a) and (16e) are the Nesterov's acceleration steps, which are motivated by (Allen-Zhu, 2018; Kovalev et al., 2020a). Steps (16c) and (16d) involve the operation of $U\mathbf{x}$, which is uncomputable for $U = \sqrt{\frac{I-W}{2}}$ in EXTRA in the distributed environment. Introducing the auxiliary variable $\widetilde{\lambda}^k = U\lambda^k$ and multiplying both sides of (16d) by $U$ leads to

$$\begin{aligned} \mathbf{z}^{k+1} &= \frac{1}{1 + \frac{\mu\alpha}{\theta_1}} \left( \frac{\mu\alpha}{\theta_1} \mathbf{y}^k + \mathbf{z}^k - \frac{1}{\theta_1} \left( \alpha \nabla^k + \widetilde{\lambda}^k + \theta_1 V^2 \mathbf{z}^k \right) \right), \\ \widetilde{\lambda}^{k+1} &= \widetilde{\lambda}^k + \theta_1 U^2 \mathbf{z}^{k+1}, \end{aligned} \tag{17}$$

in the compact form. From the definitions of $U = V = \sqrt{\frac{I-W}{2}}$, we only need to compute $W\mathbf{z}$, which corresponds to the gossip-style communications. For DIGing, we do not need such auxiliary variables.

### 3.1 Complexities

Theorem 8 gives the complexities of algorithm (16a)-(16f) in a unified way, and Corollaries 10 and 11 provide the complexities for the particular Acc-VR-EXTRA and Acc-VR-DIGing methods, respectively.

**Theorem 8** *Suppose that Assumption 1 holds, and $U$ and $V$ satisfy (7) and (15). Let $\theta_1 = \min\{\frac{1}{2}\sqrt{\frac{\kappa\mu}{L_f}}, \frac{1}{2}\}$, $\theta_2 = \frac{\overline{L}_f}{2L_f b}$, $\alpha = \frac{1}{10L_f}$, $\lambda^0 = 0$, and $b = \max\{\frac{\max\{\sqrt{n\overline{L}_f/\mu}, n\}}{\max\{\sqrt{\kappa L_f/\mu}, \kappa\}}, \frac{\overline{L}_f}{L_f}\}$.*

1. *If $\kappa \leq \frac{nL_f}{\overline{L}_f}$, such that $b = \frac{\max\{\sqrt{n\overline{L}_f/\mu}, n\}}{\max\{\sqrt{\kappa L_f/\mu}, \kappa\}}$, then algorithm (16a)-(16f) requires the time of $\mathcal{O}((\kappa + \sqrt{\kappa_b\kappa})\log\frac{1}{\epsilon})$ communication rounds and $\mathcal{O}((\sqrt{n\kappa_s} + n)\log\frac{1}{\epsilon})$ stochastic gradient evaluations to find $\mathbf{z}^k$ such that $\mathbb{E}_{\xi^k}\left[\|\mathbf{z}^k - \mathbf{x}^*\|^2\right] \leq \epsilon$.*

2. *If $\kappa \geq \frac{nL_f}{\overline{L}_f}$, such that $b = \frac{\overline{L}_f}{L_f}$, then algorithm (16a)-(16f) requires the time of $\mathcal{O}((\kappa + \sqrt{\kappa_b\kappa})\log\frac{1}{\epsilon})$ communication rounds and $\mathcal{O}(\frac{\overline{L}_f}{L_f}(\kappa + \sqrt{\kappa_b\kappa})\log\frac{1}{\epsilon})$ stochastic gradient evaluations to find $\mathbf{z}^k$ such that $\mathbb{E}_{\xi^k}\left[\|\mathbf{z}^k - \mathbf{x}^*\|^2\right] \leq \epsilon$.*

**Remark 9**

1. *As introduced in Section 1.1, we have $\kappa_b \leq \kappa_s \leq n\kappa_b$, and we always assume $\kappa_s \ll n\kappa_b$ in the analysis of stochastic algorithms. Thus we can expect $\frac{nL_f}{\overline{L}_f}$ to be large for large-scale data. On the other hand, $\kappa_c$ depends on the network scale and connectivity. For example, $\kappa_c = \mathcal{O}(1)$ for the commonly used Erdős−Rényi random graph, and $\kappa_c = \mathcal{O}(m\log m)$ for the geometric graph. In the worst case, for example, the linear graph or cycle graph, we have $\kappa_c = \mathcal{O}(m^2)$ (Nedić et al., 2018). Thus we can also expect $\kappa$ to be not very large when the number of distributed nodes is limited and the network is well connected. So we can expect that the assumption $\kappa \leq \frac{nL_f}{\overline{L}_f}$ always holds for large-scale distributed data, for example, thousands of nodes and each node with millions of data.*

2. *Similar to VR-EXTRA and VR-DIGing, both in theory and in practice, we can choose a larger mini-batch size $b$ than the particular choice given in Algorithm 2, at the expense of higher stochastic gradient computation complexities than the ones given in Theorem 8. However, the $\mathcal{O}((\sqrt{\kappa_b\kappa} + \kappa)\log\frac{1}{\epsilon})$ communication complexity remains unchanged. See the proof of Theorem 8. We take part one of Theorem 8 as an example. Recall the definition of $\tau$ in Remark 7(4). If $\max\{\sqrt{n\kappa_s}, n\} \leq \tau\max\{\sqrt{\kappa_b\kappa}, \kappa\}$, that is, communications dominate the total running time, we can increase the mini-batch size to $\frac{\max\{\sqrt{n\overline{L}_f/\mu}, n\}}{\max\{\sqrt{\kappa L_f/\mu}, \kappa\}}\frac{\tau\max\{\sqrt{\kappa_b\kappa}, \kappa\}}{\max\{\sqrt{n\kappa_s}, n\}} = \tau$, which does not increase the total running time of $\mathcal{O}(\tau(\sqrt{\kappa_b\kappa} + \kappa)\log\frac{1}{\epsilon})$.*

**Corollary 10** *Suppose that Assumptions 1 and 2 hold with $\omega = 0$. Under the parameter settings in Theorem 8 with $\kappa = 2\kappa_c$, if $2\kappa_c \leq \frac{nL_f}{\overline{L}_f}$ and $\kappa_c \leq \kappa_b$, the Acc-VR-EXTRA algorithm requires the time of $\mathcal{O}((\kappa_c + \sqrt{\kappa_b\kappa_c})\log\frac{1}{\epsilon}) = \mathcal{O}(\sqrt{\kappa_b\kappa_c}\log\frac{1}{\epsilon})$ communication rounds and $\mathcal{O}((\sqrt{n\kappa_s} + n)\log\frac{1}{\epsilon})$ stochastic gradient evaluations to find $\mathbf{z}^k$ such that $\mathbb{E}_{\xi^k}[\|\mathbf{z}^k - \mathbf{x}^*\|^2] \leq \epsilon$.*

**Corollary 11** *Suppose that Assumptions 1 and 2 hold with* $\omega = \frac{\sqrt{2}}{2}$. *Under the parameter settings in Theorem 8 with* $\kappa = \kappa_c^2$, *if* $\kappa_c^2 \leq \frac{nL_f}{L_f}$ *and* $\kappa_c^2 \leq \kappa_b$, *the Acc-VR-DIGing algorithm requires the time of* $\mathcal{O}((\kappa_c^2 + \kappa_c\sqrt{\kappa_b})\log\frac{1}{\epsilon}) = \mathcal{O}(\kappa_c\sqrt{\kappa_b}\log\frac{1}{\epsilon})$ *communication rounds and* $\mathcal{O}((\sqrt{n\kappa_s}+n)\log\frac{1}{\epsilon})$ *stochastic gradient evaluations to find* $\mathbf{z}^k$ *such that* $\mathbb{E}_{\xi^k}[\|\mathbf{z}^k - \mathbf{x}^*\|^2] \leq \epsilon$.

**Remark 12**

1. *From Table 1, we see that the communication complexity and stochastic gradient computation complexity of Acc-VR-EXTRA are both optimal under the restrictions of* $2\kappa_c \leq \frac{nL_f}{L_f}$ *and* $\kappa_c \leq \kappa_b$. *Similarly, the stochastic gradient computation complexity of Acc-VR-DIGing is also optimal. However, its communication complexity is worse than the corresponding lower bound by the* $\mathcal{O}(\sqrt{\kappa_c})$ *factor.*

2. *Running Acc-VR-EXTRA and Acc-VR-DIGing with* $mn$ *samples needs the time of* $\mathcal{O}((\sqrt{n\kappa_s}+n)\log\frac{1}{\epsilon})$ *stochastic gradient evaluations, which is the same as that of running the single-machine Katyusha with* $n$ *samples when we ignore the communication time. On the other hand, when we run the single-machine Katyusha with* $mn$ *samples, the required time increases to* $\mathcal{O}((\sqrt{mn\kappa_s}+mn)\log\frac{1}{\epsilon})$. *Since acceleration takes effect only when* $\kappa_s \gg mn$, *the parallelism speeds up Katyusha by the* $\sqrt{m}$ *factor. On the other hand, when* $\kappa_s \leq n$, *the linear speedup is achieved.*

At last, we compare VR-EXTRA and VR-DIGing with Acc-VR-EXTRA and Acc-VR-DIGing. Both the accelerated methods and non-accelerated methods have their own advantages.

1. The accelerated methods need less stochastic gradient computation evaluations than the non-accelerated methods when $\kappa_s > n$ and $\kappa \leq \frac{nL_f}{L_f}$. In this case, acceleration takes effect. Otherwise, the accelerated methods have no advantage over non-accelerated methods on the computation cost. Moreover, the non-accelerated methods have the superiority of simple implementation.

2. The accelerated methods need less communication rounds than their non-accelerated counterparts when $\kappa \leq \kappa_b$. When dealing with large-scale distributed data in machine learning, we may expect that computation often dominates the total running time. Otherwise, the full batch accelerated decentralized methods such as APAPC (Kovalev et al., 2020b) may be a better choice.

### 3.2 Chebyshev Acceleration

In this section we remove the restrictions on the size of $\kappa_c$ in Corollaries 10 and 11, which come from matrix $U$, as shown in (15). To make $\kappa$ small, our goal is to construct a new matrix $\hat{U}$ by $U$ such that $\mathrm{Ker}(\hat{U}) = \mathrm{Span}(\mathbf{1})$ and $\|\hat{U}\lambda\|^2 \geq \frac{1}{c}\|\lambda\|^2$ for all $\lambda \in \mathrm{Span}(\hat{U})$, where $c$ is a much smaller constant than $\kappa$. Moreover, the construction procedure should not take more than $\mathcal{O}(\sqrt{\kappa_c})$ time. Then we only need to replace $U$ and $V$ by $\hat{U}$ and some matrix $\hat{V}$ in algorithm (16a)-(16f), where $\hat{U}$ and $\hat{V}$ should satisfy the relations in (15). We follow (Scaman et al., 2017) to use Chebyshev acceleration to construct $\hat{U}$, which is a common acceleration scheme to minimize $c$.

### 3.2.1 Review of Chebyshev Acceleration

We first give a brief description of Chebyshev acceleration, which was first used to accelerate distributed optimization in (Scaman et al., 2017). We first introduce the Chebyshev polynomials defined as $T_0(x) = 1$, $T_1(x) = x$, and $T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x)$ for all $k \geq 1$. Given a positive semidefinite symmetric matrix $L \in \mathbb{R}^{m\times m}$ such that $\text{Ker}(L) = \text{Span}(\mathbf{1})$, denote $\lambda_1(L) \geq \lambda_2(L) \geq \cdots \geq \lambda_{m-1}(L) > \lambda_m(L) = 0$ as the eigenvalues of $L$. Following the notations in (Scaman et al., 2017), we define $\gamma(L) = \frac{\lambda_{m-1}(L)}{\lambda_1(L)}$, $c_1 = \frac{1-\sqrt{\gamma(L)}}{1+\sqrt{\gamma(L)}}$, $c_2 = \frac{1+\gamma(L)}{1-\gamma(L)}$, and $c_3 = \frac{2}{\lambda_1(L)+\lambda_{m-1}(L)}$. Then $c_3L$ has the spectrum in $[1-c_2^{-1}, 1+c_2^{-1}]$. For any polynomial $p_t(x)$ of degree at most $t$, Theorem 6.1 in (Auzinger and Melenk, 2017) tells us that the solution of the following problem

$$\min_{p_t:p_t(0)=0} \max_{x\in[1-c_2^{-1},1+c_2^{-1}]} |p_t(x) - 1|$$

is

$$P_t(x) = 1 - \frac{T_t(c_2(1-x))}{T_t(c_2)}.$$

Moreover, from Corollary 6.1 in (Auzinger and Melenk, 2017), we have

$$\min_{p_t:p_t(0)=0} \max_{x\in[1-c_2^{-1},1+c_2^{-1}]} |p_t(x) - 1| = \frac{2c_1^t}{1+c_1^{2t}},$$

which means that the spectrum of $P_t(c_3L)$ lies in $\left[1 - \frac{2c_1^t}{1+c_1^{2t}}, 1 + \frac{2c_1^t}{1+c_1^{2t}}\right]$. For the particular choice of $t = \frac{3}{\sqrt{\gamma(L)}}$, it can be checked that $c_1^t \leq \left(\left(1-\sqrt{\gamma(L)}\right)^{1/\sqrt{\gamma(L)}}\right)^3 \leq e^{-3}$, which further leads to $\frac{2c_1^t}{1+c_1^{2t}} \leq \frac{2}{e^3+e^{-3}} \leq 0.1$. Thus the spectrum of $P_t(c_3L)$ is a subinterval of $[0.9, 1.1]$. On the other hand, it can be checked that $P_t(c_3L)$ is a gossip matrix satisfying $\text{Ker}(P_t(c_3L)) = \text{Span}(\mathbf{1})$ (Scaman et al., 2017). In practice, we can compute the operation $P_t(c_3L)\mathbf{x}$ by the following procedure (Scaman et al., 2017):

Input: $\mathbf{x}$,
Initialize: $a^0 = 1$, $a^1 = c_2$, $\mathbf{z}^0 = \mathbf{x}$, $\mathbf{z}^1 = c_2(I - c_3L)\mathbf{x}$,
**for** $s = 1, 2, ..., t-1$ **do**
  $a^{s+1} = 2c_2a^s - a^{s-1}$,
  $\mathbf{z}^{s+1} = 2c_2(I - c_3L)\mathbf{z}^s - \mathbf{z}^{s-1}$.
**end for**
Output: $P_t(c_3L)\mathbf{x} = \mathbf{x} - \frac{\mathbf{z}^t}{a^t}$.

### 3.2.2 Remove the Restrictions on the Size of $\kappa_c$

For the particular choice of $U = V = \sqrt{\frac{I-W}{2}}$, define $\hat{U} = \hat{V} = \sqrt{\frac{1}{2.2}P_t(c_3U^2)}$, where $c_3 = \frac{2}{\lambda_1(U^2)+\lambda_{n-1}(U^2)}$. Then we have $\|\hat{V}\mathbf{x}\|^2 \leq \frac{1}{2}\|\mathbf{x}\|^2$ and $\|\hat{U}\lambda\|^2 \geq \frac{0.9}{2.2}\|\lambda\|^2 \geq \frac{1}{3}\|\lambda\|^2$ for all $\lambda \in \text{Span}(\hat{U})$ with $t = \frac{3}{\sqrt{\gamma(U^2)}} = \frac{3}{\sqrt{1-\sigma_2(W)}} = \mathcal{O}(\sqrt{\kappa_c})$. So $\hat{U}$ and $\hat{V}$ satisfy the relations in (15), and replacing $U$ and $V$ by $\hat{U}$ and $V$ does not destroy the proof of Theorem 8. In the

algorithm implementation, we only need to replace the operations $U^2\mathbf{z}$ and $V^2\mathbf{z}$ in (17) by $\frac{1}{2.2}P_t(c_3U^2)\mathbf{z}$. Moreover, replacing $\kappa$ by the constant 3 in Theorem 8, we can expect that the assumptions on $\kappa$ always hold. Since we need $\mathcal{O}(\sqrt{\kappa_c})$ time to construct $\hat{U}$ at each iteration, so the communication complexity remains $\left(\sqrt{\kappa_b\kappa_c}\log\frac{1}{\epsilon}\right)$.

**Corollary 13** *Suppose that Assumptions 1 and 2 hold with $\omega = 0$. Under the parameter settings in Theorem 8 with $\kappa = 3$, Acc-VR-EXTRA with Chebyshev acceleration (CA) requires the time of $\left(\sqrt{\kappa_b\kappa_c}\log\frac{1}{\epsilon}\right)$ communication rounds and $\mathcal{O}((\sqrt{n\kappa_s}+n)\log\frac{1}{\epsilon})$ stochastic gradient evaluations to find $\mathbf{z}^k$ such that $\mathbb{E}_{\xi^k}[\|\mathbf{z}^k - \mathbf{x}^*\|^2] \leq \epsilon$.*

For the particular choice of $U = I - W$ and $V = \sqrt{I - W^2}$, define $\hat{U} = \frac{2-\sqrt{2}}{2.2}P_t(c_3U)$, $\hat{W} = I - \hat{U}$, and $\hat{V} = \sqrt{I - \hat{W}^2}$, where $c_3 = \frac{2}{\lambda_1(U)+\lambda_{n-1}(U)}$. Then we have $\|\hat{V}\mathbf{x}\|^2 \leq \frac{1}{2}\|\mathbf{x}\|^2$ and $\|\hat{U}\lambda\|^2 \geq \frac{1}{20}\|\lambda\|^2$ for all $\lambda \in \text{Span}(\hat{U})$ with $t = \frac{3}{\sqrt{\gamma(U)}} = \mathcal{O}(\sqrt{\kappa_c})$. Similar to the above analysis for the Acc-VR-EXTRA-CA method, we have the following complexity corollary. Note that since we replace $\kappa$ by the constant 20 in Theorem 8, and use the fact that the construction of $\hat{U}$ needs $\mathcal{O}(\sqrt{\kappa_c})$ time at each iteration, we can reduce the communication cost from $\mathcal{O}(\kappa_c\sqrt{\kappa_b}\log\frac{1}{\epsilon})$ to $\mathcal{O}(\sqrt{\kappa_b\kappa_c}\log\frac{1}{\epsilon})$.

**Corollary 14** *Suppose that Assumptions 1 and 2 hold with $\omega = \frac{\sqrt{2}}{2}$. Under the parameter settings of Theorem 8 with $\kappa = 20$, Acc-VR-DIGing-CA requires the time of $\mathcal{O}(\sqrt{\kappa_b\kappa_c}\log\frac{1}{\epsilon})$ communication rounds and $\mathcal{O}((\sqrt{n\kappa_s}+n)\log\frac{1}{\epsilon})$ stochastic gradient evaluations to find $\mathbf{z}^k$ such that $\mathbb{E}_{\xi^k}[\|\mathbf{z}^k - \mathbf{x}^*\|^2] \leq \epsilon$.*

**Remark 15**

1. *From Corollaries 13 and 14, we see that the restrictions on $\kappa_c$ in Corollaries 10 and 11 have been removed, and the communication complexity of Acc-VR-DIGing has been improved to be optimal by Chebyshev acceleration.*

2. *We can also combine Chebyshev acceleration with the non-accelerated VR-EXTRA and VR-DIGing, and give the $\mathcal{O}\left((\kappa_s+n)\log\frac{1}{\epsilon}\right)$ stochastic gradient computation complexity and the $\mathcal{O}\left(\left(\kappa_b\sqrt{\kappa_c}\right)\log\frac{1}{\epsilon}\right)$ communication complexity, which are the same as those of DVR (Hendrikx et al., 2020).*

## 4. Proof of Theorems

We prove Theorems 2 and 8 in this section. We first introduce some useful properties. For $L$-smooth and convex function $f(\mathbf{x})$, we have (Nesterov, 2004)

$$f(\mathbf{y}) - f(\mathbf{x}) - \langle\nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x}\rangle \geq \frac{1}{2L}\|\nabla f(\mathbf{y}) - \nabla f(\mathbf{x})\|^2. \tag{18}$$

Recall that $x^*$ is the optimal solution of problem (1), then $\mathbf{x}^*$ is also the optimal solution of the following linearly constrained convex problem

$$\min_{\mathbf{x}} f(\mathbf{x}), \quad \text{s.t.} \quad U\mathbf{x} = 0, \tag{19}$$

where $U$ satisfies (7). Furthermore, there exists $\lambda^* \in \mathrm{Span}(U)$ such that

$$\nabla f(\mathbf{x}^*) + \frac{1}{\alpha} U\lambda^* = 0. \tag{20}$$

The existence of $\lambda^*$ is proved in (Shi et al., 2015, Lemma 3.1). $U\mathbf{x}^* = 0$ and (20) are the Karush-Kuhn-Tucker (KKT) optimality conditions of problem (19).

### 4.1 Non-accelerated VR-EXTRA and VR-DIGing

We first give a classical property of the VR technique (Johnson and Zhang, 2013; Xiao and Zhang, 2014).

**Lemma 16** *Suppose that Assumption 1 and condition (7) hold. Then for algorithm (9a)-(9d), we have*

$$\begin{aligned}\mathbb{E}_{\mathbb{S}^k}\left[\|\nabla^k - \nabla f(\mathbf{x}^k)\|^2\right] \leq & \frac{4\overline{L}_f}{b}\left(f(\mathbf{x}^k) - f(\mathbf{x}^*) + \frac{1}{\alpha}\left\langle \lambda^*, U\mathbf{x}^k\right\rangle\right) \\ & + \frac{4\overline{L}_f}{b}\left(f(\mathbf{w}^k) - f(\mathbf{x}^*) + \frac{1}{\alpha}\left\langle \lambda^*, U\mathbf{w}^k\right\rangle\right).\end{aligned} \tag{21}$$

**Proof** From the proof of Lemma D.2 in (Allen-Zhu, 2018), we have

$$\mathbb{E}_{\mathbb{S}^k_{(i)}}\left[\|\nabla^k_{(i)} - \nabla f_{(i)}(x^k_{(i)})\|^2\right] \leq \frac{1}{b}\mathbb{E}_{j\sim\mathcal{D}_{(i)}}\left[\left\|\frac{1}{np_{(i),j}}\left(\nabla f_{(i),j}(x^k_{(i)}) - \nabla f_{(i),j}(w^k_{(i)})\right)\right\|^2\right],$$

where $\mathcal{D}_{(i)}$ is defined in Algorithm 1. Using identity $\|a+b\|^2 \leq 2\|b\|^2 + 2\|b\|^2$ and (18), from the proof of Lemma D.2 in (Allen-Zhu, 2018) and recalling that $p_{(i),j} = \frac{L_{(i),j}}{n\overline{L}_{(i)}}$, we have

$$\begin{aligned}&\mathbb{E}_{\mathbb{S}^k_{(i)}}\left[\|\nabla^k_{(i)} - \nabla f_{(i)}(x^k_{(i)})\|^2\right] \\ &\leq \frac{2}{b}\mathbb{E}_{j\sim\mathcal{D}_{(i)}}\left[\left\|\frac{1}{np_{(i),j}}\left(\nabla f_{(i),j}(x^k_{(i)}) - \nabla f_{(i),j}(x^*)\right)\right\|^2\right] \\ &\quad + \frac{2}{b}\mathbb{E}_{j\sim\mathcal{D}_{(i)}}\left[\left\|\frac{1}{np_{(i),j}}\left(\nabla f_{(i),j}(w^k_{(i)}) - \nabla f_{(i),j}(x^*)\right)\right\|^2\right] \\ &\leq \frac{4\overline{L}_{(i)}}{b}\left(f_{(i)}(x^k_{(i)}) - f_{(i)}(x^*) - \left\langle \nabla f_{(i)}(x^*), x^k_{(i)} - x^*\right\rangle\right) \\ &\quad + \frac{4\overline{L}_{(i)}}{b}\left(f_{(i)}(w^k_{(i)}) - f_{(i)}(x^*) - \left\langle \nabla f_{(i)}(x^*), w^k_{(i)} - x^*\right\rangle\right).\end{aligned}$$

From the convexity of $f_{(i)}(x)$, the definitions in (2), (3), and (10), and the fact that $\mathbb{S}^k_{(i)}$ and $\mathbb{S}^k_{(j)}$ are selected independently for all $i$ and $j$, we have

$$\begin{aligned}\mathbb{E}_{\mathbb{S}^k}\left[\|\nabla^k - \nabla f(\mathbf{x}^k)\|^2\right] \leq & \frac{4\overline{L}_f}{b}\left(f(\mathbf{x}^k) - f(\mathbf{x}^*) - \left\langle \nabla f(\mathbf{x}^*), \mathbf{x}^k - \mathbf{x}^*\right\rangle\right) \\ & + \frac{4\overline{L}_f}{b}\left(f(\mathbf{w}^k) - f(\mathbf{x}^*) - \left\langle \nabla f(\mathbf{x}^*), \mathbf{w}^k - \mathbf{x}^*\right\rangle\right).\end{aligned}$$

From the optimality condition in (20) and $U\mathbf{x}^* = 0$, we have the conclusion. ■

The following property is also useful in the analysis of mini-batch VR methods.

**Lemma 17** *For algorithm (9a)-(9d), we have*

$$\mathbb{E}_{\mathbb{S}^k}\left[\nabla^k\right] = \nabla f(\mathbf{x}^k). \tag{22}$$

**Proof** From the definition of $\nabla^k_{(i)}$ in (9a), and the fact that the elements in $\mathbb{S}^k_{(i)}$ are selected independently with replacement, we have

$$\mathbb{E}_{\mathbb{S}^k_{(i)}}\left[\nabla^k_{(i)}\right] = \mathbb{E}_{j\sim\mathcal{D}_{(i)}}\left[\frac{1}{np_{(i),j}}\left(\nabla f_{(i),j}(x^k_{(i)}) - \nabla f_{(i),j}(w^k_{(i)})\right) + \nabla f_{(i)}(w^k_{(i)})\right] = \nabla f_{(i)}(x^k_{(i)}).$$

Using the definitions in (2) and (10), we have the conclusion. ■

The next lemma describes a progress in one iteration of (9a)-(9d).

**Lemma 18** *Suppose that Assumption 1 and conditions (7) and (15) hold. Then for algorithm (9a)-(9d), we have*

$$\begin{aligned}
&\mathbb{E}_{\mathbb{S}^k}\left[f(\mathbf{x}^{k+1}) - f(\mathbf{x}^*) + \frac{1}{\alpha}\left\langle\lambda^*, U\mathbf{x}^{k+1}\right\rangle\right]\\
&\le \left(\frac{1}{2\alpha} - \frac{\mu}{2}\right)\|\mathbf{x}^k - \mathbf{x}^*\|^2 - \frac{1}{2\alpha}\mathbb{E}_{\mathbb{S}^k}\left[\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2\right]\\
&\quad + \frac{1}{2\alpha}\left(\|\lambda^k - \lambda^*\|^2 - \mathbb{E}_{\mathbb{S}^k}\left[\|\lambda^{k+1} - \lambda^*\|^2\right]\right)\\
&\quad + \frac{1}{2\tau}\mathbb{E}_{\mathbb{S}^k}\left[\|\nabla f(\mathbf{x}^k) - \nabla^k\|^2\right] - \frac{1}{2\alpha}\|V\mathbf{x}^k\|^2 - \left(\frac{1}{4\alpha} - \frac{\tau + L_m}{2}\right)\mathbb{E}_{\mathbb{S}^k}\left[\|\mathbf{x}^{k+1} - \mathbf{x}^k\|^2\right],
\end{aligned} \tag{23}$$

*for some $\tau > 0$ and $L_m = \max\{L_f, \kappa\mu\}$.*

**Proof** From the $L_f$-smoothness of $f(\mathbf{x})$ and the definition of $L_m$, we have

$$\begin{aligned}
f(\mathbf{x}^{k+1}) \le& f(\mathbf{x}^k) + \left\langle\nabla f(\mathbf{x}^k), \mathbf{x}^{k+1} - \mathbf{x}^k\right\rangle + \frac{L_m}{2}\|\mathbf{x}^{k+1} - \mathbf{x}^k\|^2\\
=& f(\mathbf{x}^k) + \left\langle\nabla f(\mathbf{x}^k) - \nabla^k, \mathbf{x}^{k+1} - \mathbf{x}^k\right\rangle + \left\langle\nabla^k, \mathbf{x}^{k+1} - \mathbf{x}^k\right\rangle + \frac{L_m}{2}\|\mathbf{x}^{k+1} - \mathbf{x}^k\|^2\\
\overset{a}{\le}& f(\mathbf{x}^k) + \frac{1}{2\tau}\|\nabla f(\mathbf{x}^k) - \nabla^k\|^2 + \frac{\tau + L_m}{2}\|\mathbf{x}^{k+1} - \mathbf{x}^k\|^2\\
&+ \left\langle\nabla^k, \mathbf{x}^{k+1} - \mathbf{x}^*\right\rangle + \left\langle\nabla^k, \mathbf{x}^* - \mathbf{x}^k\right\rangle,
\end{aligned} \tag{24}$$

where we use Young's inequality in $\overset{a}{\le}$. Since

$$\nabla^k = \frac{1}{\alpha}(\mathbf{x}^k - \mathbf{x}^{k+1}) - \frac{1}{\alpha}U\lambda^k - \frac{1}{\alpha}V^2\mathbf{x}^k, \tag{25}$$

$$\lambda^{k+1} = \lambda^k + U\mathbf{x}^{k+1}, \tag{26}$$

from (9b) and (9c), we have

$$\begin{aligned}
&\left\langle \nabla^k, \mathbf{x}^{k+1}-\mathbf{x}^*\right\rangle\\
&\overset{b}{=} \frac{1}{\alpha}\left\langle \mathbf{x}^k-\mathbf{x}^{k+1}, \mathbf{x}^{k+1}-\mathbf{x}^*\right\rangle - \frac{1}{\alpha}\left\langle \lambda^k, U\mathbf{x}^{k+1}\right\rangle - \frac{1}{\alpha}\left\langle V\mathbf{x}^k, V\mathbf{x}^{k+1}\right\rangle\\
&\overset{c}{=} \frac{1}{\alpha}\left\langle \mathbf{x}^k-\mathbf{x}^{k+1}, \mathbf{x}^{k+1}-\mathbf{x}^*\right\rangle - \frac{1}{\alpha}\left\langle \lambda^*, U\mathbf{x}^{k+1}\right\rangle\\
&\quad - \frac{1}{\alpha}\left\langle \lambda^k-\lambda^*, \lambda^{k+1}-\lambda^k\right\rangle - \frac{1}{\alpha}\left\langle V\mathbf{x}^k, V\mathbf{x}^{k+1}\right\rangle\\
&= \frac{1}{2\alpha}\left(\|\mathbf{x}^k-\mathbf{x}^*\|^2 - \|\mathbf{x}^{k+1}-\mathbf{x}^*\|^2 - \|\mathbf{x}^{k+1}-\mathbf{x}^k\|^2\right)\\
&\quad + \frac{1}{2\alpha}\left(\|\lambda^k-\lambda^*\|^2 - \|\lambda^{k+1}-\lambda^*\|^2 + \|\lambda^{k+1}-\lambda^k\|^2\right)\\
&\quad - \frac{1}{2\alpha}\left(\|V\mathbf{x}^k\|^2 + \|V\mathbf{x}^{k+1}\|^2 - \|V\mathbf{x}^k - V\mathbf{x}^{k+1}\|^2\right) - \frac{1}{\alpha}\left\langle \lambda^*, U\mathbf{x}^{k+1}\right\rangle\\
&\overset{d}{\le} \frac{1}{2\alpha}\left(\|\mathbf{x}^k-\mathbf{x}^*\|^2 - \|\mathbf{x}^{k+1}-\mathbf{x}^*\|^2\right) + \frac{1}{2\alpha}\left(\|\lambda^k-\lambda^*\|^2 - \|\lambda^{k+1}-\lambda^*\|^2\right)\\
&\quad - \frac{1}{2\alpha}\|V\mathbf{x}^k\|^2 - \frac{1}{4\alpha}\|\mathbf{x}^{k+1}-\mathbf{x}^k\|^2 - \frac{1}{\alpha}\left\langle \lambda^*, U\mathbf{x}^{k+1}\right\rangle,
\end{aligned} \tag{27}$$

where we use $U\mathbf{x}^*=0$, $V\mathbf{x}^*=0$, and the symmetry of $U$ and $V$ in $\overset{b}{=}$, (26) in $\overset{c}{=}$, $\|\lambda^{k+1}-\lambda^k\|^2 = \|U\mathbf{x}^{k+1}\|^2 \le \|V\mathbf{x}^{k+1}\|^2$ and $\|V(\mathbf{x}^{k+1}-\mathbf{x}^k)\|^2 \le \frac{1}{2}\|\mathbf{x}^{k+1}-\mathbf{x}^k\|^2$ in $\overset{d}{\le}$. On the other hand, from (22) and the strong convexity of $f(\mathbf{x})$, we have

$$\mathbb{E}_{\mathbb{S}^k}\left[\left\langle \nabla^k, \mathbf{x}^*-\mathbf{x}^k\right\rangle\right] = \left\langle \nabla f(\mathbf{x}^k), \mathbf{x}^*-\mathbf{x}^k\right\rangle \le f(\mathbf{x}^*) - f(\mathbf{x}^k) - \frac{\mu}{2}\|\mathbf{x}^k-\mathbf{x}^*\|^2. \tag{28}$$

Plugging (27) and (28) into (24), we have

$$\begin{aligned}
&\mathbb{E}_{\mathbb{S}^k}\left[f(\mathbf{x}^{k+1})\right]\\
&\le f(\mathbf{x}^*) - \frac{\mu}{2}\|\mathbf{x}^k-\mathbf{x}^*\|^2 + \frac{1}{2\tau}\mathbb{E}_{\mathbb{S}^k}\left[\|\nabla f(\mathbf{x}^k)-\nabla^k\|^2\right]\\
&\quad + \frac{1}{2\alpha}\left(\|\mathbf{x}^k-\mathbf{x}^*\|^2 - \mathbb{E}_{\mathbb{S}^k}\left[\|\mathbf{x}^{k+1}-\mathbf{x}^*\|^2\right]\right) + \frac{1}{2\alpha}\left(\|\lambda^k-\lambda^*\|^2 - \mathbb{E}_{\mathbb{S}^k}\left[\|\lambda^{k+1}-\lambda^*\|^2\right]\right)\\
&\quad - \frac{1}{2\alpha}\|V\mathbf{x}^k\|^2 - \left(\frac{1}{4\alpha} - \frac{\tau+L_m}{2}\right)\mathbb{E}_{\mathbb{S}^k}\left[\|\mathbf{x}^{k+1}-\mathbf{x}^k\|^2\right] - \frac{1}{\alpha}\mathbb{E}_{\mathbb{S}^k}\left[\left\langle \lambda^*, U\mathbf{x}^{k+1}\right\rangle\right].
\end{aligned}$$

Rearranging the terms, we have the conclusion. ■

To prove the linear convergence, we should make the constant before $\|\lambda^k-\lambda^*\|^2$ in (23) smaller than that before $\|\lambda^{k+1}-\lambda^*\|^2$, which is established in the next lemma.

**Lemma 19** *Suppose that Assumption 1 and conditions (7) and (15) hold. Let $\alpha = \frac{1}{28L_m}$ and $\lambda^0 = 0$. Then for algorithm (9a)-(9d), we have*

$$\begin{aligned}
&\frac{1}{2}\mathbb{E}_{\mathbb{S}^k}\left[f(\mathbf{x}^{k+1}) - f(\mathbf{x}^*) + \frac{1}{\alpha}\left\langle \lambda^*, U\mathbf{x}^{k+1}\right\rangle\right]\\
&\le \left(\frac{1}{2\alpha} - \frac{\mu}{2}\right)\|\mathbf{x}^k-\mathbf{x}^*\|^2 - \frac{1}{2\alpha}\mathbb{E}_{\mathbb{S}^k}\left[\|\mathbf{x}^{k+1}-\mathbf{x}^*\|^2\right]
\end{aligned} \tag{29}$$

$$+\left(\frac{1}{2\alpha}-\frac{1-\nu}{4\kappa L_m\alpha^2}\right)\|\lambda^k-\lambda^*\|^2-\frac{1}{2\alpha}\mathbb{E}_{\mathbb{S}^k}\left[\|\lambda^{k+1}-\lambda^*\|^2\right]$$
$$+\frac{\overline{L}_f}{6L_mb}\left(f(\mathbf{x}^k)-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{x}^k\right\rangle\right)+\frac{\overline{L}_f}{6L_mb}\left(f(\mathbf{w}^k)-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{w}^k\right\rangle\right),$$

*with* $\nu=\frac{3140}{3141}$.

**Proof** From the optimality condition in (20) and the smoothness property in (18), we have

$$\begin{aligned}
&f(\mathbf{x}^{k+1})-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{x}^{k+1}\right\rangle=f(\mathbf{x}^{k+1})-f(\mathbf{x}^*)-\left\langle\nabla f(\mathbf{x}^*),\mathbf{x}^{k+1}-\mathbf{x}^*\right\rangle\\
&\geq\frac{1}{2L_m}\|\nabla f(\mathbf{x}^{k+1})-\nabla f(\mathbf{x}^*)\|^2=\frac{1}{2L_m\alpha^2}\|\alpha\nabla f(\mathbf{x}^{k+1})+U\lambda^*\|^2\\
&\overset{a}{=}\frac{1}{2L_m\alpha^2}\left\|\mathbf{x}^{k+1}-\mathbf{x}^k+U(\lambda^k-\lambda^*)+V^2\mathbf{x}^k+\alpha\nabla^k-\alpha\nabla f(\mathbf{x}^k)+\alpha\nabla f(\mathbf{x}^k)-\alpha\nabla f(\mathbf{x}^{k+1})\right\|^2\\
&\overset{b}{\geq}\frac{1-\nu}{2L_m\alpha^2}\|U(\lambda^k-\lambda^*)\|^2-\frac{1}{2L_m\alpha^2}\left(\frac{1}{\nu}-1\right)\left\|\mathbf{x}^{k+1}-\mathbf{x}^k+V^2\mathbf{x}^k\right.\\
&\qquad\qquad\left.+\alpha\nabla^k-\alpha\nabla f(\mathbf{x}^k)+\alpha\nabla f(\mathbf{x}^k)-\alpha\nabla f(\mathbf{x}^{k+1})\right\|^2\\
&\overset{c}{\geq}\frac{1-\nu}{2\kappa L_m\alpha^2}\|\lambda^k-\lambda^*\|^2-\left(\frac{2}{L_m\alpha^2}+2L_m\right)\left(\frac{1}{\nu}-1\right)\|\mathbf{x}^{k+1}-\mathbf{x}^k\|^2\\
&\quad-\frac{2}{L_m}\left(\frac{1}{\nu}-1\right)\|\nabla^k-\nabla f(\mathbf{x}^k)\|^2-\frac{2}{L_m\alpha^2}\left(\frac{1}{\nu}-1\right)\|V\mathbf{x}^k\|^2,
\end{aligned}\tag{30}$$

where we use (25) in $\overset{a}{=}$, $\|a-b\|^2\geq(1-\nu)\|a\|^2-(\frac{1}{\nu}-1)\|b\|^2$ in $\overset{b}{\geq}$ for some $0<\nu<1$, (15), $\|\sum_{i=1}^n a_i\|^2\leq n\sum_{i=1}^n\|a_i\|^2$, the $L_f$-smoothness of $f(\mathbf{x})$, and $\|V^2\mathbf{x}^k\|^2\leq\|V\mathbf{x}^k\|^2$ in $\overset{c}{\geq}$, where the requirement of $\lambda^k-\lambda^*\in\text{Span}(U)$ in (15) holds since $\lambda^0\in\text{Span}(U)$, the update in (9c), and $\lambda^*\in\text{Span}(U)$. Dividing both sides of (30) by 2 and plugging it into (23), we have

$$\begin{aligned}
&\frac{1}{2}\mathbb{E}_{\mathbb{S}^k}\left[f(\mathbf{x}^{k+1})-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{x}^{k+1}\right\rangle\right]\\
&\leq\left(\frac{1}{2\alpha}-\frac{\mu}{2}\right)\|\mathbf{x}^k-\mathbf{x}^*\|^2-\frac{1}{2\alpha}\mathbb{E}_{\mathbb{S}^k}\left[\|\mathbf{x}^{k+1}-\mathbf{x}^*\|^2\right]\\
&\quad+\left(\frac{1}{2\alpha}-\frac{1-\nu}{4\kappa L_m\alpha^2}\right)\|\lambda^k-\lambda^*\|^2-\frac{1}{2\alpha}\mathbb{E}_{\mathbb{S}^k}\left[\|\lambda^{k+1}-\lambda^*\|^2\right]\\
&\quad+\left(\frac{1}{2\tau}+\frac{1}{L_m}\left(\frac{1}{\nu}-1\right)\right)\mathbb{E}_{\mathbb{S}^k}\left[\|\nabla f(\mathbf{x}^k)-\nabla^k\|^2\right]-\left(\frac{1}{2\alpha}-\frac{1}{L_m\alpha^2}\left(\frac{1}{\nu}-1\right)\right)\|V\mathbf{x}^k\|^2\\
&\quad-\left(\frac{1}{4\alpha}-\frac{\tau+L_m}{2}-\left(\frac{1}{L_m\alpha^2}+L_m\right)\left(\frac{1}{\nu}-1\right)\right)\mathbb{E}_{\mathbb{S}^k}\left[\|\mathbf{x}^k-\mathbf{x}^{k+1}\|^2\right].
\end{aligned}$$

Letting $\tau=12.5L_m$, $\nu=\frac{3140}{3141}$, and $\alpha=\frac{1}{28L_m}$, such that $\frac{1}{2\alpha}-\frac{1}{L_m\alpha^2}(\frac{1}{\nu}-1)\geq 0$, $\frac{1}{4\alpha}-\frac{\tau+L_m}{2}-(\frac{1}{L_m\alpha^2}+L_m)(\frac{1}{\nu}-1)\geq 0$, and $\frac{1}{2\tau}+\frac{1}{L_m}(\frac{1}{\nu}-1)\leq\frac{1}{24L_m}$, and using (21), we have the conclusion. ■

Now, we are ready to prove Theorem 2.

**Proof** From step (9d), we have

$$\begin{aligned}&\mathbb{E}_{w_{(i)}^{k+1}}\left[f_{(i)}(w_{(i)}^{k+1})-\left\langle\nabla f_{(i)}(x^*),w_{(i)}^{k+1}-x^*\right\rangle\right]\\&=\frac{b}{n}\left(f_{(i)}(x_{(i)}^k)-\left\langle\nabla f_{(i)}(x^*),x_{(i)}^k-x^*\right\rangle\right)+\left(1-\frac{b}{n}\right)\left(f_{(i)}(w_{(i)}^k)-\left\langle\nabla f_{(i)}(x^*),w_{(i)}^k-x^*\right\rangle\right).\end{aligned}$$

From the definitions in (2), the optimality condition in (20), and the fact that each $w_{(i)}^{k+1}$ is computed independently at each node, we further have

$$\begin{aligned}&\mathbb{E}_{\mathbf{w}^{k+1}}\left[f(\mathbf{w}^{k+1})-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{w}^{k+1}\right\rangle\right]\\&=\frac{b}{n}\left(f(\mathbf{x}^k)-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{x}^k\right\rangle\right)\\&\quad+\left(1-\frac{b}{n}\right)\left(f(\mathbf{w}^k)-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{w}^k\right\rangle\right).\end{aligned}\tag{31}$$

Multiplying both sides of (31) by $\frac{n}{b}(\frac{1}{2}-\frac{b}{10n}-\frac{\overline{L}_f}{6L_mb})$ and adding it to (29), taking expectation with respect to $\xi^k$, from the easy-to-identity equation $\frac{n}{b}(\frac{1}{2}-\frac{b}{10n}-\frac{\overline{L}_f}{6L_mb})(1-\frac{b}{n})+\frac{\overline{L}_f}{6L_mb}\leq\frac{n}{b}(\frac{1}{2}-\frac{b}{10n}-\frac{\overline{L}_f}{6L_mb})(1-\frac{b}{10n})$ under the condition $\frac{\overline{L}_f}{L_mb}\leq 1$, we have

$$\begin{aligned}&\frac{1}{2}\mathbb{E}_{\xi^{k+1}}\left[f(\mathbf{x}^{k+1})-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{x}^{k+1}\right\rangle\right]\\&\quad+\frac{n}{b}\left(\frac{1}{2}-\frac{b}{10n}-\frac{\overline{L}_f}{6L_mb}\right)\mathbb{E}_{\xi^{k+1}}\left[f(\mathbf{w}^{k+1})-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{w}^{k+1}\right\rangle\right]\\&\quad+\frac{1}{2\alpha}\mathbb{E}_{\xi^{k+1}}\left[\|\mathbf{x}^{k+1}-\mathbf{x}^*\|^2\right]+\frac{1}{2\alpha}\mathbb{E}_{\xi^{k+1}}\left[\|\lambda^{k+1}-\lambda^*\|^2\right]\\&\leq\left(\frac{1}{2}-\frac{b}{10n}\right)\mathbb{E}_{\xi^k}\left[f(\mathbf{x}^k)-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{x}^k\right\rangle\right]\\&\quad+\frac{n}{b}\left(\frac{1}{2}-\frac{b}{10n}-\frac{\overline{L}_f}{6L_mb}\right)\left(1-\frac{b}{10n}\right)\mathbb{E}_{\xi^k}\left[f(\mathbf{w}^k)-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{w}^k\right\rangle\right]\\&\quad+\left(\frac{1}{2\alpha}-\frac{\mu}{2}\right)\mathbb{E}_{\xi^k}\left[\|\mathbf{x}^k-\mathbf{x}^*\|^2\right]+\left(\frac{1}{2\alpha}-\frac{1-\nu}{4\kappa L_m\alpha^2}\right)\mathbb{E}_{\xi^k}\left[\|\lambda^k-\lambda^*\|^2\right]\\&\overset{a}{\leq}\left\{\frac{1}{2}\mathbb{E}_{\xi^k}\left[f(\mathbf{x}^k)-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{x}^k\right\rangle\right]\right.\\&\qquad+\frac{n}{b}\left(\frac{1}{2}-\frac{b}{10n}-\frac{\overline{L}_f}{6L_mb}\right)\mathbb{E}_{\xi^k}\left[f(\mathbf{w}^k)-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{w}^k\right\rangle\right]\\&\qquad\left.+\frac{1}{2\alpha}\mathbb{E}_{\xi^k}\left[\|\mathbf{x}^k-\mathbf{x}^*\|^2\right]+\frac{1}{2\alpha}\mathbb{E}_{\xi^k}\left[\|\lambda^k-\lambda^*\|^2\right]\right\}\\&\quad\times\max\left\{1-\frac{b}{5n},1-\frac{b}{10n},1-\alpha\mu,1-\frac{1-\nu}{2\kappa L_m\alpha}\right\},\end{aligned}$$

where we use the fact $f(\mathbf{x})+\frac{1}{\alpha}\langle\lambda^*, U\mathbf{x}\rangle \geq f(\mathbf{x}^*)+\frac{1}{\alpha}\langle\lambda^*, U\mathbf{x}^*\rangle$ for any $\mathbf{x}$, and $\frac{1}{2}-\frac{b}{10n}-\frac{\overline{L}_f}{6L_m b} > 0$ in $\overset{a}{\leq}$. From the setting of $\alpha = \frac{1}{28L_m}$, we know the algorithm needs $\mathcal{O}((\frac{n}{b}+\frac{L_m}{\mu}+\kappa)\log\frac{1}{\epsilon}) \overset{b}{=} \mathcal{O}((\frac{n}{b}+\frac{L_f}{\mu}+\kappa)\log\frac{1}{\epsilon})$ iterations to find $\mathbf{x}^k$ such that $\mathbb{E}_{\xi^k}\left[\|\mathbf{x}^k-\mathbf{x}^*\|^2\right] \leq \epsilon$, where we use $L_m = \max\{L_f, \kappa\mu\}$ in $\overset{b}{=}$. Recall that each iteration requires the time of one communication round and $b$ stochastic gradient evaluations in average, see Remark 3. So the communication complexity is $\mathcal{O}((\frac{n}{b}+\frac{L_f}{\mu}+\kappa)\log\frac{1}{\epsilon})$, and the stochastic gradient computation complexity is $\mathcal{O}((n+\frac{bL_f}{\mu}+b\kappa)\log\frac{1}{\epsilon}) = \mathcal{O}((n+\frac{bL_m}{\mu})\log\frac{1}{\epsilon})$.

Case 1. If $\kappa \leq \max\{\frac{\overline{L}_f}{\mu}, n\}$, we have $\kappa\mu \leq \max\{\overline{L}_f, n\mu\}$ and $b = \frac{\max\{\overline{L}_f, n\mu\}}{\max\{L_f, \kappa\mu\}} \geq \frac{\max\{\overline{L}_f, n\mu\}}{\max\{\overline{L}_f, n\mu\}} = 1$, where we use $L_f \leq \overline{L}_f \leq \max\{\overline{L}_f, n\mu\}$. We also have $b = \frac{\max\{\overline{L}_f, n\mu\}}{\max\{L_f, \kappa\mu\}} \leq \frac{\max\{\overline{L}_f, n\mu\}}{L_f} \leq n$, where we use $\overline{L}_f \leq nL_f$ given in (4) and $L_f \geq \mu$. This verifies that the setting of $b$ is meaningful. On the other hand, since $b = \frac{\max\{\overline{L}_f, n\mu\}}{L_m}$, we have $\frac{\overline{L}_f}{L_m b} = \frac{\overline{L}_f}{\max\{\overline{L}_f, n\mu\}} \leq 1$ and $\frac{n}{b} = \frac{nL_m}{\max\{\overline{L}_f, n\mu\}} \leq \frac{nL_m}{n\mu} = \max\{\frac{L_f}{\mu}, \kappa\}$. Then the communication complexity is $\mathcal{O}((\frac{L_f}{\mu}+\kappa)\log\frac{1}{\epsilon})$, and the stochastic gradient computation complexity is $\mathcal{O}((\frac{\overline{L}_f}{\mu}+n)\log\frac{1}{\epsilon})$, where we use $\frac{bL_m}{\mu} = \frac{\max\{\overline{L}_f, n\mu\}}{\mu} \leq \frac{\overline{L}_f}{\mu}+n$.

If we choose $b \geq \frac{\max\{\overline{L}_f, n\mu\}}{L_m}$, similar to the above analysis, we also have $\frac{\overline{L}_f}{L_m b} \leq 1$ and $\frac{n}{b} \leq \max\{\frac{L_f}{\mu}, \kappa\}$. So the communication complexity remains unchanged, but the stochastic gradient computation complexity increases. This verifies Remark 7(4). Specially, if we let $b = \frac{\max\{\overline{L}_f, n\mu\}}{\max\{L_f, \kappa\mu\}}\frac{\tau(\kappa_b+\kappa)}{\kappa_s+n}$, we have $\frac{bL_m}{\mu} = \tau(\kappa_b+\kappa)$.

Case 2. If $\kappa \geq \max\{\frac{\overline{L}_f}{\mu}, n\}$, letting $b = 1$, we have $\frac{\overline{L}_f}{L_m b} = \frac{\overline{L}_f}{L_m} \leq \frac{\overline{L}_f}{\kappa\mu} \leq 1$. The communication complexity and stochastic gradient computation complexity are both $\mathcal{O}((\frac{L_f}{\mu}+n+\kappa)\log\frac{1}{\epsilon}) = \mathcal{O}((\frac{L_f}{\mu}+\kappa)\log\frac{1}{\epsilon})$, where we use $\kappa \geq n$. ■

Now, we prove Theorem 4, which reduces the stochastic gradient computation complexity from $\mathcal{O}((\frac{L_f}{\mu}+\kappa)\log\frac{1}{\epsilon})$ to $\mathcal{O}((\frac{\overline{L}_f}{\mu}+n)\log\frac{1}{\epsilon})$ in the case of $\kappa \geq \max\{\frac{\overline{L}_f}{\mu}, n\}$ by the zero-sample strategy.

**Proof** From the definitions in (14), it can be easily checked that $\overline{L}'_{(i)} = \frac{\sum_{j=1}^{n'} L_{(i),j}}{n'} = \frac{\sum_{j=1}^{n} L_{(i),j}+\sum_{j=n+1}^{n'} L_{(i),j}}{n'} = \frac{n\overline{L}_{(i)}+n\mu n'-n\overline{L}_{(i)}}{n'} = n\mu$, $\overline{L}'_f = \overline{L}'_{(i)} = n\mu$, and $b = \frac{\max\{\overline{L}'_f, n'\mu'\}}{\max\{L'_f, \kappa\mu'\}} = \frac{n\mu}{\max\{\frac{nL_f}{\kappa}, n\mu\}} = \frac{n\mu}{n\mu} = 1$, where we use $\kappa \geq \kappa_s \geq \frac{L_f}{\mu}$.

Replacing $n$, $L_f$, $\mu$, and $\overline{L}_f$ in the proof of Theorem 2 by $n'$, $L'_f$, $\mu'$, and $\overline{L}'_f$ given in (14), respectively, we know $L_m = \max\{L'_f, \kappa\mu'\} = n\mu$ and $\frac{\overline{L}'_f}{L_m b} = \frac{n\mu}{n\mu} = 1$. So the algorithm needs $\mathcal{O}((\frac{n'}{b}+\frac{L'_f}{\mu'}+\kappa)\log\frac{1}{\epsilon}) = \mathcal{O}((\kappa+\frac{L_f}{\mu})\log\frac{1}{\epsilon}) = \mathcal{O}(\kappa\log\frac{1}{\epsilon})$ iterations to find $\mathbf{x}^k$ such that $\mathbb{E}_{\xi^k}\left[\|\mathbf{x}^k-\mathbf{x}^*\|^2\right] \leq \epsilon$. So the communication complexity and the stochastic gradient computation complexity are both $\mathcal{O}(\kappa\log\frac{1}{\epsilon})$. Since we select the samples in $\left[1, n\right]$ with

probability $\frac{\sum_{j=1}^{n} L_{(i),j}}{\sum_{j=1}^{n'} L_{(i),j}} = \frac{n\overline{L}_f}{n\mu\kappa} = \frac{\overline{L}_f}{\mu\kappa}$, and the zero samples do not spend the computation time, so the valid number of stochastic gradient evaluations is $\mathcal{O}(\frac{\overline{L}_f}{\mu\kappa}\kappa\log\frac{1}{\epsilon}) = \mathcal{O}(\frac{\overline{L}_f}{\mu}\log\frac{1}{\epsilon})$. On the other hand, we compute the full batch gradient with probability $\frac{b}{n'} = \frac{1}{\kappa}$, which takes $\mathcal{O}(n\frac{1}{\kappa}\kappa\log\frac{1}{\epsilon}) = \mathcal{O}(n\log\frac{1}{\epsilon})$ valid stochastic gradient evaluations in total. So the final valid stochastic gradient computation complexity is $\mathcal{O}((\frac{\overline{L}_f}{\mu}+n)\log\frac{1}{\epsilon})$.

At last, we explain that the zero samples do not destroy the proof of Theorem 2. For the zero sample $f_{(i),j}(x) = 0$, we have $\nabla f_{(i),j}(x) = 0$. So it also satisfies the convexity and $L_{(i),j}$-smooth property (18) even for positive $L_{(i),j}$. We can check that (21) and (22) also hold. In the proofs of Lemmas 18 and 19, we use the smoothness and strong convexity of $f'_{(i)}(x)$, as explained in Section 2.3, which also hold. ■

### 4.2 Accelerated VR-EXTRA and VR-DIGing

From Lemma D.2 in (Allen-Zhu, 2018) and similar to Lemma 16, we have

$$\mathbb{E}_{\mathbb{S}^k}\left[\|\nabla^k - \nabla f(\mathbf{y}^k)\|^2\right] \le \frac{2\overline{L}_f}{b}\left(f(\mathbf{w}^k) - f(\mathbf{y}^k) - \left\langle \nabla f(\mathbf{y}^k), \mathbf{w}^k - \mathbf{y}^k\right\rangle\right). \tag{32}$$

Similar to (22), we also have

$$\mathbb{E}_{\mathbb{S}^k}\left[\nabla^k\right] = \nabla f(\mathbf{y}^k). \tag{33}$$

The following lemma is the counterpart of Lemma 18, which gives a progress in one iteration of procedure (16a)-(16f).

**Lemma 20** *Suppose that Assumption 1 and conditions (7) and (15) hold. Let $\theta_1 + \theta_2 \le 1$. Then for algorithm (16a)-(16f), we have*

$$\begin{aligned}
&\mathbb{E}_{\mathbb{S}^k}\left[f(\mathbf{x}^{k+1}) - f(\mathbf{x}^*) + \frac{1}{\alpha}\left\langle\lambda^*, U\mathbf{x}^{k+1}\right\rangle\right] \\
&\le (1-\theta_1-\theta_2)\left(f(\mathbf{x}^k) - f(\mathbf{x}^*) + \frac{1}{\alpha}\left\langle\lambda^*, U\mathbf{x}^k\right\rangle\right) \\
&\quad + \theta_2\left(f(\mathbf{w}^k) - f(\mathbf{x}^*) + \frac{1}{\alpha}\left\langle\lambda^*, U\mathbf{w}^k\right\rangle\right) \\
&\quad + \left(\frac{\overline{L}_f}{\tau b} - \theta_2\right)\left(f(\mathbf{w}^k) - f(\mathbf{y}^k) - \left\langle\nabla f(\mathbf{y}^k), \mathbf{w}^k - \mathbf{y}^k\right\rangle\right) \\
&\quad + \frac{\theta_1^2}{2\alpha}\|\mathbf{z}^k - \mathbf{x}^*\|^2 - \left(\frac{\theta_1^2}{2\alpha} + \frac{\mu\theta_1}{2}\right)\mathbb{E}_{\mathbb{S}^k}\left[\|\mathbf{z}^{k+1} - \mathbf{x}^*\|^2\right] \\
&\quad + \frac{1}{2\alpha}\|\lambda^k - \lambda^*\|^2 - \frac{1}{2\alpha}\mathbb{E}_{\mathbb{S}^k}\left[\|\lambda^{k+1} - \lambda^*\|^2\right] \\
&\quad - \frac{\theta_1^2}{2\alpha}\|V\mathbf{z}^k\|^2 - \left(\frac{\theta_1^2}{4\alpha} - \frac{\tau\theta_1^2 + L_f\theta_1^2}{2}\right)\mathbb{E}_{\mathbb{S}^k}\left[\|\mathbf{z}^{k+1} - \mathbf{z}^k\|^2\right] - \frac{\mu\theta_1}{2}\mathbb{E}_{\mathbb{S}^k}\left[\|\mathbf{z}^{k+1} - \mathbf{y}^k\|^2\right],
\end{aligned} \tag{34}$$

*for some $\tau > 0$.*

**Proof** From the $L_f$-smoothness of $f(\mathbf{x})$, similar to (24), we have

$$\begin{aligned}
f(\mathbf{x}^{k+1}) \leq & f(\mathbf{y}^k) + \left\langle \nabla f(\mathbf{y}^k), \mathbf{x}^{k+1} - \mathbf{y}^k \right\rangle + \frac{L_f}{2}\|\mathbf{x}^{k+1} - \mathbf{y}^k\|^2 \\
\leq & f(\mathbf{y}^k) + \frac{1}{2\tau}\|\nabla f(\mathbf{y}^k) - \nabla^k\|^2 + \frac{\tau + L_f}{2}\|\mathbf{x}^{k+1} - \mathbf{y}^k\|^2 + \left\langle \nabla^k, \mathbf{x}^{k+1} - \mathbf{y}^k \right\rangle \\
\overset{a}{=} & f(\mathbf{y}^k) + \frac{1}{2\tau}\|\nabla f(\mathbf{y}^k) - \nabla^k\|^2 + \frac{\tau\theta_1^2 + L_f\theta_1^2}{2}\|\mathbf{z}^{k+1} - \mathbf{z}^k\|^2 \\
& + \theta_1 \left\langle \nabla^k, \mathbf{z}^{k+1} - \mathbf{z}^* \right\rangle + \theta_1 \left\langle \nabla^k, \mathbf{z}^* - \mathbf{z}^k \right\rangle.
\end{aligned} \tag{35}$$

where we use

$$\mathbf{x}^{k+1} - \mathbf{y}^k = \theta_1(\mathbf{z}^{k+1} - \mathbf{z}^k) \tag{36}$$

in $\overset{a}{=}$, which comes from (16e). Since

$$\nabla^k = \frac{\theta_1}{\alpha}(\mathbf{z}^k - \mathbf{z}^{k+1}) + \mu(\mathbf{y}^k - \mathbf{z}^{k+1}) - \frac{1}{\alpha}U\lambda^k - \frac{\theta_1}{\alpha}V^2\mathbf{z}^k, \tag{37}$$

$$\lambda^{k+1} = \lambda^k + \theta_1 U\mathbf{z}^{k+1}, \tag{38}$$

from (16c) and (16d), similar to (27), we have

$$\begin{aligned}
& \theta_1 \left\langle \nabla^k, \mathbf{z}^{k+1} - \mathbf{x}^* \right\rangle \\
= & \frac{\theta_1^2}{\alpha} \left\langle \mathbf{z}^k - \mathbf{z}^{k+1}, \mathbf{z}^{k+1} - \mathbf{x}^* \right\rangle + \mu\theta_1 \left\langle \mathbf{y}^k - \mathbf{z}^{k+1}, \mathbf{z}^{k+1} - \mathbf{x}^* \right\rangle \\
& - \frac{\theta_1}{\alpha} \left\langle \lambda^k, U\mathbf{z}^{k+1} \right\rangle - \frac{\theta_1^2}{\alpha} \left\langle V\mathbf{z}^k, V\mathbf{z}^{k+1} \right\rangle \\
\overset{b}{=} & \frac{\theta_1^2}{\alpha} \left\langle \mathbf{z}^k - \mathbf{z}^{k+1}, \mathbf{z}^{k+1} - \mathbf{x}^* \right\rangle + \mu\theta_1 \left\langle \mathbf{y}^k - \mathbf{z}^{k+1}, \mathbf{z}^{k+1} - \mathbf{x}^* \right\rangle \\
& - \frac{\theta_1}{\alpha} \left\langle \lambda^*, U\mathbf{z}^{k+1} \right\rangle - \frac{1}{\alpha} \left\langle \lambda^k - \lambda^*, \lambda^{k+1} - \lambda^k \right\rangle - \frac{\theta_1^2}{\alpha} \left\langle V\mathbf{z}^k, V\mathbf{z}^{k+1} \right\rangle \\
= & \frac{\theta_1^2}{2\alpha} \left( \|\mathbf{z}^k - \mathbf{x}^*\|^2 - \|\mathbf{z}^{k+1} - \mathbf{x}^*\|^2 - \|\mathbf{z}^{k+1} - \mathbf{z}^k\|^2 \right) \\
& + \frac{\mu\theta_1}{2} \left( \|\mathbf{y}^k - \mathbf{x}^*\|^2 - \|\mathbf{z}^{k+1} - \mathbf{x}^*\|^2 - \|\mathbf{z}^{k+1} - \mathbf{y}^k\|^2 \right) \\
& + \frac{1}{2\alpha} \left( \|\lambda^k - \lambda^*\|^2 - \|\lambda^{k+1} - \lambda^*\|^2 + \|\lambda^{k+1} - \lambda^k\|^2 \right) \\
& - \frac{\theta_1^2}{2\alpha} \left( \|V\mathbf{z}^k\|^2 + \|V\mathbf{z}^{k+1}\|^2 - \|V\mathbf{z}^{k+1} - V\mathbf{z}^k\|^2 \right) - \frac{\theta_1}{\alpha} \left\langle \lambda^*, U\mathbf{z}^{k+1} \right\rangle \\
\overset{c}{\leq} & \frac{\theta_1^2}{2\alpha} \left( \|\mathbf{z}^k - \mathbf{x}^*\|^2 - \|\mathbf{z}^{k+1} - \mathbf{x}^*\|^2 \right) + \frac{\mu\theta_1}{2} \left( \|\mathbf{y}^k - \mathbf{x}^*\|^2 - \|\mathbf{z}^{k+1} - \mathbf{x}^*\|^2 \right) \\
& + \frac{1}{2\alpha} \left( \|\lambda^k - \lambda^*\|^2 - \|\lambda^{k+1} - \lambda^*\|^2 \right) - \frac{\theta_1^2}{2\alpha}\|V\mathbf{z}^k\|^2 - \frac{\theta_1^2}{4\alpha}\|\mathbf{z}^{k+1} - \mathbf{z}^k\|^2 \\
& - \frac{\mu\theta_1}{2}\|\mathbf{z}^{k+1} - \mathbf{y}^k\|^2 - \frac{\theta_1}{\alpha} \left\langle \lambda^*, U\mathbf{z}^{k+1} \right\rangle,
\end{aligned} \tag{39}$$

where we use (38) in $\overset{b}{=}$, $\|\lambda^{k+1}-\lambda^k\|^2=\|\theta_1 U\mathbf{z}^{k+1}\|^2\le\theta_1^2\|V\mathbf{z}^{k+1}\|^2$ and $\|V(\mathbf{z}^{k+1}-\mathbf{z}^k)\|^2\le\frac{1}{2}\|\mathbf{z}^{k+1}-\mathbf{z}^k\|^2$ in $\overset{c}{\le}$. On the other hand, from (33), we have

$$\begin{aligned}
&\theta_1\mathbb{E}_{\mathbb{S}^k}\left[\left\langle\nabla^k,\mathbf{x}^*-\mathbf{z}^k\right\rangle\right]=\theta_1\left\langle\nabla f(\mathbf{y}^k),\mathbf{x}^*-\mathbf{z}^k\right\rangle\\
&\overset{d}{=}\left\langle\nabla f(\mathbf{y}^k),\theta_1\mathbf{x}^*+\theta_2\mathbf{w}^k+(1-\theta_1-\theta_2)\mathbf{x}^k-\mathbf{y}^k\right\rangle\\
&=\theta_1\left\langle\nabla f(\mathbf{y}^k),\mathbf{x}^*-\mathbf{y}^k\right\rangle+(1-\theta_1-\theta_2)\left\langle\nabla f(\mathbf{y}^k),\mathbf{x}^k-\mathbf{y}^k\right\rangle+\theta_2\left\langle\nabla f(\mathbf{y}^k),\mathbf{w}^k-\mathbf{y}^k\right\rangle\\
&\overset{e}{\le}\theta_1\Big(f(\mathbf{x}^*)-f(\mathbf{y}^k)-\frac{\mu}{2}\|\mathbf{y}^k-\mathbf{x}^*\|^2\Big)+(1-\theta_1-\theta_2)(f(\mathbf{x}^k)-f(\mathbf{y}^k))+\theta_2\Big\langle\nabla f(\mathbf{y}^k),\mathbf{w}^k-\mathbf{y}^k\Big\rangle\\
&=\theta_1 f(\mathbf{x}^*)+(1-\theta_1-\theta_2)f(\mathbf{x}^k)-(1-\theta_2)f(\mathbf{y}^k)-\frac{\mu\theta_1}{2}\|\mathbf{y}^k-\mathbf{x}^*\|^2+\theta_2\left\langle\nabla f(\mathbf{y}^k),\mathbf{w}^k-\mathbf{y}^k\right\rangle, \quad (40)
\end{aligned}$$

where we use (16a) in $\overset{d}{=}$, and the strong convexity of $f(\mathbf{x})$ in $\overset{e}{=}$. Plugging (39) and (40) into (35), and using (32), we have

$$\begin{aligned}
&\mathbb{E}_{\mathbb{S}^k}\left[f(\mathbf{x}^{k+1})\right]\\
&\le\theta_1 f(\mathbf{x}^*)+(1-\theta_1-\theta_2)f(\mathbf{x}^k)+\theta_2 f(\mathbf{y}^k)-\frac{\mu\theta_1}{2}\|\mathbf{y}^k-\mathbf{x}^*\|^2+\theta_2\left\langle\nabla f(\mathbf{y}^k),\mathbf{w}^k-\mathbf{y}^k\right\rangle\\
&\quad+\frac{\overline{L}_f}{\tau b}\left(f(\mathbf{w}^k)-f(\mathbf{y}^k)-\left\langle\nabla f(\mathbf{y}^k),\mathbf{w}^k-\mathbf{y}^k\right\rangle\right)\\
&\quad+\frac{\theta_1^2}{2\alpha}\left(\|\mathbf{z}^k-\mathbf{x}^*\|^2-\mathbb{E}_{\mathbb{S}^k}\left[\|\mathbf{z}^{k+1}-\mathbf{x}^*\|^2\right]\right)+\frac{\mu\theta_1}{2}\left(\|\mathbf{y}^k-\mathbf{x}^*\|^2-\mathbb{E}_{\mathbb{S}^k}\left[\|\mathbf{z}^{k+1}-\mathbf{x}^*\|^2\right]\right)\\
&\quad+\frac{1}{2\alpha}\left(\|\lambda^k-\lambda^*\|^2-\mathbb{E}_{\mathbb{S}^k}\left[\|\lambda^{k+1}-\lambda^*\|^2\right]\right)-\frac{\theta_1}{\alpha}\mathbb{E}_{\mathbb{S}^k}\left[\left\langle\lambda^*,U\mathbf{z}^{k+1}\right\rangle\right]\\
&\quad-\frac{\theta_1^2}{2\alpha}\|V\mathbf{z}^k\|^2-\left(\frac{\theta_1^2}{4\alpha}-\frac{\tau\theta_1^2+L_f\theta_1^2}{2}\right)\mathbb{E}_{\mathbb{S}^k}\left[\|\mathbf{z}^{k+1}-\mathbf{z}^k\|^2\right]-\frac{\mu\theta_1}{2}\mathbb{E}_{\mathbb{S}^k}\left[\|\mathbf{z}^{k+1}-\mathbf{y}^k\|^2\right]\\
&\overset{f}{=}\theta_1 f(\mathbf{x}^*)+(1-\theta_1-\theta_2)f(\mathbf{x}^k)+\theta_2 f(\mathbf{w}^k)\\
&\quad+\left(\frac{\overline{L}_f}{\tau b}-\theta_2\right)\left(f(\mathbf{w}^k)-f(\mathbf{y}^k)-\left\langle\nabla f(\mathbf{y}^k),\mathbf{w}^k-\mathbf{y}^k\right\rangle\right)\\
&\quad+\frac{\theta_1^2}{2\alpha}\|\mathbf{z}^k-\mathbf{x}^*\|^2-\left(\frac{\theta_1^2}{2\alpha}+\frac{\mu\theta_1}{2}\right)\mathbb{E}_{\mathbb{S}^k}\left[\|\mathbf{z}^{k+1}-\mathbf{x}^*\|^2\right]\\
&\quad+\frac{1}{2\alpha}\left(\|\lambda^k-\lambda^*\|^2-\mathbb{E}_{\mathbb{S}^k}\left[\|\lambda^{k+1}-\lambda^*\|^2\right]\right)\\
&\quad-\frac{1}{\alpha}\mathbb{E}_{\mathbb{S}^k}\left[\left\langle\lambda^*,U\mathbf{x}^{k+1}-\theta_2 U\mathbf{w}^k-(1-\theta_1-\theta_2)U\mathbf{x}^k\right\rangle\right]\\
&\quad-\frac{\theta_1^2}{2\alpha}\|V\mathbf{z}^k\|^2-\left(\frac{\theta_1^2}{4\alpha}-\frac{\tau\theta_1^2+L_f\theta_1^2}{2}\right)\mathbb{E}_{\mathbb{S}^k}\left[\|\mathbf{z}^{k+1}-\mathbf{z}^k\|^2\right]-\frac{\mu\theta_1}{2}\mathbb{E}_{\mathbb{S}^k}\left[\|\mathbf{z}^{k+1}-\mathbf{y}^k\|^2\right],
\end{aligned}$$

where we use (16a) and (16e) in $\overset{f}{=}$. Rearranging the terms, we have the conclusion. ■

Similar to Lemma 19, we establish the smaller constant before $\|\lambda^k-\lambda^*\|^2$ than that before $\|\lambda^{k+1}-\lambda^*\|^2$ in the next lemma.

**Lemma 21** *Suppose that Assumption 1 and conditions (7) and (15) hold. Choose $b$ such that $\theta_2 = \frac{\overline{L}_f}{2L_f b} \leq \frac{1}{2}$. Let $\theta_1 \leq \frac{1}{2}$, $\alpha = \frac{1}{10L_f}$, and $\lambda^0 = 0$. Then for algorithm (16a)-(16f), we have*

$$\begin{aligned}
&\left(1-\frac{\theta_1}{2}\right)\mathbb{E}_{\mathbb{S}^k}\left[f(\mathbf{x}^{k+1})-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{x}^{k+1}\right\rangle\right]\\
&\leq(1-\theta_1-\theta_2)\left(f(\mathbf{x}^k)-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{x}^k\right\rangle\right)+\theta_2\left(f(\mathbf{w}^k)-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{w}^k\right\rangle\right)\\
&\quad+\frac{\theta_1^2}{2\alpha}\|\mathbf{z}^k-\mathbf{x}^*\|^2-\left(\frac{\theta_1^2}{2\alpha}+\frac{\mu\theta_1}{2}\right)\mathbb{E}_{\mathbb{S}^k}\left[\|\mathbf{z}^{k+1}-\mathbf{x}^*\|^2\right]\\
&\quad+\left(\frac{1}{2\alpha}-\frac{(1-\nu)\theta_1}{4\kappa L_f\alpha^2}\right)\|\lambda^k-\lambda^*\|^2-\frac{1}{2\alpha}\mathbb{E}_{\mathbb{S}^k}\left[\|\lambda^{k+1}-\lambda^*\|^2\right],
\end{aligned}\tag{41}$$

*with $\nu = \frac{127}{128}$.*

**Proof** From (18) and (20), similar to (30), we have

$$\begin{aligned}
&f(\mathbf{x}^{k+1})-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{x}^{k+1}\right\rangle\geq\frac{1}{2L_f\alpha^2}\|\alpha\nabla f(\mathbf{x}^{k+1})+U\lambda^*\|^2\\
&\overset{a}{=}\frac{1}{2L_f\alpha^2}\Big\|\alpha\mu(\mathbf{z}^{k+1}-\mathbf{y}^k)+\theta_1(\mathbf{z}^{k+1}-\mathbf{z}^k)+U(\lambda^k-\lambda^*)+\theta_1V^2\mathbf{z}^k\\
&\qquad\qquad+\alpha\nabla^k-\alpha\nabla f(\mathbf{y}^k)+\alpha\nabla f(\mathbf{y}^k)-\alpha\nabla f(\mathbf{x}^{k+1})\Big\|^2\\
&\geq\frac{1-\nu}{2L_f\alpha^2}\|U(\lambda^k-\lambda^*)\|^2-\frac{1}{2L_f\alpha^2}\left(\frac{1}{\nu}-1\right)\Big\|\alpha\mu(\mathbf{z}^{k+1}-\mathbf{y}^k)+\theta_1(\mathbf{z}^{k+1}-\mathbf{z}^k)\\
&\qquad\qquad+\theta_1V^2\mathbf{z}^k+\alpha\nabla^k-\alpha\nabla f(\mathbf{y}^k)+\alpha\nabla f(\mathbf{y}^k)-\alpha\nabla f(\mathbf{x}^{k+1})\Big\|^2\\
&\overset{b}{\geq}\frac{1-\nu}{2\kappa L_f\alpha^2}\|\lambda^k-\lambda^*\|^2-\frac{5\mu^2}{2L_f}\left(\frac{1}{\nu}-1\right)\|\mathbf{z}^{k+1}-\mathbf{y}^k\|^2-\frac{5\theta_1^2}{2L_f\alpha^2}\left(\frac{1}{\nu}-1\right)\|V\mathbf{z}^k\|^2\\
&\quad-\frac{5}{2L_f}\left(\frac{1}{\nu}-1\right)\|\nabla^k-\nabla f(\mathbf{y}^k)\|^2-\left(\frac{5\theta_1^2}{2L_f\alpha^2}+\frac{5L_f\theta_1^2}{2}\right)\left(\frac{1}{\nu}-1\right)\|\mathbf{z}^{k+1}-\mathbf{z}^k\|^2,
\end{aligned}\tag{42}$$

where we use (37) in $\overset{a}{=}$, (15), the $L_f$-smoothness of $f(\mathbf{x})$, (36), and $\|V^2\mathbf{z}^k\|^2 \leq \|V\mathbf{z}^k\|^2$ in $\overset{b}{\geq}$. Multiplying both sides of (42) by $\frac{\theta_1}{2}$ and plugging it into (34), using (32), we have

$$\begin{aligned}
&\left(1-\frac{\theta_1}{2}\right)\mathbb{E}_{\mathbb{S}^k}\left[f(\mathbf{x}^{k+1})-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{x}^{k+1}\right\rangle\right]\\
&\leq(1-\theta_1-\theta_2)\left(f(\mathbf{x}^k)-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{x}^k\right\rangle\right)\\
&\quad+\theta_2\left(f(\mathbf{w}^k)-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{w}^k\right\rangle\right)\\
&\quad+\left(\frac{\overline{L}_f}{\tau b}+\frac{5\overline{L}_f\theta_1}{2bL_f}\left(\frac{1}{\nu}-1\right)-\theta_2\right)\left(f(\mathbf{w}^k)-f(\mathbf{y}^k)-\left\langle\nabla f(\mathbf{y}^k),\mathbf{w}^k-\mathbf{y}^k\right\rangle\right)\\
&\quad+\frac{\theta_1^2}{2\alpha}\|\mathbf{z}^k-\mathbf{x}^*\|^2-\left(\frac{\theta_1^2}{2\alpha}+\frac{\mu\theta_1}{2}\right)\mathbb{E}_{\mathbb{S}^k}\left[\|\mathbf{z}^{k+1}-\mathbf{x}^*\|^2\right]
\end{aligned}$$

$$+\left(\frac{1}{2\alpha}-\frac{(1-\nu)\theta_1}{4\kappa L_f\alpha^2}\right)\|\lambda^k-\lambda^*\|^2-\frac{1}{2\alpha}\mathbb{E}_{\mathbb{S}^k}\left[\|\lambda^{k+1}-\lambda^*\|^2\right]$$
$$-\left(\frac{\theta_1^2}{2\alpha}-\frac{5\theta_1^3}{4L_f\alpha^2}\left(\frac{1}{\nu}-1\right)\right)\|V\mathbf{z}^k\|^2-\left(\frac{\mu\theta_1}{2}-\frac{5\mu^2\theta_1}{4L_f}\left(\frac{1}{\nu}-1\right)\right)\mathbb{E}_{\mathbb{S}^k}\left[\|\mathbf{z}^{k+1}-\mathbf{y}^k\|^2\right]$$
$$-\left(\frac{\theta_1^2}{4\alpha}-\frac{\tau\theta_1^2+L_f\theta_1^2}{2}-\left(\frac{5\theta_1^3}{4L_f\alpha^2}+\frac{5L_f\theta_1^3}{4}\right)\left(\frac{1}{\nu}-1\right)\right)\mathbb{E}_{\mathbb{S}^k}\left[\|\mathbf{z}^{k+1}-\mathbf{z}^k\|^2\right].$$

Letting $\theta_1\le\frac{1}{2}$, $\theta_2=\frac{\overline{L}_f}{2L_fb}$, $\tau=3L_f$, $\nu=\frac{127}{128}$, and $\alpha=\frac{1}{10L_f}$ such that $\frac{\overline{L}_f}{\tau b}+\frac{5\overline{L}_f\theta_1}{2L_fb}(\frac{1}{\nu}-1)-\theta_2\le 0$, $\frac{\theta_1^2}{2\alpha}-\frac{5\theta_1^3}{4L_f\alpha^2}(\frac{1}{\nu}-1)\ge 0$, $\frac{\mu\theta_1}{2}-\frac{5\mu^2\theta_1}{4L_f}(\frac{1}{\nu}-1)\ge 0$, and $\frac{\theta_1^2}{4\alpha}-\frac{\tau\theta_1^2+L_f\theta_1^2}{2}-(\frac{5\theta_1^3}{4L_f\alpha^2}+\frac{5L_f\theta_1^3}{4})(\frac{1}{\nu}-1)\ge 0$, we have the conclusion. ■

Now, we are ready to prove Theorem 8.

**Proof** Let $b\ge\max\{\frac{\max\{\sqrt{n\overline{L}_f/\mu},n\}}{\max\{\sqrt{\kappa L_f/\mu},\kappa\}},\frac{\overline{L}_f}{L_f}\}$, then we know $\theta_2=\frac{\overline{L}_f}{2L_fb}\le\frac{1}{2}$ and $b\ge 1$, where we use $\overline{L}_f\ge L_f$. We also have $\max\{\frac{\max\{\sqrt{n\overline{L}_f/\mu},n\}}{\max\{\sqrt{\kappa L_f/\mu},\kappa\}},\frac{\overline{L}_f}{L_f}\}\le\max\{\max\{\sqrt{\frac{n\overline{L}_f}{\mu}},n\}\sqrt{\frac{\mu}{\kappa L_f}},\frac{\overline{L}_f}{L_f}\}=\max\{\sqrt{\frac{n\overline{L}_f}{\kappa L_f}},\sqrt{\frac{n^2\mu}{\kappa L_f}},\frac{\overline{L}_f}{L_f}\}\overset{a}{\le}n$, where $\overset{a}{\le}$ uses $\kappa\ge 1$ and $\mu\le L_f\le\overline{L}_f\le nL_f$ given in (4). This verifies that the setting of $b$ is meaningful.

Multiplying both sides of (31) by $\frac{\theta_2}{\frac{b}{n}-\frac{\theta_1}{20\kappa}}$ and adding it to (41), we have

$$\begin{aligned}
&\left(1-\frac{\theta_1}{2}\right)\mathbb{E}_{\mathbb{S}^k}\left[f(\mathbf{x}^{k+1})-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{x}^{k+1}\right\rangle\right]\\
&\quad+\frac{\theta_2}{\frac{b}{n}-\frac{\theta_1}{20\kappa}}\mathbb{E}_{\mathbf{w}^{k+1}}\left[f(\mathbf{w}^{k+1})-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{w}^{k+1}\right\rangle\right]\\
&\le\left(1-\theta_1-\theta_2+\frac{b}{n}\frac{\theta_2}{\frac{b}{n}-\frac{\theta_1}{20\kappa}}\right)\left(f(\mathbf{x}^k)-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{x}^k\right\rangle\right)\\
&\quad+\left(\theta_2+\left(1-\frac{b}{n}\right)\frac{\theta_2}{\frac{b}{n}-\frac{\theta_1}{20\kappa}}\right)\left(f(\mathbf{w}^k)-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{w}^k\right\rangle\right)\\
&\quad+\frac{\theta_1^2}{2\alpha}\|\mathbf{z}^k-\mathbf{x}^*\|^2-\left(\frac{\theta_1^2}{2\alpha}+\frac{\mu\theta_1}{2}\right)\mathbb{E}_{\mathbb{S}^k}\left[\|\mathbf{z}^{k+1}-\mathbf{x}^*\|^2\right]\\
&\quad+\left(\frac{1}{2\alpha}-\frac{(1-\nu)\theta_1}{4\kappa L_f\alpha^2}\right)\|\lambda^k-\lambda^*\|^2-\frac{1}{2\alpha}\mathbb{E}_{\mathbb{S}^k}\left[\|\lambda^{k+1}-\lambda^*\|^2\right].
\end{aligned}\tag{43}$$

We can easily check that

$$\begin{aligned}
1-\theta_1-\theta_2+\frac{b}{n}\frac{\theta_2}{\frac{b}{n}-\frac{\theta_1}{20\kappa}}&=1-\theta_1-\theta_2+\frac{\theta_2}{1-\frac{n\theta_1}{20b\kappa}}\\
&=1-\theta_1+\frac{\frac{n\theta_1\theta_2}{20b\kappa}}{1-\frac{n\theta_1}{20b\kappa}}\overset{b}{\le}1-\theta_1+\frac{\theta_1}{39}=1-\frac{38}{39}\theta_1,
\end{aligned}$$

and

$$\theta_2+\left(1-\frac{b}{n}\right)\frac{\theta_2}{\frac{b}{n}-\frac{\theta_1}{20\kappa}}=\theta_2+\left(\frac{\theta_1}{20\kappa}-\frac{b}{n}\right)\frac{\theta_2}{\frac{b}{n}-\frac{\theta_1}{20\kappa}}+\left(1-\frac{\theta_1}{20\kappa}\right)\frac{\theta_2}{\frac{b}{n}-\frac{\theta_1}{20\kappa}}$$
$$=\left(1-\frac{\theta_1}{20\kappa}\right)\frac{\theta_2}{\frac{b}{n}-\frac{\theta_1}{20\kappa}},$$

where we check $\overset{b}{\leq}$ in the following two cases. In the first case, if $\kappa\leq\frac{L_f}{\mu}$, we have $\theta_1=\frac{1}{2}\sqrt{\frac{\kappa\mu}{L_f}}$ and $b\geq\max\{\sqrt{\frac{n\overline{L}_f}{\kappa L_f}},\sqrt{\frac{n^2\mu}{\kappa L_f}},\frac{\overline{L}_f}{L_f}\}$. So we have $\frac{n\theta_2}{20b\kappa}\overset{c}{=}\frac{n\overline{L}_f}{40L_fb^2\kappa}\leq\frac{n\overline{L}_f}{40L_f\kappa}\frac{\kappa L_f}{n\overline{L}_f}=\frac{1}{40}$ and $\frac{n\theta_1}{20b\kappa}=\frac{n}{40b\kappa}\sqrt{\frac{\kappa\mu}{L_f}}\leq\frac{n}{40\kappa}\sqrt{\frac{\kappa L_f}{n^2\mu}}\sqrt{\frac{\kappa\mu}{L_f}}=\frac{1}{40}$, where $\overset{c}{=}$ uses the setting of $\theta_2$. So we get $\overset{b}{\leq}$. In the second case, if $\kappa\geq\frac{L_f}{\mu}$, we have $\theta_1=\frac{1}{2}$ and $b\geq\max\{\sqrt{\frac{n\overline{L}_f}{\mu}}\frac{1}{\kappa},\frac{n}{\kappa},\frac{\overline{L}_f}{L_f}\}\geq\frac{n}{\kappa}$. So we have $\frac{n\theta_2}{20b\kappa}\overset{d}{\leq}\frac{n}{40b\kappa}\leq\frac{1}{40}$ and $\frac{n\theta_1}{20b\kappa}=\frac{n}{40b\kappa}\leq\frac{1}{40}$, where $\overset{d}{\leq}$ uses $\theta_2\leq\frac{1}{2}$ derived at the beginning of this proof. So we also get $\overset{b}{\leq}$.

Taking expectation with respect to $\xi^k$ on both sides of (43) and rearranging the terms, we have

$$\begin{aligned}
&\left(1-\frac{\theta_1}{2}\right)\mathbb{E}_{\xi^{k+1}}\left[f(\mathbf{x}^{k+1})-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{x}^{k+1}\right\rangle\right]\\
&\quad+\frac{\theta_2}{\frac{b}{n}-\frac{\theta_1}{20\kappa}}\mathbb{E}_{\xi^{k+1}}\left[f(\mathbf{w}^{k+1})-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{w}^{k+1}\right\rangle\right]\\
&\quad+\left(\frac{\theta_1^2}{2\alpha}+\frac{\mu\theta_1}{2}\right)\mathbb{E}_{\xi^{k+1}}\left[\|\mathbf{z}^{k+1}-\mathbf{x}^*\|^2\right]+\frac{1}{2\alpha}\mathbb{E}_{\xi^{k+1}}\left[\|\lambda^{k+1}-\lambda^*\|^2\right]\\
&\leq\left(1-\frac{38}{39}\theta_1\right)\mathbb{E}_{\xi^k}\left[f(\mathbf{x}^k)-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{x}^k\right\rangle\right]\\
&\quad+\frac{\theta_2}{\frac{b}{n}-\frac{\theta_1}{20\kappa}}\left(1-\frac{\theta_1}{20\kappa}\right)\mathbb{E}_{\xi^k}\left[f(\mathbf{w}^k)-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{w}^k\right\rangle\right]\\
&\quad+\frac{\theta_1^2}{2\alpha}\mathbb{E}_{\xi^k}\left[\|\mathbf{z}^k-\mathbf{x}^*\|^2\right]+\left(\frac{1}{2\alpha}-\frac{(1-\nu)\theta_1}{4\kappa L_f\alpha^2}\right)\mathbb{E}_{\xi^k}\left[\|\lambda^k-\lambda^*\|^2\right]\\
&\overset{f}{\leq}\left\{\left(1-\frac{\theta_1}{2}\right)\mathbb{E}_{\xi^k}\left[f(\mathbf{x}^k)-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{x}^k\right\rangle\right]\right.\\
&\qquad+\frac{\theta_2}{\frac{b}{n}-\frac{\theta_1}{20\kappa}}\mathbb{E}_{\xi^k}\left[f(\mathbf{w}^k)-f(\mathbf{x}^*)+\frac{1}{\alpha}\left\langle\lambda^*,U\mathbf{w}^k\right\rangle\right]\\
&\qquad\left.+\left(\frac{\theta_1^2}{2\alpha}+\frac{\mu\theta_1}{2}\right)\mathbb{E}_{\xi^k}\left[\|\mathbf{z}^k-\mathbf{x}^*\|^2\right]+\frac{1}{2\alpha}\mathbb{E}_{\xi^k}\left[\|\lambda^k-\lambda^*\|^2\right]\right\}\\
&\quad\times\max\left\{\frac{1-\frac{38}{39}\theta_1}{1-\frac{\theta_1}{2}},1-\frac{\theta_1}{20\kappa},\frac{1}{1+\frac{\mu\alpha}{\theta_1}},1-\frac{(1-\nu)\theta_1}{2\kappa L_f\alpha}\right\},
\end{aligned}$$

where $\overset{f}{\leq}$ uses the fact that $f(\mathbf{x})+\frac{1}{\alpha}\langle\lambda^*,U\mathbf{x}\rangle\geq f(\mathbf{x}^*)+\frac{1}{\alpha}\langle\lambda^*,U\mathbf{x}^*\rangle$ for any $\mathbf{x}$, and $\frac{n\theta_1}{20b\kappa}\leq\frac{1}{40}$ in the above analysis such that $\frac{b}{n}\geq\frac{\theta_1}{20\kappa}$.

From the settings of $\theta_1$ and $\alpha$ and $\kappa \geq 1$, we can easily check $\frac{1-\frac{38}{39}\theta_1}{1-\frac{\theta_1}{2}} \leq 1-\frac{18}{39}\theta_1 \leq 1-\frac{18}{39}\frac{\theta_1}{\kappa} = \mathcal{O}(1-\frac{\theta_1}{\kappa})$, $\frac{1}{1+\frac{\mu\alpha}{\theta_1}} \leq 1-\frac{\mu\alpha}{2\theta_1} = \mathcal{O}(1-\frac{\mu}{L_f\theta_1})$ due to $\mu\alpha = \frac{\mu}{10L_f} \leq \frac{1}{10}\sqrt{\frac{\mu}{L_f}} \leq \theta_1$, and $1-\frac{(1-\nu)\theta_1}{2\kappa L_f\alpha} = \mathcal{O}(1-\frac{\theta_1}{\kappa})$. Thus the algorithm needs $\mathcal{O}((\frac{\kappa}{\theta_1}+\frac{L_f\theta_1}{\mu})\log\frac{1}{\epsilon})$ iterations to find $\mathbf{z}^k$ such that $\mathbb{E}_{\xi^k}\left[\|\mathbf{z}^k-\mathbf{x}^*\|^2\right] \leq \epsilon$.

We first consider the communication complexity.

1. If $\kappa \leq \frac{L_f}{\mu}$, we have $\theta_1 = \frac{1}{2}\sqrt{\frac{\kappa\mu}{L_f}}$ and $\mathcal{O}((\frac{\kappa}{\theta_1}+\frac{L_f\theta_1}{\mu})\log\frac{1}{\epsilon}) = \mathcal{O}(\sqrt{\frac{\kappa L_f}{\mu}}\log\frac{1}{\epsilon})$. So the communication complexity is $\mathcal{O}(\sqrt{\frac{\kappa L_f}{\mu}}\log\frac{1}{\epsilon})$.

2. If $\kappa \geq \frac{L_f}{\mu}$, we have $\theta_1 = \frac{1}{2}$ and $\mathcal{O}((\frac{\kappa}{\theta_1}+\frac{L_f\theta_1}{\mu})\log\frac{1}{\epsilon}) = \mathcal{O}((\kappa+\frac{L_f}{\mu})\log\frac{1}{\epsilon}) = \mathcal{O}(\kappa\log\frac{1}{\epsilon})$. So the communication complexity is $\mathcal{O}(\kappa\log\frac{1}{\epsilon})$.

So the algorithm needs the time of $\mathcal{O}(\max\{\sqrt{\frac{\kappa L_f}{\mu}},\kappa\}\log\frac{1}{\epsilon})$ communication rounds to find an $\epsilon$-optimal solution $\mathbf{z}^k$ such that $\mathbb{E}_{\xi^k}\left[\|\mathbf{z}^k-\mathbf{x}^*\|^2\right] \leq \epsilon$.

Next, we consider the stochastic gradient computation complexity.

1. If $\max\{\sqrt{\frac{n\overline{L}_f}{\mu}},n\}\frac{L_f}{\overline{L}_f} \geq \max\{\sqrt{\frac{\kappa L_f}{\mu}},\kappa\}$ such that $b = \frac{\max\{\sqrt{n\overline{L}_f/\mu},n\}}{\max\{\sqrt{\kappa L_f/\mu},\kappa\}}$, the stochastic gradient computation complexity is $\mathcal{O}(b\max\{\sqrt{\frac{\kappa L_f}{\mu}},\kappa\}\log\frac{1}{\epsilon}) = \mathcal{O}(\max\{\sqrt{\frac{n\overline{L}_f}{\mu}},n\}\log\frac{1}{\epsilon})$.

2. If $\max\{\sqrt{\frac{n\overline{L}_f}{\mu}},n\}\frac{L_f}{\overline{L}_f} \leq \max\{\sqrt{\frac{\kappa L_f}{\mu}},\kappa\}$ such that $b = \frac{\overline{L}_f}{L_f}$, the stochastic gradient computation complexity is $\mathcal{O}(b\max\{\sqrt{\frac{\kappa L_f}{\mu}},\kappa\}\log\frac{1}{\epsilon}) = \mathcal{O}(\frac{\overline{L}_f}{L_f}\max\{\sqrt{\frac{\kappa L_f}{\mu}},\kappa\}\log\frac{1}{\epsilon})$.

3. If we choose $b > \max\{\frac{\max\{\sqrt{n\overline{L}_f/\mu},n\}}{\max\{\sqrt{\kappa L_f/\mu},\kappa\}},\frac{\overline{L}_f}{L_f}\}$, the stochastic gradient computation complexity is higher than the above ones. But the communication complexity remains unchanged. This verifies Remark 9(2).

At last, we discuss the condition $\max\{\sqrt{\frac{n\overline{L}_f}{\mu}},n\}\frac{L_f}{\overline{L}_f} \geq \max\{\sqrt{\frac{\kappa L_f}{\mu}},\kappa\}$. We know that $\phi(\kappa) \equiv \max\{\sqrt{\frac{\kappa L_f}{\mu}},\kappa\}$ is a piece-wise increasing function with respect to $\kappa$ such that

$$\phi(\kappa) = \begin{cases} \sqrt{\frac{\kappa L_f}{\mu}}, & \text{if } 0 \leq \kappa \leq \frac{L_f}{\mu}, \\ \kappa, & \text{if } \kappa \geq \frac{L_f}{\mu}, \end{cases} \quad \text{and } \phi(\kappa) \begin{cases} \leq \frac{L_f}{\mu}, & \text{if } \phi(\kappa) = \sqrt{\frac{\kappa L_f}{\mu}}, \\ \geq \frac{L_f}{\mu}, & \text{if } \phi(\kappa) = \kappa. \end{cases}$$

1. If $n \geq \frac{\overline{L}_f}{\mu}$, we have $\max\{\sqrt{\frac{n\overline{L}_f}{\mu}},n\}\frac{L_f}{\overline{L}_f} = \frac{nL_f}{\overline{L}_f} \geq \frac{L_f}{\mu}$. So the condition $\max\{\sqrt{\frac{n\overline{L}_f}{\mu}},n\}\frac{L_f}{\overline{L}_f} \geq \max\{\sqrt{\frac{\kappa L_f}{\mu}},\kappa\}$ is equivalent to $\frac{nL_f}{\overline{L}_f} \geq \kappa$.

2. If $n \leq \frac{\overline{L}_f}{\mu}$, we have $\max\{\sqrt{\frac{n\overline{L}_f}{\mu}},n\}\frac{L_f}{\overline{L}_f} = \sqrt{\frac{nL_f^2}{\mu\overline{L}_f}} \leq \frac{L_f}{\mu}$. So $\max\{\sqrt{\frac{n\overline{L}_f}{\mu}},n\}\frac{L_f}{\overline{L}_f} \geq \max\{\sqrt{\frac{\kappa L_f}{\mu}},\kappa\}$ is equivalent to $\sqrt{\frac{nL_f^2}{\mu\overline{L}_f}} \geq \sqrt{\frac{\kappa L_f}{\mu}}$, that is, $\frac{nL_f}{\overline{L}_f} \geq \kappa$. ■

## 5. Numerical Experiments

Consider the following decentralized regularized logistic regression problem:

$$\min_{x\in\mathbb{R}^p}\sum_{i=1}^m f_{(i)}(x),\quad \text{where}\quad f_{(i)}(x)=\frac{\mu}{2}\|x\|^2+\frac{1}{n}\sum_{j=1}^n\log\left(1+\exp(-y_{(i),j}A_{(i),j}^Tx)\right),$$

where the pairs $(A_{(i),j},y_{(i),j})\in\mathbb{R}^p\times\{1,-1\}$ are taken from the RCV1 dataset[8] with $p=47236$, $m=49$, and $n=500$. Denote $A_{(i)}=[A_{(i),1},A_{(i),2},\cdots,A_{(i),n}]\in\mathbb{R}^{p\times n}$ as the data matrix on the $i$th node. For this special problem and dataset, we observe $L_f=\max_i\frac{\|A_{(i)}\|_2^2}{4n}+\mu\approx 0.016+\mu$ and $\overline{L}_f=\max_i\frac{\|A_{(i)}\|_F^2}{4n}+\mu=\frac{1}{4}+\mu$, respectively. We test the performance of the proposed algorithms on different ratios between $\kappa_s$ and $n$. Specifically, we test on $\mu=5\times10^{-5}$, $\mu=5\times10^{-6}$, and $\mu=5\times10^{-7}$, which correspond to $\kappa_s=\frac{\overline{L}_f}{\mu}\approx5\times10^3$, $\kappa_s\approx5\times10^4$, and $\kappa_s\approx5\times10^5$, respectively. Note that $n=500$. We also observe $\frac{n\kappa_b}{\kappa_s}\approx31.9$.

We test the performance on two kinds of networks: the Erdős−Rényi random graph and the two-dimensional grid graph, where each pair of nodes has a connection with the ratio of 0.2 for the first graph, and $m$ nodes are placed in the $\sqrt{m}\times\sqrt{m}$ grid and each node is connected with its neighbors around it for the second graph. Theoretically, $\kappa_c=\mathcal{O}(1)$ for the first graph, and $\kappa_c=\mathcal{O}(m\log m)$ for the second graph. Practically, we observe $\kappa_c=4.62$ and $\kappa_c=19.9$ for the two graphs, respectively. We set the weight matrix as $W=\frac{M-\lambda_{\min}I}{1-\lambda_{\min}}$ for both graphs, where $M$ is the Metropolis weight matrix (Boyd et al., 2004) with $M_{ij}=\begin{cases}\frac{1}{\max\{d(i),d(j)\}} & \text{if } (i,j)\in\mathcal{E},\\ 1-\sum_{l\in\mathcal{N}_{(i)}}M_{il} & \text{if } i=j,\\ 0 & \text{if } (i,j)\notin\mathcal{E} \text{ and } i\neq j.\end{cases}$, $d(i)$ is the degree of node $i$, and $\lambda_{\min}<0$ is the smallest negative eigenvalue of $M$.

We compare the proposed VR-EXTRA, Acc-VR-EXTRA, Acc-VR-EXTRA-CA, VR-DIGing, Acc-VR-DIGing, and Acc-VR-DIGing-CA with EXTRA (Shi et al., 2015), DIGing (Nedić et al., 2017), GT-SVRG (Xin et al., 2020a), APAPC (Kovalev et al., 2020b), and accelerated DVR (Acc-DVR) (Hendrikx et al., 2020). For the case of $\mu=5\times10^{-6}$, we choose the best step-sizes $\alpha=\frac{1}{L_f}$ for Acc-VR-EXTRA, Acc-VR-EXTRA-CA, Acc-VR-DIGing, and Acc-VR-DIGing-CA, $\alpha=\frac{3}{L_f}$ for VR-EXTRA, VR-DIGing, and GT-SVRG, and $\alpha=\frac{7}{L_f}$ for EXTRA and DIGing. The other parameters are chosen according to the theories. We set the parameters of APAPC according to Theorem 2 in (Kovalev et al., 2020b). For Acc-DVR, we follow the suggestions in the experimental section in (Hendrikx et al., 2020) to set the number of inner iterations to $\frac{n}{1-p_{comm}}$ (one pass over the local dataset), and the batch Lipschitz constant as $L_f=0.01\overline{L}_f$, which leads to better performance of Acc-DVR than the theoretical setting of $L_f=\max_i\frac{\|A_{(i)}\|_2^2}{4n}+\mu$. We follow Algorithm 2 and Theorem 5 in (Hendrikx et al., 2020) to choose other parameters of Acc-DVR, that is, $\alpha=\frac{2}{L_f\kappa_c}$, $\eta=\min\{p_{comm}(\beta+\mu),\frac{p_{ij}}{\alpha(1+1/(4n\mu))}\}$, $\beta=\frac{\overline{L}_f}{n}-\mu$, $p_{comm}=(1+\frac{n+\kappa_s^\beta}{\kappa_b^\beta\kappa_c})^{-1}$, $p_{ij}=\frac{1-p_{comm}}{n}$, $\kappa_s^\beta=\frac{\overline{L}_f}{\beta+\mu}$, and $\kappa_b^\beta=\frac{L_f}{\beta+\mu}$[9]. For the case of $\mu=5\times10^{-7}$, we choose the same parameters

8. https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/binary.html
9. We do not find the parameters in APAPC and Acc-DVR in the role as step-sizes in the form of $\mathcal{O}(\frac{1}{L_f})$, thus we set the parameters according to their theories directly.

as above. For the case of $\mu = 5 \times 10^{-5}$, we set the step-sizes $\alpha = \frac{2}{L_f}$ for VR-EXTRA, VR-DIGing, and GT-SVRG, and $\alpha = \frac{3}{L_f}$ for EXTRA and DIGing, and other parameters are chosen the same as above. Specially, since $\frac{n\kappa_b}{\kappa_s} < 2\kappa_c$ and $\frac{n\kappa_b}{\kappa_s} < \kappa_c^2$ for the grid graph, we set the mini-batch size $b = \frac{\overline{L}_f}{L_f}$ for Acc-VR-EXTRA and Acc-VR-DIGing according to Remark 9(1).

Figures 1 and 2 plot the results on the Erdős−Rényi random graph and grid graph, respectively, where $f^*$ is approximated by the minimum value of the objective function over all iterations of all the compared algorithms. We have the following observations:

1. VR-EXTRA and VR-DIGing need less gradient computations than EXTRA and DIGing to reach the same precision for all cases of $\mu$. This verifies the efficiency of variance reduction in decentralized optimization to reduce the computation cost.

2. When considering the communication cost, VR-EXTRA and VR-DIGing perform worse than EXTRA and DIGing in practice, although they have the same communication complexities theoretically. This is reasonable since EXTRA and DIGing go through all the data at each communication round, while VR-EXTRA and VR-DIGing only use a mini-batch. We observe the mini-batch size of $b = \frac{\overline{L}_f}{L_f} \approx 16$ in our experiment.

3. Acc-VR-EXTRA and Acc-VR-DIGing perform better than VR-EXTRA and VR-DIGing on both computations and communications when $\kappa_s$ is much larger than $n$. For example, $\kappa_s = 1000n$ in the top plots and $\kappa_s = 100n$ in the middle plots. Otherwise, as seen in the bottom plots with $\kappa_s = 10n$, Acc-VR-EXTRA and Acc-VR-DIGing perform quite similarly to VR-EXTRA and VR-DIGing, especially on the computations. This verifies that acceleration only takes effect to reduce the computation cost when $\kappa_s \gg n$.

4. When considering the communication cost, Acc-VR-EXTRA performs similarly to the optimal full batch method of APAPC. This observation matches the theory that the two methods have the same communication complexity.

5. Acc-VR-EXTRA-CA and Acc-VR-DIGing-CA do not perform well in practice, although they are theoretically optimal. Thus Acc-VR-EXTRA-CA and Acc-VR-DIGing-CA are more interesting in theory, but they are not suggested in practice.

## 6. Conclusion and Future Research

This paper extends the widely used EXTRA and DIGing methods with variance reduction, and four VR-based stochastic decentralized algorithms are proposed. The proposed VR-EXTRA has the $\mathcal{O}((\kappa_s + n)\log\frac{1}{\epsilon})$ stochastic gradient computation complexity and the $\mathcal{O}((\kappa_b + \kappa_c)\log\frac{1}{\epsilon})$ communication complexity. The proposed VR-DIGing has a little worse communication complexity of $\mathcal{O}((\kappa_b + \kappa_c^2)\log\frac{1}{\epsilon})$. Our stochastic gradient computation complexities keep the same as the single-machine VR methods such as SVRG, and our communication complexities are the same as those of EXTRA and DIGing, respectively. The proposed accelerated VR-EXTRA and VR-DIGing achieve both the optimal $\mathcal{O}((\sqrt{n\kappa_s} + n)\log\frac{1}{\epsilon})$ stochastic gradient computation complexity and $\mathcal{O}(\sqrt{\kappa_b\kappa_c}\log\frac{1}{\epsilon})$ communication complexity. They are also the same as the ones of single-machine accelerated VR methods

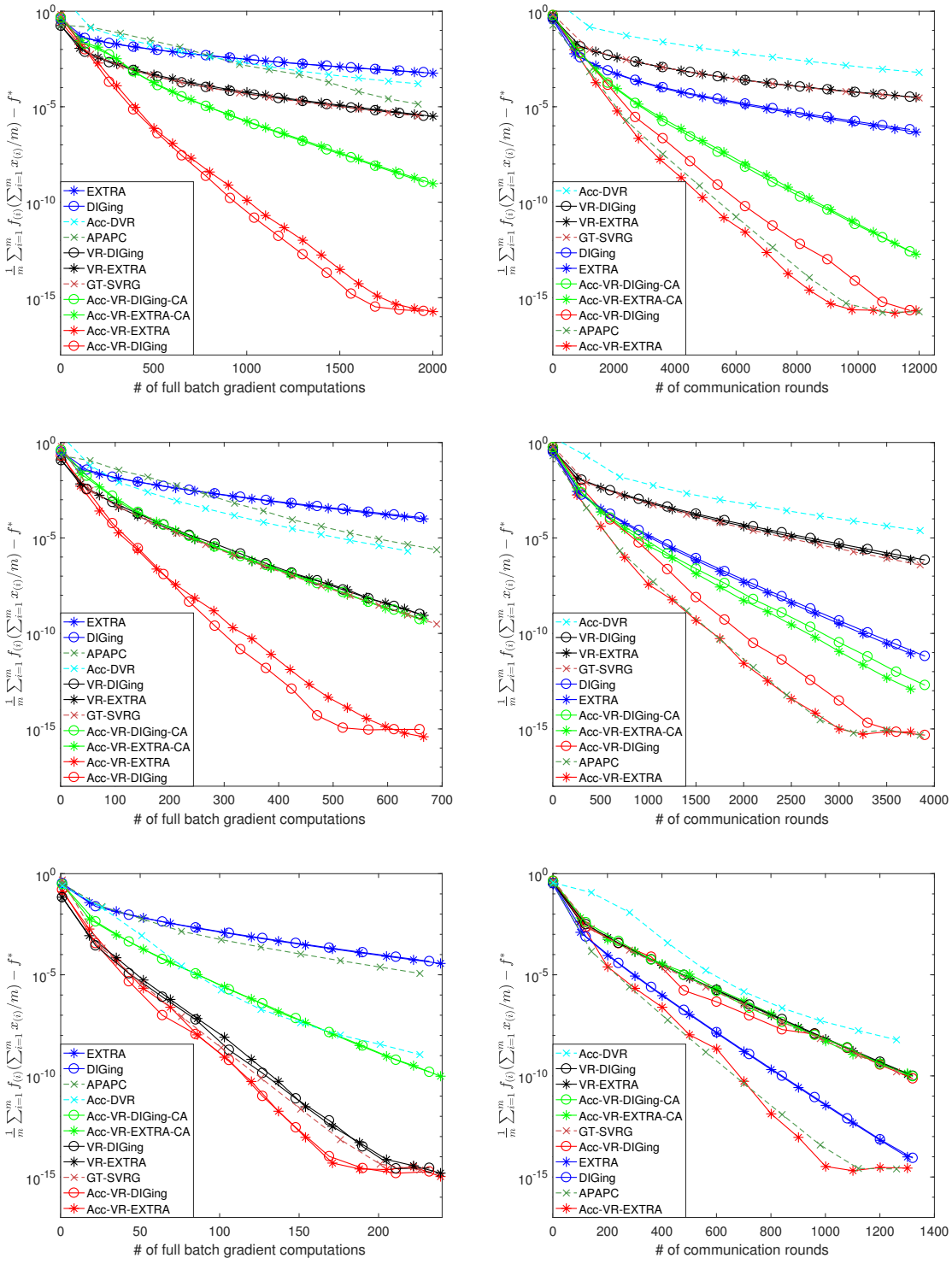


Figure 1: Comparisons on Erdős−Rényi random graph with $\mu = 5\times10^{-7}$ (top), $\mu = 5\times10^{-6}$ (middle), and $\mu = 5\times10^{-5}$ (bottom).

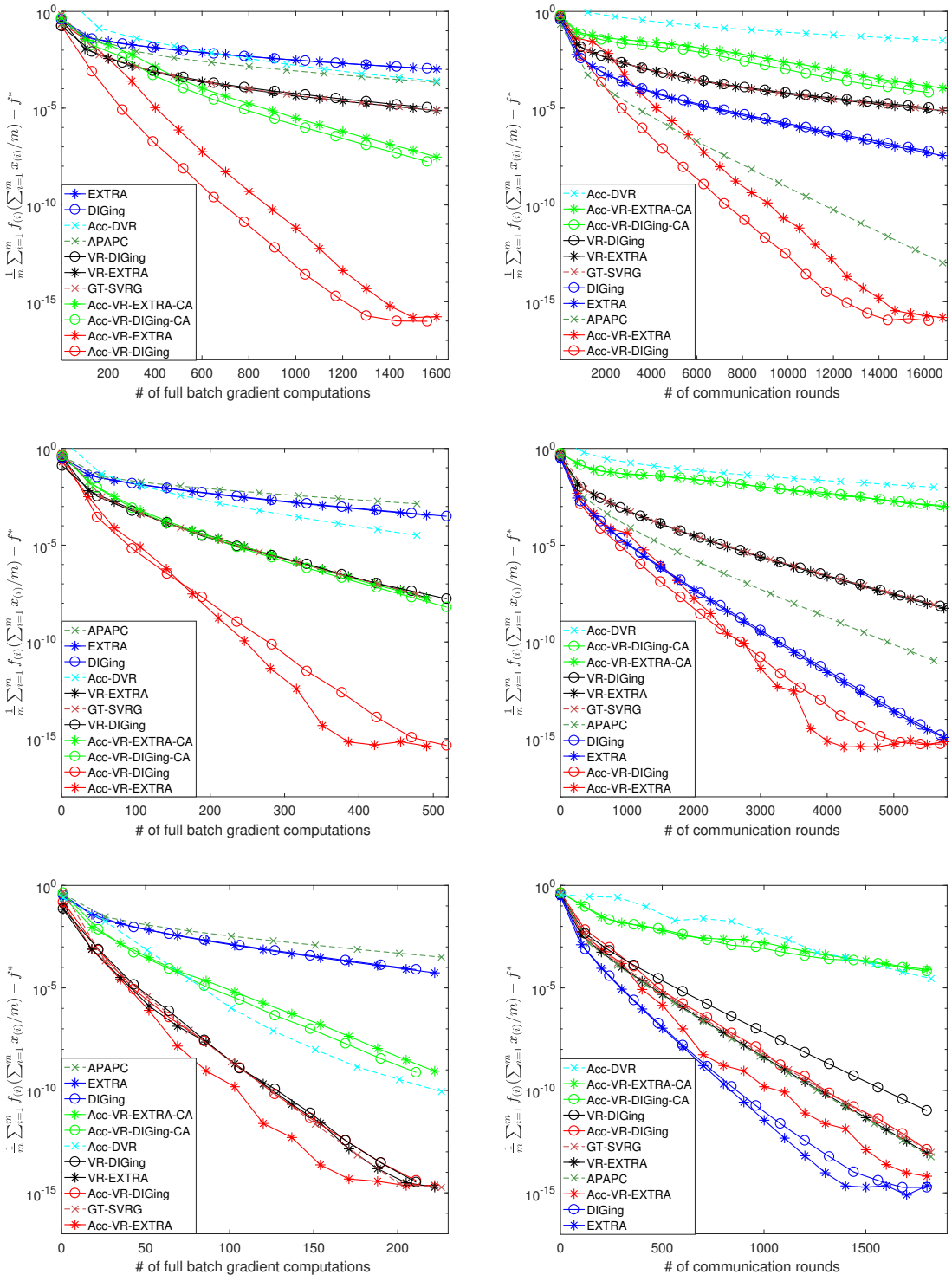


Figure 2: Comparisons on grid graph with $\mu = 5 \times 10^{-7}$ (top), $\mu = 5 \times 10^{-6}$ (middle), and $\mu = 5 \times 10^{-5}$ (bottom).

such as Katyusha, and the accelerated full batch decentralized methods such as MSDA, respectively.

DIGing, called gradient tracking in other literatures, is a fundamental decentralized algorithm in the distributed optimization community. However, its state-of-the-art complexity is $\mathcal{O}((\kappa_b+\kappa_c^2)\log\frac{1}{\epsilon})$, which is worse than the $\mathcal{O}((\kappa_b+\kappa_c)\log\frac{1}{\epsilon})$ one of EXTRA. An open problem is that can we improve it from $\mathcal{O}((\kappa_b+\kappa_c^2)\log\frac{1}{\epsilon})$ to $\mathcal{O}((\kappa_b+\kappa_c)\log\frac{1}{\epsilon})$? Recently, Koloskova et al. (2021) gave some potential directions, where the complexity is improved from $\mathcal{O}(\kappa_b^2\kappa_c^2\log\frac{1}{\epsilon})$ (Qu and Li, 2018) to $\mathcal{O}(\kappa_b\kappa_c\log\frac{1}{\epsilon})$. That is, the dependence on $\kappa_c^2$ has been reduced to $\kappa_c$. Currently, it is unclear whether the complexity can be further improved to $\mathcal{O}((\kappa_b+\kappa_c)\log\frac{1}{\epsilon})$.

In our Algorithms 1 and 2, we set the parameters dependent on $\kappa_c$, $L_f$, and $\overline{L}_f$, where $\kappa_c$ needs the global knowledge of the network, while $L_f$ and $\overline{L}_f$ needs to know the parameters of the other nodes. It is important to design practical algorithms only dependent on the local parameters, such as the Lipschitz constant $L_{(i)}$ and strong-convexity constant $\mu_{(i)}$, while still keep the optimal complexities. Another open problem is whether the reformulation (6) can be extended to directed graphs, where $U$ and $V$ cannot simply take the ones in this paper. Other interesting extensions include compression, asynchrony, and so on.

## Acknowledgments

Huan Li is sponsored by NSF China (grant no. 62006116) and Jiangsu Province Basic Research Program (grant no. BK20200439). Zhouchen Lin is supported by the major key project of PCL (grant no. PCL2021A12), the NSF China (grant no. 61731018), and Project 2020BD006 supported by PKU-Baidu Fund. Yongchun Fang is supported by NSF China (grant no. 61873132).

## References

Sulaiman A. Alghunaim, Ernest K. Ryu, Kun Yuan, and Ali H.Sayed. Decentralized proximal gradient algorithms with linear covnergence rates. *IEEE Transactions on Automatic Control*, 66(6):2787–2794, 2021.

Zeyuan Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. *Journal of Machine Learning Research*, 18(221):1–51, 2018.

Mario Arioli and Jennifer Scott. Chebyshev acceleration of iterative refinement. *Numerical Algorithms*, 66(3):591–608, 2014.

Winfried Auzinger and Jens Markus Melenk. Iterative solution of large linear systems. *Lecture notes, TU Wien*, 2017.

Necdet Serhat Aybat, Zi Wang, Tianyi Lin, and Shiqian Ma. Distributed linearized alternating direction method of multipliers for composite convex consensus optimization. *IEEE Transactions on Automatic Control*, 63(1):5–20, 2018.

Dimitri P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Athena Scientific, Belmont, Massachusetts, 1982.

Dimitri P. Bertsekas. Distributed asynchromous computation of fixed points. *Mathmatical Programming*, 27:107–120, 1983.

Stephen Boyd, Persi Diaconis, and Lin Xiao. Fastest mixing markov chain on a graph. *SIAM Review*, 46(4):667–689, 2004.

Jianshu Chen and Ali H. Sayed. Diffusion adaptation strategies for distributed optimization and learning over networks. *IEEE Transactions on Signal Processing*, 60(8):4289–4305, 2012.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1646–1654, 2014.

Darina Dvinskikh and Alexander Gasnikov. Decentralized and parallelized primal and dual accelerated methods for stochastic convex programming problems. *Journal of Inverse and Ill-posed Problems*, 29(3):385–405, 2021.

Alireza Fallah, Mert Gürbüzbalaban, Asuman Ozdaglar, Umut Simsekli, and Lingjiong Zhu. Robust distributed accelerated stochastic gradient methods for multi-agent networks. *preprint arXiv:1910.08701*, 2019.

Olivier Fercoq and Peter Richtárik. Accelerated, parallel, and proximal coordinate descent. *SIAM Journal on Optimization*, 25(4):1997–2023, 2015.

Eduard Gorbunov, Darina Dvinskikh, and Alexander Gasnikov. Optimal decentralized distributed algorithms for stochastic convex optimization. *preprint arXiv:1911.07363*, 2019.

Eduard Gorbunov, Alexander Rogozin, Aleksandr Beznosikov, Darina Dvinskikh, and Alexander Gasnikov. Recent theoretical advances in decentralized distributed convex optimization. In *High-Dimensional Optimization and Probability*, pages 253–325. Springer, 2022.

Hadrien Hendrikx, Francis Bach, and Laurent Massoulié. Dual-free stochastic decentralized optimization with variance reduction. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 19455–19466, 2020.

Hadrien Hendrikx, Francis Bach, and Laurent Massoulié. An optimal algorithm for decentralized finite-sum optimization. *SIAM Journal on Optimization*, 31(4):2753–2783, 2021.

Mingyi Hong, Davood Hajinezhad, and Ming-Min Zhao. Prox-PDA: The proximal primal-dual algorithm for fast distributed nonconvex optimization and learning over networks. In *International Conference on Machine Learning (ICML)*, pages 1529–1538, 2017.

Franck Iutzeler, Pascal Bianchi, Philippe Ciblat, and Walid Hachem. Explicit convergence rate of a distributed alternating direction method of multipliers. *IEEE Transactions on Automatic Control*, 61(4):892–904, 2016.

Dusan Jakovetić. A unification and generatliztion of exact distributed first order methods. *IEEE Transactions on Signal and Information Processing over Networks*, 5(1):31–46, 2019.

Dusan Jakovetić, Joao Xavier, and José M. F. Moura. Fast distributed gradient methods. *IEEE Transactions on Automatic Control*, 59(5):1131–1146, 2014.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems (NIPS)*, pages 315–323, 2013.

Anastasia Koloskova, Tao Lin, and Sebastian U. Stich. An improved analysis of gradient tracking for decentralized machine learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 11422–11435, 2021.

Dmitry Kovalev, Samuel Horváth, and Peter Richtárik. Don't jump through hoops and remove those loops: SVRG and Katyusha are better without the outer loop. In *International Conference on Algorithmic Learning Theory (ALT)*, pages 451–467, 2020a.

Dmitry Kovalev, Adil Salim, and Peter Richtárik. Optimal and practical algorithms for smooth and strongly convex decentralized optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 18342–18352, 2020b.

Guanghui Lan and Yi Zhou. An optimal randomized incremental gradient method. *Mathematical Programming*, 171:167–215, 2018.

Guanghui Lan, Soomin Lee, and Yi Zhou. Communication-efficient algorithms for decentralized and stochastic optimization. *Mathematical Programming*, 180:237–284, 2020.

Boyue Li, Shicong Cen, Yuxin Chen, and Yuejie Chi. Communication-efficient distributed optimization in networks with gradient tracking and variance reduction. *Journal of Machine Learning Research*, 21(180):1–51, 2020a.

Huan Li and Zhouchen Lin. Revisiting EXTRA for smooth distributed optimization. *SIAM Journal on Optimization*, 30(3):1795–1821, 2020.

Huan Li, Cong Fang, Wotao Yin, and Zhouchen Lin. Decentralized accelerated gradient methods with increasing penalty parameters. *IEEE transactions on Signal Processing*, 68: 4855–4870, 2020b.

Zhi Li, Wei Shi, and Ming Yan. A decentralized proximal-gradient method with network independent step-sizes and separated convergence rates. *IEEE Transactions on Signal Processing*, 67(17):4494–4506, 2019.

Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. Catalyst acceleration for first-order convex optimization: from theory to practice. *Journal of Machine Learning Research*, 18 (212):1–54, 2018.

Qihang Lin, Zhaosong Lu, and Lin Xiao. An accelerated randomized proximal coordinate gradient method and its application to regularized empirical risk minimization. *SIAM Journal on Optimization*, 25(4):2244–2273, 2015.

Ali Makhdoumi and Asuman Ozdaglar. Convergence rate of distributed ADMM over networks. *IEEE Transactions on Automatic Control*, 62(10):5082–5095, 2017.

Aryan Mokhtari and Alejandro Ribeiro. DSA: Decenrtalized double stochastic averaging gradient algorithm. *Journal of Machine Learning Research*, 17(61):1–35, 2016.

Agelia Nedić and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

Angelia Nedić. Asynchronous broadcast-based convex optimization over a network. *IEEE Transactions on Automatic Control*, 56(6):1337–1351, 2011.

Angelia Nedić, Alex Olshevsky, and Wei Shi. Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM Journal on Optimization*, 27(4):2597–2633, 2017.

Angelia Nedić, Alex Olshevsky, and Michael G. Rabbat. Network topology and communication-computation tradeoffs in decentralized optimization. *Proceedings of the IEEE*, 106(5):953–976, 2018.

Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic, Boston, 2004.

Shi Pu and Angelia Nedić. Distributed stochastic gradient tracking methods. *Mathematical Programming*, 187:409–457, 2021.

Guannan Qu and Na Li. Harnessing smoothness to accelerate distributed optimization. *IEEE Transactions on Control of Network Systems*, 5(3):1245–1260, 2018.

Guannan Qu and Na Li. Accelerated distributed Nesterov gradient descent. *IEEE Transactions on Automatic Control*, 65(6):2566–2581, 2020.

S. Sundhar Ram, Angelia Nedić, and Venugopal V. Veeravalli. Distributed stochastic subgradient projection algorithms for convex optimization. *Journal of Optimization Theory and Applications*, 147:516–545, 2010.

Kevin Scaman, Francis Bach, Sebastien Bubeck, Yin Tat Lee, and Laurent Massoulié. Optimal algorithms for smooth and strongly convex distributed optimization in networks. In *International Conference on Machine Learning (ICML)*, pages 3027–3036, 2017.

Kevin Scaman, Francis Bach, Sebastien Bubeck, Yin Tat Lee, and Laurent Massoulié. Optimal algorithms for non-smooth distributed optimization in networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2740–2749, 2018.

Kevin Scaman, Francis Bach, Sebastien Bubeck, Yin Tat Lee, and Laurent Massoulié. Optimal convergence rates for convex distributed optimization in networks. *Journal of Machine Learning Research*, 20(159):1–31, 2019.

Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162:83–112, 2017.

Wei Shi, Qing Ling, Gang Wu, and Wotao Yin. EXTRA: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015.

Håkan Terelius, Ufuk Topcu, and Richard M. Murray. Decentralized multi-agent optimization via dual decomposition. *IFAC proceedings volumes*, 44(1):11245–11251, 2011.

John N. Tsitsiklis, Dimitri P. Bertsekas, and Michael Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transaction on Automatic Control*, 31(9):803–812, 1986.

Cesar A. Uribe, Soomin Lee, Alexander Gasnikov, and Angelia Nedić. A dual approach for optimal algorithms in distributed optimization over networks. In *Information Theory and Applications Workshop (ITA)*, pages 1–37, 2020.

Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

Ran Xin, Usman A. Khan, and Soummya Kar. A linear algorithm for optimization over directed graphs with geometric convergence. *IEEE Control Systems Letters*, 2(3):315–320, 2018.

Ran Xin, Soummya Kar, and Usman A. Khan. Decentralized stochastic optimization and machine learning: A unified variance-reduction framework for robust performance and fast convergence. *IEEE Signal Processing Magazine*, 37(3):102–113, 2020a.

Ran Xin, Usman A. Khan, and Soummya Kar. Variance-reduced decentralized stochastic optimization with accelerated convergence. *IEEE Transactions on Signal Processing*, 68: 6255–6271, 2020b.

Jinming Xu, Shanying Zhu, Yeng Chai Soh, and Lihua Xie. Augmented distributed gradient methods for multi-agent optimization under uncoordinated constant stepsizes. In *IEEE Conference on Decision and Control (CDC)*, pages 2055–2060, 2015.

Kun Yuan, Qing Ling, and Wotao Yin. On the convergence of decentralized gradient descent. *SIAM Journal on Optimization*, 26(3):1835–1854, 2016.

Kaiwen Zhou, Fanhua Shang, and James Cheng. A simple stochastic variance reduced algorithm with fast convergence rates. In *International Conference on Machine Learning (ICML)*, pages 5975–5984, 2019.