Mitigating Data Heterogeneity Effect in Federated Learning with Client Reshuffling

Anonymous Authors[1]

Abstract

Federated learning is a distributed paradigm that facilitates collaborative model training across decentralized data sources. As the number of participating devices increases, it becomes impractical to require every client to participate in the entire training process. Most algorithms address this challenge by using partial attendance with sampling with replacement, but this approach results in low attendance efficiency at large scales. Current methods adopt client reshuffling to improve partial attendance efficiency in practice. However, the convergence guarantees of these methods typically depend on the level of data heterogeneity among clients. The greater the heterogeneity, the stricter the conditions required for convergence. To overcome these limitations, we propose a novel federated algorithm that incorporates client shuffling to enhance client sampling efficiency and reduce variance caused by client sampling. Unlike existing methods, our approach achieves convergence regardless of the degree of data heterogeneity among clients. Experimentally, our method also outperforms other federated learning techniques that use client reshuffling in the presence of data heterogeneity.

1. Introduction

Federated learning (FL) problems and algorithms have gained significant attention in machine learning research in recent years. Modern problems often require training deep neural networks with billions of parameters on large datasets (Brown et al., 2020). Distributed algorithms are employed to accelerate large-scale model training, i.e., applying federated learning algorithms. Another key motivation for federated learning is that data can be naturally distributed across multiple devices and be private. Due to these advantages, federated learning is now deployed in various applications (Hard et al., 2018). For instance, it is a promising approach in smart healthcare (Rieke et al., 2020; Sheller et al., 2020), where privacy is of paramount importance.

We assume a large number of small clients and a central server. This scenario, known as cross-device federated learning, leverages millions of edge devices, such as smartphones, to train federated learning models (Bonawitz et al., 2017; Bhowmick et al., 2018; Yan et al., 2020). As the number of personal and home smart devices capable of generating, capturing, and processing data continues to grow, this scenario is becoming increasingly essential. Compared to standard federated learning, this setting introduces unique challenges. For instance, methods designed for cross-device federated learning must address issues such as participant selection (Ribero & Vikalo, 2020; Yang et al., 2021; Chen et al., 2022), system heterogeneity (Diao et al., 2021; Horváth et al., 2022), and edge computing (Xia et al., 2021).

In this paper, we focus on the challenge of *participant selection*, i.e., clients only intermittently participate in the collaborative training process (Bonawitz et al., 2017). As previously mentioned, our scenario involves millions of participating devices. At this scale, client sampling for partial participation is essential, as it is impractical for millions of devices to engage in every communication round. This impracticality arises due to the high computational and communication costs, which could also lead to network congestion. Furthermore, in real-world settings, devices are only available at certain times. For instance, if the devices are personal smartphones, users may prefer to participate in FL only when their devices are inactive, charging, and connected to high-speed WiFi to avoid a negative user experience. Additionally, each device may participate only once or a few times throughout the entire learning process. Therefore, it is crucial to explore methods that effectively support clients' partial participation.

A classical approach to limiting the number of clients participating in each round is to employ uniform sampling (McMahan et al., 2017b). In each round, the global server samples C clients uniformly at random to perform local

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

training. A more general method is to select C clients using importance sampling methods (Fraboni et al., 2021a;b; Chen et al., 2022) instead of uniform sampling. However, such sampling methods result in low client utilization efficiency. To address this issue, Malinovsky et al. (2023a) introduce a client reshuffling method, where each client participates once per meta-epoch. This approach improves the efficiency of partial participation in practical scenarios and meets real-world requirements, such as accommodating part-time device availability and ensuring that each device participates in training at least once or only a few times.

However, the theoretical analysis of client reshuffling in federated learning has certain limitations. Malinovsky et al. (2023a) discuss the theoretical convergence rate only in convex settings. Demidovich et al. (2024) extend the analysis to nonconvex settings. Nevertheless, previous methods require a small step size to mitigate the effects of client heterogeneity. When data heterogeneity among clients becomes severe, the convergence of these methods may slow down. Such a noticeable gap in the literature leads us to the question:

Is it possible to design a federated learning method with client reshuffling with provable convergence guarantee independent of the influences of client heterogeneity without additional restrictive assumptions?

In this paper, we give a positive answer to the question.

1.1. Contributions

New method. We propose a new method called **FedCDR** (**Fed**erated **C**lient Reshuffling **D**ouglas-**R**achford Method) (*Algorithm 1*) for nonconvex federated learning problems. This method integrates the client reshuffling technique into the Douglas-Rachford (DR) method in nonconvex federated learning. **FedCDR** is designed for cross-device settings and aims to mitigate data heterogeneity among clients while allowing efficient partial participation during training.

Convergence analysis. We provide a convergence analysis in two cases: *Shuffle Once* and *Random Reshuffling*.

- In both cases, our method achieves the best-known $O(\epsilon^{-2})$ communication complexity for finding a stationary point under standard assumptions.
- Our analysis shows that the upper bound of the proposed method is independent of data heterogeneity across different clients. This addresses a key limitation in the convergence analysis of previous federated learning methods that involve client shuffling. Specifically, in cross-device settings where heterogeneity is high, prior methods suffer from slow convergence (see Remark 5.2 for a detailed discussion). In contrast, the convergence rate of our method remains unaffected even in the presence of significant heterogeneity.

Technical Novelty A key challenge in analyzing the convergence rate of our method is that, unlike sampling with replacement, reshuffling-based sampling is not identically distributed across different communication rounds within a single meta-epoch. As a result, we cannot explicitly compute the expectation expressions. To address this challenge, we first estimate the error between the aggregated updates and the local updates. Then, we bound the descent of the potential function between successive rounds using this error. Due to the dependence between rounds, our analysis heavily relies on conditional expectation, which is fundamentally different from the analysis of methods that use sampling with replacement.

Experimental Validation The theoretical analysis of our method is supported by experimental evaluations that validate our findings. We conduct numerical experiments on both synthetic and real-world datasets. The results demonstrate that our method outperforms the baseline and effectively mitigates data heterogeneity issues in client reshuffling compared to the baseline.

2. Related Work

Federated learning and data heterogeneity. Federated Averaging (FedAvg) is one of the earliest methods used in FL (McMahan et al., 2017a). In FedAvg, clients perform stochastic gradient descent (SGD) updates for a number of epochs and then send the updated models to the server for aggregation. The practical performance of FedAvg has been demonstrated in many early studies (Konečný et al., 2016; McMahan et al., 2017b). However, analyzing the convergence of FedAvg was challenging in its early stages due to data heterogeneity. For the i.i.d. setting, Stich (2019) was one of the earliest works to analyze FedAvg in convex settings. Yu et al. (2019) extended this analysis to nonconvex settings. In the non-i.i.d. setting, Khaled et al. (2019) studied the convergence properties of local gradient descent (GD) in FL. Additionally, Li et al. (2020b) analyzed the convergence of SGD in convex FL settings. Moreover, Pathak & Wainwright (2020) and Zhang et al. (2020) demonstrated that FedAvg may fail to converge even in convex settings due to data heterogeneity.

Several studies have addressed the data heterogeneity problem in FL. FedProx (Li et al., 2020a) introduces a proximal term in local updates to improve the stability of the training process, demonstrating superior performance to FedAvg in heterogeneous settings. SCAFFOLD (Karimireddy et al., 2020) is another approach that mitigates data heterogeneity by using a control variate to correct client drift in local updates. Additionally, FedDR (Tran-Dinh et al., 2021) employs a nonconvex Douglas-Rachford splitting method to

address the client drift problem in FL. However, previous methods select clients via uniform sampling in each round, which results in low client utilization efficiency and poses practical challenges, particularly in cross-device FL scenarios.

Unlike these prior works, our method introduces client reshuffling to improve client utilization efficiency and better meet practical requirements.

Federated learning with reshuffling. Although Stochastic Gradient Descent (SGD) has been extensively analyzed in theoretical studies (Rakhlin et al., 2012; Nguyen et al., 2019; Gower et al., 2019), many widely used machine learning frameworks and deep learning algorithms rely on reshuffling methods (sampling without replacement). This approach is simple and efficient for deep neural network training (Recht & Ré, 2013; Bengio, 2012). Reshuffling ensures that each data point is used exactly once per epoch, thereby leveraging the finite-sum optimization structure.

However, reshuffling introduces sampling bias: individual sample steps do not necessarily reflect the average batch gradient descent update. This creates challenges in optimization convergence analysis, requiring more advanced techniques. Mishchenko et al. (2020) and Nguyen et al. (2021) analyze the convergence rate of reshuffling methods and demonstrate their superiority over traditional SGD in terms of convergence speed.

In FL, reshuffling methods can be categorized into data reshuffling and client reshuffling. Malinovsky et al. (2023b) first introduced data reshuffling at local clients in FL, while Malinovsky et al. (2023a) first introduced client reshuffling in FL. Both works empirically demonstrate the effectiveness of reshuffling methods through experiments. Demidovich et al. (2024) propose a method that reshuffles both data and clients, analyzing its convergence rate under the assumption of generalized smoothness. However, all the aforementioned methods require a small step size to mitigate the effects of client heterogeneity and ensure convergence. When client heterogeneity is severe, these methods require an extremely small step size, which significantly slows down the optimization process.

Unlike previous methods, our approach achieves a convergence rate independent of data heterogeneity. Our convergence rate depends only on the objective function value in the optimization problem and the algorithm's hyperparameters. This key distinction explains why our method effectively mitigates the heterogeneity problem in federated reshuffling methods.

3. Preliminaries

We denote $\mathbb{R}^n$ as the n-dimensional Euclidean space with inner product $\langle \cdot, \cdot \rangle$ and Euclidean norm $\| \cdot \|$. We denote the unit ball in $\mathbb{R}^n$ as $\mathcal{B}(0,1)$. We denote the set of positive real value as $\mathbb{R}_{++}$. Given a point $x \in \mathbb{R}^n$ and a set A, we denote the distance from x to A as $\mathrm{dist}(x, A)$. An extended-real-valued function $f : \mathbb{R}^n \to [-\infty, \infty]$ is said to be proper if $\mathrm{dom}\, f := \{x \in \mathbb{R}^n : f(x) < \infty\}$ is not empty and f never equals $-\infty$. We say a proper function f is closed if it is lower semicontinuous.

Our method uses the proximal operator in Douglas-Rachford iterations. Therefore, we give the definition.

Definition 3.1. (Proximal Operator). For a proper function $f : \mathbb{R}^n \to [-\infty, \infty]$, we denote the proximal operator of f as

$$\mathrm{prox}_{\beta f}(x) := \arg\min_{z \in \mathbb{R}^n} \left\{ f(z) + \frac{1}{2\beta} \|z - x\|^2 \right\}.$$

In this paper, we consider the following nonconvex optimization problem:

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

where f_i is differentiable and probably non-convex setting.

In federated learning, n is the total number of clients. $x \in \mathbb{R}^d$ corresponds to the parameters of the model we want to train. $f_i : \mathbb{R}^d \to \mathbb{R}$ is emperical loss of the model x defined by the local training data stored by the i_{th} client. In cross-device settings, n is extremely large.

To analyze the optimzation problem (1), we make the following assumptions.

Assumption 3.2. (Well-defined Optimzation Problem) $\mathrm{dom}(f) \neq \emptyset$ and $f^* := \inf_{x \in \mathbb{R}^d} f(x) > -\infty$

In all our theoretical results, we rely on the smoothness property of f_i.

Definition 3.3. (L-smoothness). Function $h : \mathbb{R}^d \to \mathbb{R}$ is called L-smooth if it is continuously differentiable and its gradient is L-Lipschitz continuous for some $L > 0$

$$\|\nabla h(x) - \nabla h(y)\| \leq L\|x - y\| \ \ \forall x, y \in \mathbb{R}^d. \tag{2}$$

For (1), we made the following assumption.

Assumption 3.4. The $f_1, f_2, .., f_n$ in (1) are L-smooth functions.

The assumption 3.2 and 3.4 are widely used in non-convex optimization problems. Finally, we define the ϵ-stationary point for (1).

Definition 3.5. (ϵ-approximate stationary point) Consider (1). If $\tilde{x} \in \mathrm{dom}(F)$ satisfies $\mathbb{E}[\|\nabla f(\tilde{x})\|^2] \leq \epsilon^2$, then $\tilde{x}$ is called ϵ-approximate stationary point, where we use expectation to eliminate the randomness of the algorithm.

4. Algorithm

In this section, we propose the Federated Client Reshuffling Douglas-Rachford Method (**FedCDR**). The complete algorithm is presented in *Algorithm 1*, and its full derivation is provided in Appendix A.1. We explain the details of **FedCDR** as follows:

Algorithm 1 **Fed**erated **C**lient Reshuffling **D**ouglas-**R**achford Method (**FedCDR**)

Choose $x_0 \in \mathrm{dom}(F)$, $\eta > 0$ and $\alpha > 0$.
Initialize the server with $x_0^0 = x^0$.
Initialize each client $i \in [n]$ with $y_i^0 = x^0$, $x_i^0 := \mathrm{prox}_{\eta f_i}(y_i^0)$, and $\widehat{x}_i^0 = 2x_i^0 - y_i^0$.
for meta-epoch $t = 0, ..., T-1$ **do**
  Client-Reshuffling: sample a permutation $S^t = (S_0^t, S_1^t, ..., S_{R-1}^t)$ of $[n]$.
  for communication rounds $r = 0, ..., R-1$ **do**
    Each client $i \in S_r^t$ receives x_r^t from the server.
    For each client $i \in S_r^t$ **do:**
    Update $y_{i,r+1}^t = y_{i,r}^t + \alpha(x_r^t - x_{i,r}^t)$,
    $x_{i,r+1}^t = \mathrm{prox}_{\eta f_i}(y_{i,r+1}^t)$,
    $\widehat{x}_{i,r+1}^t := 2x_{i,r+1}^t - y_{i,r+1}^t$,
    and $g_{i,r}^t = \widehat{x}_{i,r+1}^t - \widehat{x}_{i,r}^t$.
    The server aggregates $g_r^t = \frac{1}{n}\sum_{i \in S_r^t} g_{i,r}^t$.
    Then, the server updates $x_{r+1}^t = x_r^t + g_r^t$.
  end for
end for

Firstly, we initialize both the server and clients at the same point x^0.

At each meta-epoch t, suppose we have a permutation $(\pi_1, \ldots, \pi_n)$ of $\{1, 2, \ldots, n\}$. We then divide $(\pi_1, \ldots, \pi_n)$ into batches, each containing C clients. Specifically, we define the batch S_r^t as

$$S_r^t := \{\pi_{rC}, \ldots, \pi_{(r+1)C-1}\}.$$

Let $R := \frac{n}{C}$ denote the total number of batches, and define $S^t := (S_0^t, ..., S_{R-1}^t)$.

Now, we perform R rounds of inner training within meta-epoch t, where only the clients in batch S_r^t participate in training at round r.

For each communication round r, the server sends the current aggregated model parameter x_r^t to the clients in S_r^t. Then, for each client $i \in S_r^t$, we perform Douglas-Rachford updates as follows:

$$\begin{cases} y_{i,r+1}^t := x_{i,r}^t + \alpha(x_r^t - x_{i,r}^t) \\ x_{i,r+1}^t := \mathrm{prox}_{n\eta f}(y_{i,r+1}^t) \\ \hat{x}_{i,r+1}^t := 2x_{i,r+1}^t - y_{i,r+1}^t \end{cases},$$

where $\alpha > 0$ and $\eta > 0$ are hyperparameters.

Once the local updates are finished, each client i send

$$g_{i,r}^t := \hat{x}_{i,r+1}^t - \hat{x}_{i,r}^t$$

back to the server.

Let $g_r^t = \frac{1}{n}\sum_{i \in S_r^t} g_{i,r}^t$. Then, the server aggregate $g_{i,r}^t$'s as follows:

$$x_{r+1}^t = x_r^t + g_r^t \tag{3}$$

and start the next round.

Note that there are different ways to generate the permutation at the beginning of each meta-epoch. In this paper, we consider two cases: *Shuffle Once* and *Random Reshuffling*.

- *Shuffle Once*: Clients are randomly shuffled at the beginning of training, but the order remains fixed for the entire training process.
- *Random Reshuffling*: At the beginning of each meta-epoch, clients are randomly reshuffled.

We analyze both cases in our theoretical framework.

In summary, we leverage client reshuffling for more efficient client sampling and employ the Douglas-Rachford method to address client heterogeneity.

5. Convergence Analysis

In this section, we analyze Algorithm 1 in different sampling cases. We start with the *Random Reshuffling* case.

Theorem 5.1. *Consider* (1). *Suppose that Assumptions 3.2 and 3.4 holds. Let* $\{(x_r^t, x_{i,r}^t, y_{i,r}^t, \hat{x}_{i,r}^t)\}$ *be generated by Algorithm 1 using stepsizes* α *and* η *and Random Reshuffling method. Let* $\tilde{x}$ *be a random variable chosen uniformly at random from* $\{x_0^0, x_1^0, ..., x_{R-1}^0, x_0^1, ..., x_{R-1}^{T-1}\}$ *as the output of Algorithm 1. Assume stepsizes satisfy* $2 - \alpha - \alpha\eta L - 2\eta^2 L^2 > 0$. *Then, the following holds*

$$\mathbb{E}[\|\nabla f(\tilde{x})\|^2] \leq \frac{\mathcal{A}[f(x^0) - f^*]}{T} \tag{4}$$

where $\mathcal{A}$ *is defined as follows:*

$$\mathcal{A} := \frac{2(1+\eta L)^4}{\alpha\eta(2 - \alpha - \alpha\eta L - 2\eta^2 L^2)}$$

If we run Algorithm 1 for at most

$$K := \lceil \frac{\mathcal{A}[f(x^0) - f^*]}{\epsilon^2} \rceil$$

communication rounds, then $\tilde{x}$ is an ϵ-approximate stationary point of (1) defined by 3.5.

Remark 5.2. The proof of this theorem is presented in Appendix **??**.

- *(Optimal complexity)* We achieve the state-of-the-art communication complexity of $O(\epsilon^{-2})$ for solving the nonconvex optimization problem (1) using federated learning methods. This result matches the lower-bound complexity up to a constant factor.
- *(Heterogeneity bound free)* Our convergence rate in the upper bound is **NOT** dependent on client heterogeneity. This is a key merit of our method compared with previous work. For example, in Theorem 6.2 of Malinovsky et al. (2023a), the convergence rate is related to the data heterogeneity, i.e. $\tilde{\sigma}_*^2 := \frac{1}{n}\sum_{i=1}^n \|\nabla f_i(x^*)\|^2$. Moreover, the error term containing $\tilde{\sigma}_*^2$ grows **quadratically** with the number of clients. This implies that as the number of clients increases, their method requires an extremely small step size to guarantee convergence, potentially slowing down the training process. In contrast, our method does not suffer from this issue. In our experiments with 1000 clients, the results demonstrate the superiority of our approach. This finding provides theoretical justification for why our method effectively mitigates data heterogeneity problems among clients.

The following corollary gives a specific choice of stepsizes in *Algorithm 1*.

Corollary 5.3. *Consider (1). Suppose that Assumptions 3.2 and 3.4 hold. Let $\{(x_r^t, x_{i,r}^t, y_{i,r}^t, \hat{x}_{i,r}^t)\}$ be generated by Algorithm 1 using stepsizes α and η and Random Reshuffling method. Let $\tilde{x}$ be a random variable chosen uniformly at random from $\{x_0^0, x_1^0, ..., x_{R-1}^0, x_0^1, ..., x_{R-1}^{T-1}\}$ as the output of Algorithm 1. Assume stepsizes $\alpha = 1$ and $\eta = \frac{1}{4L}$. Then, the following holds*

$$\mathbb{E}[\|\nabla f(\tilde{x})\|^2] \leq \frac{125L[f(x^0) - f^*]}{4T} \tag{5}$$

If we run Algorithm 1 for at most

$$K := \lceil \frac{125L[f(x^0) - f^*]}{4\epsilon^2} \rceil$$

communication rounds, then $\tilde{x}$ is an ϵ-approximate stationary point of (1) defined by 3.5.

To meet practical requirements, such as client availability only at specific times, we need to shuffle clients before the training process and fix their participation times. In this case, we adopt the *Shuffle Once* method. The following presents the convergence results for *Algorithm 1* under the *Shuffle Once* setting.

Theorem 5.4. *Consider (1). Suppose that Assumptions 3.2 and 3.4 holds. Let $\{(x_r^t, x_{i,r}^t, y_{i,r}^t, \hat{x}_{i,r}^t)\}$ be generated by Algorithm 1 using stepsizes α and η and Shuffle Once method. Let $\tilde{x}$ be a random variable chosen uniformly at random from $\{x_0^0, x_1^0, ..., x_{R-1}^0, x_0^1, ..., x_{R-1}^{T-1}\}$ as the output of Algorithm 1. Assume stepsizes satisfy $2 - \alpha - \alpha\eta L - 2\eta^2 L^2 > 0$. Then, the following holds*

$$\mathbb{E}[\|\nabla f(\tilde{x})\|^2] \leq \frac{\mathcal{B}[f(x^0) - f^*]}{T} \tag{6}$$

where $\mathcal{B}$ is defined as follows:

$$\mathcal{B} := \frac{2(1+\eta L)^4}{\alpha\eta(2 - \alpha - \alpha\eta L - 2\eta^2 L^2)}$$

If we run Algorithm 1 for at most

$$K := \lceil \frac{\mathcal{B}[f(x^0) - f^*]}{\epsilon^2} \rceil$$

communication rounds, then $\tilde{x}$ is an ϵ-approximate stationary point of (1) defined by 3.5.

Remark 5.5. In *Algorithm 1* with *Shuffle Once* method, we can see that it achieves the same communication complexity $O(\epsilon^{-2})$ as *Algorithm 1* with *Random Reshuffling* method. We can still see the independence of the convergence rate with the heterogeneity between clients.

6. Numerical Experiments

To evaluate the performance of **FedCDR**, we conduct a series of experiments on both synthetic and real-world datasets. To demonstrate the effectiveness of our method, we also implement **RR-CLI** by Malinovsky et al. (2023a) as a baseline for comparison. For client shuffling, we adopt the *Random Reshuffling* strategy. We evaluate the performance using training loss, training accuracy, and test accuracy as our key metrics.

We conduct our experiments on a Linux-based server running Ubuntu 20.04. Each experiment is executed on a compute node equipped with a 24-core 2.80GHz Intel processor, an NVIDIA 3090 GPU, and 40GB of virtual memory allocated for the Python interpreter.

6.1. Datasets, Models, and Hyperparameters

We use both synthetic and real-world datasets to evaluate our method. For the synthetic dataset, we follow the construction method proposed by Li et al. (2020a). Specifically, for each client i, we generate samples (x_i, y_i) based

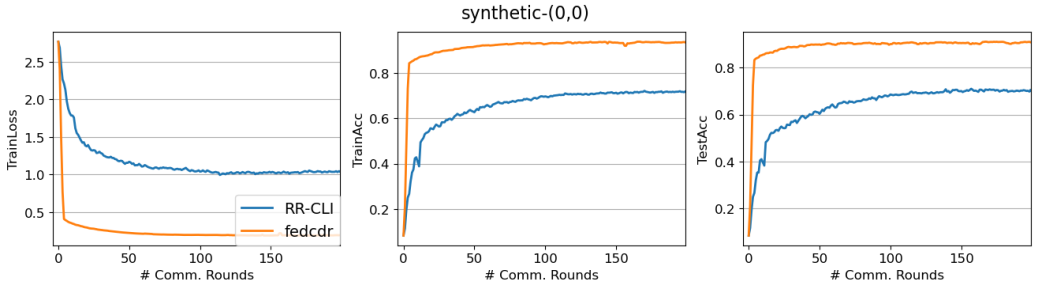

Figure 1: Results with Synthetic-$(0, 0)$ Dataset and 100 Clients.

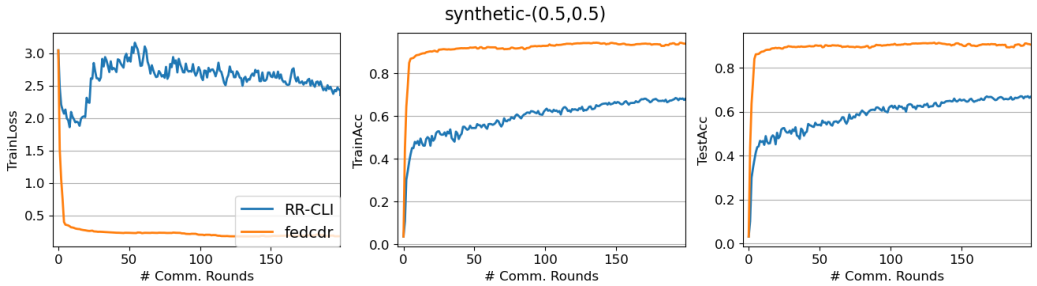

Figure 2: Results with Synthetic-$(0.5, 0.5)$ Dataset and 100 Clients.

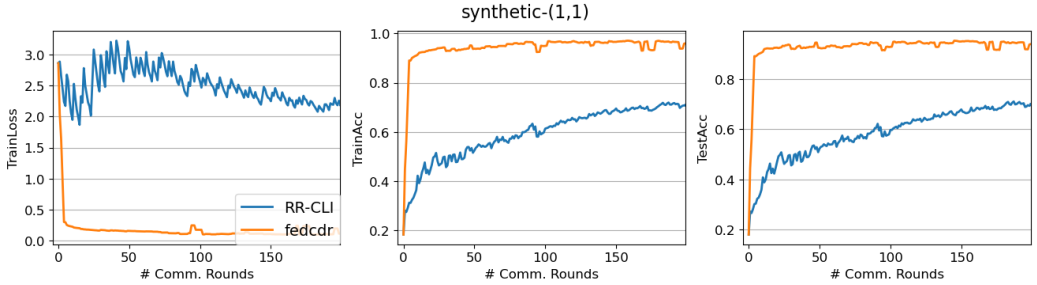

Figure 3: Results with Synthetic-$(1, 1)$ Dataset and 100 Clients.

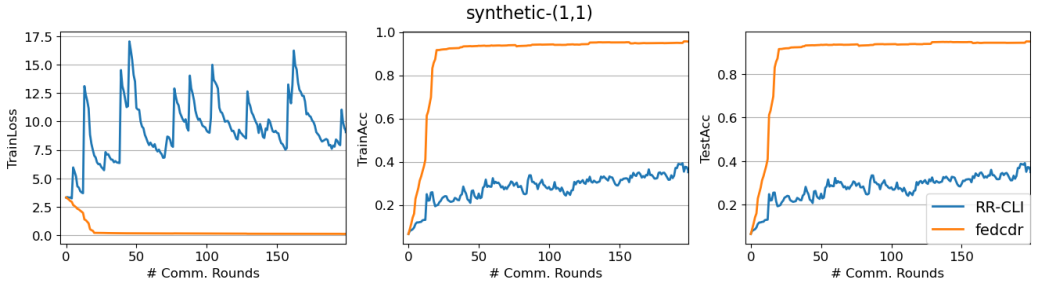

Figure 4: Results with Synthetic-$(1, 1)$ Dataset and 1000 Clients.

on the model: $y = \arg\max(\text{softmax}(Wx + b))$, $x \in \mathbb{R}^{60}$, $W \in \mathbb{R}^{10\times 60}$, $b \in \mathbb{R}^{10}$. The softmax parameters are modeled as follows: $W_k \sim \mathcal{N}(u_k, 1)$, $b_k \sim \mathcal{N}(u_k, 1)$, $u_k \sim \mathcal{N}(0, \alpha)$; $x \sim \mathcal{N}(v_k, \Sigma)$, $v_k \sim \mathcal{N}(B_k, 1)$, $B_k \sim \mathcal{N}(0, \beta)$ where the covariance matrix Σ is defined as a diagonal matrix with $\Sigma_{j,j} = j^{-1.2}$. The parameter α controls the variation between local models, while β determines how much the local data distribution on each device differs from that of other devices. We generate different heterogeneous datasets using various pairs

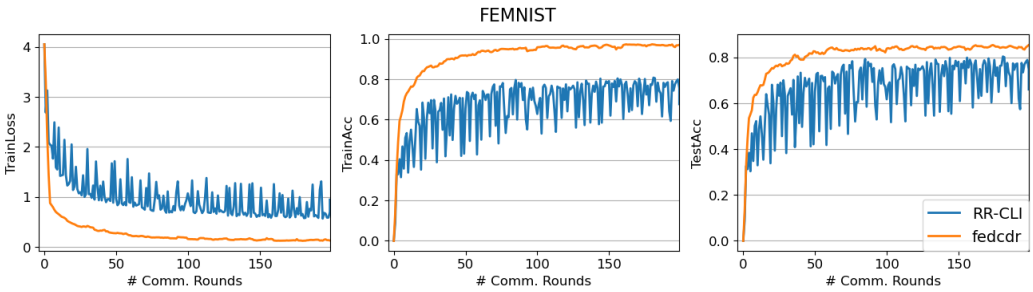

Figure 5: FEMNIST Dataset Results

of (α, β), denoted as Synthetic-(α, β). In our experiments, we use the Synthetic-$(0, 0)$ and Synthetic-$(1, 1)$ datasets. Additionally, we generate an IID synthetic dataset by setting the same W and b across all clients and ensuring that all x values follow the same distribution. Our goal is to learn a global W, b for prediction. For real-world datasets, we use the Federated Extended MNIST (FEMNIST) dataset (Cohen et al., 2017). FEMNIST consists of 200 clients, 62 classes, and a total of 18,345 samples.

We use fully connected neural networks as our models. For all synthetic datasets, we employ a network of size $60 \times 32 \times 10$, where the format represents *input size* $\times$ *hidden layer size* $\times$ *output size*. For the FEMNIST dataset, we use a network of size $784 \times 128 \times 26$, following the architecture used by Li et al. (2020b). Similar to the experiments in Li et al. (2020a), we use the same local solver (SGD) for all algorithms and perform local updates for 20 epochs. We tune the following hyperparameters: the learning rate for the inner SGD solver and the parameters α and η in **FedCDR**. For each dataset, we select the best hyperparameters for the algorithm and report its performance based on the chosen values. The hyperparameter tuning ranges are $\eta \in [1, 1000]$, $\alpha \in [0, 1.99]$ for **FedCDR** and learning rate $lr \in [0.01, 0.001]$ for the local solver.

6.2. Results on synthetic datasets

We compare these algorithms under non-IID synthetic settings. To evaluate the impact of data heterogeneity across clients, we generate three non-IID synthetic datasets with varying degrees of heterogeneity. Specifically, we create Synthetic-(α, β) datasets with $(\alpha, \beta) = (0, 0),\ (0.5, 0.5),\ (1, 1)$ respectively.

We first evaluate the algorithms in a federated learning setting with 100 clients, where 25 clients are selected per round to perform updates. We present the performance of **FedCDR** and the baseline **RR-CLI** on different synthetic datasets: Synthetic-$(0, 0)$ in Figure 1, Synthetic-$(0.5, 0.5)$ in Figure 2, and Synthetic-$(1, 1)$ in Figure 3. As the degree of data heterogeneity increases, **RR-CLI** exhibits slower convergence in both training loss and test accuracy. By mitigating the challenges posed by data heterogeneity, **FedCDR** outperforms **RR-CLI** across different non-IID synthetic datasets. Our method achieves faster and more stable convergence across all evaluation metrics.

Second, we consider a more extreme cross-device federated learning setting with 1000 clients and the Synthetic-$(1, 1)$ dataset, which has the highest degree of heterogeneity. In each communication round, we sample 50 clients to perform updates. The results are presented in Figure 4. **RR-CLI** performs poorly due to the high heterogeneity and the large number of participating clients. This aligns with the theoretical results of **RR-CLI** (Malinovsky et al., 2023a), which state that as the total number of participating clients n increases, the error term related to data heterogeneity in the upper bound increases with n. In contrast, our method demonstrates superior accuracy and convergence rate while maintaining a more stable convergence process. Compared to the results in Figure 3, where the number of clients is only 100, the advantage of our method becomes even more obvious in the cross-device FL setting.

6.3. Results on FEMNIST datasets

In the FEMNIST dataset, there are 200 clients, and in each round, we sample 50 clients to do local training. We present the performance of **FedCDR** and the baseline **RR-CLI** on the FEMNIST dataset in Figure 5. Our method achieves lower training loss, higher training accuracy, and higher test accuracy compared to the baseline. In summary, **FedCDR** demonstrates superior performance on the real-world FEMNIST dataset.

7. Conclusion

In this work, we propose a new method to address the problem of client data heterogeneity in federated learning with client reshuffling, particularly in cross-device federated learning. Our method mitigates the impact of data heterogeneity in federated learning with client reshuffling. We theoretically show that the convergence of our method

remains unaffected by data heterogeneity across clients, eliminating the need to shrink the step size to ensure convergence—an issue present in previous federated learning methods that shuffle clients. For future work, we are interested in studying the theoretical convergence of our method under generalized smoothness conditions. We are also interested in analyzing the last-iterate convergence of the proposed method.

8. Impact Statement

This paper introduces a novel method for federated learning with client reshuffling, particularly in cross-device settings, to address the challenge of data heterogeneity. By improving the convergence of client reshuffling methods in federated learning, our approach enhances their effectiveness, broadens access to advanced AI tools, and contributes to bridging a critical gap in federated learning theory.

References

Bengio, Y. Practical recommendations for gradient-based training of deep architectures. In Montavon, G., Orr, G. B., and Müller, K. (eds.), *Neural Networks: Tricks of the Trade - Second Edition*, volume 7700 of *Lecture Notes in Computer Science*, pp. 437–478. Springer, 2012. doi: 10.1007/978-3-642-35289-8_26. URL https://doi.org/10.1007/978-3-642-35289-8_26.

Bhowmick, A., Duchi, J. C., Freudiger, J., Kapoor, G., and Rogers, R. Protection against reconstruction and its applications in private federated learning. *CoRR*, abs/1812.00984, 2018. URL http://arxiv.org/abs/1812.00984.

Bonawitz, K. A., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. Practical secure aggregation for privacy-preserving machine learning. In Thuraisingham, B., Evans, D., Malkin, T., and Xu, D. (eds.), *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pp. 1175–1191. ACM, 2017. doi: 10.1145/3133956.3133982. URL https://doi.org/10.1145/3133956.3133982.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Chen, W., Horváth, S., and Richtárik, P. Optimal client sampling for federated learning. *Trans. Mach. Learn. Res.*, 2022, 2022. URL https://openreview.net/forum?id=8GvRCWKHIL.

Cohen, G., Afshar, S., Tapson, J., and van Schaik, A. EMNIST: an extension of MNIST to handwritten letters. *CoRR*, abs/1702.05373, 2017. URL http://arxiv.org/abs/1702.05373.

Demidovich, Y., Ostroukhov, P., Malinovsky, G., Horváth, S., Takáč, M., Richtárik, P., and Gorbunov, E. Methods with local steps and random reshuffling for generally smooth non-convex federated optimization, 2024. URL https://arxiv.org/abs/2412.02781.

Diao, E., Ding, J., and Tarokh, V. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=TNkPBBYFkXg.

Fraboni, Y., Vidal, R., Kameni, L., and Lorenzi, M. Clustered sampling: Low-variance and improved representativity for clients selection in federated learning. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 3407–3416. PMLR, 2021a. URL http://proceedings.mlr.press/v139/fraboni21a.html.

Fraboni, Y., Vidal, R., Kameni, L., and Lorenzi, M. On the impact of client sampling on federated learning convergence. *CoRR*, abs/2107.12211, 2021b. URL https://arxiv.org/abs/2107.12211.

Gower, R. M., Loizou, N., Qian, X., Sailanbayev, A., Shulgin, E., and Richtárik, P. SGD: general analysis and improved rates. *CoRR*, abs/1901.09401, 2019. URL http://arxiv.org/abs/1901.09401.

Hard, A., Rao, K., Mathews, R., Beaufays, F., Augenstein, S., Eichner, H., Kiddon, C., and Ramage, D. Federated learning for mobile keyboard prediction. *CoRR*, abs/1811.03604, 2018. URL http://arxiv.org/abs/1811.03604.

Horváth, S., Sanjabi, M., Xiao, L., Richtárik, P., and Rabbat, M. G. Fedshuffle: Recipes for better use of local work in federated learning. *Trans. Mach. Learn. Res.*, 2022, 2022. URL https://openreview.net/forum?id=Lgs5pQ1v30.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., and Suresh, A. T. SCAFFOLD: Stochastic controlled averaging for federated learning. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5132–5143. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/karimireddy20a.html.

Khaled, A., Mishchenko, K., and Richtárik, P. First analysis of local GD on heterogeneous data. *CoRR*, abs/1909.04715, 2019. URL http://arxiv.org/abs/1909.04715.

Konečný, J., McMahan, H. B., Ramage, D., and Richtárik, P. Federated optimization: Distributed machine learning for on-device intelligence. *CoRR*, abs/1610.02527, 2016. URL http://arxiv.org/abs/1610.02527.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. In Dhillon, I. S., Papailiopoulos, D. S., and Sze, V. (eds.), *Proceedings of the Third Conference on Machine Learning and Systems, MLSys 2020, Austin, TX, USA, March 2-4, 2020*. mlsys.org, 2020a. URL https://proceedings.mlsys.org/paper_files/paper/2020/hash/1f5fe83998a09396ebe6477d9475ba0c-Abstract.html.

Li, X., Huang, K., Yang, W., Wang, S., and Zhang, Z. On the convergence of fedavg on non-iid data. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020b. URL https://openreview.net/forum?id=HJxNAnVtDS.

Lions, P.-L. and Mercier, B. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16:964–979, 1979. URL https://api.semanticscholar.org/CorpusID:120554278.

Malinovsky, G., Horváth, S., Burlachenko, K., and Richtárik, P. Federated learning with regularized client participation. *CoRR*, abs/2302.03662, 2023a. doi: 10.48550/ARXIV.2302.03662. URL https://doi.org/10.48550/arXiv.2302.03662.

Malinovsky, G., Mishchenko, K., and Richtárik, P. Server-side stepsizes and sampling without replacement provably help in federated optimization. In Laskaridis, S., Tumanov, A., Baracaldo, N., and Vytiniotis, D. (eds.), *Proceedings of the 4th International Workshop on Distributed Machine Learning, DistributedML 2023, Paris, France, 8 December 2023*, pp. 85–104. ACM, 2023b. doi: 10.1145/3630048.3630187. URL https://doi.org/10.1145/3630048.3630187.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In Singh, A. and Zhu, X. J. (eds.), *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282. PMLR, 2017a. URL http://proceedings.mlr.press/v54/mcmahan17a.html.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In Singh, A. and Zhu, X. J. (eds.), *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282. PMLR, 2017b. URL http://proceedings.mlr.press/v54/mcmahan17a.html.

Mishchenko, K., Khaled, A., and Richtárik, P. Random reshuffling: Simple analysis with vast improvements. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/c8cc6e90ccbff44c9cee23611711cdc4-Abstract.html.

Nguyen, L. M., Tran-Dinh, Q., Phan, D. T., Nguyen, P. H., and van Dijk, M. A unified convergence analysis for shuffling-type gradient methods. *J. Mach. Learn. Res.*, 22:207:1–207:44, 2021. URL https://jmlr.org/papers/v22/20-1238.html.

Nguyen, P. H., Nguyen, L. M., and van Dijk, M. Tight dimension independent lower bound on the expected convergence rate for diminishing step sizes in SGD. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December*

8-14, 2019, Vancouver, BC, Canada, pp. 3660–3669, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/deb54ffb41e085fd7f69a75b6359c989-Abstract.html.

Pathak, R. and Wainwright, M. J. Fedsplit: an algorithmic framework for fast federated optimization. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4ebd440d99504722d80de606ea8507da-Abstract.html.

Rakhlin, A., Shamir, O., and Sridharan, K. Making gradient descent optimal for strongly convex stochastic optimization. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress, 2012. URL http://icml.cc/2012/papers/261.pdf.

Recht, B. and Ré, C. Parallel stochastic gradient algorithms for large-scale matrix completion. *Math. Program. Comput.*, 5(2):201–226, 2013. doi: 10.1007/S12532-013-0053-8. URL https://doi.org/10.1007/s12532-013-0053-8.

Ribero, M. and Vikalo, H. Communication-efficient federated learning via optimal client sampling. *CoRR*, abs/2007.15197, 2020. URL https://arxiv.org/abs/2007.15197.

Rieke, N., Hancox, J., Li, W., Milletarì, F., Roth, H. R., Albarqouni, S., Bakas, S., Galtier, M. N., Landman, B. A., Maier-Hein, K. H., Ourselin, S., Sheller, M. J., Summers, R. M., Trask, A., Xu, D., Baust, M., and Cardoso, M. J. The future of digital health with federated learning. *npj Digit. Medicine*, 3, 2020. doi: 10.1038/S41746-020-00323-1. URL https://doi.org/10.1038/s41746-020-00323-1.

Sheller, M. J., Edwards, B., Reina, G. A., Martin, J., Pati, S., Kotrotsou, A., Milchenko, M., Xu, W., Marcus, D., Colen, R. R., and Bakas, S. Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. *Scientific Reports*, 10, 2020. URL https://api.semanticscholar.org/CorpusID:220812287.

Stich, S. U. Local SGD converges fast and communicates little. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=S1g2JnRcFX.

Themelis, A. and Patrinos, P. Douglas-rachford splitting and ADMM for nonconvex optimization: Tight convergence results. *SIAM J. Optim.*, 30(1):149–181, 2020. doi: 10.1137/18M1163993. URL https://doi.org/10.1137/18M1163993.

Tran-Dinh, Q., Pham, N. H., Phan, D. T., and Nguyen, L. M. Feddr - randomized douglas-rachford splitting algorithms for nonconvex federated composite optimization. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 30326–30338, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/fe7ee8fc1959cc7214fa21c4840dff0a-Abstract.html.

Xia, Q., Ye, W., Tao, Z., Wu, J., and Li, Q. A survey of federated learning for edge computing: Research problems and solutions. *High-Confidence Computing*, 1(1):100008, 2021. ISSN 2667-2952. doi: https://doi.org/10.1016/j.hcc.2021.100008. URL https://www.sciencedirect.com/science/article/pii/S266729522100009X.

Yan, Y., Niu, C., Ding, Y., Zheng, Z., Wu, F., Chen, G., Tang, S., and Wu, Z. Distributed non-convex optimization with sublinear speedup under intermittent client availability. *CoRR*, abs/2002.07399, 2020. URL https://arxiv.org/abs/2002.07399.

Yang, H., Fang, M., and Liu, J. Achieving linear speedup with partial worker participation in non-iid federated learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=jDdzh5ul-d.

Yu, H., Yang, S., and Zhu, S. Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 5693–5700. AAAI Press, 2019. doi: 10.1609/AAAI.V33I01.33015693. URL https://doi.org/10.1609/aaai.v33i01.33015693.

Zhang, X., Hong, M., Dhople, S. V., Yin, W., and Liu, Y. Fedpd: A federated learning framework with optimal rates and adaptivity to non-iid data. *CoRR*, abs/2005.11418, 2020. URL https://arxiv.org/abs/2005.11418.

A. Appendix

In the Appendix, we first derive *Algorithm 1* (**FedCDR**) in detail. Then, we give the full proofs of the convergence analysis of *Algorithm 1*.

A.1. Derivation of *Agorithm 1* (FedCDR)

Distributed Douglas-Rachford We first derive the equivalence problems of the standard federated learning with the constraint. Then, we use the Douglas-Rachford method to split the problem into distributed versions instead of centralized ones. At last, we derive the expression of *Algorithm 1* by adding client reshuffling.

Since the standard federated learning problems formulate as (1), we can equivalently write (1) into the following constrained minimization problem by adding the $x_1 = x_2 = ... = x_n$ constraint:

$$\min_{x_1,x_2,...,x_n} f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(x_i) \ \text{ s.t. } \ x_1 = x_2 = ... = x_n \tag{7}$$

where $\mathbf{x} := [x_1, x_2, ..., x_n]$ concatenates n clients' parameters. Note that $\mathbf{x} \in \mathbb{R}^{nd}$. This constraint $x_1 = x_2 = ... = x_n$ can be considered as a solution space of a particular linear equation, and we can also use a particular linear subspace $\mathcal{H}$ to describe, where $\mathcal{H}$ is defined as $\{\mathbf{x} \in \mathbb{R}^{nd} | x_1 = x_2 = ... = x_n\}$. We can use the indicator function to eliminate the constrained expression in (7). Define $I_{\mathcal{H}}$ be the indicator function of linear subspace $\mathcal{H}$. i.e., $I_{\mathcal{H}} = 0$ if $\mathbf{x} \in \mathcal{H}$, and $I_{\mathcal{H}} = +\infty$ if $\mathbf{x} \notin \mathcal{H}$. Using the indicator function, we can reformulate the optimization problems without any constraint.

$$\min_{x_1,x_2,...,x_n} F(\mathbf{x}) := f(\mathbf{x}) + I_{\mathcal{H}}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(x_i) + I_{\mathcal{H}}(\mathbf{x}) \tag{8}$$

where f is differential while $I_{\mathcal{H}}$ is not differentiable. In this settings, we can use the Douglas-Rachford method to split the optimization of f and $I_{\mathcal{H}}$ to simplify the optimization process. The Douglas-Rachford method for (8) include the following steps:

$$\begin{cases} \mathbf{y}^{k+1} := (1-\alpha)\mathbf{x}^k + \alpha \mathbf{z}^k \\ \mathbf{x}^{k+1} := \text{prox}_{n\eta f}(\mathbf{y}^{k+1}) \\ \mathbf{z}^{k+1} := \text{prox}_{n\eta I_{\mathcal{H}}}(2\mathbf{x}^{k+1} - \mathbf{y}^{k+1}) \end{cases} \tag{9}$$

where k represents the update iterations, $\eta > 0$ can be seen as a update step size and $\alpha \in (0, 2]$ is the relaxation parameter of iterations by Themelis & Patrinos (2020). In classical Douglas-Rachford research (Lions & Mercier, 1979), α is set to be 1.

Since $I_{\mathcal{H}}$ is an indicator of a linear subspace, we can simplify the second prox optimization problem as follows:

$$\begin{aligned} \mathbf{z}^{k+1} &= \text{prox}_{n\eta I_{\mathcal{H}}}(2\mathbf{x}^{k+1} - \mathbf{y}^{k+1}) \\ &\overset{\text{def}}{=} \min_{\mathbf{z}} [n\eta I_{\mathcal{H}}(\mathbf{z}) + \frac{1}{2}\|\mathbf{z} - (2\mathbf{x}^{k+1} - \mathbf{y}^{k+1})\|^2] \\ &\overset{(a)}{=} \min_{z_1,z_2,...,z_n} \|\mathbf{z} - (2\mathbf{x}^{k+1} - \mathbf{y}^{k+1})\|^2 \ \text{ s.t. } \ z_1 = z_2 = ... = z_n \\ &\overset{(b)}{=} \min_{z_1,z_2,...,z_n} \sum_{i=1}^{n} \|z_i - (2x_i^{k+1} - y_i^{k+1})\|^2 \ \text{ s.t. } \ z_1 = z_2 = ... = z_n \end{aligned} \tag{10}$$

Where (a) reformulates the problem by constraint, and (b) calculates the norm by coordinate. Defining $z := z_1$, then the solution of (10) is $\underbrace{(z, \ldots, z)}_{n's}$ with

$$z = \arg\min_{z} \sum_{i=1}^{n} \|z - (2x_i^{k+1} - y_i^{k+1})\|^2 \Rightarrow z = \frac{1}{n}\sum_{i=1}^{n}(2x_i^{k+1} - y_i^{k+1}) \tag{11}$$

Also, since $f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(x_i)$, we can simplify the first prox optimization problem as follows:

$$\begin{aligned}
\mathbf{x}^{k+1} &= \text{prox}_{n\eta f}(\mathbf{y}^{k+1}) \\
&= \arg\min_{\mathbf{x}}[n\eta f(\mathbf{x}) + \frac{1}{2}\|\mathbf{x} - \mathbf{y}^{k+1}\|^2] \\
&= \arg\min_{\mathbf{x}}[f(\mathbf{x}) + \frac{1}{2n\eta}\|\mathbf{x} - \mathbf{y}^{k+1}\|^2] \\
&\overset{(a)}{=} \arg\min_{\mathbf{x}}\{\frac{1}{n}\sum_{i=1}^{n}[f_i(x_i) + \frac{1}{2\eta}\|x_i - y_i^{k+1}\|^2]\} \\
&= \left(\arg\min_{x_1}[f_1(x_1) + \frac{1}{2\eta}\|x_1 - y_1^{k+1}\|^2], \ldots, \arg\min_{x_n}[f_n(x_n) + \frac{1}{2\eta}\|x_n - y_n^{k+1}\|^2]\right) \\
&= \left(\text{prox}_{\eta f_1}(y_1^{k+1}), \ldots, \text{prox}_{\eta f_n}(y_n^{k+1})\right)
\end{aligned} \tag{12}$$

where (a) is because f is separable.

Therefore, we can use distributed algorithms to calculate the centralized Douglas-Rachford iteration method (9), especially in the federated learning framework. We can let distributed clients calculate $x_i^{k+1} = \text{prox}_{\eta f_i}(y_i^{k+1})$ for $i \in [n]$ i.e. (12) and the server aggregates all the pseudo gradients to calculate (10).

To simplify the iteration, we introduce a new variable $\hat{x}_i^k = 2x_i^k - y_i^k$ and we simplify the centralized Douglas-Rachford (9) into decentralized form:

$$\begin{cases}
y_i^{k+1} := (1-\alpha)x_i^k + \alpha z^k, \text{ for } i = 1, \ldots, n \\
x_i^{k+1} := \text{prox}_{n\eta f}(y_i^{k+1}) \text{ for } i = 1, \ldots, n \\
\hat{x}_i^{k+1} := 2x_i^{k+1} - y_i^{k+1} \text{ for } i = 1, \ldots, n \\
z^{k+1} := \frac{1}{n}\sum_{i=1}^{n} \hat{x}_i^{k+1}
\end{cases} \tag{13}$$

The first three equations of (13) are calculated by each client. The last equation of (13) is aggregated by the central server. Decentralized optimization problem (13) is equivalent to centralized optimization problem (9).

Client reshuffling Each meta epoch t starts with the partitioning of all n clients into $R := \frac{n}{C}$ batches $S^t := (S_0^t, S_1^t, ..., S_{R-1}^t)$. The size of each S_r^t is C (batch size). There are two ways to generate the batch sequence. We can use sampling without replacement of clients before each meta epoch (*Random Reshuffling*) or arrange the availability pf client before the whole training process (*Shuffle Once*).

Then, we will use several notations as follows and adapt algorithm (13) with client reshuffling.

We define that meta-epoch variable t satisfies $t \in \{0, 1, ..., T-1\}$. In t-th meta-epoch, we do client reshuffling and generate a sequence of batches with length R. i.e. In t-th meta-epoch, we have R communication round. We define that inner communication round as $r \in \{0, 1, ..., R-1\}$. In t-th meta-epoch and r-th inner communication round, we have the participating client set S_r^t. For each client $m \in S_r^t$, we perform the first steps in (13). Using the notation, we can replace the whole update iteration k with a pair of (t, r) with a transform equation as follows.

First, within some meta-epoch t, at round r, for all chosen client i, we update

$$\begin{cases} y_{i,r+1}^t := x_{i,r}^t + \alpha(x_r^t - x_{i,r}^t) \\ x_{i,r+1}^t := \text{prox}_{n\eta f}(y_{i,r+1}^t) \\ \hat{x}_{i,r+1}^t := 2x_{i,r+1}^t - y_{i,r+1}^t \end{cases} \tag{14}$$

and then the server update

$$x_{r+1}^t := \frac{1}{n}\sum_{i=1}^n \hat{x}_{i,r+1}^t \tag{15}$$

Then in the beginning of the next meta-epoch, we define

$$\begin{aligned} x_0^{t+1} &:= x_R^t \\ x_{i,0}^{t+1} &:= x_{i,R}^t \\ y_{i,0}^{t+1} &:= y_{i,R}^t \\ \hat{x}_{i,0}^{t+1} &:= \hat{x}_{i,R}^t \end{aligned} \tag{16}$$

Instead of performing update for all clients $i \in [n]$, we use client reshuffling to perform *partial participation*. For all participating clients $i \in S_r^t$, the algorithm do iteration as (14). For all inactive clients $i \notin S_r^t$, clients do nothing. We have

$$\begin{cases} y_{i,r+1}^t := y_{i,r}^t \\ x_{i,r+1}^t := x_{i,r}^t \\ \hat{x}_{i,r+1}^t := \hat{x}_{i,r}^t \end{cases} \tag{17}$$

Therefore, we can describe the whole process of our algorithm as follows.

Initialization: Choose the initial parameter $x_0 \in \text{dom}(F)$, hyperparameter $\eta > 0$ and $\alpha > 0$.
First, initialize the server with $x_0^0 = x^0$.
Then, initialize each client $i \in [n]$ with $y_{i,0}^0 := x^0$, $x_{i,0}^0 := \text{prox}_{\eta f_i}(y_{i,0}^0)$, and $\hat{x}_{i,0}^0 := 2x_{i,0}^0 - y_{i,0}^0$.
The t-th meta epoch: Sample a permutation of clients $S^t = (S_0^t, S_1^t, ..., S_{R-1}^t)$ of $[n]$.
The r-th communication round: For all active clients $i \in S_r^t$:
(Communication) Each client $i \in S_r^t$ receives x_r^t from the server.
(Active Client Update) Each client $i \in S_r^t$, it updates

$$\begin{cases} y_{i,r+1}^t := x_{i,r}^t + \alpha(x_r^t - x_{i,r}^t) \\ x_{i,r+1}^t := \text{prox}_{n\eta f}(y_{i,r+1}^t) \\ \hat{x}_{i,r+1}^t := 2x_{i,r+1}^t - y_{i,r+1}^t \end{cases}$$

(Inactive Client Update) Each client $i \notin S_r^t$, it does nothing

$$\begin{cases} y_{i,r+1}^t := y_{i,r}^t \\ x_{i,r+1}^t := x_{i,r}^t \\ \hat{x}_{i,r+1}^t := \hat{x}_{i,r}^t \end{cases}$$

(Communication) Each client $i \in S_r^t$ sends $\hat{x}_{i,r+1}^t$ to the server.
(Server Aggregation) The server aggregates $x_{r+1}^t := \frac{1}{n}\sum_{i=1}^n \hat{x}_{i,r+1}^t$.

To simplify the server aggregation step of our algorithm, we have

$$\begin{aligned} x_{r+1}^t &= \frac{1}{n}\sum_{i=1}^n \hat{x}_{i,r+1}^t \\ &= \frac{1}{n}\sum_{i\in S_r^t} \hat{x}_{i,r+1}^t + \frac{1}{n}\sum_{i\notin S_r^t} \hat{x}_{i,r+1}^t \\ &\overset{(17)}{=} \frac{1}{n}\sum_{i\in S_r^t} \hat{x}_{i,r}^t + \frac{1}{n}\sum_{i\notin S_r^t} \hat{x}_{i,r}^t + \frac{1}{n}\sum_{i\in S_r^t} (\hat{x}_{i,r+1}^t - \hat{x}_{i,r}^t) \\ &= \frac{1}{n}\sum_{i=1}^n \hat{x}_{i,r}^t + \frac{1}{n}\sum_{i\in S_r^t} (\hat{x}_{i,r+1}^t - \hat{x}_{i,r}^t) \\ &\overset{(15)}{=} x_r^t + \frac{1}{n}\sum_{i\in S_r^t} g_{i,r}^t = x_r^t + g_r^t \end{aligned} \tag{18}$$

where we define $g_{i,r}^t := \hat{x}_{i,r+1}^t - \hat{x}_{i,r}^t$ and $g_r^t := \frac{1}{n}\sum_{i\in S_r^t} g_{i,r}^t$. This step is implemented in *Algorithm 1*.

A.2. Proofs of Convergence Analysis of *Algorithm 1*

We first propose an important lemma to show the optimal condition of the inner proximal operator and the relationship between $x_{i,r}^t$, $y_{i,r}^t$ and $\hat{x}_{i,r}^t$.

Lemma A.1. *Let $\{(x_{i,r}^t, y_{i,r}^t, \hat{x}_{i,r}^t)\}$ be generated by Algorithm 1 and starting from initial state $x_i^0 = prox_{\eta f_i}(y_i^0)$ for all $i \in [n]$. Then, for all $i \in [n]$, $r \geq 0$ and $t \geq 0$, we have*

$$y_{i,r}^t = x_{i,r}^t + \eta\nabla f_i(x_{i,r}^t) \text{ and } \hat{x}_{i,r}^t = 2x_{i,r}^t - y_{i,r}^t \tag{19}$$

Proof. We prove (19) by induction for t.

First, we prove (19) for $t = 0$.

For $r = 0$, because of the optimal condition of optimization problem $x_i^0 = \text{prox}_{\eta f_i}(y_i^0)$ for all $i \in [n]$ and initialization of *Algorithm 1*, we have both $y_{i,0}^0 = x_{i,0}^0 + \eta\nabla f_i(x_{i,0}^0)$ and $\hat{x}_{i,0}^0 = 2x_{i,0}^0 - y_{i,0}^0$.

Suppose that (19) holds for $t = 0$ and all $r \geq 0$, i.e., $y_{i,r}^0 = x_{i,r}^0 + \eta\nabla f_i(x_{i,r}^0)$ and $\hat{x}_{i,r}^0 = 2x_{i,r}^0 - y_{i,r}^0$. Then, we will show that (19) holds for $r + 1$. We have two cases:

If $i \in S_r^0$, since the optimality condition of proximal operator, we have $\nabla f_i(x_{i,r+1}^0) + \frac{1}{\eta}(x_{i,r+1}^0 - y_{i,r+1}^0) = 0$. Therefore, $y_{i,r+1}^0 = x_{i,r+1}^0 + \eta\nabla f_i(x_{i,r+1}^0)$. Moreover, due to (14), $\hat{x}_{i,r+1}^0 = 2x_{i,r+1}^0 - y_{i.r+1}^0$.

If $i \notin S_r^0$, since (17) and the induction assumption, we have $y_{i,r+1}^0 = x_{i,r+1}^0 + \eta\nabla f_i(x_{i,r+1}^0)$ and $\hat{x}_{i,r+1}^0 = 2x_{i,r+1}^0 - y_{i.r+1}^0$.

Therefore, both cases above show (19) holds for all $i \in [n]$ when $t = 0$ and all $r \geq 0$.

Second, suppose that (19) holds for $t \geq 0$ and $r \geq 0$, i.e. $y_{i,r}^t = x_{i,r}^t + \eta\nabla f_i(x_{i,r}^t)$ and $\hat{x}_{i,r}^t = 2x_{i,r}^t - y_{i,r}^t$. Then, we will show that (19) holds for $t + 1$.

Since (16) and the induction assumption, i.e. $y_{i,n}^t = x_{i,n}^t + \eta\nabla f_i(x_{i,n}^t)$ and $\hat{x}_{i,n}^t = 2x_{i,n}^t - y_{i,n}^t$. Therfore, we have $y_{i,0}^{t+1} = x_{i,0}^{t+1} + \eta\nabla f_i(x_{i,0}^{t+1})$ and $\hat{x}_{i,0}^{t+1} = 2x_{i,0}^{t+1} - y_{i,0}^{t+1}$. Then we can use a similar proof as case $t = 0$ to show when $t + 1$, for all $r \geq 0$, (19) holds. □

Second, the goal of the following lemma is to bound $\|x_r^t - x_{i,r}^t\|^2$.

Lemma A.2. *Let $\{(x_r^t, x_{i,r}^t)\}$ be generated by Algorithm 1, and $\alpha > 0$. Then, for all $i \in S_t^T$, we have*

$$\|x_r^t - x_{i,r}^t\|^2 \leq \frac{(1+\eta L)^2}{\alpha^2}\|x_{i,r}^t - x_{i,r+1}^t\|^2 \tag{20}$$

Proof. Since the update step of $y_{i,r+1}^t$ and Lemma A.1. For $i \in S_t^r$, we have

$$x_r^t - x_{i,r}^t = \frac{1}{\alpha}(y_{i,r+1}^t - y_{i,r}^t) = \frac{1}{\alpha}(x_{i,r+1}^t - x_{i,r}^t) + \frac{\eta}{\alpha}(\nabla f_i(x_{i,r+1}^t) - \nabla f_i(x_{i,r}^t)) \tag{21}$$

Then we use this expression and $\|a+b\| \leq \|a\| + \|b\|$, we have

$$\begin{aligned}
\|x_r^t - x_{i,r}^t\|^2 &= \|\frac{1}{\alpha}(x_{i,r+1}^t - x_{i,r}^t) + \frac{\eta}{\alpha}(\nabla f_i(x_{i,r+1}^t) - \nabla f_i(x_{i,r}^t))\|^2 \\
&\leq \frac{1}{\alpha^2}(\|x_{i,r+1}^t - x_{i,r}^t\| + \|\nabla f_i(x_{i,r+1}^t) - \nabla f_i(x_{i,r}^t)\|)^2 \\
&\overset{\text{(a)}}{\leq} \frac{1}{\alpha^2}(\|x_{i,r+1}^t - x_{i,r}^t\| + \eta L\|x_{i,r+1}^t - x_{i,r}^t\|)^2 \\
&= \frac{(1+\eta L)^2}{\alpha^2}\|x_{i,r}^t - x_{i,r+1}^t\|^2
\end{aligned}$$

(a) uses the L-smoothness of f_i. □

Third, we use the following lemma to bound the norm of the full gradient $\|\nabla f(x_r^t)\|^2$.

Lemma A.3. *Let $\{(x_r^t, x_{i,r}^t)\}$ be generated by Algorithm 1 and $\alpha > 0$. Then we have*

$$\|\nabla f(x_r^t)\|^2 \leq \frac{1}{n\eta^2}(1+\eta L)^2 \sum_{i=1}^n \|x_{i,r}^t - x_r^t\|^2 \tag{22}$$

Proof. From the aggregation step of *Algorithm 1* and Lemma A.1, we have

$$x_r^t = \frac{1}{n}\sum_{i=1}^n \hat{x}_{i,r}^t = \frac{1}{n}\sum_{i=1}^n (2x_{i,r}^t - y_{i,r}^t) = \frac{1}{n}\sum_{i=1}^n (x_{i,r}^t - \eta \nabla f_i(x_{i,r}^t)) \tag{23}$$

Using (23) and $\|a+b\| \leq \|a\| + \|b\|$, we have

$$\begin{aligned}
\|\nabla f(x_r^t)\|^2 &= \|\sum_{i=1}^n \nabla f_i(x_r^t)\|^2 = \|\frac{1}{\eta}(x_r^t - x_r^t) + \sum_{i=1}^n \nabla f_i(x_r^t)\|^2 \\
&\overset{(23)}{=} \frac{1}{\eta^2}\|\frac{1}{n}\sum_{i=1}^n (x_{i,r}^t - x_r^t + \eta \nabla f_i(x_r^t) - \eta \nabla f_i(x_{i,r}^t))\|^2 \\
&\overset{\text{(a)}}{\leq} \frac{1}{n^2\eta^2}\|\sum_{i=1}^n (\|x_{i,r}^t - x_r^t\| + \eta L\|x_{i,r}^t - x_r^t\|)\|^2 \\
&\leq \frac{1}{n\eta^2}\sum_{i=1}^n (1+\eta L)^2 \|x_{i,r}^t - x_r^t\|^2
\end{aligned}$$

(a) uses the L-smoothness of f_i and $\|a+b\| \leq \|a\| + \|b\|$. □

Fourth, we use the potential function $\mathcal{D}_r^t$ as follows to show the sure descent lemma (Lemma A.5).

Definition A.4. Define potential function $\mathcal{D}$ as follows when $0 \leq r \leq n-1$.

$$\mathcal{D}_r^t := \frac{1}{n}\sum_{i=1}^n [f_i(x_{i,r}^t) + \langle \nabla f_i(x_{i,r}^t), x_r^t - x_{i,r}^t \rangle + \frac{1}{2\eta}\|x_r^t - x_{i,r}^t\|^2] \tag{24}$$

For the continuity of definition and convenience, when $r = n$ we define $\mathcal{D}_n^t := \mathcal{D}_0^{t+1}$.

Lemma A.5. *Let $\{(x_r^t, x_{i,r}^t, y_{i,r}^t, \hat{x}_{i,r}^t)\}$ be generated by Algorithm 1 and $\mathcal{D}_t^r$ be defined by Definition A.4. Then, we have*

$$\mathcal{D}_{r+1}^t \leq \mathcal{D}_r^t - \frac{2-\alpha-\alpha\eta L-2\eta^2L^2}{2n\alpha\eta}\sum_{i\in S_r^t}\|x_{i,r+1}^t - x_{i,r}^t\|^2 \tag{25}$$

Proof. First, we have

$$\|x_{r+1}^t - x_{i,r+1}^t\|^2 - \|x_r^t - x_{i,r+1}^t\|^2 - \|x_{r+1}^t - x_r^t\|^2 = 2\left\langle x_r^t - x_{i,r+1}^t, x_{r+1}^t - x_r^t\right\rangle \tag{26}$$

Using (26), we can derive

$$\begin{aligned}
\mathcal{D}_{r+1}^t &= \frac{1}{n}\sum_{i=1}^n[f_i(x_{i,r+1}^t) + \left\langle \nabla f_i(x_{i,r+1}^t), x_{r+1}^t - x_{i,r+1}^t\right\rangle + \frac{1}{2\eta}\|x_{i,r+1}^t - x_{r+1}^t\|^2] \\
&= \frac{1}{n}\sum_{i=1}^n[f_i(x_{i,r+1}^t) + \left\langle \nabla f_i(x_{i,r+1}^t), x_r^t - x_{i,r+1}^t\right\rangle + \frac{1}{2\eta}\|x_{i,r+1}^t - x_r^t\|^2] \\
&+ \frac{1}{n}\sum_{i=1}^n\left\langle \nabla f_i(x_{i,r+1}^t), x_{r+1}^t - x_r^t\right\rangle + \frac{1}{2\eta}(\|x_{r+1}^t - x_{i,r+1}^t\|^2 - \|x_{r+1}^t - x_r^t\|^2) \\
&\overset{(26)}{=} \frac{1}{n}\sum_{i=1}^n[f_i(x_{i,r+1}^t) + \left\langle \nabla f_i(x_{i,r+1}^t), x_r^t - x_{i,r+1}^t\right\rangle + \frac{1}{2\eta}\|x_{i,r+1}^t - x_r^t\|^2] \\
&+ \frac{1}{n\eta}\sum_{i=1}^n[\left\langle x_r^t - 2x_{i,r+1}^t + (x_{i,r+1}^t + \eta\nabla f_i(x_{i,r+1}^t)), x_{r+1}^t - x_r^t\right\rangle] + \frac{1}{2\eta}\|x_{r+1}^t - x_r^t\|^2 \\
&\overset{(a)}{=} \frac{1}{n}\sum_{i=1}^n[f_i(x_{i,r+1}^t) + \left\langle \nabla f_i(x_{i,r+1}^t), x_r^t - x_{i,r+1}^t\right\rangle + \frac{1}{2\eta}\|x_{i,r+1}^t - x_r^t\|^2] \\
&+ \frac{1}{n\eta}\sum_{i=1}^n[\left\langle x_r^t - 2x_{i,r+1}^t + y_{i,r+1}^t, x_{r+1}^t - x_r^t\right\rangle] + \frac{1}{2\eta}\|x_{r+1}^t - x_r^t\|^2 \\
&\overset{(14)}{=} \frac{1}{n}\sum_{i=1}^n[f_i(x_{i,r+1}^t) + \left\langle \nabla f_i(x_{i,r+1}^t), x_r^t - x_{i,r+1}^t\right\rangle + \frac{1}{2\eta}\|x_{i,r+1}^t - x_r^t\|^2] \\
&+ \frac{1}{n\eta}\sum_{i=1}^n\left\langle x_r^t - \hat{x}_{i,r+1}^t, x_{r+1}^t - x_r^t\right\rangle + \frac{1}{2\eta}\|x_{r+1}^t - x_r^t\|^2 \\
&\overset{(15)}{=} \frac{1}{n}\sum_{i=1}^n[f_i(x_{i,r+1}^t) + \left\langle \nabla f_i(x_{i,r+1}^t), x_r^t - x_{i,r+1}^t\right\rangle + \frac{1}{2\eta}\|x_{i,r+1}^t - x_r^t\|^2] - \frac{1}{2\eta}\|x_{r+1}^t - x_r^t\|^2,
\end{aligned}$$

where (a) uses (19) in Lemma A.1.

Therefore, we have

$$\begin{aligned}
\mathcal{D}_{r+1}^t &= \frac{1}{n}\sum_{i=1}^n[f_i(x_{i,r+1}^t) + \left\langle \nabla f_i(x_{i,r+1}^t), x_r^t - x_{i,r+1}^t\right\rangle + \frac{1}{2\eta}\|x_r^t - x_{i,r+1}^t\|^2] - \frac{1}{2\eta}\|x_{r+1}^t - x_r^t\|^2 \\
&= \frac{1}{n}\sum_{i\in S_r^t} f_i(x_{i,r+1}^t) + \frac{1}{n}\sum_{i\in S_r^t}\left\langle \nabla f_i(x_{i,r+1}^t), x_{i,r}^t - x_{i,r+1}^t\right\rangle + \frac{1}{n}\sum_{i\in S_r^t}\left\langle \nabla f_i(x_{i,r+1}^t), x_r^t - x_{i,r}^t\right\rangle \\
&+ \frac{1}{2n\eta}\sum_{i\in S_r^t}\|x_r^t - x_{i,r+1}^t\|^2 + \frac{1}{n}\sum_{i\notin S_r^t} f_i(x_{i,r}^t) + \frac{1}{n}\sum_{i\notin S_r^t}\left\langle \nabla f_i(x_{i,r}^t), x_r^t - x_{i,r}^t\right\rangle \\
&+ \frac{1}{2n\eta}\sum_{i\notin S_r^t}\|x_r^t - x_{i,r}^t\|^2 - \frac{1}{2\eta}\|x_r^t - x_{r+1}^t\|^2
\end{aligned}$$

Since the L-smoothness property of f_i, we have

$$f_i(x_{i,r+1}^t) + \left\langle \nabla f_i(x_{i,r+1}^t), x_{i,r}^t - x_{i,r+1}^t \right\rangle \leq f_i(x_{i,r}^t) + \frac{L}{2}\|x_{i,r}^t - x_{i,r+1}^t\|^2 \tag{27}$$

Therefore,

$$\begin{aligned}
\mathcal{D}_{r+1}^t &\overset{(27)}{\leq} \frac{1}{n}\sum_{i\in S_r^t}[f_i(x_{i,r}^t) + \left\langle \nabla f_i(x_{i,r+1}^t), x_r^t - x_{i,r}^t \right\rangle + \frac{1}{2\eta}\|x_r^t - x_{i,r+1}^t\|^2] + \frac{L}{2n}\sum_{i\in S_r^t}\|x_{i,r}^t - x_{i,r+1}^t\|^2 \\
&+ \frac{1}{n}\sum_{i\notin S_r^t}[f_i(x_{i,r}^t) + \left\langle \nabla f_i(x_{i,r}^t), x_r^t - x_{i,r}^t \right\rangle + \frac{1}{2\eta}\|x_r^t - x_{i,r}^t\|^2] - \frac{1}{2\eta}\|x_r^t - x_{r+1}^t\|^2 \\
&= \frac{1}{n}\sum_{i\in S_r^t}[f_i(x_{i,r}^t) + \left\langle \nabla f_i(x_{i,r}^t), x_r^t - x_{i,r}^t \right\rangle + \frac{1}{2\eta}\|x_r^t - x_{i,r+1}^t\|^2] + \frac{L}{2n}\sum_{i\in S_r^t}\|x_{i,r}^t - x_{i,r+1}^t\|^2 \\
&+ \frac{1}{n}\sum_{i\notin S_r^t}[f_i(x_{i,r}^t) + \left\langle \nabla f_i(x_{i,r}^t), x_r^t - x_{i,r}^t \right\rangle + \frac{1}{2\eta}\|x_r^t - x_{i,r}^t\|^2] - \frac{1}{2\eta}\|x_r^t - x_{r+1}^t\|^2 \\
&+ \frac{1}{n}\sum_{i\in S_r^t}\left\langle \nabla f_i(x_{i,r+1}^t) - \nabla f_i(x_{i,r}^t), x_r^t - x_{i,r}^t \right\rangle
\end{aligned}$$

Moreover, according to

$$\|x_r^t - x_{i,r+1}^t\|^2 = \|x_r^t - x_{i,r}^t\|^2 + 2 < x_r^t - x_{i,r}^t, x_{i,r}^t - x_{i,r+1}^t > + \|x_{i,r}^t - x_{i,r+1}^t\|^2 \tag{28}$$

We have

$$\begin{aligned}
\mathcal{D}_{r+1}^t &\overset{(28)}{\leq} \frac{1}{n}\sum_{i=1}^n[f_i(x_{i,r}^t) + \left\langle \nabla f_i(x_{i,r}^t), x_r^t - x_{i,r}^t \right\rangle + \frac{1}{2\eta}\|x_r^t - x_{i,r}^t\|^2] - \frac{1}{2\eta}\|x_r^t - x_{r+1}^t\|^2 \\
&+ \frac{L}{2n}\sum_{i\in S_r^t}\|x_{i,r}^t - x_{i,r+1}^t\|^2 + \frac{1}{n}\sum_{i\in S_r^t}\left\langle \nabla f_i(x_{i,r+1}^t) - \nabla f_i(x_{i,r}^t), x_r^t - x_{i,r}^t \right\rangle \\
&+ \frac{1}{n\eta}\sum_{i\in S_r^t}\left\langle x_r^t - x_{i,r}^t, x_{i,r}^t - x_{i,r+1}^t \right\rangle + \frac{1}{2n\eta}\sum_{i\in S_r^t}\|x_{i,r}^t - x_{i,r+1}^t\|^2 \\
&= \mathcal{D}_r^t - \frac{1}{2\eta}\|x_r^t - x_{r+1}^t\|^2 + \frac{1+\eta L}{2n\eta}\sum_{i\in S_r^t}\|x_{i,r}^t - x_{i,r+1}^t\|^2 \\
&+ \frac{1}{n}\sum_{i\in S_r^t}\left\langle \nabla f_i(x_{i,r+1}^t) - \nabla f_i(x_{i,r}^t), x_r^t - x_{i,r}^t \right\rangle + \frac{1}{n\eta}\sum_{i\in S_r^t}\left\langle x_r^t - x_{i,r}^t, x_{i,r}^t - x_{i,r+1}^t \right\rangle
\end{aligned}$$

For $i \in S_r^t$, using (21) and the L-smoothness of f_i, we have

$$\begin{aligned}
\mathcal{D}_{r+1}^t &\leq \mathcal{D}_r^t - \frac{1}{2\eta}\|x_r^t - x_{r+1}^t\|^2 + \frac{1+\eta L}{2n\eta}\sum_{i\in S_r^t}\|x_{i,r}^t - x_{i,r+1}^t\|^2 + \frac{\eta}{n\alpha}\sum_{i\in S_r^t}\|\nabla f_i(x_{i,r+1}^t) - \nabla f_i(x_{i,r}^t)\|^2 \\
&+ \frac{1}{n\alpha}\sum_{i\in S_r^t}\left\langle \nabla f_i(x_{i,r+1}^t) - \nabla f_i(x_{i,r}^t), x_{i,r+1}^t - x_{i,r}^t \right\rangle + \frac{1}{n\alpha}\sum_{i\in S_r^t}\left\langle \nabla f_i(x_{i,r+1}^t) - \nabla f_i(x_{i,r}^t), x_{i,r}^t - x_{i,r+1}^t \right\rangle \\
&- \frac{1}{n\alpha\eta}\sum_{i\in S_r^t}\|x_{i,r+1}^t - x_{i,r}^t\|^2 \\
&\overset{(a)}{\leq} \mathcal{D}_r^t - \frac{2-\alpha-\alpha\eta L - 2\eta^2L^2}{2n\alpha\eta}\sum_{i\in S_r^t}\|x_{i,r+1}^t - x_{i,r}^t\|^2
\end{aligned}$$

where (a) uses the L-smoothness of f_i and omit the $\|x_r^t - x_{r+1}^t\|^2$. □

Lemma A.6. *Let $\{(x_r^t, x_{i,r}^t)\}$ be generated by Algorithm 1 and $\mathcal{D}_r^t$ be defined by Definition A.4. Then we have*

$$\mathbb{E}_{(S_0^t,\ldots,S_r^t)}[\mathcal{D}_{r+1}^t] \leq \mathbb{E}_{(S_0^t,\ldots,S_{r-1}^t)}[\mathcal{D}_r^t] - \frac{\alpha C[2-\alpha-\alpha\eta L-2\eta^2L^2]}{2n^2\eta(1+\eta L)^2}\sum_{i=0}^{n}\|x_{i,r}^t - x_r^t\|^2 \tag{29}$$

Proof. According to Lemma A.2 and Lemma A.5, we have

$$\begin{aligned}\mathcal{D}_{r+1}^t &\leq \mathcal{D}_r^t - \frac{2-\alpha-\alpha\eta L-2\eta^2L^2}{2n\alpha\eta}\sum_{i\in S_r^t}\|x_{i,r+1}^t - x_{i,r}^t\|^2 \\ &\leq \mathcal{D}_r^t - \frac{\alpha[2-\alpha-\alpha\eta L-2\eta^2L^2]}{2n\eta(1+\eta L)^2}\sum_{i\in S_r^t}\|x_{i,r}^t - x_r^t\|^2\end{aligned} \tag{30}$$

Taking expectation of $(S_0^t, ..., S_r^t)$, we have

$$\mathbb{E}_{(S_0^t,\ldots,S_r^t)}[\mathcal{D}_{r+1}^t] \leq \mathbb{E}_{(S_0^t,\ldots,S_r^t)}[\mathcal{D}_r^t] - \frac{\alpha[2-\alpha-\alpha\eta L-2\eta^2L^2]}{2n\eta(1+\eta L)^2}\mathbb{E}_{(S_0^t,\ldots,S_r^t)}\sum_{i\in S_r^t}\|x_{i,r}^t - x_r^t\|^2$$

Since $\mathcal{D}_r^t$ only depends on $(\pi_0^t, ..., \pi_{r-1}^t)$, we have

$$\mathbb{E}_{(S_0^t,\ldots,S_r^t)}[\mathcal{D}_r^t] = \mathbb{E}_{(S_0^t,\ldots,S_{r-1}^t)}[\mathcal{D}_r^t] \tag{31}$$

Using conditional expectation of $(S_0^t, ..., S_{r-1}^t)$, we have

$$\begin{aligned}\mathbb{E}_{(S_0^t,\ldots,S_r^t)}[\sum_{i\in S_r^t}\|x_{i,r}^t - x_r^t\|^2] &= \mathbb{E}_{(S_0^t,\ldots,S_{r-1}^t)}\{\mathbb{E}_{S_r^t}[\sum_{i\in S_r^t}\|x_{i,r}^t - x_r^t\|^2|S_0^t,...,S_{r-1}^t]\} \\ &\overset{(a)}{=} \mathbb{E}_{(S_0^t,\ldots,S_{r-1}^t)}[\frac{C}{n-rC}\sum_{i\notin(S_0^t,\ldots,S_{r-1}^t)}\|x_{i,r}^t - x_r^t\|^2] \\ &= \frac{(C!)^r(n-rC)!}{n!}\sum_{(S_0^t,\ldots,S_{r-1}^t)}[\frac{C}{n-rC}\sum_{i\notin(S_0^t,\ldots,S_{r-1}^t)}\|x_{i,r}^t - x_r^t\|^2] \\ &= \frac{(C!)^r(n-rC)!C}{n!(n-rC)}\sum_{(S_0^t,\ldots,S_{r-1}^t)}[\sum_{i\notin(S_0^t,\ldots,S_{r-1}^t)}\|x_{i,r}^t - x_r^t\|^2]\end{aligned}$$

where (a) uses the first equation in Lemma 1 by Malinovsky et al. (2023b).

To simplify the last equation, we can fix $i = k$ and count the item $\|x_{k,r}^t - x_r^t\|^2$. The item $\|x_{k,r}^t - x_r^t\|^2$ appears in the summation if and only if $k \notin (S_0^t, ..., S_{r-1}^t)$. In other words, we can choose a permutation of length rC in $(0, 1, ..., k-1, k+1, ..., n-1)$ for $(S_0^t, ..., S_{r-1}^t)$ and remove the repetition of the inner permutation order in $S_i^t (0 \leq r \leq n-1)$ in order to get the item $\|x_{k,r}^t - x_r^t\|^2$. In fact, the number of such permutations is $\frac{(n-1)!}{(n-rC-1)!}$ and the repetition of the inner permutation order is $(C!)^r$. So in the summation, the number of item $\|x_{k,r}^t - x_r^t\|^2$ is $\frac{(n-1)!}{(n-rC-1)!(C!)^r}$. So we have

$$\begin{aligned}\mathbb{E}_{(S_0^t,\ldots,S_r^t)}[\sum_{i\in S_r^t}\|x_{i,r}^t - x_r^t\|^2] &= \frac{(C!)^r(n-rC)!C}{n!(n-rC)}\sum_{(S_0^t,\ldots,S_{r-1}^t)}[\sum_{i\notin(S_0^t,\ldots,S_{r-1}^t)}\|x_{i,r}^t - x_r^t\|^2] \\ &= \frac{(C!)^r(n-rC)!C}{n!(n-rC)}\sum_{i=0}^{n}[\frac{(n-1)!}{(n-rC-1)!(C!)^r}\|x_{i,r}^t - x_r^t\|^2] \\ &= \frac{C}{n}\sum_{i=0}^{n}[\|x_{i,r}^t - x_r^t\|^2]\end{aligned} \tag{32}$$

Using (30), (31) and (32), we have

$$\mathbb{E}_{(S_0^t,\ldots,S_r^t)}[\mathcal{D}_{r+1}^t] \leq \mathbb{E}_{(S_0^t,\ldots,S_{r-1}^t)}[\mathcal{D}_r^t] - \frac{\alpha C[2-\alpha-\alpha\eta L-2\eta^2L^2]}{2n^2\eta(1+\eta L)^2}\sum_{i=0}^{n}\|x_{i,r}^t - x_r^t\|^2 \tag{33}$$

□

Finally, we discuss the results with either *Random Reshuffling* method (Lemma A.7) or *Shuffle Once* method (Lemma A.8), and prove the main theorems. In *Shuffle Once* case, since we once shuffle client before the training process, we define

$$S := S^0 = S^1 = \ldots = S^{T-1} \tag{34}$$

Lemma A.7. *Let $\{(x_r^t, x_{i,r}^t, y_{i,r}^t)\}$ be generated by Algorithm 1, we use Random Reshuffling method and $\mathcal{D}_r^t$ be defined by Definition A.4. Then we have*

$$\frac{\alpha\eta(2-\alpha-\alpha\eta L-2\eta^2L^2)}{2(1+\eta L)^4}\mathbb{E}_{(S^0,\ldots,S^{t-1})}[\frac{1}{R}\sum_{r=0}^{R-1}\|\nabla f(x_r^t)\|^2] \leq \mathbb{E}_{(S^0,\ldots,S^{t-1})}[\mathcal{D}_0^t] - \mathbb{E}_{(S^0,\ldots,S^t)}[\mathcal{D}_0^{t+1}] \tag{35}$$

Proof. Using Lemma A.6, we have

$$\frac{\alpha C[2-\alpha-\alpha\eta L-2\eta^2L^2]}{2n^2\eta(1+\eta L)^2}\sum_{i=1}^{n}\|x_{i,r}^t - x_r^t\|^2 \leq \mathbb{E}_{(S_0^t,\ldots,S_{r-1}^t)}[\mathcal{D}_r^t] - \mathbb{E}_{(S_0^t,\ldots,S_r^t)}[\mathcal{D}_{r+1}^t] \tag{36}$$

Adding up (36) from $r=0$ to $r=n-1$, we have

$$\frac{\alpha C[2-\alpha-\alpha\eta L-2\eta^2L^2]}{2n^2\eta(1+\eta L)^2}\sum_{r=0}^{R-1}\sum_{i=1}^{n}\|x_{i,r}^t - x_r^t\|^2 \leq \mathcal{D}_0^t - \mathbb{E}_{(S_0^t,\ldots,S_{R-1}^t)}[\mathcal{D}_R^t] \overset{(a)}{=} \mathcal{D}_0^t - \mathbb{E}_{S^t}[\mathcal{D}_0^{t+1}] \tag{37}$$

where (a) uses Definition A.4.

Using Lemma A.3 and $R = \frac{n}{C}$, we have

$$\frac{\alpha\eta(2-\alpha-\alpha\eta L-2\eta^2L^2)}{2(1+\eta L)^4}\frac{1}{R}\sum_{r=0}^{R-1}\|\nabla f(x_r^t)\|^2 \leq \mathcal{D}_0^t - \mathbb{E}_{S^t}[\mathcal{D}_0^{t+1}]$$

Taking expectation of $(S^0, \ldots, S^{t-1})$, we have

$$\frac{\alpha\eta(2-\alpha-\alpha\eta L-2\eta^2L^2)}{2(1+\eta L)^4}\mathbb{E}_{(S^0,\ldots,S^{t-1})}[\frac{1}{R}\sum_{r=0}^{R-1}\|\nabla f(x_r^t)\|^2] \leq \mathbb{E}_{(S^0,\ldots,S^{t-1})}[\mathcal{D}_0^t] - \mathbb{E}_{(S^0,\ldots,S^t)}[\mathcal{D}_0^{t+1}]$$

□

Lemma A.8. *Let $\{(x_r^t, x_{i,r}^t, y_{i,r}^t)\}$ be generated by Algorithm 1, we use Shuffle Once method and $\mathcal{D}_r^t$ be defined by Definition A.4. Then we have*

$$\frac{\alpha\eta(2-\alpha-\alpha\eta L-2\eta^2L^2)}{2(1+\eta L)^4}\mathbb{E}_S[\frac{1}{R}\sum_{r=0}^{R-1}\|\nabla f(x_r^t)\|^2] \leq \mathbb{E}_S[\mathcal{D}_0^t] - \mathbb{E}_S[\mathcal{D}_0^{t+1}] \tag{38}$$

Proof. Using Lemma A.6 and (34), we have

$$\frac{\alpha C[2-\alpha-\alpha\eta L-2\eta^2L^2]}{2n^2\eta(1+\eta L)^2}\sum_{i=1}^{n}\|x_{i,r}^t - x_r^t\|^2 \leq \mathbb{E}_{(S_0,\ldots,S_{r-1})}[\mathcal{D}_r^t] - \mathbb{E}_{(S_0,\ldots,S_r)}[\mathcal{D}_{r+1}^t] \tag{39}$$

Adding up (39) from $r=0$ to $r=n-1$, we have

$$\frac{\alpha C[2-\alpha-\alpha\eta L-2\eta^2L^2]}{2n^2\eta(1+\eta L)^2}\sum_{r=0}^{R-1}\sum_{i=1}^{n}\|x_{i,r}^t-x_r^t\|^2\leq\mathcal{D}_0^t-\mathbb{E}_{(S_0,\ldots,S_{R-1})}[\mathcal{D}_R^t]\overset{(a)}{=}\mathcal{D}_0^t-\mathbb{E}_S[\mathcal{D}_0^{t+1}] \tag{40}$$

where (a) uses Definition A.4.

Using Lemma A.3 and $R=\frac{n}{C}$, we have

$$\frac{\alpha\eta(2-\alpha-\alpha\eta L-2\eta^2L^2)}{2(1+\eta L)^4}\frac{1}{R}\sum_{r=0}^{R-1}\|\nabla f(x_r^t)\|^2\leq\mathcal{D}_0^t-\mathbb{E}_S[\mathcal{D}_0^{t+1}]$$

Taking expectation of S, we have

$$\frac{\alpha\eta(2-\alpha-\alpha\eta L-2\eta^2L^2)}{2(1+\eta L)^4}\mathbb{E}_S[\frac{1}{R}\sum_{r=0}^{R-1}\|\nabla f(x_r^t)\|^2]\leq\mathbb{E}_S[\mathcal{D}_0^t]-\mathbb{E}_S[\mathcal{D}_0^{t+1}]$$

□

The average of gradients in the whole meta epoch which will help us do theoretical analysis.

Definition A.9. (Meta-epoch Average Gradient Norm)

$$\mathcal{G}^t=\frac{1}{n}\sum_{r=0}^{R-1}\|\nabla f(x_r^t)\|^2 \tag{41}$$

,

where x_r^t is the model of the server at the t-th meta epoch and the r-th inner communication round. $\mathcal{G}^t$ repersents the average norm in the whole meta epoch.

Proof. (**Theorem 5.1**)

First, we define constant $\mathcal{A}$ as follows.

$$\mathcal{A}:=\frac{2(1+\eta L)^4}{\alpha\eta(2-\alpha-\alpha\eta L-2\eta^2L^2)} \tag{42}$$

Using Lemma A.7, (42), and Definition A.9, we have

$$\frac{1}{\mathcal{A}}\mathbb{E}_{(S^0,\ldots,S^{t-1})}[\mathcal{G}^t]\leq\mathbb{E}_{(S^0,\ldots,S^{t-1})}[\mathcal{D}_0^t]-\mathbb{E}_{(S^0,\ldots,S^t)}[\mathcal{D}_0^{t+1}] \tag{43}$$

Using (43) and adding up from $t=0$ to $t=T-1$, we have

$$\frac{1}{\mathcal{A}}\sum_{t=0}^{T-1}\mathbb{E}_{(S^0,\ldots,S^{t-1})}[\mathcal{G}^t]\leq\mathcal{D}_0^0-\mathbb{E}_{(S^0,\ldots,S^t)}[\mathcal{D}_0^T] \tag{44}$$

Due to initialization of *Algorithm 1*, i.e., $x_{i,0}^0=x^0$ and $x_0^0=x^0$, we have

$$\begin{aligned}\mathcal{D}_0^0&=\frac{1}{n}\sum_{i=1}^{n}[f_i(x_{i,0}^0)+\langle\nabla f_i(x_{i,0}^0),x_0^0-x_{i,0}^0\rangle+\frac{1}{2\eta}\|x_0^0-x_{i,0}^0\|^2]\\&=\frac{1}{n}\sum_{i=1}^{n}f_i(x^0)\overset{(1)}{=}f(x^0)\end{aligned} \tag{45}$$

Moreover, due to the L-smoothness of f_i, we have

$$\begin{aligned}\mathcal{D}_r^t &= \frac{1}{n}\sum_{i=1}^{n}[f_i(x_{i,r}^t) + \langle \nabla f_i(x_{i,r}^t), x_r^t - x_{i,r}^t\rangle + \frac{1}{2\eta}\|x_r^t - x_{i,r}^t\|^2] \\ &\overset{(27)}{\geq} \frac{1}{n}\sum_{i=1}^{n}[f_i(x_r^t) - \frac{L}{2}\|x_r^t - x_{i,r}^t\|^2 + \frac{1}{2\eta}\|x_r^t - x_{i,r}^t\|^2] \\ &= \frac{1}{n}\sum_{i=1}^{n}[f_i(x_r^t) + (\frac{L}{2} - \frac{1}{2\eta})\|x_r^t - x_{i,r}^t\|^2] \\ &\overset{(a)}{\geq} \frac{1}{n}\sum_{i=1}^{n} f_i(x_r^t) = f(x_r^t) \geq f^*\end{aligned} \tag{46}$$

where (a) uses $\eta L \leq 1$. Using the condition of Theorem 5.1, this is shown as follows.

$$2 - \alpha - \alpha\eta L - 2\eta^2 L^2 > 0 \Rightarrow 2 - 2\eta^2 L^2 > 0 \Rightarrow \eta L \leq 1 \tag{47}$$

Therefore, we have

$$\mathbb{E}_{(S^0,\ldots,S^t)}[\mathcal{D}_0^T] \geq f^* \tag{48}$$

Using (43), (45), and (48), we have

$$\frac{1}{\mathcal{A}}\sum_{t=0}^{T-1}\mathbb{E}_{(S^0,\ldots,S^{t-1})}[\mathcal{G}^t] \leq f(x^0) - f^*$$

i.e.,

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\mathcal{G}^t] \leq \frac{\mathcal{A}[f(x^0) - f^*]}{T}$$

Finally, Let $\tilde{x}$ be selected uniformly at random from $\{x_0^0, x_1^0, ..., x_{R-1}^0, x_0^1, ..., x_{R-1}^{T-1}\}$ as the output of *Algorithm 1*. We have

$$\begin{aligned}\mathbb{E}[\|\nabla f(\tilde{x})\|^2] &= \mathbb{E}\{\frac{1}{T}\sum_{t=0}^{T-1}\frac{1}{R}\sum_{r=0}^{R-1}[\|\nabla f(x_r^t)\|^2]\} \\ &\overset{(a)}{=} \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\mathcal{G}^t] \leq \frac{\mathcal{A}[f(x^0) - f^*]}{T}\end{aligned} \tag{49}$$

where (a) uses Definition A.9. Therefore, to ensure $\tilde{x}$ is a ϵ-approximate stationary point, according to Definition 3.5, we need to guarantee $\mathbb{E}[\|\nabla f(\tilde{x})\|^2] \leq \epsilon^2$. If we select K as follows.

$$K \geq \frac{\mathcal{A}(f(x^0) - f^*)}{\epsilon^2} \tag{50}$$

Hence, we can select $K := \lceil \frac{\mathcal{A}(f(x^0) - f^*)}{\epsilon^2} \rceil = O(\epsilon^{-2})$ as its lower bound. □

Proof. (**Theorem 5.4**) First, we define constant $\mathcal{B}$ as follows.

$$\mathcal{B} := \frac{2(1+\eta L)^4}{\alpha\eta(2 - \alpha - \alpha\eta L - 2\eta^2 L^2)} \tag{51}$$

Using Lemma A.8, (51), and Definition A.9, for $t > 0$, we have

$$\frac{1}{\mathcal{B}}\mathbb{E}_S[\mathcal{G}^t] \leq \mathbb{E}_S[\mathcal{D}_0^t] - \mathbb{E}_S[\mathcal{D}_0^{t+1}] \tag{52}$$

Particularly, for $t = 0$, we have

$$\frac{1}{\mathcal{B}}\mathbb{E}_S[\mathcal{G}^0] \leq \mathcal{D}_0^0 - \mathbb{E}_S[\mathcal{D}_0^1] \tag{53}$$

Using (52), (53), and adding up from $t = 0$ to $t = T - 1$, we have

$$\frac{1}{\mathcal{B}} \sum_{t=0}^{T-1} \mathbb{E}_S[\mathcal{G}^t] \leq \mathcal{D}_0^0 - \mathbb{E}_S[\mathcal{D}_0^T] \tag{54}$$

According to (46), we have

$$\mathbb{E}_S[\mathcal{D}_0^T] \geq f^* \tag{55}$$

Using (45), (53), and (55), we have

$$\frac{1}{\mathcal{B}} \sum_{t=0}^{T-1} \mathbb{E}_S[\mathcal{G}^t] \leq f(x^0) - f^*$$

i.e.,

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\mathcal{G}^t] \leq \frac{\mathcal{B}[f(x^0) - f^*]}{T}$$

Finally, Let $\tilde{x}$ be selected uniformly at random from $\{x_0^0, x_1^0, ..., x_{R-1}^0, x_0^1, ..., x_{R-1}^{T-1}\}$ as the output of *Algorithm 1*. We have

$$\begin{aligned} \mathbb{E}[\|\nabla f(\tilde{x})\|^2] &= \mathbb{E}\{\frac{1}{T} \sum_{t=0}^{T-1} \frac{1}{R} \sum_{r=0}^{R-1} [\|\nabla f(x_r^t)\|^2]\} \\ &\overset{(a)}{=} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\mathcal{G}^t] \leq \frac{\mathcal{B}[f(x^0) - f^*]}{T} \end{aligned} \tag{56}$$

where (a) uses Definition A.9. Therefore, to ensure $\tilde{x}$ is a ϵ-approximate stationary point, according to Definition 3.5, we need to guarantee $\mathbb{E}[\|\nabla f(\tilde{x})\|^2] \leq \epsilon^2$. If we select K as follows.

$$K \geq \frac{\mathcal{B}(f(x^0) - f^*)}{\epsilon^2} \tag{57}$$

Hence, we can select $K := \lceil \frac{\mathcal{B}(f(x^0) - f^*)}{\epsilon^2} \rceil = O(\epsilon^{-2})$ as its lower bound. □