Multi-Job Intelligent Scheduling with Cross-Device Federated Learning

Ji Liu*‡, Juncheng Jia*§, Beichen Ma‡, Chendi Zhou§, Jingbo Zhou‡, Yang Zhou¶, Huaiyu Dai‖, and Dejing Dou‡

Abstract—Recent years have witnessed a large amount of decentralized data in various (edge) devices of end-users, while the decentralized data aggregation remains complicated for machine learning jobs because of regulations and laws. As a practical approach to handling decentralized data, Federated Learning (FL) enables collaborative global machine learning model training without sharing sensitive raw data. The servers schedule devices to jobs within the training process of FL. In contrast, device scheduling with multiple jobs in FL remains a critical and open problem. In this paper, we propose a novel multi-job FL framework, which enables the training process of multiple jobs in parallel. The multi-job FL framework is composed of a system model and a scheduling method. The system model enables a parallel training process of multiple jobs, with a cost model based on the data fairness and the training time of diverse devices during the parallel training process. We propose a novel intelligent scheduling approach based on multiple scheduling methods, including an original reinforcement learning-based scheduling method and an original Bayesian optimization-based scheduling method, which corresponds to a small cost while scheduling devices to multiple jobs. We conduct extensive experimentation with diverse jobs and datasets. The experimental results reveal that our proposed approaches significantly outperform baseline approaches in terms of training time (up to 12.73 times faster) and accuracy (up to 46.4% higher).

Index Terms—Federated learning, Scheduling, Multi-job, Parallel execution, Distributed learning.

1 Introduction

In recent years, we have witnessed a large amount of decentralized data over various Internet of Things (IoT) devices, mobile devices, etc. [1], which can be exploited to train machine learning models of high accuracy for diverse artificial intelligence applications. Since the data contain sensitive information of end-users, a few stringent legal restrictions [2], [3], [4], [5] have been put in place to protect data security and privacy. In this case, it is difficult or even impossible to aggregate the decentralized data into a single server or a data center to train machine learning models. To enable collaborative training with decentralized data, Federated Learning (FL) [6], which does not transfer raw data, have emerged as a practical approach.

FL was first introduced to collaboratively train a global model with non-Independent and Identically Distributed (non-IID) data distributed across mobile devices [6]. During the training process of FL, the raw data remains decentralized without being transferred to a single server or a single data center [7], [8]. FL only allows the intermediate data to be transferred from the distributed devices, which can be the weights or the gradients of a model. FL generally utilizes a parameter server architecture [9], [10], [11], where a server (or a group of servers) coordinates the training process with numerous devices. To collaboratively train a global model, the server selects (schedules) several devices to perform local model updates based on their local data, and then it aggregates the local models to obtain a new global model. This process is repeated multiple times to generate a global model of high accuracy.

While current FL solutions [6], [12] focus on a single-task job or a multi-task job [13], FL with multiple jobs [14] remains an open problem. The major difference between the multi-task job and multiple jobs is that the tasks of the multi-task job share some common parts of the model, while the multiple jobs do not interact with each other in terms of the model. The multi-job FL deals with the simultaneous training process of multiple independent jobs. Each job corresponds to multiple updates during the training process of a global model with the corresponding decentralized data. While the FL with a single job generally selects a portion of devices to update the model, the other devices remain idle, and the efficiency thus is low. The multi-job FL can well exploit diverse devices for multiple jobs simultaneously, which brings high efficiency. The available devices are generally heterogeneous [15], [16], i.e., the computing and communication capacity of each device is different, and the data in each device may also differ. For instance, multiple machine learning jobs, e.g., CTR models [17], [18], mobile keyboard prediction [19], and travel time prediction [20], may be concurrently executed with FL. The concurrent execution can be carried out with the same group of users. In addition, industry-level machine learning jobs, e.g., recommendation system jobs [21], speech recognition [22], etc., may be adapted to be executed in parallel with FL for privacy issues.

During the training process of multiple jobs, the devices need to be scheduled for each job. At a given time, a device can be scheduled to one job. However, only a portion of the available devices are scheduled to one job to reduce the influence of stragglers [6]. Powerful devices should be scheduled to jobs to accelerate the training process, while

* *Corresponding author.*
‡ *J. Liu, B. Ma, J. Zhou, and D. Dou are with Baidu Inc., Beijing, China.*
§ *J. Jia and C. Zhou are with Soochow University, China.*
¶ *Y. Zhou is with Auburn University, United States.*
‖ *H. Dai is with North Carolina State University, United States.*

other eligible devices should also participate in the training process to increase the fairness of data to improve the accuracy of the final global models. The fairness of data refers to the fair participation of the data in the training process of FL, which can be indicated by the standard deviation of the times to be scheduled to a job [23], [24].

While the scheduling problem of devices is typical NP-hard [25], [26], some solutions have already been proposed for the training process of FL [16], [27], [28], [29] or distributed systems [30], which generally only focus on a single job with FL. In addition, these methods either cannot address the heterogeneity of devices [27], or do not consider the data fairness during the training process [16], [28], [29], which may lead to low accuracy.

In this paper, we propose a Multi-Job Federated Learning (MJ-FL) framework to enable the efficient training of multiple jobs with heterogeneous edge devices. The MJ-FL framework consists of a system model and a novel intelligent scheduling approach. The system model enables the parallel training process of multiple jobs. With the consideration of both the efficiency of the training process, i.e., the time to execute an iteration, and the data fairness of each job for the accuracy of final models, we propose a cost model based on the training time and the data fairness within the system model. We propose an intelligent scheduling approach based on multiple scheduling methods, including two original scheduling methods, i.e., reinforcement learning-based and Bayesian optimization-based, to schedule the devices for each job. To the best of our knowledge, we are among the first to study FL with multiple jobs. This paper is an extension of a conference version [31], with an extra meta-scheduling approach, additional theoretical proof, and extensive experimental results. We summarize our contributions as follows:

- We propose MJ-FL, a multi-job FL framework composed of a parallel training process for multiple jobs and a cost model for scheduling methods. We propose combining the capability and data fairness in the cost model to improve the efficiency of the training process and the accuracy of the global model.
- We propose two scheduling methods, i.e., Reinforcement Learning (RL)-based and Bayesian Optimization (BO)-based methods, to schedule the devices to diverse jobs (more details including the method to estimate the loss in Section 3.3 are added compared with [31]). Each method has advantages in a specific situation. The BO-based method performs better for simple jobs, while the RL-based method is more suitable for complex jobs. In addition, we provide theoretical convergence analysis in Section 5.
- We propose a novel intelligent scheduling approach based on multiple scheduling methods (extra contribution compared with [31]). The novel intelligent scheduling approach is a meta-scheduling approach that coordinates multiple scheduling methods to achieve excellent performance with an adapted dynamic cost model.
- We carry out extensive experimentation to validate the proposed approach. We exploit multiple jobs, composed of Resnet18, CNN, AlexNet, VGG, and LeNet, to demonstrate the advantages of our proposed approach using both IID and non-IID datasets (with extra extensive experimental results for the meta-scheduling approach).

The rest of the paper is organized as follows. We present the related work in Section 2. Then, we explain the system model and formulate the problem with a cost model in Section 3. We present the scheduling methods in Section 4. We provide theoretical convergence analysis in Section 5. The experimental results with diverse models and datasets are given in Section 6. Finally, Section 7 concludes the paper.

2 Related Work

In order to protect the security and privacy of decentralized raw data, FL emerges as a promising approach, which enables training a global model with decentralized data [1], [6], [8], [15]. Based on the data distribution, FL can be classified into three types, i.e., horizontal, vertical, and hybrid [1], [8]. The horizontal FL addresses the decentralized data of the same features, while the identifications are different. The vertical FL handles the decentralized data of the same identifications with different features. The hybrid FL deals with the data of different identifications and different features. In addition, FL includes two variants: cross-device FL and cross-silo FL [7]. The cross-device FL trains global machine learning models with a huge number of mobile or IoT devices, while the cross-silo FL handles the collaborative training process with the decentralized data from multiple organizations or geo-distributed datacenters. In this paper, we focus on the horizontal and cross-device FL.

Current FL approaches [32], [33], [34], [35], [36], [37] generally deal with a single job, i.e., with a single global model. While some FL approaches have been proposed to handle multiple tasks [13], [38], the tasks share some common parts of a global model and deal with the same types of data. In addition, the devices are randomly selected (scheduled) in these approaches.

A few scheduling approaches [16], [16], [27], [28], [28], [29], [29], [30], [39] exist for single-job scheduling while the device scheduling with multi-job FL is rarely addressed. The scheduling methods in the above works are mainly based on some heuristics. For instance, the greedy method [40] and the random scheduling method [27] are proposed for FL, while genetic algorithms [30] are exploited for distributed systems. However, these methods do not consider the fairness of data, which may lead to low accuracy for multi-job FL. The black-box optimization-based methods, e.g., RL [39], BO [41], and deep neural network [42], have been proposed to improve the efficiency, i.e., the reduction of execution time, in distributed systems. They do not consider data fairness either, which may lead to low accuracy for multi-job FL. Although ensemble learning or ensemble method consisting of multiple models [43], has been exploited for scheduling parallel tasks [44], proper cost models and ensemble mechanism should be well designed.

Different from all existing works, we propose a system model for the multi-job FL with the consideration of both efficiency and accuracy. In addition, we propose a novel intelligent scheduling approach based on multiple scheduling methods. To improve the efficiency of the training process,

TABLE 1: Summary of Main Notations

Notation	Definition
$\mathcal{K}$; $\|\mathcal{K}\|$	Set of all devices; size of $\mathcal{K}$
M; m; T	The total number of jobs; index of jobs; total training time
$\mathcal{D}_k^m$; D_k^m; d_k^m	Local dataset of Job m on Device k; size of $\mathcal{D}_k^m$; batch size of the local update of Device k
$\mathcal{D}^m$; D^m	Global dataset of Job m; size of $\mathcal{D}^m$
$F_k^m(\boldsymbol{w})$; $F^m(\boldsymbol{w})$	Local loss function of Job m in Device k; global loss function of Job m
$\boldsymbol{w}_{k,r}^m(j)$	Local model of Job m in Device k in the j-th local update of Round r
R_m	The maximum rounds for Job m during the execution
R'_m	The minimum rounds for Job m to achieve the required performance (loss value or accuracy)
l_m	The desired loss value for Job m
τ_m; C_m	Number of local epochs of Job m; the ratio between the number of devices scheduled to Job m and $\|\mathcal{K}\|$
S_m, $s_{k,m}^r$	The frequency vector for Job m; the frequency of Device k scheduled to Job m at Round r
$\mathcal{V}_m^r$	A set of devices scheduled to Job m at Round r
$\mathcal{V}_o$; $\mathcal{V}_o^r$	A set of occupied devices; the set of occupied devices in Round r
$\zeta_{k,j}^{m,r}$	The sampled dataset for Job m at local iteration h Round r on Device k

we propose two original scheduling methods, i.e., RL and BO, for multi-job FL, which are suitable for diverse models and for both IID and non-IID datasets.

3 System Model and Problem Formulation

In this section, we first explain the motivation for multi-job FL. Then, we propose our multi-job FL framework , consisting of a multi-job FL process and a cost model. Afterward, we formally define the problem to address in this paper. Please see the meanings of the major notations in Table 1.

3.1 Motivation for Multi-Job Federated Learning

Let us assume a scenario where multiple FL jobs are processed simultaneously, e.g., image classification, speech recognition, and text generation. These jobs can be trained in parallel to exploit the available devices efficiently. However, while each device can only update the model of one job at a given time slot, it is critical to schedule devices to different jobs during the training process. As the devices are generally heterogeneous, some may possess high computation or communication capability while others may not. In addition, the data fairness of multiple devices may also impact the convergence speed of the training process. For instance, if only specific powerful devices are scheduled for a job, the model can only learn from the data stored on these devices, while the knowledge from the data stored on other devices may be missed. In order to accelerate the training process of multiple jobs with high accuracy, it is critical to consider how to schedule devices while considering both the computing and communication capability and the data fairness.

A straightforward approach is to train each job separately using the mechanism explained in [27], while exploiting the existing scheduling of single-job FL, e.g., FedAvg [27]. In this way, simple parallelism is considered while the devices are not fully utilized and the system is of low efficiency. In addition, a direct adaptation of existing scheduling methods to multi-job FL cannot address the efficiency and the accuracy at the same time. Thus, it is critical to propose a reasonable and effective approach for the multi-job FL.

3.2 Multi-job Federated Learning Framework

In this paper, we focus on an FL environment composed of a server module and multiple devices. The server module (Server) may consist of a single parameter server or a group of parameter servers [45]. In this section, we present a multi-job FL framework, which is composed of a process for the multi-job execution and a cost model to estimate the cost of the execution.

3.2.1 Multi-job FL Process

Within the multi-job FL process, we assume that K devices, denoted by the set $\mathcal{K}$, collaboratively train machine learning models for M jobs, denoted by the set $\mathcal{M}$. Each Device k is assumed to have M local datasets corresponding to the M jobs without loss of generality, and the dataset of the m-th job on Device k is expressed as $\mathcal{D}_k^m = \{\boldsymbol{x}_{k,d}^m \in \mathbb{R}^{n_m}, y_{k,d}^m \in \mathbb{R}\}_{d=1}^{D_k^m}$ with $D_k^m = |\mathcal{D}_k^m|$ as the number of data samples, $\boldsymbol{x}_{k,d}^m$ representing the d-th n_m-dimentional input data vector of Job m at Device k, and $y_{k,d}^m$ denoting the labeled output of $\boldsymbol{x}_{k,d}^m$. The whole dataset of Job m is denoted by $\mathcal{D}^m = \bigcup_{k\in\mathcal{K}} \mathcal{D}_k^m$ with $D^m = \sum_{k\in\mathcal{K}} D_k^m$. The objective of multi-job FL is to learn respective model parameters $\{\boldsymbol{w}^m\}$ based on the decentralized datasets.

The global learning problem of multi-job FL can be expressed by the following formulation:

$$\min_{\boldsymbol{W}} \sum_{m=1}^{M} \mathbb{L}_m, \text{ with } \mathbb{L}_m = \sum_{k=1}^{K} \frac{D_k^m}{D^m} F_k^m(\boldsymbol{w}^m), \tag{1}$$

where $\mathbb{L}_m$ is the loss value of Job m, $F_k^m(\boldsymbol{w}^m) = \frac{1}{D_k^m} \sum_{\{\boldsymbol{x}_{k,d}^m, y_{k,d}^m\}\in\mathcal{D}_k^m} f^m(\boldsymbol{w}^m; \boldsymbol{x}_{k,d}^m, y_{k,d}^m)$ is the loss value of Job m at Device k, $\boldsymbol{W} := \{\boldsymbol{w}^1, \boldsymbol{w}^2, ..., \boldsymbol{w}^M\}$ is the set of weight vectors for all jobs, and $f^m(\boldsymbol{w}^m; \boldsymbol{x}_{k,d}^m, y_{k,d}^m)$ captures the error of the model parameter $\boldsymbol{w}^m$ on the data pair $\{\boldsymbol{x}_{k,d}^m, y_{k,d}^m\}$.

In order to solve the problem defined in Formula 1, the Server needs to continuously schedule devices for different jobs to iteratively update the global models until the training processes of the corresponding job converge or achieve the target performance requirement (in terms of accuracy or loss value). We design a multi-job FL process as shown in Figure 1. The Server first initializes a global model for each job. The initialization can be implemented randomly or from the pre-training process with public data. To know the current status of devices, the Server sends requests to available devices in Step ①. Then, in Step ②, the Server schedules devices for the current job based on the scheduling plan

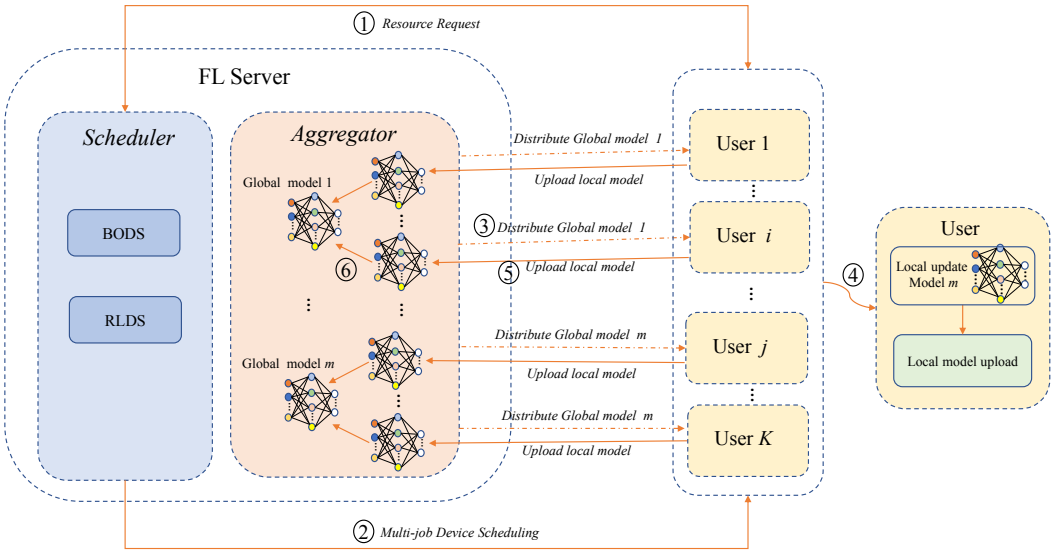


Fig. 1: The training process within the Multi-job Federated Learning Framework.

generated by the scheduling method (see details in Section 4). The scheduling plan is a set of devices selected to execute the local training process of the current job. Note that the scheduling process generates a scheduling plan for each job during the training process of multiple jobs, i.e., with an online strategy, while the scheduling processes for multiple jobs are carried out in parallel. The Server distributes the latest global model for the current job to the scheduled devices in Step ③, and then the model is updated in each device based on the local data in Step ④. Afterward, each device sends the updated model to the Server after its local training in Step ⑤. Finally, the Server aggregates the models of scheduled devices to generate a new global model in Step ⑥. The combination of Steps ① - ⑥ is denoted by a round, which is repeated for each job until the corresponding global model reaches the expected performance (accuracy, loss value, or convergence). Please note that multiple jobs are executed in parallel asynchronously, while a device can only be scheduled to one job at a given time. In addition, we assume that each job is of equal importance.

As an FL environment may contain GPUs or other high-performance chips, it is beneficial to train multiple jobs simultaneously to reduce training time while achieving target accuracy. Within each round, Step ⑥ exploits FedAvg [27] to aggregate multiple models within each job, which can ensure the optimal convergence [46], [47].Within our framework, the sensitive raw data is kept within each device, while only the models are allowed to be transferred. Other methods, e.g., homomorphic encryption [48] and differential privacy [49], can be exploited to protect the privacy of sensitive data.

3.2.2 Cost Model

In order to measure the performance of each round, we exploit a cost model defined in Formula 2, which is composed of time cost and data fairness cost. The data fairness has a significant impact on convergence speed.

$$Cost_m^r(\mathcal{V}_m^r) = \alpha * \mathscr{T}_m^r(\mathcal{V}_m^r) + \beta * \mathscr{F}_m^r(\mathcal{V}_m^r), \tag{2}$$

where α and β are the weights of time cost and fairness cost respectively, $\mathscr{T}_m^r(\cdot)$ represents the execution time of the training process in Round r with the set of scheduled devices $\mathcal{V}_m^r$, and $\mathscr{F}_m^r(\cdot)$ is the corresponding data fairness cost. We choose the linear combination because of its convenience and excellent performance. In practice, we empirically set α and β based on the information from previous execution and adjust them using small epochs. We increase α for fast convergence and increase β mainly for high accuracy.

As defined in Formula 3, the execution time of a round depends on the slowest device in the set of scheduled devices.

$$\mathscr{T}_m^r(\mathcal{V}_m^r) = \max_{k \in \mathcal{V}_m^r} \{t_m^k\}, \tag{3}$$

where t_m^k is the execution time of Round r in Device k for Job m. t_m^k is composed of the communication time and the computation time, which is complicated to estimate and differs for different devices. In this study, we assume that the execution time of each device follows the shift exponential distribution as defined in Formula 4 [50], [51]:

$$P[t_m^k < t] = \begin{cases} 1 - e^{-\frac{\mu_k}{\tau_m D_k^m}(t - \tau_m a_k D_k^m)}, & t \geq \tau_m a_k D_k^m, \\ 0, & \text{otherwise}, \end{cases} \tag{4}$$

where the parameters $a_k > 0$ and $\mu_k > 0$ are the maximum and fluctuation of the computation and communication capability, which is combined into one quantity, of Device k, respectively. Moreover, we assume that the calculation time of model aggregation has little impact on the training process because of the strong computation capability of the Server and the low complexity of the model.

The data fairness of Round r corresponding to Job m is indicated by the deviation of the frequency of each device to be scheduled to Job m defined in Formula 5.

$$\mathscr{F}_m^r(\mathcal{V}_m^r) = \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} (s_{k,m}^r - \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} s_{k,m}^r)^2, \tag{5}$$

where $s_{k,m}^r$ is the frequency of Device k to be scheduled to Job m, and $\mathcal{K}$ and $|\mathcal{K}|$ are the set of all devices and the size, respectively. $s_{k,m}^r$ is calculated by counting the total number of the appearance of Device k to be scheduled to Job m in

the set of scheduling plans for Job m, i.e., $\{\mathcal{V}_m^1, ..., \mathcal{V}_m^r\}$. In particular, $S_m^r = \{s_{1,m}^r, ..., s_{|\mathcal{K}|,m}^r\}$ represents the frequency vector of Job m at Round r. At the beginning, i.e., Round 0, each $s_{k,m}^0 \in S_m^0$ is 0. $s_{k,m}^r$ represents the frequency of Device k scheduled to Job m at Round r. Then, we can calculate $s_{k,m}^{r+1}$ using the following formula:

$$s_{k,m}^{r+1} = \begin{cases} s_{k,m}^r + 1, & \text{if Device } k \in \mathcal{V}_m^r \\ s_{k,m}^r, & \text{otherwise} \end{cases} \tag{6}$$

Please note that the "data fairness" is reflected by the deviation of the frequency of each device scheduled to a job [52], which is different from the "fairness" (the bias of the machine learning models concerning certain features) in machine learning [53]. Formula 5 is inspired by [52], and we are among the first to extend this idea from distributed or network systems to FL. When the devices are non-uniformly sampled with low data fairness, the convergence is slowed down [46], [47]. In addition, data fairness is important due to the underlying data heterogeneity across the devices. Data fairness can help arbitrarily select devices without harming the learning performance.

3.3 Problem Formulation

The problem we address is how to reduce the training time when given a loss value for each job. While the execution of each job is carried out in parallel, the problem can be formulated as follows:

$$\begin{aligned} &\min_{\mathcal{V}_m^r} \Big\{ \sum_{m=1}^{M} \sum_{r=1}^{R'_m} \mathscr{T}_m^r(\mathcal{V}_m^r) \Big\} \\ &\text{s.t.} \begin{cases} \mathbb{L}_m(R'_m) \leq l_m, \\ \mathcal{V}_m^r \subset \mathcal{K}, \forall m \in \{1, 2, ..., M\}, \forall r \in \{1, 2, ..., R'_m\}, \end{cases} \end{aligned} \tag{7}$$

where l_m is the given loss value of Job m, R'_m represents the minimum number of rounds to achieve the given loss in the real execution, and $\mathbb{L}_m(R'_m)$ is the loss value of the trained model at Round R'_m, defined in Formula 1.

We assume Stochastic Gradient Descent (SGD) is utilized to train models, which converges at a rate of $O(r)$ with r representing the number of rounds [54]. Inspired by [46], we exploit Formula 8 to roughly estimate the loss value of the global model for Job m at Round r.

$$Loss_m(r) = \frac{1}{\gamma_m^0 r + \gamma_m^1} + \gamma_m^2, \tag{8}$$

where γ_m^0, γ_m^1 and γ_m^2 represent non-negative coefficients of the convergence curve of Job m. γ_m^0, γ_m^1 and γ_m^2 can be calculated based on previous execution. In addition, we assume that the real number of rounds corresponding to the same loss value has 30% error compared with r (from the observation of multiple executions). Given a loss value of a model, we exploit this loss estimation method to calculate the maximum rounds for each job. Given a loss value of a model, we utilize this loss estimation method to calculate the number of rounds as R_m^c and take $(1 + 0.3) * R_m^c$ as R_m defined in Table 1. Please note that this estimation is different from the loss value during the real execution; i.e., R'_m can be different from R_m.

As it requires the global information of the whole training process, which is hard to predict, to solve the problem, we transform the problem to the following one, which can be solved with limited local information of each round. In addition, in order to achieve the given loss value of Job m within a short time (the first constraint in Formula 7), we need to consider the data fairness within the total cost in Formula 9, within which the data fairness can help reduce R'_m to minimize the total training time.

$$\begin{aligned} &\min_{\mathcal{V}_m^r} \Big\{ TotalCost(\mathcal{V}_m^r) \Big\}, \\ &TotalCost(\mathcal{V}_m^r) = \sum_{m'=1}^{M} Cost_{m'}^r(\mathcal{V}_{m'}^r), \\ &s.t. \qquad \mathcal{V}_{m'}^r \subset \mathcal{K}, \forall m' \in \{1, 2, ..., M\}, \end{aligned} \tag{9}$$

where $Cost_m^r(\mathcal{V}_m^r)$ can be calculated based on Formula 2 with a set of scheduled devices $\mathcal{V}_m^r$ to be generated using a scheduling method for Job m. Since the scheduling results of one job may potentially influence the scheduling of other jobs, we consider the cost of other jobs when scheduling devices to the current job in this problem. As the search space is $O(2^{|\mathcal{K}|})$, this scheduling problem is a combinatorial optimization problem [55] and NP-hard [25].

4 Device Scheduling for Multi-job FL

In this section, we propose two original scheduling methods, i.e., BO-based and RL-based, and a novel intelligent scheduling approach, i.e., meta-greedy, to address the problem defined in Formula 9. The scheduling plan generated by a scheduling method is defined in Formula 10:

$$\mathcal{V'}_m^r = \underset{\mathcal{V}_m^r \subset \{\mathcal{K} \backslash \mathcal{V}_o^r\}}{\operatorname{argmin}} TotalCost(\mathcal{V}_m^r), \tag{10}$$

where $\mathcal{V'}_m^r$ is a scheduling plan, $\mathcal{K} \backslash \mathcal{V}_o^r$ represents the set of available devices to schedule, $TotalCost(\mathcal{V}_m^r)$ is defined in Formula 9, and $\mathcal{K}$ and $\mathcal{V}_o^r$ are the set of all devices and the set of occupied devices in Round r, respectively.

The BO-based and RL-based methods are designed for different model complexities, and we choose the better one based on known profiling information with small tests (a few epochs) to avoid possible limitations. RLDS favors complex jobs, as it can learn the influence among diverse devices. The influence refers to the concurrent, complementary, and latent impacts of the data in multiple devices for diverse jobs. However, BODS favors simple jobs, while it relies on simple statistical knowledge. The complexity of jobs is determined by the number of parameters of models and the size of the training dataset. We consider the probability to release the devices in $\mathcal{V}_o$ in BODS and RLDS, and possible concurrent occupation of other devices for other jobs, which is not explained in the paper to simplify the explanation. During the execution, we assume that a fraction of the devices is sampled for each job and some devices can be unavailable for the execution. In order to optimize the scheduling process for diverse types of models, we further propose a meta-greedy scheduling approach, which takes advantage of multiple scheduling methods and chooses the most appropriate scheduling plan from the results of the scheduling methods.

4.1 Bayesian Optimization-Based Scheduling

As the Gaussian Process (GP) [56] can well represent linear and non-linear functions, BO-based methods [57] can exploit

Algorithm 1: Bayesian Optimization-Based Scheduling

Input:

$\mathcal{V}_o$: A set of occupied devices

S_m : A vector of the frequency of each device scheduled to Job m

R_m : The maximum round of the current Job m

l_m : The desired loss value for Job m.

Output:

$\mathcal{V}_m = \{\mathcal{V}_m^{*1}, ..., \mathcal{V}_m^{*R_m}\}$: a set of scheduling plans, each with the size $|\mathcal{K}| \times C_m$

1: $\Pi_L \leftarrow$ Randomly generate a set of observation points and calculate the cost

2: **for** $r \in \{1, ...R_m\}$ and l_m is not achieved **do**

3: $\Pi' \leftarrow$ Randomly generate a set of observation points with the devices within $\mathcal{K} \backslash \mathcal{V}_o$

4: $\mathcal{V}_m^{*r} \leftarrow \underset{\mathcal{V} \subset \Pi'}{arg\,\max} \alpha_{\mathrm{EI}}(\mathcal{V}; \Pi')$

5: FL training of Job m with $\mathcal{V}_m^{*r}$ and update S_m, $\mathcal{V}_o$

6: $\mathbb{C}_r = TotalCost(\mathcal{V}_m^{*r})$

7: $\Pi_{L+r} \leftarrow \Pi_{L+r-1} \cup (\mathcal{V}_m^{*r}, \mathbb{C}_r)$

8: **end for**

a GP to find a near-optimal solution for the problem defined in Formula 10. In this section, we propose a Bayesian Optimization-based Device Scheduling method (BODS).

We adjust a GP to fit the cost function $TotalCost(\cdot)$. The GP is composed of a mean function μ defined in Formula 11 and a covariance function K defined in Formula 12 with a $Matern$ kernel [58].

$$\mu(\mathcal{V}_m^r) = \mathop{\mathbb{E}}_{\mathcal{V}_m^r \subset \{\mathcal{K} \backslash \mathcal{V}_o^r\}} [TotalCost(\mathcal{V}_m^r)] \tag{11}$$

$$\mathrm{K}(\mathcal{V}_m^r, \mathcal{V}_m'^r) = \mathop{\mathbb{E}}_{\mathcal{V}_m^r \subset \{\mathcal{K} \backslash \mathcal{V}_o^r\}, \mathcal{V}_m'^r \subset \{\mathcal{K} \backslash \mathcal{V}_o^r\}} [(TotalCost(\mathcal{V}_m^r) - \mu(\mathcal{V}_m^r))(TotalCost(\mathcal{V}_m'^r) - \mu(\mathcal{V}_m'^r))] \tag{12}$$

The BODS is explained in **Algorithm 1**. First, we generate a random set of observation points and calculate the cost according to the Formula 2 (Line 1). Each observation point is a pair of scheduling plan and cost for the estimation of mean function and the covariance function. Then, in each round, we randomly sample a set of scheduling plans (Line 3), within which we use updated μ and K based on Π_{L+r-1} (Line 4) to select the one with the largest reward. Afterward, we perform the FL training for Job m with the generated scheduling plan (Line 5). In the meanwhile, we calculate the cost corresponding to the actual execution (Line 6) according to Formula 9 and update the observation point set (Line 7).

Let $(\mathcal{V}_l^r, \mathbb{C}_l)$ denote an observation point l for Job m in Round r, where $\mathcal{V}_l^r = \{\mathcal{V}_{l,1}^r, ..., \mathcal{V}_{l,M}^r\}$ and $\mathbb{C}_l$ is the cost value of $TotalCost(\mathcal{V}_{l,m}^r)$ while the scheduling plans of other jobs are updated with the ones in use in Round r. At a given time, we have a set of observations $\Pi_{L-1} = \{(\mathcal{V}_1^r, \mathbb{C}_1), ..., (\mathcal{V}_{L-1}^r, \mathbb{C}_{L-1})\}$ composed of $L-1$ observation points. We denote the minimum cost value within the $L-1$ observations by $\mathbb{C}_{L-1}^+$. Then, we exploit Expected Improvement(EI) [59] to select a new scheduling plan $\mathcal{V}_m^{*r}$ in Round r what improves $\mathbb{C}_{L-1}^+$ the most, which is the utility function. Please note that this is not an exhaustive search since we randomly select several observation points (a subset of the whole search space) at the beginning and add new observation points using the EI method. The utility function is defined in Formula 13.

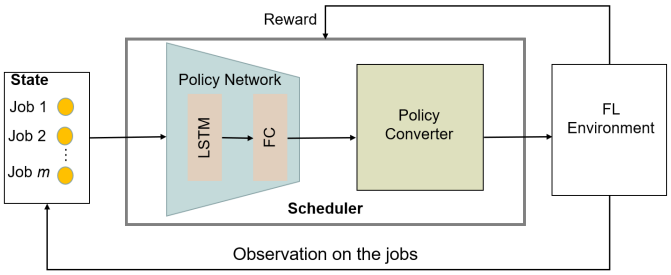


Fig. 2: The architecture of the RLDS.

$$u(\mathcal{V}_m^{*r}) = max(0, \mathbb{C}_{L-1}^+ - TotalCost(\mathcal{V}_m^{*r})), \tag{13}$$

where we receive a reward $\mathbb{C}_{L-1}^+ - TotalCost(\mathcal{V}_m^{*r})$ if $TotalCost(\mathcal{V}_m^{*r})$ turns out to be less than $\mathbb{C}_{L-1}^+$, and no reward otherwise. Then, we use the following formula, which is also denoted an acquisition function, to calculate the expected reward of a given scheduling plan $\mathcal{V}$.

$$\begin{aligned} \alpha_{\mathrm{EI}}(\mathcal{V}; \Pi_{L-1}) =& \mathbb{E}[u(\mathcal{V})|\mathcal{V}, \Pi_{L-1}] \\ =& (\mathbb{C}_{L-1}^+ - \mu(\mathcal{V}))\Phi(\mathbb{C}_{L-1}^+; \mu(\mathcal{V}), \mathrm{K}(\mathcal{V}, \mathcal{V})) \\ &+ \mathrm{K}(\mathcal{V}, \mathcal{V})\mathcal{N}(\mathbb{C}_{L-1}^+; \mu(\mathcal{V}), \mathrm{K}(\mathcal{V}, \mathcal{V})), \end{aligned} \tag{14}$$

where Φ is the Cumulative Distribution Function (CDF) of the standard Gaussian distribution. Finally, we can choose the scheduling plan with the largest reward as the next observation point, i.e., $\mathcal{V}_{L,m}^{*r}$.

4.2 Reinforcement Learning-Based Scheduling

In order to learn more information about the near-optimal scheduling patterns for complex jobs, we further propose a Reinforcement Learning-based Device Scheduling (RLDS) method as shown in Figure 2. In addition, the method is inspired by [39], [60]. The scheduler of RLDS consists of a policy network and a policy converter. During the process of device scheduling, RLDS collects the status information of jobs as the input of the policy network. Afterwards, the policy network generates a list of probabilities on all devices as the output. Finally, the policy converter converts the list into a scheduling plan.

4.2.1 *Policy Network*

The policy network is implemented using a Long Short-Term Memory (LSTM) network followed by a fully connected layer, which can learn the device sharing relationship among diverse jobs. We take the computation and communication capability of available devices to be used in Formula 4, and the data fairness of each job defined in Formula 5 as the input. The network calculates the probability of each available device to be scheduled for a job.

4.2.2 *Policy Converter*

The Policy Converter generates a scheduling plan based on the probability of each available device which is calculated by the policy network with the ϵ-greedy strategy [61].

4.2.3 *Training*

In the training process of RLDS, we define the reward as $\mathscr{R}^m = -1 * TotalCost(\mathcal{V}_m^r)$. Inspired by [62], [63], we exploit Formula 15 to update the policy network:

Algorithm 2: Reinforcement Learning Based Pre-Training

Input:

$\mathcal{V}_o$: A set of occupied devices

S_m : A vector of the frequency of each device scheduled to Job m

N : The number of scheduling plans used to train the network for each round

R_m : The maximum round of the current Job m

l_m : The desired loss value for Job m.

Output:

θ : Parameters of the pre-trained policy network

1: $\theta \leftarrow$ randomly initialize the policy network, $b_m \leftarrow 0$
2: **for** $r \in \{1, 2, ..., R_m\}$ and l_m is not achieved **do**
3: $\quad \mathcal{V}_m^r \leftarrow$ generate a set of N scheduling plans
4: $\quad$ **for** $\mathcal{V}_{n,m}^r \in \mathcal{V}_m^r$ **do**
5: $\quad\quad \mathscr{R}_n^m \leftarrow -1 * TotalCost(\mathcal{V}_{n,m}^r)$
6: $\quad$ **end for**
7: $\quad$ Update θ according to Formula 15
8: $\quad b_m \leftarrow$ (1 - γ) * b_m + $\frac{\gamma}{N}$ * $\sum_{n=1}^{N} \mathscr{R}_n^m$
9: $\quad \mathcal{V}_m^{*r} \leftarrow \text{argmin}_{\mathcal{V}_{n,m}^r \in \mathcal{V}_m^r}$ TotalCost($\mathcal{V}_{n,m}^r$)
10: $\quad$ Update S_m, $\mathcal{V}_o$ with $\mathcal{V}_m^{*r}$
11: **end for**

$$\theta^{'} = \theta + \frac{\eta}{N} \sum_{n=1}^{N} \sum_{k \in \mathcal{V}_{n,m}^r}^{\mathcal{V}_{n,m}^r \subset \mathcal{K} \setminus \mathcal{V}_o^r} \nabla_\theta \quad \log P(\mathscr{S}_k^m | \mathscr{S}_{(k-1):1}^m; \theta) \\ (\mathscr{R}_n^m - b_m), \tag{15}$$

where θ' and θ represent the updated parameters and the current parameters of the policy network, respectively, η represents the learning rate, N is the number of scheduling plans to update the model in Round r ($N > 1$ in the pre-training process and $N = 1$ during the execution of multiple jobs), P represents the probability calculated based on the RL model, $\mathscr{S}_k^m = 1$ represents that Device k is scheduled to Job m, and b_m is the baseline value for reducing the variance of the gradient.

We pre-train the policy network using **Algorithm 2**. First, we randomly initialize the policy network (Line 1). We use the latest policy network and the ϵ-Greedy method to generate N scheduling plans (Line 5). The parameters are updated based on the Formula 15 (Line 7), and the baseline value b_m is also updated with the consideration of the historical value (Line 8). Afterward, we choose the best scheduling plan that corresponds to the minimum total cost, i.e., the maximum reward (Line 9). Finally, we update the frequency matrix S_m and the set of occupied devices $\mathcal{V}_o$, while assuming that the best scheduling plan is used for the multi-job FL (Line 10).

After the pre-training, we exploit RLDS during the training process of multiple jobs within the MJ-FL framework as shown in **Algorithm 3**. First, we load the pre-trained policy network and initialize the parameters $\Delta\theta$, b_m (Line 1). When generating a scheduling plan for Job m, the latest policy network is utilized (Line 3). We perform the FL training for Job m with the generated scheduling plan and update the frequency matrix S_m and the set of occupied devices $\mathcal{V}_o$ (Line 4). Afterward, we calculate the reward corresponding to the real execution (Line 5). The parameters are updated based on the Formula 15 (Line 6), while the baseline value b_m is updated with the consideration of the historical value (Line 7).

Algorithm 3: Reinforcement Learning-Based Scheduling

Input:

$\mathcal{V}_o$: A set of occupied devices

S_m : A vector of the frequency of each device scheduled to Job m

R_m : The maximum round of the current Job m

l_m : The desired loss value for Job m.

Output:

$\mathcal{V}_m = \{\mathcal{V}_m^1, ..., \mathcal{V}_m^{R_m}\}$: a set of scheduling plans, each with the size $|\mathcal{K}| \times C_m$

1: $\theta \leftarrow$ pre-trained policy network, $\Delta\theta \leftarrow 0$, $b_m \leftarrow 0$
2: **for** $r \in \{1, 2, ..., R_m\}$ and l_m is not achieved **do**
3: $\quad \mathcal{V}_m^r \leftarrow$ generate a scheduling plan using the policy network
4: $\quad$ FL training of Job m and update S_m, $\mathcal{V}_o$
5: $\quad$ Compute $\mathscr{R}^m$
6: $\quad$ Update θ according to Formula 15
7: $\quad b_m \leftarrow$ (1 - γ) * b_m + γ * $\mathscr{R}_n^m$
8: **end for**

4.3 Meta-Greedy Scheduling

Multiple jobs of diverse structures and layers exist within the multi-job federated learning environment. Some scheduling methods, e.g., BODS, favor simple jobs, while some other scheduling methods, e.g., RLDS, prefer complex jobs. In addition, heuristic scheduling methods, e.g., Greedy and Genetic, can be exploited to schedule devices, as well. In this case, we take advantage of the existing scheduling methods, and further propose a meta-greedy scheduling approach as shown in Figure 3. In Round r, the Meta-Greedy executes six scheduling methods in parallel, appends the solution generated by each method to the candidate scheduling solutions set Θ_m^r, and selects a scheduling plan from the set of candidate device scheduling solutions according to Formula 16 as the solution for the r-th round under the current job. The meta-greedy scheduling approach chooses the most appropriate one from the scheduling plans generated by multiple methods, which can be closer to the optimal solution compared with that of a single method.

The Meta-Greedy algorithm is shown in **Algorithm 4**. First, for each round (Line 2), we exploit diverse scheduling methods, e.g., BODS, RLDS, Random [27], FedCS [28], Genetic [30], and Greedy [40], to generate scheduling plan candidates (Line 3). As each scheduling method may have superior performance in a specific environment (Random corresponds to high final accuracy; Genetic corresponds to high accuracy at the beginning of the training process for the jobs of moderate complexity; Greedy corresponds to high accuracy at the beginning of the training process of simple jobs; FedCS corresponds to high accuracy at the middle of the the training process; see details in Section 6), we take these 6 methods in Meta-Greedy. Then, we choose a scheduling plan that corresponds to the smallest total cost according to Formula 16 (Line 4). Finally, we utilize the selected scheduling plan for the training process (Line 5).

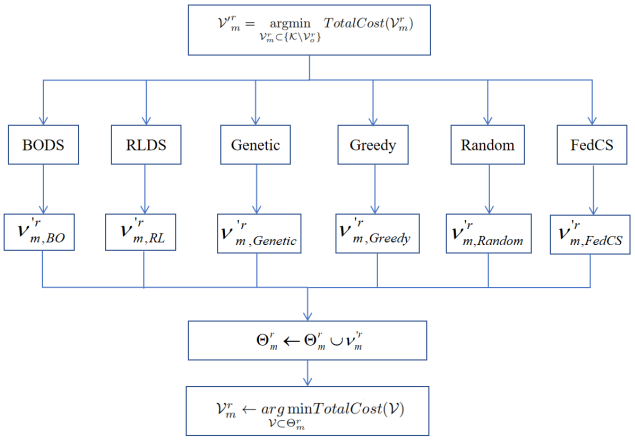


Fig. 3: The architecture of the Meta-Greedy.

TABLE 2: Experimental Setup of Group A. Size represents the size of training samples and test samples (number of training samples/number of test samples). "Emnist-L" represents "Emnist-Letters" and "Emnist-D" represents "Emnist-Digitals".

datasets	Cifar10	Emnist-L	Emnist-D
Features	32x32	28x28	28x28
Network model	VGG16	CNN	LeNet5
Parameters	26,233K	3,785K	62K
Size	50k/10k	124.8k/20.8k	240k/40k
Local epochs	5	5	5
Mini-batch size	30	10	64

We propose a dynamic cost model for the selection in Line 4. While Formula 2 can also be exploited for the selection, the influence of the data fairness becomes smaller in the later round of FL as the frequency of the participation has little change with a small increase of one. We reconstruct the cost model using Formula 16 to replace Formula 2.

$$ReCost(\mathcal{V}^r_m) = \alpha * \mathscr{T}^r_m(\mathcal{V}^r_m) + \beta^r * \mathscr{F}^r_m(\mathcal{V}^r_m), \tag{16}$$

where β^r dynamically changes according to r and is defined in Formula 17.

$$\beta^r = \beta * \Omega(r), r \geq 1 \tag{17}$$

where $\Omega(r)$ is a function based on Round r. $\Omega(r)$ should enhance the impact of data fairness when r becomes significant. We take $\Omega(r) = \sqrt{r}$ in our algorithm because of its excellent performance (see details in Section 6).

5 Convergence Analysis

In this section, we present the theoretical convergence proof for our multi-job FL with arbitrary scheduling methods. We first introduce the assumptions and then present the convergence theorem and corollary.

Assumption 1. *Lipschitz gradient: The function F^m_k is L-smooth for each device $k \in \mathcal{N}$ i.e., $\| \nabla F^m_k(x) - \nabla F^m_k(y) \| \leq L \| x - y \|$.*

Assumption 2. *Unbiased stochastic gradient: $\mathbb{E}_{\zeta^{m,r}_{k,h} \sim \mathcal{D}_i}[\nabla f^m_k(w^m; \zeta^{m,r}_{k,h})] = \nabla F^m_k(w^m)$.*

Assumption 3. *Bounded local variance: For each device $k \in \mathcal{N}$, the variance of its stochastic gradient is bounded: $\mathbb{E}_{\zeta^{m,r}_{k,h} \sim \mathcal{D}_i} \| \nabla f^m_k(w^m, \zeta^{m,r}_{k,h}) - \nabla F^m_k(w^m) \|^2 \leq \sigma^2$.*

Algorithm 4: Meta-Greedy Scheduling

Input:
$\mathcal{V}_o$: A set of occupied devices
S_m : A vector of the frequency of each device scheduled to Job m
R_m : The maximum round of the current Job m
l_m : The desired loss value for Job m.

Output:
$\mathcal{V}_m = \{\mathcal{V}^1_m, ..., \mathcal{V}^{R_m}_m\}$: a set of scheduling plans, each with the size $|\mathcal{K}| \times C_m$

1: Initialize $\mathcal{V}_m \leftarrow \varnothing$
2: **for** $r \in \{1, 2, ..., R_m\}$ and l_m is not achieved **do**
3: $\Theta^r_m \leftarrow$ generate a set of candidate scheduling plans using BODS, RLDS, Genetic, Greedy, FedCS and Random within $\mathcal{K} \setminus \mathcal{V}_o$
4: $\mathcal{V}^r_m \leftarrow arg\min_{\mathcal{V} \subset \Theta^r_m} ReCost(\mathcal{V})$
5: $\mathcal{V}_m \leftarrow \mathcal{V}_m \cup \mathcal{V}^r_m$
6: **end for**

TABLE 3: Experimental Setup of Group B. Size represents the size of training samples and test samples (number of training samples/number of test samples).

datasets	Fashion_mnist	Cifar10	Mnist
Features	28x28	32x32	28x28
Network model	CNN	ResNet18	AlexNet
Parameters	225K	598K	3,275K
Size	60K/10K	50K/10K	60K/10K
Local epochs	5	5	5
Mini-batch size	10	30	64

Assumption 4. *Bounded local gradient: For each device $k \in \mathcal{N}$, the expected squared of stochastic gradient is bounded: $\mathbb{E}_{\zeta^{m,r}_{k,h} \sim \mathcal{D}_i} \| \nabla f^m_k(w^m, \zeta^{m,r}_{k,h}) \|^2 \leq G^2$.*

Assumption 1 has been made in [46], [64], while Assumptions 2, 3, 4, have been exploited in [65]. When Assumptions 1 - 4 hold, we get the following theorem.

Theorem 5.1. *Suppose that Assumptions 1 to 4 hold, and consider that F^m_k is a non-convex function. When the learning rate satisfies $0 < \eta^m \leq \frac{1}{L}$, then for all $R \geq 1$ we have:*

$$\begin{aligned} &\frac{1}{RH} \sum_{r=1}^{R} \sum_{h=1}^{H} \mathbb{E} \| \nabla F^m(\bar{w}^m_{r,h}) \|^2 \\ &\leq \frac{2}{\eta^m_{r,h} RH} (F^m(\bar{w}^m_{1,1}) - F^{m*}) + L \eta^m_{r,h} \sigma^2 \\ &\quad + L^2 Q^2 (H-1)^2 {\eta^m_{r,h}}^2 G^2 \end{aligned} \tag{18}$$

where F^{m*} is the local optimal value, R refers to the number of rounds for Job m during the execution, H represents the number of local iterations, and Q is the upper bound ratio between the learning rate at local iteration 0 and h', i.e., $\eta^m_{r,0} \leq Q \eta^m_{r,h'}$.

Proof. The proof can be found in Appendix. □

Then, we can get the following corollary:

Corollary 5.1.1. *When we choose $\eta^m_r = \frac{1}{L\sqrt{RH}}$ and $Q \leq$*

TABLE 4: The convergence accuracy and the time required to achieve the target accuracy for different methods in Group A. The numbers in parentheses represent the target accuracy, and "/" represents that the target accuracy is not achieved.

	Convergence Accuracy								Time (min)						
	Random	Genetic	FedCS	Greedy	BODS	RLDS	Meta-Greedy		Random	Genetic	FedCS	Greedy	BODS	RLDS	Meta-Greedy
Non-IID															
VGG	0.55	0.54	0.55	0.43	0.57	0.56	**0.577**	VGG (0.55)	2486	1164.3	1498.5	/	455.1	406.8	**291.3**
Cnn	**0.90**	0.80	0.80	0.83	**0.90**	0.897	**0.90**	Cnn (0.80)	44.25	95.85	27.39	43.04	15.88	17.6	**11.9**
LeNet	0.990	0.988	0.990	0.986	**0.991**	**0.991**	**0.991**	LeNet (0.984)	43.81	30.15	33.37	43.76	28.84	22.54	**12.37**
IID															
VGG	**0.610**	0.558	0.603	0.522	0.603	0.605	0.602	VGG (0.55)	126.9	231.4	87.5	/	57.7	43.81	**30.34**
Cnn	**0.943**	0.928	0.942	0.928	**0.943**	0.935	0.936	Cnn (0.930)	52.05	176.85	27.45	26.48	19.25	13.0	**12.88**
LeNet	0.9945	0.9928	0.9934	0.990	**0.9946**	**0.9946**	0.9936	LeNet (0.99)	14.94	6.03	7.12	17.98	5.18	4.02	**2.02**

TABLE 5: The convergence accuracy and the time required to achieve the target accuracy for different methods in Group B. The numbers in parentheses represent the target accuracy, and "/" represents that the target accuracy is not achieved.

	Convergence Accuracy								Time (min)						
	Random	Genetic	FedCS	Greedy	BODS	RLDS	Meta-Greedy		Random	Genetic	FedCS	Greedy	BODS	RLDS	Meta-Greedy
Non-IID															
ResNet	0.546	0.489	0.523	0.403	0.583	0.562	**0.590**	ResNet (0.50)	852.9	621.5	402.3	/	219.8	168.9	**149.8**
Cnn	0.824	0.767	0.823	0.764	0.836	0.830	**0.845**	Cnn (0.73)	47.1	22.0	18.5	70.8	13.8	15.5	**13.0**
AlexNet	0.989	0.986	0.987	0.871	0.990	0.990	**0.990**	AlexNet (0.976)	140.57	60.0	82.87	181.2	59.2	53.87	**45.0**
IID															
ResNet	0.787	0.754	0.782	0.743	0.791	0.785	**0.799**	ResNet (0.740)	65.93	32.51	31.4	52.93	15.9	12.82	**10.6**
Cnn	0.868	0.867	0.868	0.868	0.869	0.869	**0.871**	Cnn (0.867)	120.19	38.99	89.6	36.13	32.83	19.7	**16.13**
AlexNet	0.9938	0.9938	0.9939	0.9935	0.9939	**0.9943**	0.9940	AlexNet (0.9933)	35.08	19.44	20.97	/	21.65	12.9	**10.8**

TABLE 6: The time required to achieve the target accuracy for jobs executed sequentially with FedAvg. "*" indicates that it fails to achieve the target accuracy.

	non-IID/IID			non-IID/IID		
Job	VGG	CNN	LeNet	ResNet	CNN	AlexNet
Target Accuracy	0.55/0.55	0.80/0.93	0.984/0.99	0.50/0.74	0.73/0.867	0.976/0.9933
Time (min)	2483.4/133.3	53.1/45.5	50.5/18.01	897.2/*	35.8/322.6	115.8/65.16

$(RH)^{\frac{1}{4}}$, *we have:*

$$\begin{aligned}&\frac{1}{RH}\sum_{r=1}^{R}\sum_{h=1}^{H}\mathbb{E}\parallel\nabla F^m(\bar{w}_{r,h}^m)\parallel^2\\ &\leq\frac{2L}{\sqrt{RH}}(F^m(\bar{w}_{1,1}^m)-F^{m*})+\frac{1}{\sqrt{RH}}\sigma^2+\frac{1}{\sqrt{RH}}(H-1)^2G^2\\ &=\mathcal{O}(\frac{1}{\sqrt{RH}})\end{aligned}\tag{19}$$

We can find that, the training process of multi-job FL converges to a stationary point of $f(w^*)$ with a convergence rate of $\mathcal{O}(\frac{1}{\sqrt{RH}})$ for each job.

6 Experiments

In this section, we present the experimental results to show the efficiency of our proposed scheduling methods within MJ-FL. We compared the performance of Meta-Greedy, RLDS, and BODS with six baseline methods, i.e., Random [27], FedCS [28], Genetic [30], Greedy [40], Deep Neural Network (DNN), and Simulated Annealing (SA) [66].

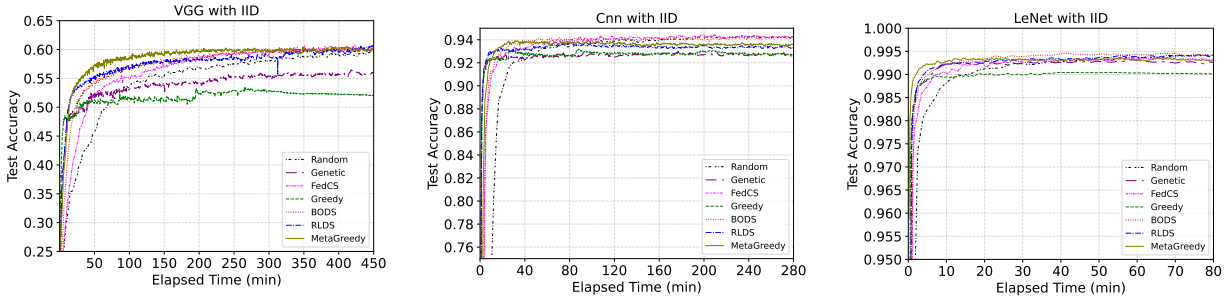
6.1 Federated Learning Setups

In the experiment, we take three jobs as a group to be executed in parallel. We carry out the experiments with two groups, i.e., Group A with VGG-16 (VGG) [67], CNN (CNN-A-IID and CNN-A-non-IID) [68], and LeNet-5 (LeNet) [68], and Group B with Resnet-18 (ResNet) [69], CNN (CNN-B) [68], and Alexnet [70], while each model corresponds to one job. The complexity of the models is as follows: AlexNet $<$ CNN-B $<$ ResNet and LeNet $<$ CNN (CNN-A-IID and CNN-A-non-IID) $<$ VGG. We exploit the datasets of Cifar-10 [71], emnist-letters [72], emnist-digital [72], Fashion-MNIST [73], and MNIST [68] in the training process.

CNN-A-IID comprises of two 3×3 convolution layers, one with 32 channels and the other with 64 channels. Each layer is followed by one batch normalization layer and 2×2 max pooling. Then, there are one flatten layer and three fully-connected layers (1568, 784, and 26 units) after the two convolution layers. In addition, we make a simple modification of CNN-A-IID to CNN-A-non-IID since the convergence behavior of CNN on non-IID in Group A is not satisfiable. CNN-A-non-IID consists of three 3×3 convolution layers (32, 64, 64 channels, each of them exploits ReLU activations, and each of the first two convolution layers is followed by 2×2 max pooling), followed by one flatten layer and two fully-connected layers (64, 26 units). CNN-B consists of two 2×2 convolution layers (64, 32 channels, each of them exploits ReLU activations) followed by a flatten layer and a fully-connected layer, and each convolution layer is followed by a dropout layer with 0.05. In addition, the other parameters are shown in Tables 2 and 3.

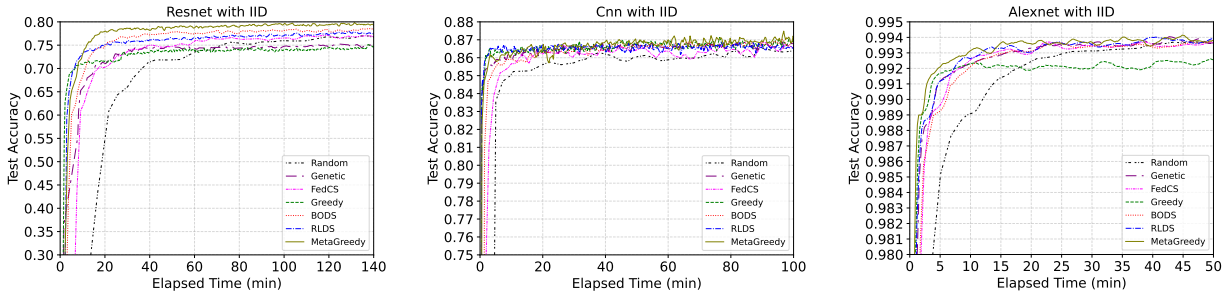
DNN comprises of a flatten layer and 40 hidden layers before an Alpha Dropout layer and a fully-connected layer. The hidden layers consist of 20 units of a fully-connected layer and a batch normalization layer.

For the non-IID setting of each dataset, the training set is classified by category, and the samples of each category



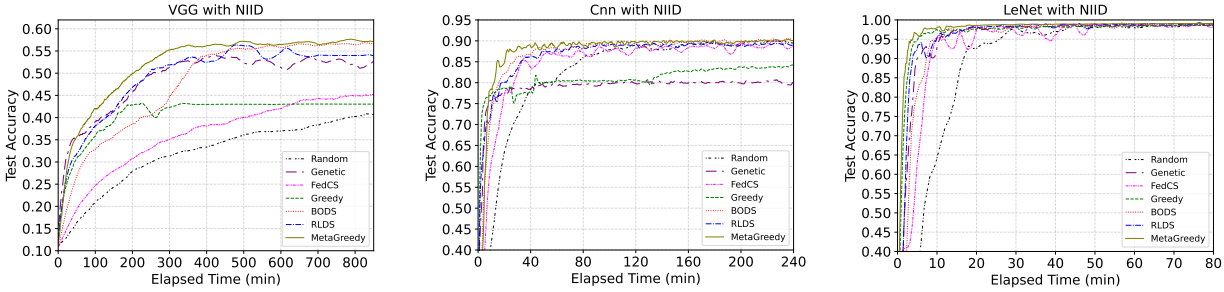
(a) (b) (c)

Fig. 4: The convergence accuracy of different jobs in Group A changes over time with the IID distribution.



(a) (b) (c)

Fig. 5: The convergence accuracy of different jobs in Group B changes over time with the IID distribution.



(a) (b) (c)

Fig. 6: The convergence accuracy of different jobs in Group A changes over time with the non-IID distribution.

are divided into 20 parts. Each device randomly selects two categories and then selects one part from each category to produce its local training set. For the IID setting, each device randomly samples a specified number of images from each training set. In addition, we use 12 Tesla V100 GPUs to simulate an FL environment composed of a parameter server and 100 devices. We exploit the execution time, including the training time and communication time, measured in real edge devices as the parameters in Formula 4 to simulate the real execution. We use Formula 4 to simulate the capabilities of devices in terms of training time with the uniform sampling strategy, while the accuracy is the results from the actual training processes. In the experimentation, we use corresponding target accuracy (for ease of comparison) in the place of target loss value.

6.2 Experimental Results

We first present the comparison between MJ-FL and Single-Job FL (SJ-FL). Then, we compare the proposed methods with baselines, e.g., Random [27], FedCS [28], Genetic [30], Greedy [40], DNN and SA, in both IID and non-IID settings. Afterward, we present the ablation experiments to show the impact of execution time and the data fairness in the cost model and the influence of $\Omega(r)$.

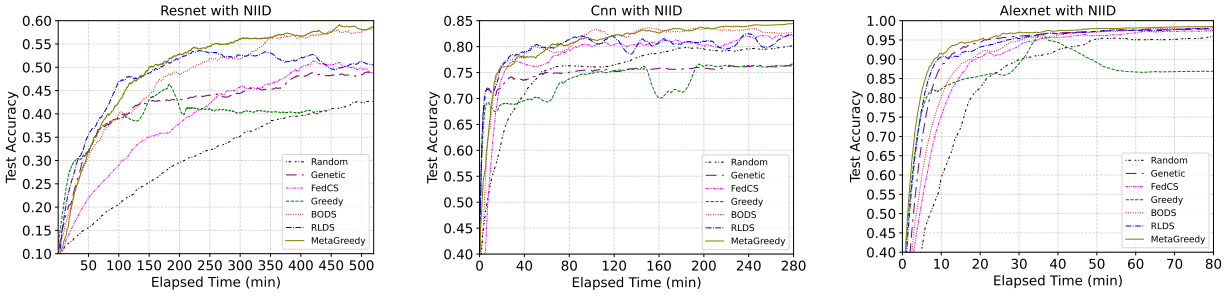
6.2.1 *Comparison with Single-Job FL*

In order to demonstrate the effectiveness of our proposed framework, i.e., MJ-FL, over the SJ-FL, we execute each group of jobs sequentially with FedAvg, which is denoted the Random method when adapted to multi-job FL. As shown in Tables 4, 5, and 6, MJ-FL outperforms SJ-FL (up to 1.68 times faster) with Random and the same accuracy. RLDS with MJ-FL outperforms Random with SJ-FL up to 15.38 times faster, and the advantage of BODS can be up to 8.83 times faster. Furthermore, Meta-Greedy can achieve much better performance, i.e., 19 times faster.

6.2.2 *Comparison within MJ-FL*

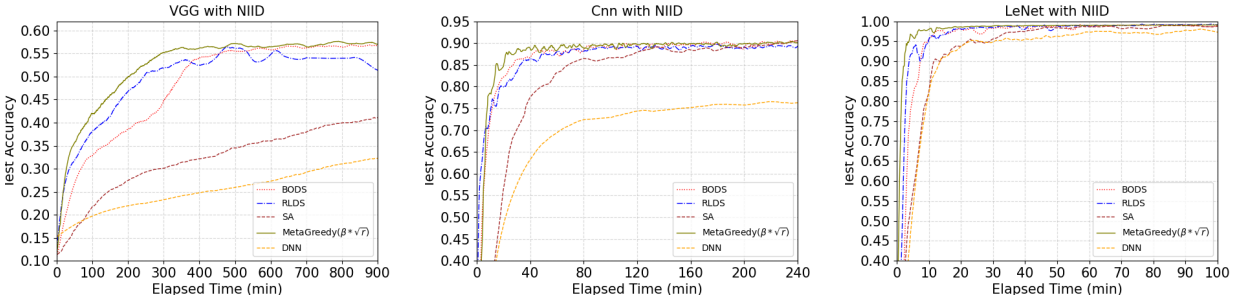
In this section, we present the experimental results with the IID setup and the non-IID setup.

Evaluation with the IID setting: As shown in Figures 4 and 5, the convergence speed of our proposed methods (i.e., BODS, RLDS and Meta-Greedy) is significantly faster

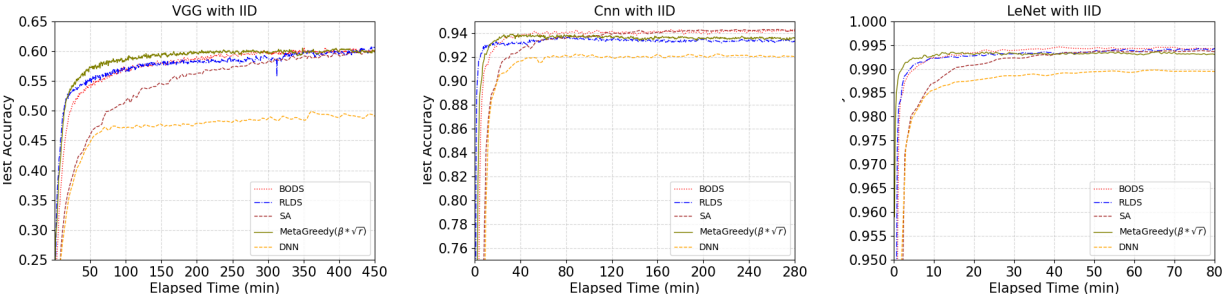


(a) (b) (c)

Fig. 7: The convergence accuracy of different jobs in Group B changes over time with the non-IID distribution.



(a) (b) (c)



(d) (e) (f)

Fig. 8: The convergence accuracy of different jobs changes over time compared with DNN and SA

than other methods. In addition, the convergence speed of RLDS has significant advantages in terms of both complex and simple jobs compared to BODS, while Meta-Greedy outperforms all others in terms of convergence speed. Besides, Meta-Greedy can achieve higher accuracy within less time compared to other methods for both complex jobs (ResNet in Figure 5(a)) and simple jobs (AlexNet in Figure 5(c)). Tables 4 and 5 reveal that IID data correspond to high convergence accuracy, and for VGG in IID setups, the final accuracy of our proposed methods, i.e., BODS (up to 15.52% compared to Greedy), RLDS (up to 15.90% compared to Greedy), and Meta-Greedy (up to 15.33% compared to Greedy), significantly surpasses that of the other methods. In addition, the training time for a target accuracy of our proposed methods is significantly shorter than baseline methods, including the time for an individual job, i.e., the training time of each job (up to 8.19 times faster for BODS, 12.6 times faster for RLDS, and 12.73 times faster for Meta-Greedy), and the time for the whole training process, i.e., the total time calculated according to Formula 7, (up to 4.04 times for BODS, 5.81 times for RLDS and 8.16 times for Meta-Greedy). Slightly different from Non-IID setups results, RLDS performs better than BODS for both complex and simple jobs in terms of the convergence speed, while Meta-Greedy still converges the fastest to the target accuracy.

Evaluation with the non-IID setting: When the decentralized data is of non-IID, the data fairness defined in Formula 5 has a significant impact on the accuracy. As shown in Figures 6 and 7, the convergence speed of our proposed methods, i.e., RLDS, BODS and Meta-Greedy, is significantly faster than other methods. RLDS shows a significant advantage for complex jobs (VGG in Figure 6(a)), while BODS shows advantage for relatively simple jobs in Groups A and B (please see details in Figures 6 and 7). This is reasonable as we have much more parameters to adjust for complex jobs with RLDS, e.g., the learning rate, the ϵ decay rate, the structure of the LSTM etc, which can fit into a complex jobs. However, it may be complicated to fine-tune the parameters for a simple job with RLDS. In contrary, the execution of simple jobs can be directly well addressed by the Bayesian optimization. Meta-Greedy can lead to a good performance for both simple and complex jobs. As shown in Tables 4 and 5, the final accuracy of RLDS, BODS and Meta-Greedy significantly outperforms other methods (up to 44.6% for BODS, 39.4% for RLDS and

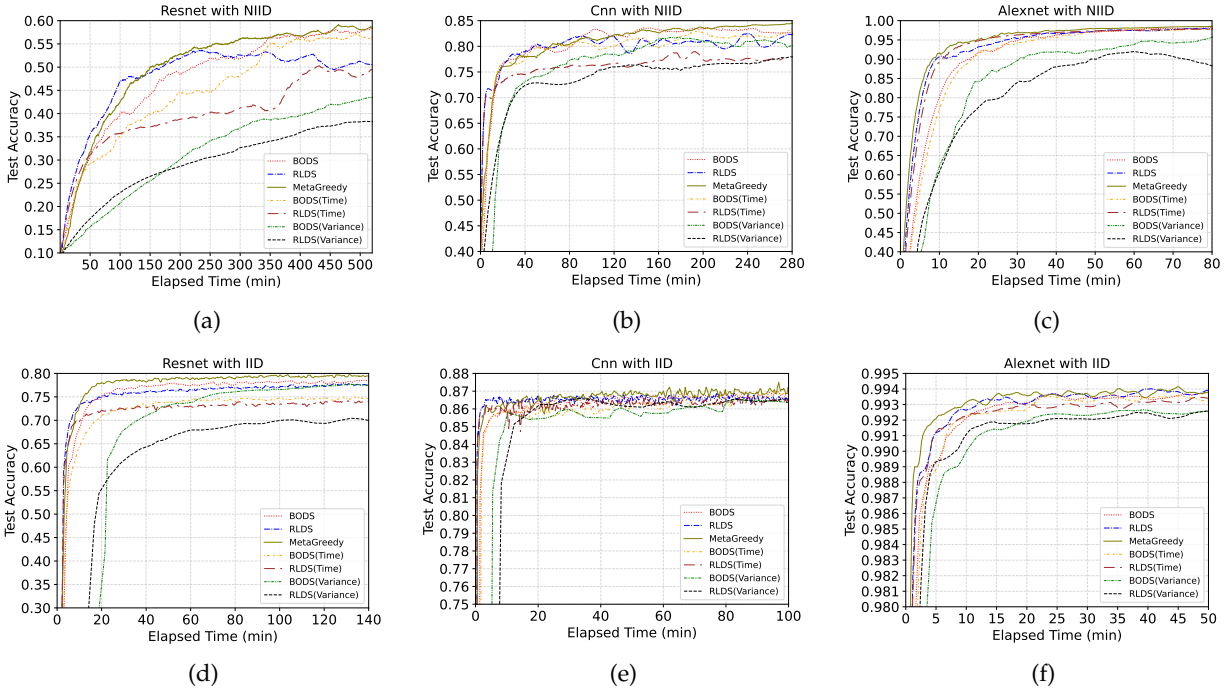


Fig. 9: The convergence accuracy of different jobs in Group B changes over time with the ablation setting, i.e., "time" represents the cost model with execution time and "variance" represents the cost model with data fairness.

46.4% for Meta-Greedy), as well. Given a target accuracy, our proposed methods can achieve the accuracy within a shorter time, compared with baseline methods, in terms of the time for a single job, i.e., the training time of each job (up to 5.04 times shorter for BODS, 5.11 times shorter for RLDS and 7.53 times shorter for Meta-Greedy), and the time for the whole training process, i.e., the total time calculated based on Formula 7 (up to 4.15 times for BODS, 4.67 times for RLDS and 7.16 times for Meta-Greedy), for Groups A and B. In addition, among our proposed methods, Meta-Greedy always has the shortest time to achieve the target accuracy. We have similar observations with IID, while the advantage of Meta-Greedy is much more significant (up to 19.0 times shorter in terms of the time for a single job) than that of non-IID as shown in Tables 4 and 5. Besides, we can find that our proposed method (i.e., Meta-Greedy) leads to a better performance in the case of non-IID data due to dynamically enhancing the impact of data fairness to decrease the imbalance of devices.

Comparison in other cases: As shown in Figures 5 and 7, the convergence speed corresponding to RLDS, BODS and Meta-Greedy is much higher than baseline methods with different target accuracy. The advantage of RLDS and BODS is up to 8.77 times faster for Target 1 (0.845), 13.96 times faster for Target 2 (0.856), and 27.04 times faster for Target 3 (0.865), compared with the baselines. In addition, Meta-Greedy outperforms the other six methods (four baselines and two scheduling methods, i.e., RLDS and BODS) up to 4.42 times faster for Target 1 (0.845), 6.65 times faster for Target 2 (0.856), and 5.61 times faster for Target 3 (0.865). Furthermore, According to Figure 8, SA [66] corresponds to much worse performance (up to 91.4% slower and 3.5% lower accuracy) compare with our methods. We carry out experiments to compare out methods with DNNs [42], the performance of which is significantly worse (up to 90.5% slower and 26.3% lower accuracy) than our methods. In addition, we test other combinations of the two costs, which correspond to worse performance (up to 37.1% slower and 3.5% lower accuracy for the sum of squared costs, and 64.4% slower and 3.3% lower accuracy for multiplication) compared to the linear one (Formula 2).

As RLDS can learn more information through a complex neural network, RLDS outperforms BODS for complex jobs (0.008 and 0.029 in terms of accuracy with VGG19 and ResNet18, and 46.7% and 34.8% faster for the target accuracy of 0.7 with VGG19 and 0.5 with ResNet18; see details in Appendix). Due to the emphasis on the combination of data fairness and device capabilities, i.e.,computation and communication capabilities, BODS can lead to high convergence accuracy and fast convergence speed for simple jobs (0.018 in terms of accuracy and 38% faster for the target accuracy of 0.97 with CNN; see details in Appendix). Meta-Greedy reconstructs the cost model based on six scheduling schemes and dynamically adjusts the parameters so that the impact of data fairness increases with the number of rounds, which leads to high convergence accuracy and fast convergence speed in both complex and simple jobs. BODS, RLDS and Meta-Greedy significantly outperform the baseline methods, while there are also differences among the four methods. The Greedy method prefers devices with high capacity, which leads to a significant decline in terms of the final convergence accuracy. The Genetic method can exploit randomness to achieve data fairness while generating scheduling plans, and the convergence performance is better than the Greedy method. The FedCS method optimizes the scheduling plan by randomly selecting devices, which improves the fairness of the device to a certain extent, and convergences faster than the Random method. We carry out

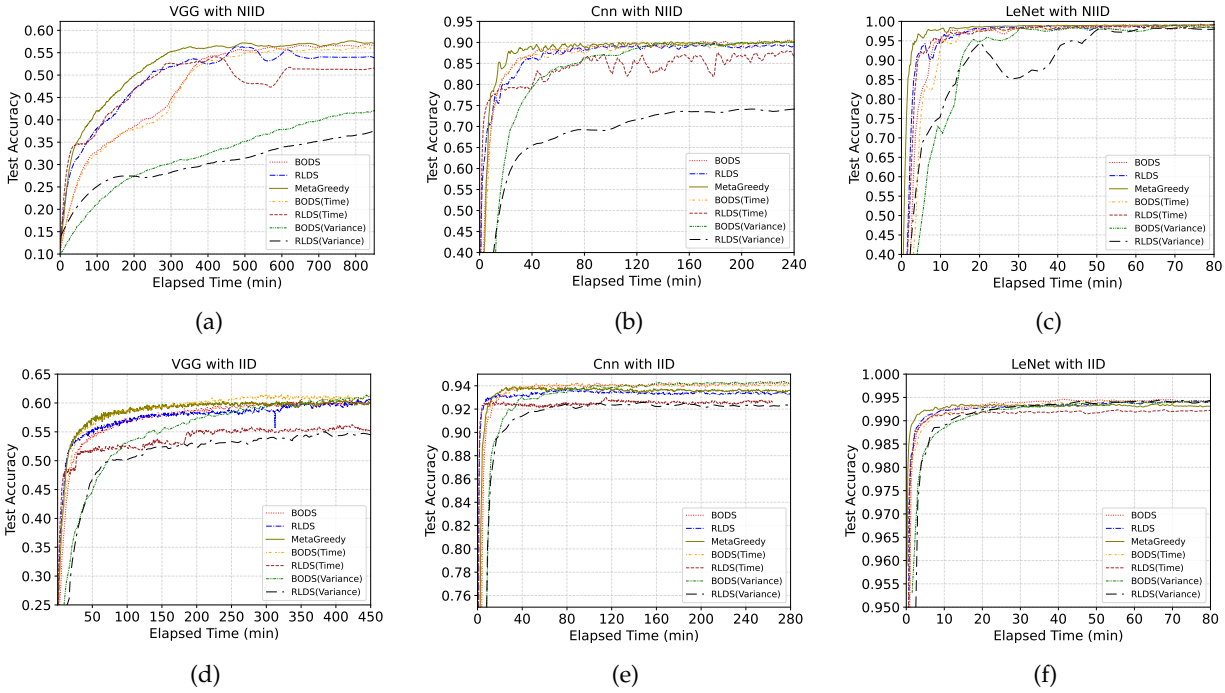


Fig. 10: The convergence accuracy of different jobs in Group A changes over time with the ablation setting, i.e., "time" represents the cost model with execution time and "variance" represents the cost model with data fairness.

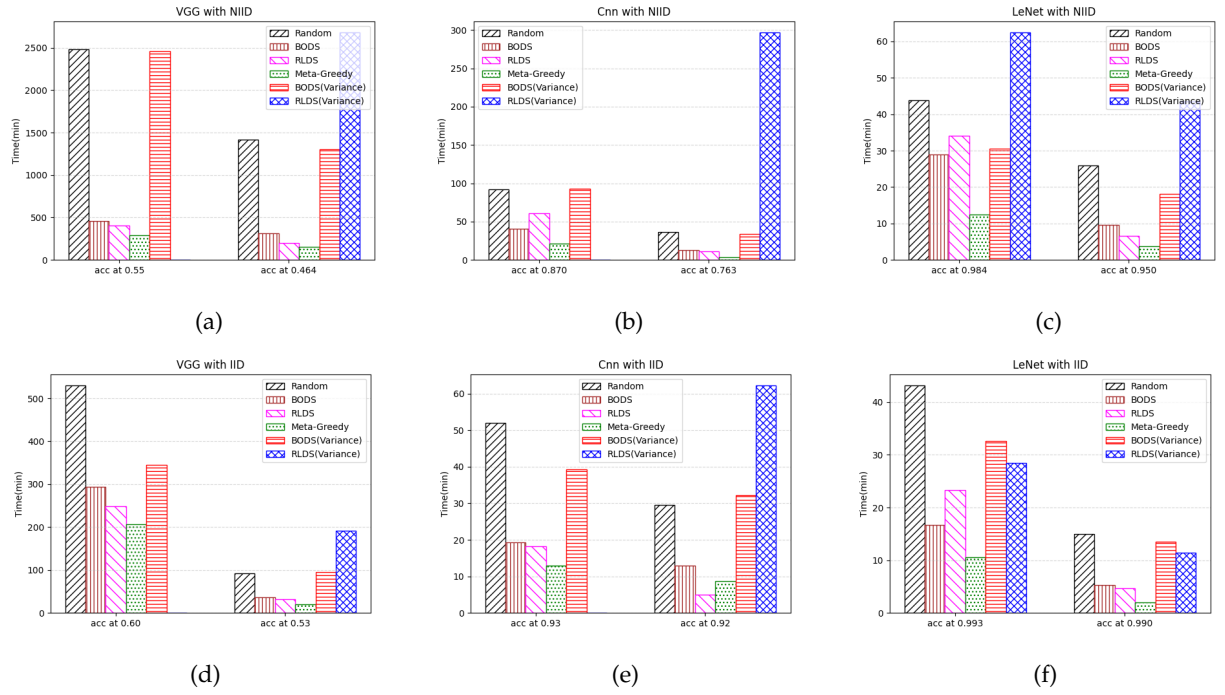


Fig. 11: The time required for each job of Group A to achieve the target convergence accuracy on the non-IID and IID distribution with the ablation setting, i.e., "time" represents the cost model with execution time and "variance" represents the cost model with data fairness.

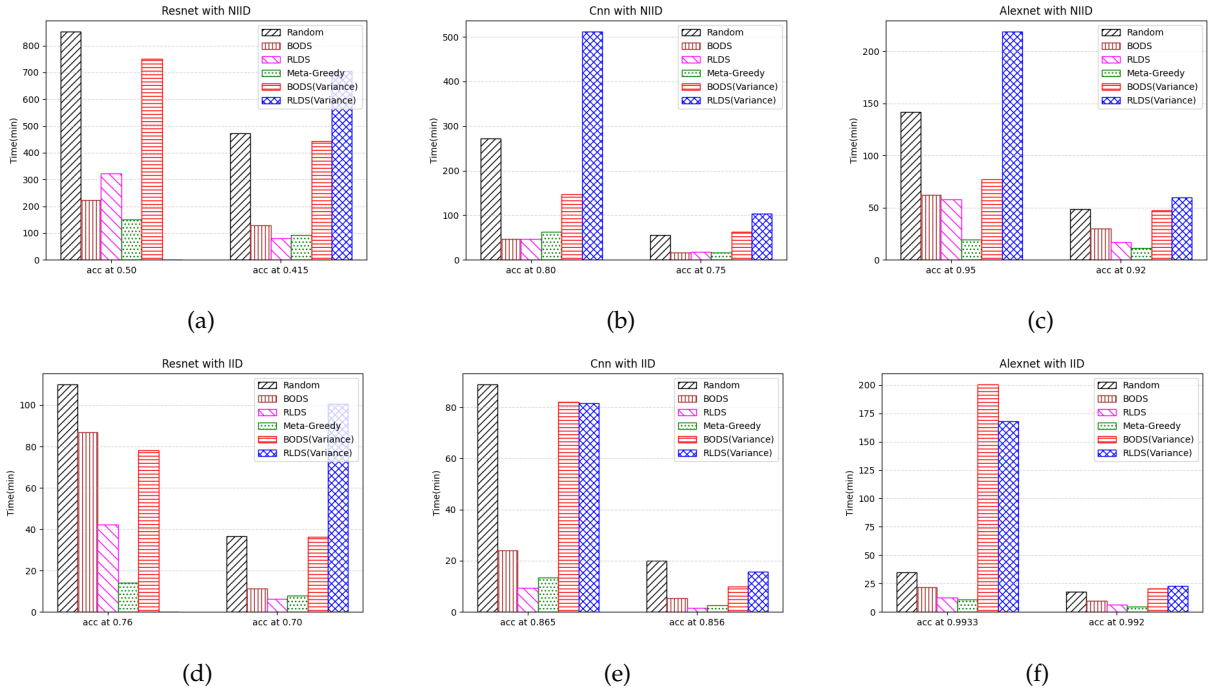


Fig. 12: The time required for each job of Group B to achieve the target convergence accuracy on the non-IID and IID distribution with the ablation setting, i.e., "time" represents the cost model with execution time and "variance" represents the cost model with data fairness.

an experiment with Group A with both IID and non-IID distribution, and find that Meta-Greedy with 6 methods significantly outperforms that with 2 methods in terms of accuracy (up to 0.091) and training speed (up to 65%) (see details in Appendix). In addition, after analyzing the log of the training process (see details in Appendix), we find Greedy and Genetic are extensively exploited, RLDS is selected at the beginning of the training process, BODS is chosen at the end of the training process, FedCS participates with less frequency, and Random is seldomly utilized. As Meta-Greedy can intelligently select a proper scheduling plan based on available methods, it results in a more efficient training process.

6.2.3 Ablation Study

In this section, we first present the ablation study to show the impact of the execution time and the data fairness. Then, we analyze the influence of $\Omega(r)$.

Impact of data fairness: As shown in Figures 10 and 9, we conduct ablation experiments and find an evident decline in convergence accuracy when the cost model is only composed of execution time cost. From Figures 10 and 9, we can see that the accuracy of the cost model composed of execution time (i.e., BODS (Time) and RLDS (Time)) is worse than that of both data fairness and execution time (i.e., BODS and RLDS) in most cases (up to 35.83% lower for RLDS (Time)). The abnormal case, when the convergence accuracy of BODS (Time) is slightly higher (about 1%) than the convergence accuracy of BODS (please see details in Figure 10 (d)) should be caused by randomness. Besides, from the Figures 10 and 9, we can find that the impact of execution time on complex jobs (VGG in Figure 10 and Resnet in Figure 9) is significantly greater than that on simple jobs (LeNet in Figure 10 and AlexNet in Figure 9).

Impact of execution time: As shown in Figures 10 and 9, the data fairness improves both the convergence speed (up to 9.35 times faster) and the accuracy (up to 15.3%). From Figures 11 and 12, we can find that the convergence speed significantly decreases (up to 25.46 times slower for RLDS (Variance) compared with RLDS) when given a target accuracy with the cost model composed of only data fairness. In most cases, considering data fairness reduces the convergence speed and essentially does not affect the convergence accuracy. There are few cases that the job fails to achieve the convergence accuracy. In addition, from the results shown in Figures 10 and 9, we find that the convergence accuracy in complex jobs (VGG in Figure 11 and ResNet in Figure 12) are more likely to decline when the cost model is only composed of data fairness, compared with that with simple jobs. Besides, the performance corresponding to the cost model with only data fairness is close to that of the 'Random' method in most setups.

Influence of $\Omega(r)$: As shown in Figures of Appendix, Meta-Greedy with $\Omega(r) = \sqrt{r}$ outperforms other methods, i.e., $\Omega(r) = r$ and $\Omega(r) = \log r$, in terms of both convergence accuracy and convergence speed. Although the convergence accuracy of Meta-Greedy with $\Omega(r) = r$ is slightly higher than that with $\Omega(r) = \sqrt{r}$ in a few experiments, the convergence accuracy of Meta-Greedy with $\Omega(r) = \sqrt{r}$ in complex jobs is significantly lower than that with $\Omega(r) = r$. In addition, Meta-Greedy with $\Omega(r) = \sqrt{r}$ takes less time to reach the target accuracy in previous rounds compared with Meta-Greedy with $\Omega(r) = r$. This indicates that when round r is linearly related to $\Omega(r)$, the data fairness in later rounds influences the cost model too much and may lead to a lower convergence accuracy. Therefore, we avoid using $\Omega(r) = r$ due to the dramatic changes of the magnitude in the linear relationship. Meta-Greedy with $\Omega(r) = \sqrt{r}$

takes less time to reach the target accuracy in previous rounds compared with Meta-Greedy with $\Omega(r) = \log r$. In the meanwhile, the convergence accuracy of Meta-Greedy with $\Omega(r) = \log r$ is lower than that of Meta-Greedy with $\Omega(r) = \sqrt{r}$, which implies that the relatively gentle variation of $\Omega(r)$ with round r leads to lower convergence accuracy and slower convergence speed. By contrast, Meta-Greedy with $\Omega(r) = \sqrt{r}$ performs best among them in terms of both convergence accuracy and convergence speed. Thus, we adopt Meta-Greedy with $\Omega(r) = \sqrt{r}$, which dynamically changes the influence of data fairness to obtain relatively optimal solution in both complex and simple jobs.

7 Conclusion

In this work, we proposed a new Multi-Job Federated Learning framework, i.e., MJ-FL. The framework is composed of a system model and three device scheduling methods. The system model is composed of a process for the parallel execution of multiple jobs and a cost model based on the capability of devices and data fairness. We proposed three device scheduling methods, i.e., RLDS for complex jobs and BODS for simple jobs, while Meta-Greedy for both complex and simple jobs, to efficiently schedule proper devices for each job based on the cost model. We carried out extensive experimentation with six real-life models and four datasets with IID and non-IID distribution. The experimental results show that MJ-FL outperforms the single-job FL, and that our proposed scheduling methods, i.e., BODS and RLDS, significantly outperform baseline methods (up to 44.6% in terms of accuracy, 12.6 times faster for a single job and 5.81 times faster for the total time). In addition, Meta-Greedy, the intelligent scheduling approach based on multiple scheduling methods (including the two proposed methods, i.e., RODS and BODS) significantly outperforms other methods (up to 46.4% in terms of accuracy, 12.73 times faster for a single job and 8.16 times faster for the total time).

References

[1] J. Liu, J. Huang, Y. Zhou, X. Li, S. Ji, H. Xiong, and D. Dou, "From distributed machine learning to federated learning: a survey," *Knowledge and Information Systems*, vol. 64, no. 4, pp. 885–917, 2022.

[2] Official Journal of the European Union, "General data protection regulation," https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679, 2016, online; accessed 12/02/2021.

[3] "Cybersecurity law of the people's republic of china," https://www.newamerica.org/cybersecurity-initiative/digichina/blog/translation-cybersecurity-law-peoples-republic-china/, 2018, online; accessed 22/02/2021.

[4] "California consumer privacy act home page," https://www.caprivacy.org/, 2018, online; accessed 14/02/2021.

[5] W. B. Chik, "The singapore personal data protection act and an assessment of future trends in data privacy reform," *Computer Law & Security Review*, vol. 29, no. 5, pp. 554–575, 2013.

[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2017, pp. 1273–1282.

[7] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.

[8] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[9] A. Smola and S. Narayanamurthy, "An architecture for parallel topic models," *Very Large Data Bases Conference (VLDB) Endowment*, vol. 3, no. 1-2, pp. 703–710, 2010.

[10] J. Liu, D. Dong, X. Wang, A. Qin, X. Li, P. Valduriez, D. Dou, and D. Yu, "Large-scale knowledge distillation with elastic heterogeneous computing resources," *Concurrency and Computation: Practice and Experience*, pp. 1–16, 2022.

[11] J. Liu, Z. Wu, D. Yu, Y. Ma, D. Feng, M. Zhang, X. Wu, X. Yao, and D. Dou, "Heterps: Distributed deep learning with reinforcement learning based scheduling in heterogeneous environments," *arXiv preprint arXiv:2111.10635*, pp. 1–14, 2021.

[12] L. L. Pilla, "Optimal task assignment for heterogeneous federated learning devices," in *IEEE Int. Parallel and Distributed Processing Symposium (IPDPS)*, 2021, pp. 661–670.

[13] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," in *Annual Conf. on Neural Information Processing Systems (NeurIPS)*, 2017, p. 4427–4437.

[14] J. Han, M. M. Rafique, L. Xu, A. R. Butt, S.-H. Lim, and S. S. Vazhkudai, "Marble: A multi-gpu aware job scheduler for deep learning on hpc systems," in *IEEE/ACM Int. Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, 2020, pp. 272–281.

[15] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[16] L. Li, D. Shi, R. Hou, H. Li, M. Pan, and Z. Han, "To talk or to work: Flexible communication compression for energy efficient federated learning over heterogeneous mobile edge devices," in *IEEE Conf. on Computer Communications (INFOCOM)*, 2021, pp. 1–10.

[17] W. Zhao, J. Zhang, D. Xie, Y. Qian, R. Jia, and P. Li, "Aibox: CTR prediction model training on a single node," in *ACM Int. Conf. on Information and Knowledge Management, (CIKM)*, 2019, pp. 319–328.

[18] H. Guo, W. Guo, Y. Gao, R. Tang, X. He, and W. Liu, *ScaleFreeCTR: MixCache-Based Distributed Training System for CTR Models with Huge Embedding Table*. Association for Computing Machinery, 2021, p. 1269–1278.

[19] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[20] X. Fang, J. Huang, F. Wang, L. Liu, Y. Sun, and H. Wang, "Ssml: Self-supervised meta-learner for en route travel time estimation at baidu maps," in *ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining (SIGKDD)*, 2021, pp. 2840–2848.

[21] C. Qin, H. Zhu, C. Zhu, T. Xu, F. Zhuang, C. Ma, J. Zhang, and H. Xiong, "Duerquiz: A personalized question recommender system for intelligent job interview," in *ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining (SIGKDD)*, 2019, pp. 2165–2173.

[22] M. Masterson, "Baidu's deep speech recognition bests google, apple, and bing," *Speech Technology Magazine*, vol. 20, no. 1, pp. 12–13, 2015.

[23] T. Pitoura and P. Triantafillou, "Load distribution fairness in p2p data management systems," in *IEEE Int. Conf. on Data Engineering (ICDE)*, 2007, pp. 396–405.

[24] A. Finkelstein, M. Harman, S. A. Mansouri, J. Ren, and Y. Zhang, ""fairness analysis" in requirements assignments," in *IEEE Int. Requirements Engineering Conf.*, 2008, pp. 115–124.

[25] J. Du and J. Y.-T. Leung, "Complexity of scheduling parallel task systems," *SIAM Journal on Discrete Mathematics*, vol. 2, no. 4, p. 473–487, 1989.

[26] L. Liu, H. Yu, G. Sun, L. Luo, Q. Jin, and S. Luo, "Job scheduling for distributed machine learning in optical wan," *Future Generation Computer Systems (FGCS)*, vol. 112, pp. 549–560, 2020.

[27] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[28] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *IEEE Int. Conf. on Communications (ICC)*, 2019, pp. 1–7.

[29] S. Abdulrahman, H. Tout, A. Mourad, and C. Talhi, "Fedmccs: Multicriteria client selection model for optimal iot federated learning," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4723–4735, 2021.

[30] M. Barika, S. Garg, A. Chan, and R. Calheiros, "Scheduling algorithms for efficient execution of stream workflow applications in multicloud environments," *IEEE trans. on Services Computing*, 2019.

[31] C. Zhou, J. Liu, J. Jia, J. Zhou, Y. Zhou, H. Dai, and D. Dou, "Efficient device scheduling with multi-job federated learning," *AAAI Conf. on Artificial Intelligence*, 2022, to appear.

[32] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecný, S. Mazzocchi, B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," in *Machine Learning and Systems (MLSys)*, 2019.

[33] Y. Liu, A. Huang, Y. Luo, H. Huang, Y. Liu, Y. Chen, L. Feng, T. Chen, H. Yu, and Q. Yang, "Fedvision: An online visual object detection platform powered by federated learning," in *AAAI Conf. on Artificial Intelligence*, vol. 34, no. 08, 2020, pp. 13 172–13 179.

[34] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, N. Hoang, and Y. Khazaeni, "Bayesian nonparametric federated learning of neural networks," in *Int. Conf. on Machine Learning (ICML)*, 2019, pp. 7252–7261.

[35] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," in *Int. Conf. on Learning Representations (ICLR)*, 2020.

[36] H. Zhang, J. Liu, J. Jia, Y. Zhou, and H. Dai, "Fedduap: Federated learning with dynamic update and adaptive pruning using shared data on the server," in *Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2022, pp. 1–7, to appear.

[37] J. Jin, J. Ren, Y. Zhou, L. Lv, J. Liu, and D. Dou, "Accelerated federated learning with decoupled adaptive optimization," in *Int. Conf. on Machine Learning (ICML)*, vol. 162, 2022, pp. 10 298–10 322.

[38] D. Chen, C. S. Hong, L. Wang, Y. Zha, Y. Zhang, X. Liu, and Z. Han, "Matching theory based low-latency scheme for multi-task federated learning in mec networks," *IEEE Internet of Things Journal*, 2021.

[39] P. Sun, Z. Guo, J. Wang, J. Li, J. Lan, and Y. Hu, "Deepweave: Accelerating job completion time with deep reinforcement learning-based coflow scheduling." in *Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2020, pp. 3314–3320.

[40] W. Shi, S. Zhou, and Z. Niu, "Device scheduling with fast convergence for wireless federated learning," in *IEEE Int. Conf. on Communications (ICC)*, 2020, pp. 1–6.

[41] K.-r. Kim, Y. Kim, and S. Park, "A probabilistic machine learning approach to scheduling parallel loops with bayesian optimization," *IEEE trans. on Parallel and Distributed Systems (TPDS)*, vol. 32, no. 7, pp. 1815–1827, 2020.

[42] Z. Zang, W. Wang, Y. Song, L. Lu, W. Li, Y. Wang, and Y. Zhao, "Hybrid deep neural network scheduler for job-shop problem based on convolution two-dimensional transformation," *Computational intelligence and neuroscience*, vol. 2019, 2019.

[43] Q. Chen, Z. Zheng, C. Hu, D. Wang, and F. Liu, "On-edge multi-task transfer learning: Model and practice with data-driven task allocation," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 31, no. 6, pp. 1357–1371, 2019.

[44] M. K. Emani and M. O'Boyle, "Celebrating diversity: A mixture of experts approach for runtime mapping in dynamic environments," in *ACM SIGPLAN Conf. on Programming Language Design and Implementation*, vol. 50, no. 6, 2015, pp. 499–508.

[45] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2014, pp. 583–598.

[46] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *Int. Conf. on Learning Representations (ICLR)*, 2020.

[47] F. Zhou and G. Cong, "On the convergence properties of a k-step averaging stochastic gradient descent algorithm for nonconvex optimization," in *Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 7 2018, pp. 3219–3227.

[48] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Int. Conf. on the theory and applications of cryptographic techniques*, 1999, pp. 223–238.

[49] C. Dwork, "Differential privacy: A survey of results," in *Int. conf. on theory and applications of models of computation*, 2008, pp. 1–19.

[50] W. Shi, S. Zhou, Z. Niu, M. Jiang, and L. Geng, "Joint device scheduling and resource allocation for latency constrained wireless federated learning," *IEEE Trans. on Wireless Communications*, vol. 20, no. 1, pp. 453–467, 2021.

[51] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. on Information Theory*, vol. 64, no. 3, pp. 1514–1529, 2018.

[52] S. Petrangeli, M. Claeys, S. Latré, J. Famaey, and F. De Turck, "A multi-agent q-learning-based framework for achieving fairness in http adaptive streaming," in *IEEE Network Operations and Management Symposium (NOMS)*, 2014, pp. 1–9.

[53] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–35, 2021.

[54] Y. Peng, Y. Bao, Y. Chen, C. Wu, and C. Guo, "Optimus: an efficient dynamic resource scheduler for deep learning clusters," in *EuroSys Conf.*, 2018, pp. 1–14.

[55] P. Toth, "Optimization engineering techniques for the exact solution of np-hard combinatorial optimization problems," *European Journal of Operational Research (EJOR)*, vol. 125, no. 2, pp. 222–238, 2000.

[56] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," in *Int. Conf. on Machine Learning (ICML)*, 2010, pp. 1015–1022.

[57] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[58] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2, no. 3.

[59] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[60] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing clusters," in *ACM Special Interest Group on Data Communication (SIGCOMM)*, J. Wu and W. Hall, Eds. ACM, 2019, pp. 270–288.

[61] Z. Xia and D. Zhao, "Online reinforcement learning by bayesian inference," in *Int. Joint Conf. on Neural Networks (IJCNN)*, 2015, pp. 1–6.

[62] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[63] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Int. Conf. on Learning Representations (ICLR)*, 2017.

[64] G. Li, Y. Hu, M. Zhang, J. Liu, Q. Yin, Y. Peng, and D. Dou, "Fedhisyn: A hierarchical synchronous federated learning framework for resource and data heterogeneity," in *Int. Conf. on Parallel Processing (ICPP)*, 2022, pp. 1–10, to appear.

[65] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," *arXiv preprint arXiv:2003.00295*, 2020.

[66] P. J. Van Laarhoven and E. H. Aarts, "Simulated annealing," in *Simulated annealing: Theory and applications*, 1987, pp. 7–15.

[67] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Int. Conf. on Learning Representations (ICLR)*, 2015.

[68] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[69] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[70] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Annual Conf. on Neural Information Processing Systems (NeurIPS)*, 2012, pp. 1106–1114.

[71] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," University of Toronto, Toronto, Ontario, Tech. Rep., 2009.

[72] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "Emnist: Extending mnist to handwritten letters," in *Int. Joint Conf. on Neural Networks (IJCNN)*, 2017, pp. 2921–2926.

[73] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[74] Baidu, "Baidu ai cloud," https://login.bce.baidu.com/, 2022, online; accessed 18/11/2022.

APPENDIX

Proof of Theorem 1

For simplicity, we take $\nabla f_k^m(w_{r,h}^{m,k})$ instead of $\nabla f_k^m(w_{r,h}^{m,k};\zeta_{k,h}^{m,r})$ in the proof. By the smoothness of F^m, we have

$$\begin{aligned}&\mathbb{E}[F^m(\bar{w}_{r,h+1}^m)]\\&\leq \mathbb{E}[F^m(\bar{w}_{r,h}^m)] + \underbrace{\mathbb{E}[\langle \nabla F^m(\bar{w}_{r,h}^m), \bar{w}_{r,h+1}^m - \bar{w}_{r,h}^m\rangle]}_{A}\\&\quad + \frac{L}{2}\underbrace{\mathbb{E}[\parallel \bar{w}_{r,h+1}^m - \bar{w}_{r,h}^m \parallel^2]}_{B}\end{aligned} \tag{20}$$

Where for B we have, $\eta_{r,h}^m$ is learning rate.

$$\begin{aligned}&\mathbb{E}[\parallel \bar{w}_{r,h+1}^m - \bar{w}_{r,h}^m \parallel^2]\\&= \mathbb{E} \parallel \eta_{r,h}^m \sum_{k\in V_m^r} p_{k,r}^m g_{r,h}^{m,k} \parallel^2\\&= {\eta_{r,h}^m}^2 \mathbb{E} \parallel \sum_{k\in V_m^r} p_{k,r}^m g_{r,h}^{m,k} \parallel^2\\&= {\eta_{r,h}^m}^2 \mathbb{E} \parallel \sum_{k\in V_m^r} p_{k,r}^m (g_{r,h}^{m,k} - \nabla F_k^m(w_{r,h}^{m,k})) \parallel^2\\&\quad + {\eta_{r,h}^m}^2 \mathbb{E} \parallel \sum_{k\in V_m^r} p_{k,r}^m \nabla F_k^m(w_{r,h}^{m,k})) \parallel^2\\&\leq {\eta_{r,h}^m}^2 \sum_{k\in V_m^r} p_{k,r}^m \mathbb{E} \parallel g_{r,h}^{m,k} - \nabla F_k^m(w_{r,h}^{m,k}) \parallel^2\\&\quad + {\eta_{r,h}^m}^2 \mathbb{E} \parallel \sum_{k\in V_m^r} p_{k,r}^m \nabla F_k^m(w_{r,h}^{m,k})) \parallel^2\\&\leq {\eta_{r,h}^m}^2 \sum_{k\in V_m^r} p_{k,r}^m \sigma^2 + {\eta_{r,h}^m}^2 \mathbb{E} \parallel \sum_{k\in V_m^r} p_{k,r}^m \nabla F_k^m(w_{r,h}^{m,k})) \parallel^2\end{aligned} \tag{21}$$

where the first inequality results from Jensen's inequality. The last inequality results from Assumption 3. Next, for A we have

$$\begin{aligned}&\mathbb{E}[\langle \nabla F^m(\bar{w}_{r,h}^m), \bar{w}_{r,h+1}^m - \bar{w}_{r,h}^m\rangle]\\&= \mathbb{E}[\langle \nabla F^m(\bar{w}_{r,h}^m), -\eta_{r,h}^m \sum_{k\in V_m^r} p_{k,r}^m g_{r,h}^{m,k}\rangle]\\&= -\eta_{r,h}^m \mathbb{E}[\langle \nabla F^m(\bar{w}_{r,h}^m), \sum_{k\in V_m^r} p_{k,r}^m \nabla F_k^m(w_{r,h}^{m,k})\rangle]\\&= -\frac{\eta_{r,h}^m}{2}\mathbb{E}[\parallel \nabla F^m(\bar{w}_{r,h}^m) \parallel^2 + \parallel \sum_{k\in V_m^r} p_{k,r}^m \nabla F_k^m(w_{r,h}^{m,k}) \parallel^2\\&\quad - \parallel \nabla F^m(\bar{w}_{r,h}^m) - \sum_{k\in V_m^r} p_{k,r}^m \nabla F_k^m(w_{r,h}^{m,k}) \parallel^2]\\&= -\frac{\eta_{r,h}^m}{2}\mathbb{E} \parallel \nabla F^m(\bar{w}_{r,h}^m) \parallel^2\\&\quad - \frac{\eta_{r,h}^m}{2}\mathbb{E} \parallel \sum_{k\in V_m^r} p_{k,r}^m \nabla F_k^m(w_{r,h}^{m,k}) \parallel^2\\&\quad + \frac{\eta_{r,h}^m}{2} \parallel \nabla F^m(\bar{w}_{r,h}^m) - \sum_{k\in V_m^r} p_{k,r}^m \nabla F_k^m(w_{r,h}^{m,k}) \parallel^2\end{aligned}$$

where the fourth equality results from the basic identity $\langle a, b\rangle = \frac{1}{2}(\parallel a \parallel^2 + \parallel b \parallel^2 - \parallel a - b \parallel^2)$.

Then, combine A and B together

$$\begin{aligned}&\mathbb{E}[F^m(\bar{w}_{r,h+1}^m)]\\&\leq \mathbb{E}[F^m(\bar{w}_{r,h}^m)] - \frac{\eta_{r,h}^m}{2}\mathbb{E} \parallel \nabla F^m(\bar{w}_{r,h}^m) \parallel^2\\&\quad - \frac{\eta_{r,h}^m}{2}\mathbb{E} \parallel \sum_{k\in V_m^r} p_{k,r}^m \nabla F_k^m(w_{r,h}^{m,k}) \parallel^2 + \frac{L}{2}{\eta^m}^2 \sum_{k\in V_m^r} p_m^k \sigma^2\\&\quad + \frac{\eta_{r,h}^m}{2} \parallel \nabla F^m(\bar{w}_{r,h}^m) - \sum_{k\in V_m^r} p_{k,r}^m \nabla F_k^m(w_{r,h}^{m,k}) \parallel^2\\&\quad + \frac{{\eta^m}^2 L}{2}\mathbb{E} \parallel \sum_{k\in V_m^r} p_{k,r}^m \nabla F_k^m(w_{r,h}^{m,k})) \parallel^2\\&= \mathbb{E}[F^m(\bar{w}_{r,h}^m)] - \frac{\eta_{r,h}^m}{2}\mathbb{E} \parallel \nabla F^m(\bar{w}_{r,h}^m) \parallel^2 + \frac{L}{2}{\eta^m}^2 \sum_{k\in V_m^r} p_m^k \sigma^2\\&\quad - \frac{\eta_{r,h}^m - {\eta_{r,h}^m}^2 L}{2}\underbrace{\mathbb{E} \parallel \sum_{k\in V_m^r} p_{k,r}^m \nabla F_k^m(w_{r,h}^{m,k}) \parallel^2}_{C}\\&\quad + \frac{1}{2}\eta_{r,h}^m \underbrace{\mathbb{E} \parallel \nabla F^m(\bar{w}_{r,h}^m) - \sum_{k\in V_m^r} p_{k,r}^m \nabla F_k^m(w_{r,h}^{m,k}) \parallel^2}_{D}\end{aligned} \tag{22}$$

We take $0 < \eta_{r,h}^m \leq \frac{1}{L}$. As C in (22) is positive, we have

$$\begin{aligned}&\mathbb{E}[F^m(\bar{w}_{r,h+1}^m)]\\&\leq \mathbb{E}[F^m(\bar{w}_{r,h}^m)] - \frac{\eta_{r,h}^m}{2}\mathbb{E} \parallel \nabla F^m(\bar{w}_{r,h}^m) \parallel^2 + \frac{L}{2}{\eta^m}^2 \sum_{k\in V_m^r} p_m^k \sigma^2\\&\quad + \frac{\eta_{r,h}^m}{2}\underbrace{\mathbb{E} \parallel \nabla F^m(\bar{w}_{r,h}^m) - \sum_{k\in V_m^r} p_{k,r}^m \nabla F_k^m(w_{r,h}^{m,k}) \parallel^2}_{D}\end{aligned} \tag{23}$$

For D we have,

$$\begin{aligned}&\mathbb{E} \parallel \nabla F^m(\bar{w}_{r,h}^m) - \sum_{k\in V_m^r} p_{k,r}^m \nabla F_k^m(w_{r,h}^{m,k}) \parallel^2\\&= \mathbb{E} \parallel \sum_{k\in V_m^r} p_{k,r}^m \nabla F_k^m(\bar{w}_{r,h}^m) - \sum_{k\in V_m^r} p_{k,r}^m \nabla F_k^m(w_{r,h}^{m,k}) \parallel^2\\&= \mathbb{E} \parallel \sum_{k\in V_m^r} p_{k,r}^m (\nabla F_k^m(\bar{w}_{r,h}^m) - \nabla F_k^m(w_{r,h}^{m,k})) \parallel^2\\&\leq \mathbb{E}[\sum_{k\in V_m^r} p_{k,r}^m \parallel \nabla F_k^m(\bar{w}_{r,h}^m) - \nabla F_k^m(w_{r,h}^{m,k}) \parallel^2]\\&\leq L^2 \mathbb{E} \sum_{k\in V_m^r} p_{k,r}^m \parallel \bar{w}_{r,h}^m - w_{r,h}^{m,k} \parallel^2\end{aligned} \tag{24}$$

where the first inequality results from Jensen's inequality. The second inequality results from Assumption 1. Then we

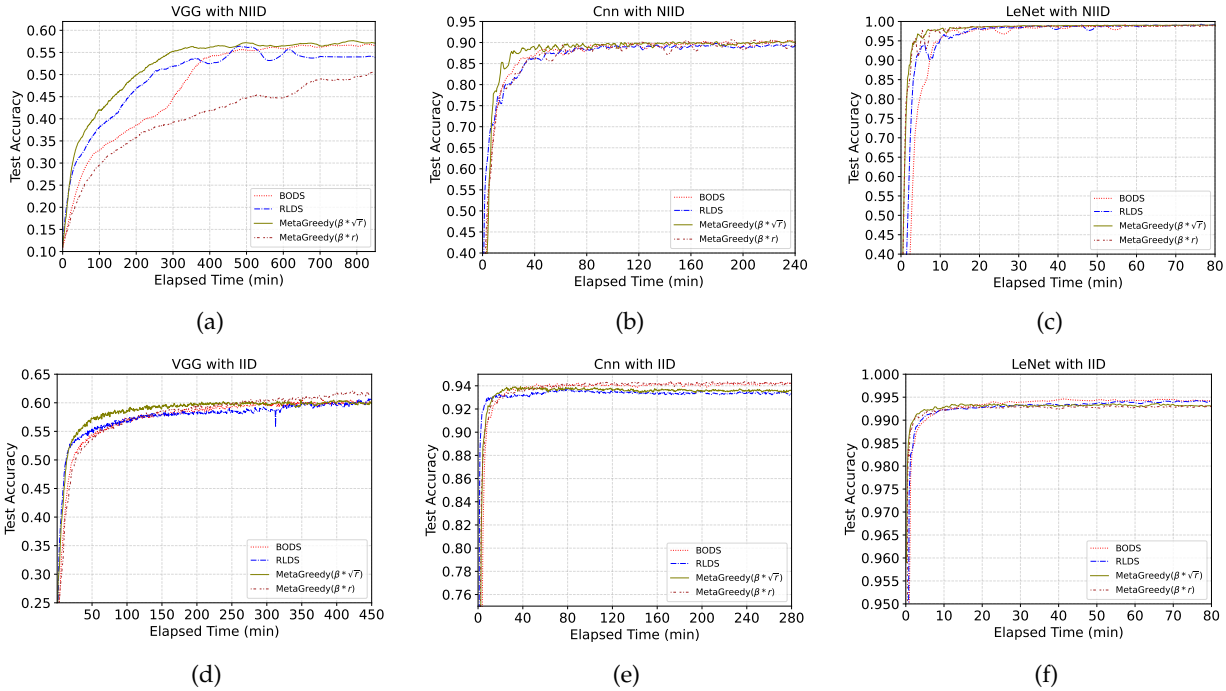


Fig. 13: The convergence accuracy of different jobs in Group A changes over time with diverse settings of β.

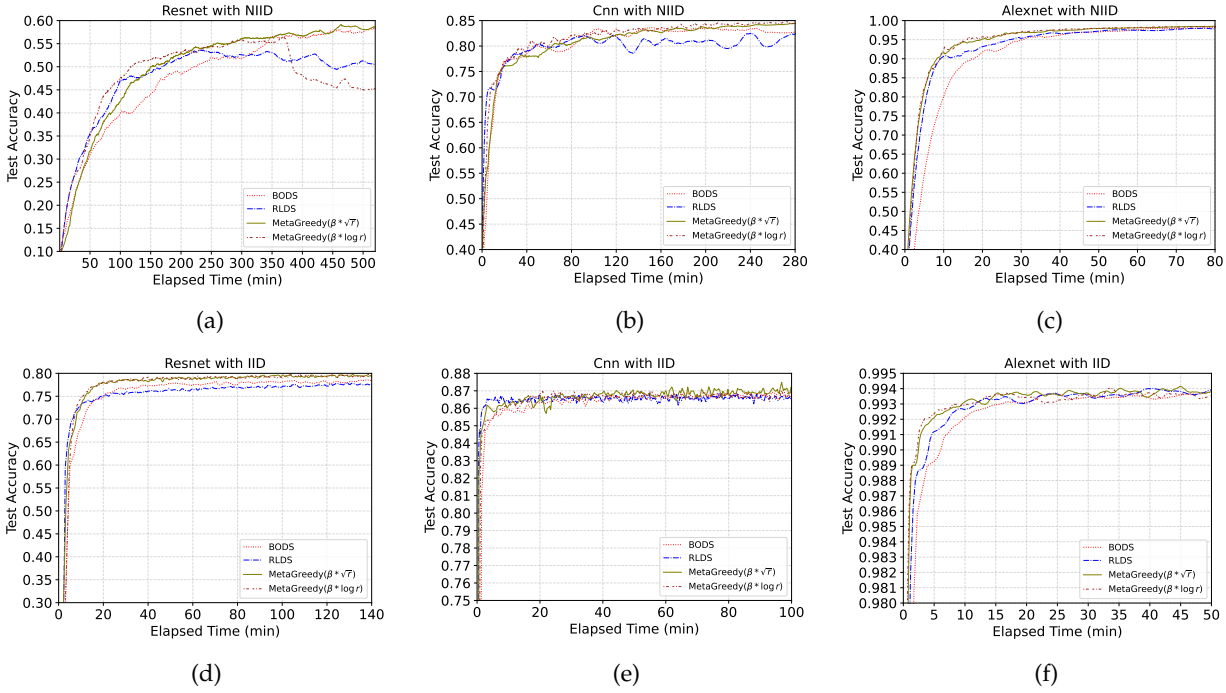


Fig. 14: The convergence accuracy of different jobs in Group B changes over time with diverse settings of β.

have,

$$\begin{aligned}
&\mathbb{E}\sum_{k\in V_m^r} p_{k,r}^m \parallel w_{r,h}^{m,k} - \bar{w}_{r,h}^m \parallel^2 \\
&= \mathbb{E}\sum_{k\in V_m^r} p_{k,r}^m \parallel (w_{r,h}^{m,k} - w_{r,0}^{m,k}) - (\bar{w}_{r,h}^m - w_{r,0}^{m,k}) \parallel^2 \\
&\leq \mathbb{E}\sum_{k\in V_m^r} p_{k,r}^m \parallel w_{r,h}^{m,k} - w_{r,0}^{m,k} \parallel^2 \\
&= \mathbb{E}\sum_{k\in V_m^r} p_{k,r}^m \parallel \sum_{h'=0}^{h-1} \eta_{r,h'}^m g_{r,h'}^{m,k} \parallel^2 \\
&= \mathbb{E}\sum_{k\in V_m^r} p_{k,r}^m \parallel \sum_{h'=0}^{h-1} \eta_{r,h'}^m \nabla f_k^m(w_{r,h'}^{m,k}) \parallel^2 \\
&\leq \sum_{k\in V_m^r} p_{k,r}^m (H-1) \sum_{h'=0}^{h-1} {\eta_{r,h'}^m}^2 \mathbb{E} \parallel \nabla f_k^m(w_{r,h}^{m,k}) \parallel^2 \\
&\leq \sum_{k\in V_m^r} p_{k,r}^m (H-1)^2 {\eta_{r,0}^m}^2 G^2 \\
&\leq Q^2(H-1)^2 {\eta_{r,h}^m}^2 \sum_{k\in V_m^r} p_{k,r}^m G^2
\end{aligned} \tag{25}$$

In the first inequality, we use $\mathbb{E}||X - \mathbb{E}X||^2 \leq \mathbb{E}||X||^2$ where $X = w_{r,h}^{m,k} - w_{r,0}^{m,k}$. In the second inequality from, We use the following steps:

$$\begin{aligned}
Var(x_k) &= \mathbb{E}({x_k}^2) - (\mathbb{E}(x_k))^2 \geq 0 \\
\frac{1}{h}\sum_{h'=0}^{h-1} \parallel x_k \parallel^2 - \parallel \frac{1}{h}\sum_{h'=0}^{h-1} x_k \parallel^2 &\geq 0 \\
\frac{1}{h^2} \parallel \sum_{h'=0}^{h-1} x_k \parallel^2 \leq \frac{1}{h}\sum_{h'=0}^{h-1} &\parallel x_k \parallel^2 \\
\parallel \sum_{h'=0}^{h-1} x_k \parallel^2 \leq h \sum_{h'=0}^{h-1} &\parallel x_k \parallel^2
\end{aligned}$$

we take $\eta_{r,h'}^m \leq \eta_{r,0}^m = \eta_{r-1,H}^m$ with $0 \leq h' \leq H-1$ and $0 \leq h \leq H-1$. The third inequality results from Assumption 4. The last inequality, we assume that $\eta_{r,0}^m \leq Q\eta_{r,h'}^m$. Therefore, we have

$$\begin{aligned}
&\mathbb{E} \parallel \nabla F^m(\bar{w}_{r,h}^m) - \sum_{k\in V_m^r} p_{k,r}^m \nabla F_k^m(w_{r,h}^{m,k}) \parallel^2 \\
&\leq L^2Q^2(H-1)^2 {\eta_{r,h}^m}^2 \sum_{k\in V_m^r} p_{k,r}^m G^2
\end{aligned} \tag{26}$$

Then, we have

$$\begin{aligned}
\mathbb{E}[F^m(\bar{w}_{r,h+1}^m)] \leq \mathbb{E}[F^m(\bar{w}_{r,h}^m)] &- \frac{\eta_{r,h}^m}{2}\mathbb{E} \parallel \nabla F^m(\bar{w}_{r,h}^m) \parallel^2 \\
&+ \frac{L}{2}{\eta_{r,h}^m}^2 \sum_{k\in V_m^r} p_{k,r}^m \sigma^2 \\
&+ \frac{L^2}{2}Q^2(H-1)^2 {\eta_{r,h}^m}^3 \sum_{k\in V_m^r} p_{k,r}^m G^2
\end{aligned} \tag{27}$$

Divide (27) both sides by $\frac{\eta_{r,h}^m}{2}$ and rearrange it yields,

$$\begin{aligned}
\mathbb{E} \parallel \nabla F^m(\bar{w}_{r,h}^m) \parallel^2 \leq &\frac{2}{\eta_{r,h}^m}(\mathbb{E}[F^m(\bar{w}_{r,h}^m)] - \mathbb{E}[F^m(\bar{w}_{r,h+1}^m)]) \\
&+ L\eta_{r,h}^m \sum_{k\in V_m^r} p_{k,r}^m \sigma^2 \\
&+ L^2Q^2(H-1)^2 {\eta_{r,h}^m}^2 \sum_{k\in V_m^r} p_{k,r}^m G^2
\end{aligned} \tag{28}$$

Summing from $r=1, h=1$ to $r=R, h=H$ and dividing both sides by RH yields

$$\begin{aligned}
&\frac{1}{RH}\sum_{r=1}^{R}\sum_{h=1}^{H} \mathbb{E} \parallel \nabla F^m(\bar{w}_{r,h}^m) \parallel^2 \\
&\leq \frac{2}{\eta_{r,h}^m RH}(F^m(\bar{w}_{1,1}^m) - \mathbb{E}[F^m(\bar{w}_{R,H}^m)]) \\
&\quad + L\eta_{r,h}^m \sum_{k\in V_m^r} p_{k,r}^m \sigma^2 + L^2Q^2(H-1)^2 {\eta_{r,h}^m}^2 \sum_{k\in V_m^r} p_{k,r}^m G^2 \\
&\leq \frac{2}{\eta_{r,h}^m RH}(F^m(\bar{w}_{1,1}^m) - \mathbb{E}[F^m(\bar{w}_{R,H}^m)]) + L\eta_{r,h}^m \sigma^2 \\
&\quad + L^2Q^2(H-1)^2 {\eta_{r,h}^m}^2 G^2 \\
&\leq \frac{2}{\eta_{r,h}^m RH}(F^m(\bar{w}_{1,1}^m) - F^{m*}) + L^2Q^2(H-1)^2 {\eta_{r,h}^m}^2 G^2 \\
&\quad + L\eta_{r,h}^m \sigma^2
\end{aligned} \tag{29}$$

We choose $\eta_r^m = \frac{1}{L\sqrt{RH}}$. Then we have

$$\begin{aligned}
&\frac{1}{RH}\sum_{r=1}^{R}\sum_{h=1}^{H} \mathbb{E} \parallel \nabla F^m(\bar{w}_{r,h}^m) \parallel^2 \\
&\leq \frac{2L}{\sqrt{RH}}(F^m(\bar{w}_{1,1}^m) - F^{m*}) + \frac{1}{\sqrt{RH}}\sigma^2 + \frac{Q^2}{RH}(H-1)^2G^2
\end{aligned} \tag{30}$$

If we further choose $Q \leq (RH)^{\frac{1}{4}}$, we have

$$\begin{aligned}
&\frac{1}{RH}\sum_{r=1}^{R}\sum_{h=1}^{H} \mathbb{E} \parallel \nabla F^m(\bar{w}_{r,h}^m) \parallel^2 \\
&\leq \frac{2L}{\sqrt{RH}}(F^m(\bar{w}_{1,1}^m) - F^{m*}) + \frac{1}{\sqrt{RH}}\sigma^2 \\
&\quad + \frac{1}{\sqrt{RH}}(H-1)^2G^2 \\
&= \mathcal{O}(\frac{1}{\sqrt{RH}})
\end{aligned} \tag{31}$$

Experimental Results

Experimentation with Real Mobile Devices

We carried out an experimentation with 20 real devices (mobile devices) and a parameter server on the Baidu AI Cloud [74]. The devices are summarized in Table 8. We carry out the experimentation with a synthetic CNN model of 4 layers, a VGG model of 6 layers, and a ResNet model of 13 layers.

The time to achieve target accuracy is shown in Table 7. From the table, we can find that Meta-Greedy corresponds to the shortest time (up to 42.4% shorter than others) to achieve the target accuracy of simple models, i.e. CNN and VGG, while BODS outperforms baseline methods (up to

TABLE 7: The time to achieve target accuracy for divers models and methods. The "()" after the model represents the target accuracy. "/" represents that the training cannot achieve the target accuracy with the corresponding scheduling method while the "()" represents the highest accuracy during the training process.

	Time (s)						
	Random	Genetic	FedCS	Greedy	BODS	RLDS	Meta-Greedy
CNN (0.928)	/ (0.908)	1997.71	1909.72	/ (0.900)	1369.10	/ (0.915)	**1351.37**
VGG (0.870)	2414.70	1714.55	/ (0.848)	/ (0.848)	1553.10	2591.16	**1493.06**
ResNet (0.680)	2533.00	2839.45	2474.19	2876.11	2209.64	2446.88	**2198.79**
ResNet (0.808)	5552.51	5117.59	/ (0.776)	/ (0.806)	4553.23	**3539.88**	4983.88

TABLE 8: Summary of devices.

Device type	RAM size
HUAWEI Mate20	6G
OPPO A72	8G
Galaxy M11	8G
Redmi Note9 Pro	8G
HUAWEI P40 Pro	8G
Realme GT2	8G
Smartisan R2	8G
HUAWEI nova2	4G
Redmi K20	6G
HUAWEI MatePad	8G
HONOR 60	8G
HUAWEI M6	4G
Galaxy 20U	8G
HONOR V10	4G
Redmi Note11	8G
HUAWEI nova5i	6G
Redmi K50 Pro	12G
Galaxy S21	8G
HONOR Play4	8G
HUAWEI MatPad	6G

35.7%). With a complex model, i.e., ResNet, Meta-Greedy corresponds to excellent efficient training (up to 23.5% compared with others) for a low target accuracy (0.680) while RLDS significantly outperforms baseline methods (up to 36.2%) for a high target accuracy (0.808). This result implies that RLDS favors complex models while BODS favors simple models.

Impact of $\Omega(r)$

As shown in Figures 13 and 14, Meta-Greedy with $\Omega(r) = \sqrt{r}$ outperforms other methods, i.e., $\Omega(r) = r$ and $\Omega(r) = \log r$, in terms of both convergence accuracy and convergence speed.

RLDS & BODS with Simple and Complex Jobs

As shown in Figure 15, RLDS favors complex jobs (VGG and ResNet) while BOSD corresponds to better performance for a simple job (CNN). We exploit VGG19 (21,240,010) [67] and ResNet18 (595,466) [69] to train models with non-IID Cifar10 [71] dataset. We exploit CNN with 491,920 parameters to train a model with the emnist-digital dataset [72].

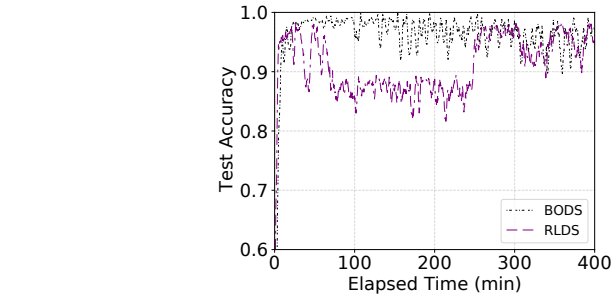
As RLDS can learn more information through a complex neural network, RLDS outperforms BODS for complex jobs (0.008 and 0.029 in terms of accuracy with VGG19 and ResNet18, and 46.7% and 34.8% faster for the target accuracy of 0.7 with VGG19 and 0.5 with ResNet18). Due to the emphasis on the combination of data fairness and device capabilities, i.e.,computation and communication capabilities, BODS can lead to high convergence accuracy and fast convergence speed for simple jobs (0.018 in terms of accuracy and 38% faster for the target accuracy of 0.97 with CNN; see details in Appendix).

Meta-Greedy with 2 Methods and 6 Methods

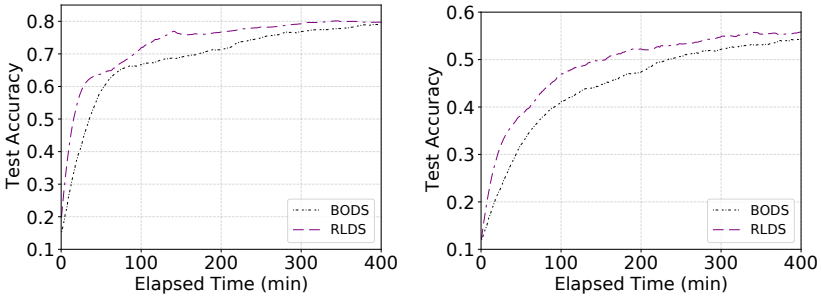
As shown in Figure 16, Meta-Greedy with 2 methods (BODS and RLDS) significantly outperforms that with 6 methods (BODS, RLDS, Genetic, Greedy, Random, and FedCS).

Frequency of Each Method in Meta-Greedy

As shown in Figure 17, Meta-Greedy with 2 methods (BODS and RLDS) significantly outperforms that with 6 methods (BODS, RLDS, Genetic, Greedy, Random, and FedCS). We find that Greedy and Genetic are extensively exploited, RLDS is selected at the beginning of the training process, BODS is chosen at the end of the training process, FedCS participates with less frequency, and Random is seldomly utilized. This result shows that when there are Greedy, Genetic, FedCS, and RLDS, Meta-Greedy can combine them to generate better scheduling plans for a simple job (LeNet with IID). When the job becomes complex, Meta-Greedy combines Greedy, Genetic, FedCS, RLDS, and BODS to generate proper scheduling plans (LeNet with non-IID, CNN with both IID and non-IID). However, when the job becomes even more complex, Meta-Greedy exploits more frequently RLDS and BODS. RLDS is utilized at the beginning of the training process because of its superior performance while BODS is exploited at the end. As BODS may introduce some randomness to the scheduling process, it may correspond to better data fairness and higher accuracy at the end. Please note that does not contradict with the claim "BODS favors simple jobs and RLDS favors complex jobs" as the combination of Greedy and Genetic can well address the simple jobs and the training process is quite different from that of a single scheduling method. As Meta-Greedy can intelligently select a proper scheduling plan based on method, it corresponds to efficiency training process.

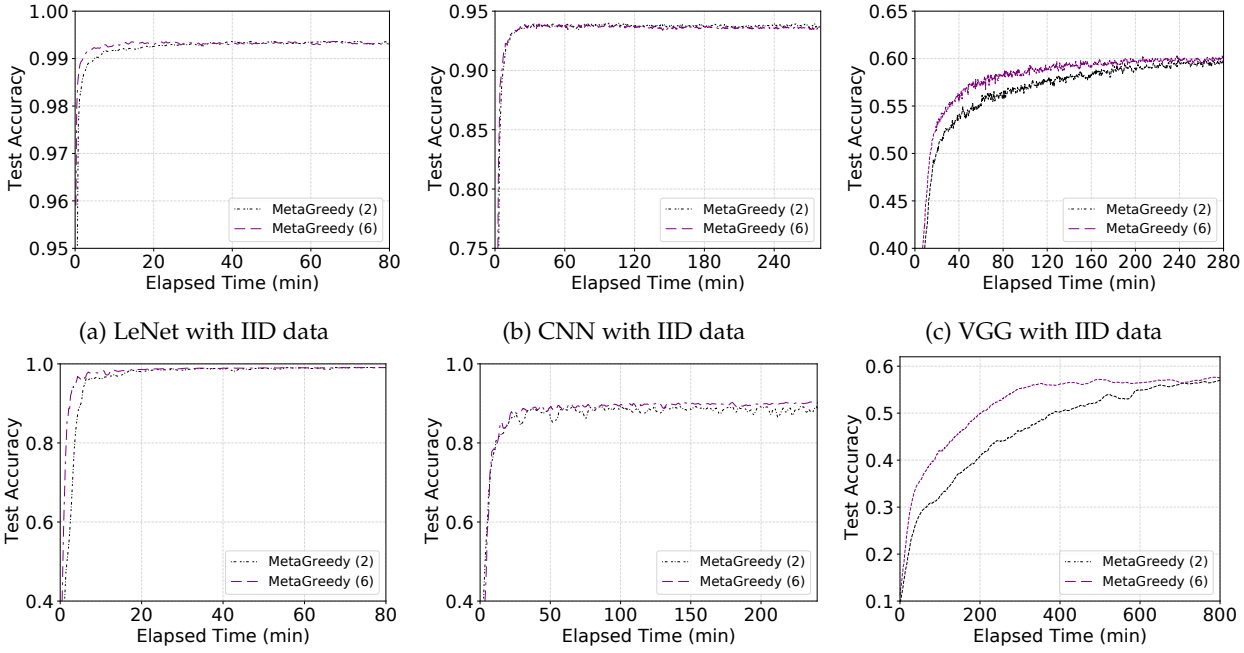


(a) CNN



(b) VGG (c) ResNet

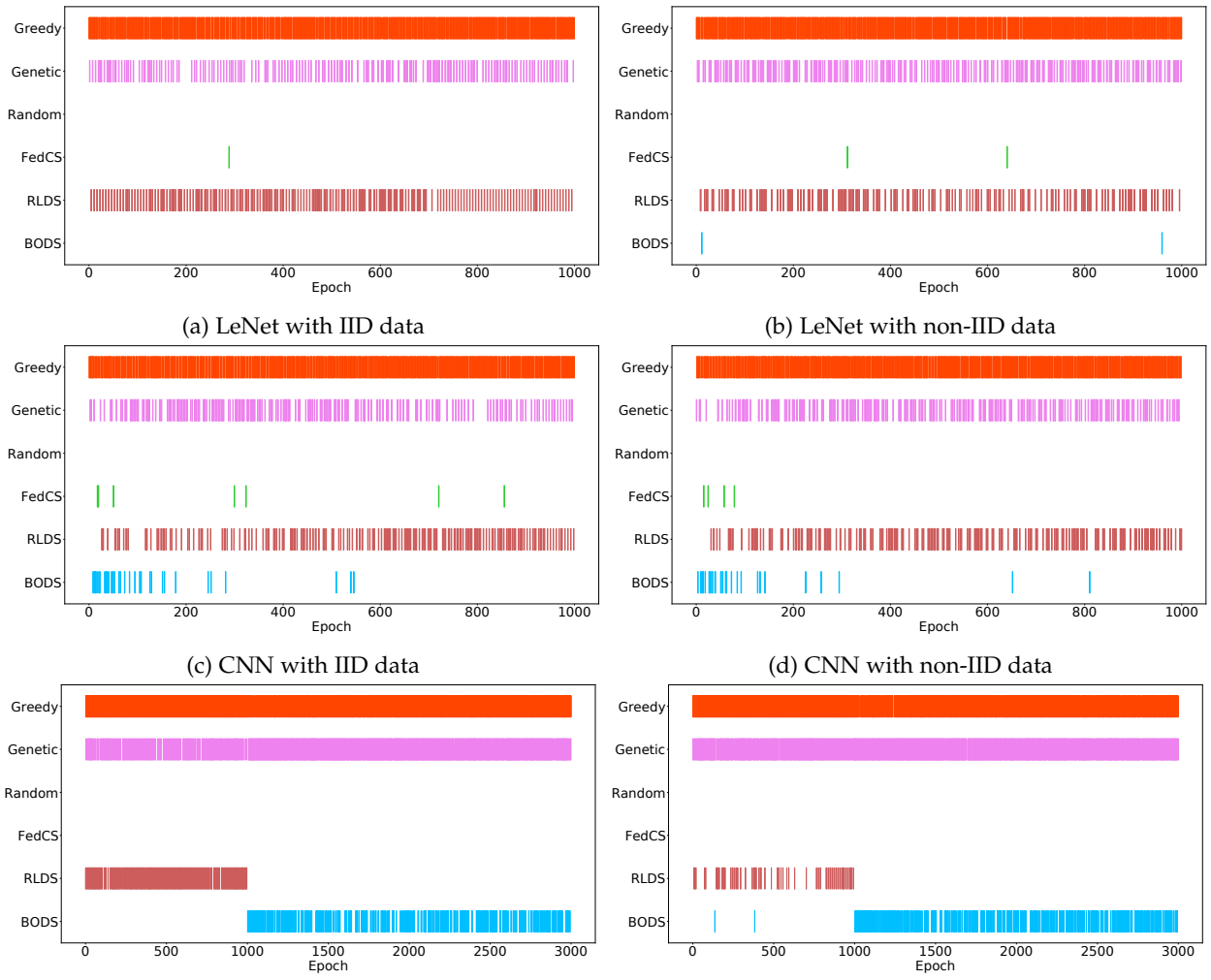
Fig. 15: The accuracy of different jobs (CNN, VGG, and ResNet) with non-IID data.



(a) LeNet with IID data (b) CNN with IID data (c) VGG with IID data

(d) LeNet with non-IID data (e) CNN with non-IID data (f) VGG with non-IID data

Fig. 16: The accuracy of different jobs in Group A with MetaGreedy (2) and MetaGreedy (6). MetaGreedy (2) represents Meta-Greedy with 2 methods and MetaGreedy (6) represents Meta-Greedy with 6 methods.



(a) LeNet with IID data

(b) LeNet with non-IID data

(c) CNN with IID data

(d) CNN with non-IID data

(e) VGG with IID data

(f) VGG with non-IID data

Fig. 17: The participation frequency of diverse methods in the training process of Group A with Meta-Greedy and 6 methods.