Efficient Device Scheduling with Multi-Job Federated Learning

Chendi Zhou[1†], Ji Liu[2†*], Juncheng Jia[1], Jingbo Zhou[2], Yang Zhou[3], Huaiyu Dai[4], Dejing Dou[2]

[1]Soochow University, [2]Baidu Inc., China
[3]Auburn University, [4]North Carolina State University, United States

Abstract

Recent years have witnessed a large amount of decentralized data in multiple (edge) devices of end-users, while the aggregation of the decentralized data remains difficult for machine learning jobs due to laws or regulations. Federated Learning (FL) emerges as an effective approach to handling decentralized data without sharing the sensitive raw data, while collaboratively training global machine learning models. The servers in FL need to select (and schedule) devices during the training process. However, the scheduling of devices for multiple jobs with FL remains a critical and open problem. In this paper, we propose a novel multi-job FL framework to enable the parallel training process of multiple jobs. The framework consists of a system model and two scheduling methods. In the system model, we propose a parallel training process of multiple jobs, and construct a cost model based on the training time and the data fairness of various devices during the training process of diverse jobs. We propose a reinforcement learning-based method and a Bayesian optimization-based method to schedule devices for multiple jobs while minimizing the cost. We conduct extensive experimentation with multiple jobs and datasets. The experimental results show that our proposed approaches significantly outperform baseline approaches in terms of training time (up to 8.67 times faster) and accuracy (up to 44.6% higher).

1 Introduction

Recent years have witnessed a large amount of decentralized data over various Internet of Things (IoT) devices, mobile devices, etc. (Liu et al. 2021), 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 (Official Journal of the European Union 2016; CCL 2018; CCP 2018; Chik 2013) have been put into practice 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 distributed data, federated learning (FL) (McMahan et al. 2017a), which does not transfer raw data, emerges as an effective approach.

FL was first introduced to collaboratively train a global model with non-Independent and Identically Distributed (non-IID) data distributed on mobile devices (McMahan et al. 2017a). During the training process of FL, the raw data is kept decentralized without being moved to a single server or a single data center (Kairouz et al. 2019; Yang et al. 2019). 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 exploits a parameter server architecture (Smola and Narayanamurthy 2010), 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) a number of 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 so as to generate a global model of high accuracy.

While current FL solutions (McMahan et al. 2017a; Pilla 2021) focus on a single-task job or a multi-task job (Smith et al. 2017), FL with multiple jobs (Han et al. 2020) 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 have interaction between 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 chooses a portion of devices to update the model, the other devices remain idle, and the efficiency 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 (Li et al. 2020a, 2021), i.e., the computing and communication capability of each device is different, and the data in each device may also differ. During the training process of multiple jobs, the devices need to be scheduled to each job. At a given time, a device can be scheduled to only one job. However, only a portion of the available devices is scheduled to one job in order to reduce the influence of stragglers (McMahan et al. 2017a). Powerful devices should be scheduled to jobs in order to accelerate the training process, while other eligible devices should also participate in the training process to increase the fairness of data so

[†]C. Zhou and J. Liu contributed equally to the paper. This work was done when C. Zhou was an intern at Baidu Inc.
[*]Corresponding author (liuji04@baidu.com).

To appear in AAAI 2022

as 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 (Pitoura and Triantafillou 2007; Finkelstein et al. 2008).

While the scheduling problem of devices is typical NP-hard (Du and Leung 1989; Liu et al. 2020a), some solutions have already been proposed for the training process of FL (McMahan et al. 2017b; Nishio and Yonetani 2019; Li et al. 2021; Abdulrahman et al. 2021) or distributed systems (Barika et al. 2019), which generally only focus on a single job with FL. In addition, these methods either cannot address the heterogeneity of devices (McMahan et al. 2017b), or do not consider the data fairness during the training process (Nishio and Yonetani 2019; Li et al. 2021; Abdulrahman et al. 2021), 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 two scheduling methods. 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 two 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. We summarize our contributions as follows:

- We propose MJ-FL, a multi-job FL framework consisting of a parallel training process for multiple jobs and a cost model for the 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. 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.
- 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.

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. The experimental results with diverse models and datasets are given in Section 5. Finally, Section 6 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 (McMahan et al. 2017a; Yang et al. 2019; Li et al. 2020a; Liu et al. 2021). Based on the data distribution, FL can be classified into three types, i.e., horizontal, vertical, and hybrid (Yang et al. 2019; Liu et al. 2021). 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 (Kairouz et al. 2019). 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 (Bonawitz et al. 2019; Liu et al. 2020b; Yurochkin et al. 2019; Wang et al. 2020) generally deal with a single job, i.e., with a single global model. While some FL approaches have been proposed to handle multiple tasks (Smith et al. 2017; Chen et al. 2021), 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 (McMahan et al. 2017b; Nishio and Yonetani 2019; Li et al. 2021; Abdulrahman et al. 2021; Barika et al. 2019; Nishio and Yonetani 2019; Li et al. 2021; Abdulrahman et al. 2021; Sun et al. 2020) 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 (Shi, Zhou, and Niu 2020) and the random scheduling method (McMahan et al. 2017b) are proposed for FL, while genetic algorithms (Barika et al. 2019) 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 (Sun et al. 2020), BO (Kim, Kim, and Park 2020), and deep neural network (Zang et al. 2019), 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.

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 two scheduling methods, one based on RL and the other based on 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 multi-job FL process and a cost model. Afterward, we formally define the problem to address in this paper.

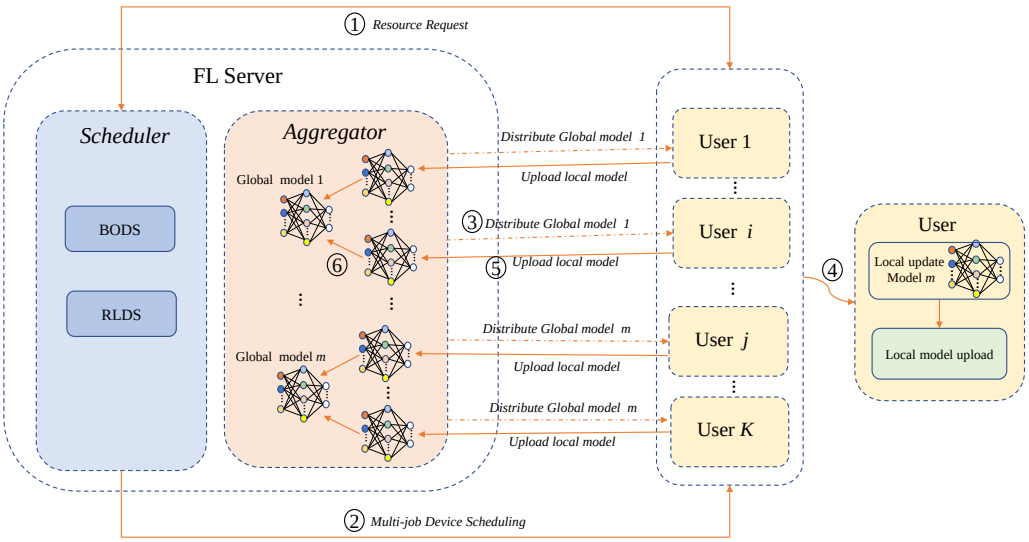


Figure 1: The training process within the Multi-job Federated Learning Framework.

Motivation for Multi-Job Federated Learning

Let us assume a scenario where there are multiple FL jobs to be processed at the same time, e.g., image classification, speech recognition, and text generation. These jobs can be trained in parallel so as to efficiently exploit the available devices. 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 devices 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 certain powerful devices are scheduled to a job, the model can only learn the knowledge 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 taking into consideration both the computing and communication capability and the data fairness.

A straightforward approach is to train each job separately using the mechanism explained in (McMahan et al. 2017b), while exploiting the existing scheduling of single-job FL, e.g., FedAvg (McMahan et al. 2017b). 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.

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 (Li et al. 2014). 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.

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:

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

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 update the global models iteratively until the training processes of the corresponding job converge or achieve a target performance requirement (in terms of accuracy or loss value). We design a multi-job FL process as shown in Fig.

1. The Server first initializes a global model for each job. The initialization can be realized randomly or from the pre-training process with public data. In order 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 to the current job, according to a scheduling plan generated from a scheduling method (see details in Section 4). The scheduling plan is a set of devices that are selected to perform the local training process for the current job. Please 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 of multiple jobs are carried out in parallel. The Server distributes the latest global model of 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 uploads the updated model to the Server after its local training in Step ⑤. Finally, 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 the importance of each job is the same.

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.

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

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.

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

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

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 (Shi et al. 2021; Lee et al. 2018):

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

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.

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

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., $\{\mathcal{V}_m^1, ..., \mathcal{V}_m^r\}$.

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:

$$\min_{\mathcal{V}_m^r} \Big\{ \sum_{m=1}^{M} \sum_{r=1}^{R'_m} \mathscr{T}_m^r(\mathcal{V}_m^r) \Big\} \quad (6)$$

$$\text{s.t.} \begin{cases} \mathbb{L}_m(R'_m) \leq l_m, \\ \mathcal{V}_m^r \subset \mathcal{K}, \forall m \in \{1, 2, ..., M\}, \forall r \in \{1, 2, ..., R'_m\}, \end{cases}$$

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. 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 6), we need to consider the data fairness within the total cost in Formula 7, within which the data fairness can help reduce R'_m so as to minimize the total training time.

$$\min_{\mathcal{V}_m^r} \Big\{ TotalCost(\mathcal{V}_m^r) \Big\}, \quad (7)$$

$$TotalCost(\mathcal{V}_m^r) = \sum_{m'=1}^{M} Cost_{m'}^r(\mathcal{V}_{m'}^r), \quad (8)$$

$$s.t. \qquad \mathcal{V}_{m'}^r \subset \mathcal{K}, \forall m' \in \{1, 2, ..., M\},$$

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 have a potential influence on 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 still a combinatorial optimization problem (Toth 2000) and NP-hard (Du and Leung 1989; Liu et al. 2020a).

4 Device Scheduling for Multi-job FL

In this section, we propose two scheduling methods, i.e., BO-based and RL-based, to address the problem defined in Formula 7. The scheduling plan generated by a scheduling method is defined in Formula 9:

Algorithm 1: Bayesian Optimization-Based Scheduling

Input:

$\mathcal{V}_o$: A set of occupied devices
S_m : A matrix 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: $\quad \Pi' \leftarrow$ Randomly generate a set of observation points with the devices within $\mathcal{K} \backslash \mathcal{V}_o$
4: $\quad \mathcal{V}_m^{*r} \leftarrow \underset{\mathcal{V} \subset \Pi'}{arg\,\max} \alpha_{\text{EI}}(\mathcal{V}; \Pi')$
5: $\quad$ FL training of Job m with $\mathcal{V}_m^{*r}$ and update S_m, $\mathcal{V}_o$
6: $\quad \mathbb{C}_r = TotalCost(\mathcal{V}_m^{*r})$
7: $\quad \Pi_{L+r} \leftarrow \Pi_{L+r-1} \cup (\mathcal{V}_m^{*r}, \mathbb{C}_r)$
8: **end for**

$$\mathcal{V'}_m^r = \underset{\mathcal{V}_m^r \subset \{\mathcal{K} \backslash \mathcal{V}_o^r\}}{\text{argmin}} \; TotalCost(\mathcal{V}_m^r), \tag{9}$$

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 8, and $\mathcal{K}$ and $\mathcal{V}_o^r$ are the set of all devices and the set of occupied devices in Round r, respectively.

Bayesian Optimization-Based Scheduling

While the Gaussian Process (GP) (Srinivas et al. 2010) can well represent linear and non-linear functions, BO-based methods (Shahriari et al. 2016) can exploit a GP to find a near-optimal solution for the problem defined in Formula 9. 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 10 and a covariance function K defined in Formula 11 with a $Matern$ kernel (Williams and Rasmussen 2006).

$$\mu(\mathcal{V}_m^r) = \underset{\mathcal{V}_m^r \subset \{\mathcal{K} \backslash \mathcal{V}_o^r\}}{\mathbb{E}} [TotalCost(\mathcal{V}_m^r)] \tag{10}$$

$$\text{K}(\mathcal{V}_m^r, \mathcal{V}_m'^r) = \underset{\mathcal{V}_m^r \subset \{\mathcal{K} \backslash \mathcal{V}_o^r\}, \mathcal{V}_m'^r \subset \{\mathcal{K} \backslash \mathcal{V}_o^r\}}{\mathbb{E}} [(TotalCost(\mathcal{V}_m^r) - \mu(\mathcal{V}_m^r))(TotalCost(\mathcal{V}_m'^r) - \mu(\mathcal{V}_m'^r))] \tag{11}$$

The BODS is explained in **Algorithm 1**. First, we randomly generate a set of observation points and calculate the cost based on 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, within each round, we randomly sample a set of scheduling plans (Line 3), within which we select the one with the biggest reward using updated μ and K based on Π_{L+r-1} (Line 4). Afterward, we perform the FL training for Job m with the generated scheduling plan (Line 5), and calculate the cost corresponding to the real execution (Line 6) according to Formula 8 and update the observation point set (Line 7).

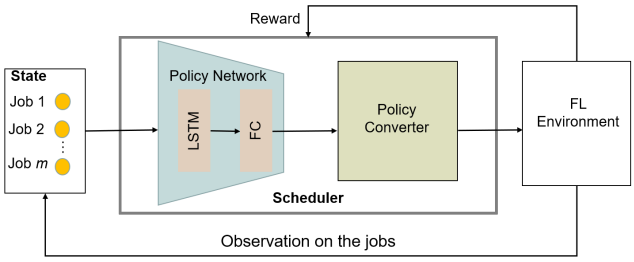


Figure 2: The architecture of the RLDS.

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) (Jones, Schonlau, and Welch 1998) to choose a new scheduling plan $\mathcal{V}_m^{*r}$ in Round r that improves $\mathbb{C}_{L-1}^+$ the most, which is the utility function. Please note that this is not an exhaustive search as 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.

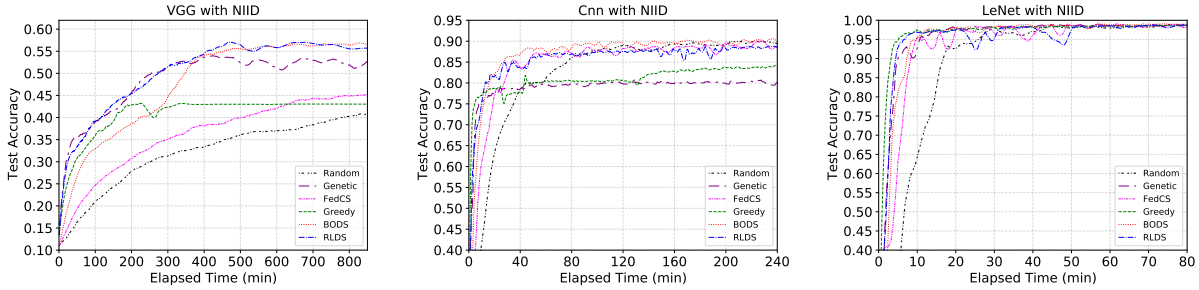
Reinforcement Learning-Based Scheduling

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 Fig. 2, which is inspired by (Mao et al. 2019; Sun et al. 2020). The scheduler of RLDS consists of a policy network and a policy converter. In the process of device scheduling, RLDS collects the status information of jobs as the input to the policy network. Then, 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.

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 sharing relationship of devices 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 to a job.

Policy Converter The Policy Converter generates a scheduling plan based on the probability of each available device calculated by the policy network with the ϵ-greedy strategy (Xia and Zhao 2015).

Training In the training process of RLDS, we define the reward as $\mathscr{R}^m = -1 * TotalCost(\mathcal{V}_m^r)$. Inspired by (Williams 1992; Zoph and Le 2017), we exploit Formula 12 to update the policy network:



(a) (b) (c)

Figure 3: The accuracy of different jobs in Group A over time with the non-IID distribution.

Algorithm 2: 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: $\mathcal{V}_m^r \leftarrow$ generate a scheduling plan using the policy network
4: FL training of Job m and update S_m, $\mathcal{V}_o$
5: Compute $\mathscr{R}^m$
6: Update θ according to Formula 12
7: $b_m \leftarrow$ (1 - γ) * b_m + γ * $\mathscr{R}_n^m$
8: **end for**

$$\theta' = \theta + \frac{\eta}{N} \sum_{n=1}^{N} \sum_{k \in \mathcal{V}_{n,m}^r}^{\mathcal{V}_{n,m}^r \subset \mathcal{K} \setminus \mathcal{V}_o^r} \nabla_\theta \quad \log P(\mathscr{S}_k^m | \mathscr{S}_{(k-1):1}^m; \theta) \\ (\mathscr{R}_n^m - b_m), \tag{12}$$

where θ' and θ represent the updated parameters and the current parameters of the policy network, respectively, η is 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 exploit RLDS during the training process of multiple jobs within the MJ-FL framework as shown in **Algorithm 2**. We pre-train the policy network with randomly generated scheduling plans (see details in Appendix) (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 12 (Line 6), while the baseline value b_m is updated while considering the historical value (Line 7).

5 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 RLDS and BODS with four baseline methods, i.e., Random (McMahan et al. 2017b), FedCS (Nishio and Yonetani 2019), Genetic (Barika et al. 2019), and Greedy (Shi, Zhou, and Niu 2020).

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) (Simonyan and Zisserman 2015), CNN (CNN-A-IID and CNN-A-non-IID) (LeCun et al. 1998), and LeNet-5 (LeNet) (LeCun et al. 1998), and Group B with Resnet-18 (ResNet) (He et al. 2016), CNN (CNN-B) (LeCun et al. 1998), and Alexnet (Krizhevsky, Sutskever, and Hinton 2012), 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 (Krizhevsky, Hinton et al. 2009), emnist-letters (Cohen et al. 2017), emnist-digital (Cohen et al. 2017), Fashion-MNIST (Xiao, Rasul, and Vollgraf 2017), and MNIST (LeCun et al. 1998) in the training process. Please see details of the models and datasets in Appendix. For the non-IID setting of each dataset, the training set is classified by category, and the samples of each category are divided into 20 parts. Each device randomly selects two categories and then selects one part from each category to form 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 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 real training processes. In the experimentation, we use corresponding target accuracy (for ease of comparison) in the place of target loss value.

Evaluation on the non-IID setting: When the decentralized data is of non-IID, the data fairness defined in Formula 5 has a significant influence on the accuracy. As shown in Fig. 3, the convergence speed of our proposed methods, i.e.,

Table 1: 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		Random	Genetic	FedCS	Greedy	BODS	RLDS
Non-IID													
VGG	0.55	0.54	0.55	0.43	**0.57**	**0.57**	VGG(0.55)	2486	1164.3	1498.5	/	455.1	**406.8**
CNN	**0.90**	0.80	0.80	0.83	**0.90**	0.88	CNN(0.80)	44.25	95.85	27.39	43.04	15.88	**12.75**
LeNet	0.990	0.988	0.990	0.986	**0.991**	0.990	LeNet(0.984)	43.81	30.15	33.37	43.76	**28.93**	34.08
IID													
VGG	**0.614**	0.558	0.603	0.522	0.603	**0.614**	VGG(0.60)	529.9	/	322.5	/	293.6	**249.2**
CNN	**0.943**	0.928	**0.943**	0.928	**0.943**	0.937	CNN(0.930)	52.05	176.85	27.45	26.48	19.25	**18.29**
LeNet	0.9945	0.9928	0.9934	0.99	**0.9946**	0.9933	LeNet(0.993)	43.15	57.53	27.31	/	**16.73**	23.31

Table 2: 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		Random	Genetic	FedCS	Greedy	BODS	RLDS
Non-IID													
ResNet	0.546	0.489	0.523	0.403	**0.583**	0.537	ResNet(0.45)	571.0	307.2	279.5	174.2	157.5	**137.6**
CNN	0.821	0.767	0.821	0.764	**0.836**	0.823	CNN(0.73)	47.1	22.0	18.5	70.8	13.8	**4.8**
AlexNet	0.989	0.986	0.987	0.871	**0.990**	0.989	AlexNet(0.978)	141.85	77.74	84.8	/	61.91	**57.97**
IID													
ResNet	0.787	0.754	0.782	0.743	**0.791**	0.771	ResNet(0.740)	65.93	32.51	31.4	52.93	15.9	**11.96**
CNN	0.867	0.867	0.868	0.868	**0.869**	**0.869**	CNN(0.865)	88.81	23.89	26.06	21.42	23.99	**9.3**
AlexNet	0.9938	0.9938	0.9939	0.9935	0.9939	**0.9943**	AlexNet(0.9933)	35.08	19.44	20.97	/	21.65	**12.58**

RLDS and BODS, is significantly faster than other methods. RLDS has a significant advantage for complex jobs (VGG in Fig. 3(a)), while BODS can lead to good performance for relatively simple jobs in Groups A and B (please see details of Group B in Fig. 6 in Appendix). In addition, as shown in Tables 1 and 2, the final accuracy of RLDS and BODS outperforms other methods (up to 44.6% for BODS and 33.3% for RLDS), 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 and 5.11 times shorter for RLDS), and the time for the whole training process, i.e., the total time calculated based on Formula 6 (up to 4.15 times for BODS and 4.67 times for RLDS), for Groups A and B. We have similar observations with IID, while the advantage of RLDS is much more significant (up to 8.67 times shorter in terms of the time for a single job) than that of non-IID as shown in Tables 1 and 2. We also find that MJ-FL outperforms (up to 5.36 faster and 12.5% higher accuracy) sequential execution of single-job FL (see details in Appendix).

As RLDS can learn more information with a complex neural network, RLDS outperforms BODS for complex jobs. BODS can lead to high convergence accuracy and fast convergence speed thanks to the emphasis on the combination of the data fairness and the capability of the device, i.e., computation and communication capability. Both RLDS and BODS significantly outperform the baseline methods, while there are also differences among the four methods. The Greedy method is more inclined to schedule the devices with high capability, which leads to a significant decrease in 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 with random selection, which improves the fairness of the device to a certain extent, and the convergence speed is faster than the Random method.

6 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 two 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 two device scheduling methods, i.e., RLDS for complex jobs and BODS for simple jobs, to efficiently select 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 significantly outperform baseline methods (up to 44.6% in terms of accuracy, 8.67 times faster for a single job, and 4.67 times faster for the total time).

References

2018. California Consumer Privacy Act Home Page. https://www.caprivacy.org/. Online; accessed 14/02/2021.

2018. Cybersecurity Law of the People's Republic of China. https://www.newamerica.org/cybersecurity-initiative/digichina/blog/translation-cybersecurity-law-peoples-republic-china/. Online; accessed 22/02/2021.

Abdulrahman, S.; Tout, H.; Mourad, A.; and Talhi, C. 2021. FedMCCS: Multicriteria Client Selection Model for Optimal IoT Federated Learning. *IEEE Internet of Things Journal*, 8(6): 4723–4735.

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

Bonawitz, K.; Eichner, H.; Grieskamp, W.; Huba, D.; Ingerman, A.; Ivanov, V.; Kiddon, C.; Konecný, J.; Mazzocchi, S.; McMahan, B.; Overveldt, T. V.; Petrou, D.; Ramage, D.; and Roselander, J. 2019. Towards Federated Learning at Scale: System Design. In *Machine Learning and Systems (MLSys)*.

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

Chik, W. B. 2013. The Singapore Personal Data Protection Act and an assessment of future trends in data privacy reform. *Computer Law & Security Review*, 29(5): 554–575.

Cohen, G.; Afshar, S.; Tapson, J.; and Van Schaik, A. 2017. EMNIST: Extending MNIST to handwritten letters. In *Int. Joint Conf. on Neural Networks (IJCNN)*, 2921–2926.

Du, J.; and Leung, J. Y.-T. 1989. Complexity of Scheduling Parallel Task Systems. *SIAM Journal on Discrete Mathematics*, 2(4): 473–487.

Dwork, C. 2008. Differential privacy: A survey of results. In *Int. conf. on theory and applications of models of computation*, 1–19.

Finkelstein, A.; Harman, M.; Mansouri, S. A.; Ren, J.; and Zhang, Y. 2008. "Fairness analysis" in requirements assignments. In *IEEE Int. Requirements Engineering Conf.*, 115–124.

Han, J.; Rafique, M. M.; Xu, L.; Butt, A. R.; Lim, S.-H.; and Vazhkudai, S. S. 2020. 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)*, 272–281.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

Jones, D. R.; Schonlau, M.; and Welch, W. J. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4): 455–492.

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

Kim, K.-r.; Kim, Y.; and Park, S. 2020. A probabilistic machine learning approach to scheduling parallel loops with bayesian optimization. *IEEE trans. on Parallel and Distributed Systems (TPDS)*, 32(7): 1815–1827.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Annual Conf. on Neural Information Processing Systems (NeurIPS)*, 1106–1114.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Lee, K.; Lam, M.; Pedarsani, R.; Papailiopoulos, D.; and Ramchandran, K. 2018. Speeding Up Distributed Machine Learning Using Codes. *IEEE Trans. on Information Theory*, 64(3): 1514–1529.

Li, L.; Shi, D.; Hou, R.; Li, H.; Pan, M.; and Han, Z. 2021. 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)*, 1–10.

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

Li, T.; Sahu, A. K.; Talwalkar, A.; and Smith, V. 2020a. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3): 50–60.

Li, X.; Huang, K.; Yang, W.; Wang, S.; and Zhang, Z. 2019. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*.

Li, X.; Huang, K.; Yang, W.; Wang, S.; and Zhang, Z. 2020b. On the Convergence of FedAvg on Non-IID Data. In *Int. Conf. on Learning Representations (ICLR)*.

Liu, J.; Huang, J.; Zhou, Y.; Li, X.; Ji, S.; Xiong, H.; and Dou, D. 2021. From Distributed Machine Learning to Federated Learning: A Survey. *arXiv preprint arXiv:2104.14362*.

Liu, L.; Yu, H.; Sun, G.; Luo, L.; Jin, Q.; and Luo, S. 2020a. Job scheduling for distributed machine learning in optical WAN. *Future Generation Computer Systems (FGCS)*, 112: 549–560.

Liu, Y.; Huang, A.; Luo, Y.; Huang, H.; Liu, Y.; Chen, Y.; Feng, L.; Chen, T.; Yu, H.; and Yang, Q. 2020b. Fedvision: An online visual object detection platform powered by federated learning. In *AAAI Conf. on Artificial Intelligence*, volume 34, 13172–13179.

Mao, H.; Schwarzkopf, M.; Venkatakrishnan, S. B.; Meng, Z.; and Alizadeh, M. 2019. Learning scheduling algorithms for data processing clusters. In Wu, J.; and Hall, W., eds., *ACM Special Interest Group on Data Communication (SIGCOMM)*, 270–288. ACM.

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

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017b. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, 1273–1282. PMLR.

Mehrabi, N.; Morstatter, F.; Saxena, N.; Lerman, K.; and Galstyan, A. 2021. A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)*, 54(6): 1–35.

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

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

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

Peng, Y.; Bao, Y.; Chen, Y.; Wu, C.; and Guo, C. 2018. Optimus: an efficient dynamic resource scheduler for deep learning clusters. In *EuroSys Conf.*, 1–14.

Petrangeli, S.; Claeys, M.; Latré, S.; Famaey, J.; and De Turck, F. 2014. A multi-agent Q-Learning-based framework for achieving fairness in HTTP Adaptive Streaming. In *IEEE Network Operations and Management Symposium (NOMS)*, 1–9.

Pilla, L. L. 2021. Optimal Task Assignment for Heterogeneous Federated Learning Devices. In *IEEE Int. Parallel and Distributed Processing Symposium (IPDPS)*, 661–670.

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

Shahriari, B.; Swersky, K.; Wang, Z.; Adams, R. P.; and de Freitas, N. 2016. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE*, 104(1): 148–175.

Shi, W.; Zhou, S.; and Niu, Z. 2020. Device scheduling with fast convergence for wireless federated learning. In *IEEE Int. Conf. on Communications (ICC)*, 1–6.

Shi, W.; Zhou, S.; Niu, Z.; Jiang, M.; and Geng, L. 2021. Joint Device Scheduling and Resource Allocation for Latency Constrained Wireless Federated Learning. *IEEE Trans. on Wireless Communications*, 20(1): 453–467.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Int. Conf. on Learning Representations (ICLR)*.

Smith, V.; Chiang, C.-K.; Sanjabi, M.; and Talwalkar, A. 2017. Federated Multi-Task Learning. In *Annual Conf. on Neural Information Processing Systems (NeurIPS)*, 4427–4437.

Smola, A.; and Narayanamurthy, S. 2010. An architecture for parallel topic models. *Very Large Data Bases Conference (VLDB) Endowment*, 3(1-2): 703–710.

Srinivas, N.; Krause, A.; Kakade, S. M.; and Seeger, M. W. 2010. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. In *Int. Conf. on Machine Learning (ICML)*, 1015–1022.

Sun, P.; Guo, Z.; Wang, J.; Li, J.; Lan, J.; and Hu, Y. 2020. DeepWeave: Accelerating Job Completion Time with Deep Reinforcement Learning-based Coflow Scheduling. In *Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 3314–3320.

Toth, P. 2000. Optimization engineering techniques for the exact solution of NP-hard combinatorial optimization problems. *European Journal of Operational Research (EJOR)*, 125(2): 222–238.

Van Laarhoven, P. J.; and Aarts, E. H. 1987. Simulated annealing. In *Simulated annealing: Theory and applications*, 7–15.

Wang, H.; Yurochkin, M.; Sun, Y.; Papailiopoulos, D.; and Khazaeni, Y. 2020. Federated Learning with Matched Averaging. In *Int. Conf. on Learning Representations (ICLR)*.

Williams, C. K.; and Rasmussen, C. E. 2006. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4): 229–256.

Xia, Z.; and Zhao, D. 2015. Online reinforcement learning by bayesian inference. In *Int. Joint Conf. on Neural Networks (IJCNN)*, 1–6.

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

Yang, Q.; Liu, Y.; Chen, T.; and Tong, Y. 2019. Federated machine learning: Concept and applications. *ACM Trans. on Intelligent Systems and Technology (TIST)*, 10(2): 1–19.

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

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

Zhou, F.; and Cong, G. 2018. On the Convergence Properties of a K-step Averaging Stochastic Gradient Descent Algorithm for Nonconvex Optimization. In *Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 3219–3227.

Zoph, B.; and Le, Q. V. 2017. Neural Architecture Search with Reinforcement Learning. In *Int. Conf. on Learning Representations (ICLR)*.

Appendix

Algorithm 3: 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: $\quad\quad Rd_n^m \leftarrow \text{TotalCost}(\mathcal{V}_{n,m}^r)$
6: $\quad$ **end for**
7: $\quad$ Update θ according to Formula 12
8: $\quad b_m \leftarrow (1 - \gamma) * b_m + \frac{\gamma}{N} * \sum_{n=1}^{N} \mathscr{R}_n^m$
9: $\quad \mathcal{V}_m^{*r} \leftarrow \text{argmin}_{\mathcal{V}_{n,m}^r \in \mathcal{V}_m^r} \text{TotalCost}(\mathcal{V}_{n,m}^r)$
10: $\quad$ Update S_m, $\mathcal{V}_o$ with $\mathcal{V}_m^{*r}$
11: **end for**

Loss Estimation

We assume exploiting stochastic gradient descent (SGD) to train models, which converges at a rate of $O(r)$ with r representing the number of rounds (Peng et al. 2018). Inspired by (Li et al. 2019), we exploit Formula 13 to roughly estimate the loss value of the global model for Job m at Round r.

$$Loss_m(r) = \frac{1}{\beta_m^0 r + \beta_m^1} + \beta_m^2, \tag{13}$$

where β_m^0, β_m^1 and β_m^2 represent non-negative coefficients of the convergence curve of Job m. β_m^0, β_m^1 and β_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 execution). Given a loss value of a model, we use this loss estimation method to calculate the maximum rounds for each job. Given a loss value of a model, we use this loss estimation method to calculate the number of rounds as R_m^c and use $(1 + 0.3) * R_m^c$ as R_m defined in Table 6. 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.

Details for Bayesian Optimization-Based Scheduling

The utility function is defined in Formula 14.

$$u(\mathcal{V}_m^{*r}) = max(0, \mathbb{C}_{L-1}^{+} - TotalCost(\mathcal{V}_m^{*r})), \tag{14}$$

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 $\mathcal{V}$.

$$\begin{aligned}\alpha_{\text{EI}}(\mathcal{V}; \Pi_{L-1}) =& \mathbb{E}[u(\mathcal{V})|\mathcal{V}, \Pi_{L-1}] \\ =& (\mathbb{C}_{L-1}^{+} - \mu(\mathcal{V}))\Phi(\mathbb{C}_{L-1}^{+}; \mu(\mathcal{V}), \text{K}(\mathcal{V}, \mathcal{V})) \\ &+ \text{K}(\mathcal{V}, \mathcal{V})\mathcal{N}(\mathbb{C}_{L-1}^{+}; \mu(\mathcal{V}), \text{K}(\mathcal{V}, \mathcal{V})),\end{aligned} \tag{15}$$

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., $\mathcal{V}_{L,m}^{*r}$.

Training Process of Reinforcement Learning-Based Device Scheduling

We pre-train the policy network using **Algorithm 3**. 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 12 (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).

Details of Experimental Setup

CNN-A-IID is composed of two 3×3 convolution layers, one with 32 channels and the other with 64 channels. Each

Table 3: 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

Table 4: 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

Table 5: The time required to achieve the target accuracy for jobs executed sequentially with FedAvg. "*" indicates that it fails to achieve the target accuracy.

	NIID/IID			NIID/IID		
Job	VGG	CNN	LeNet	ResNet	CNN	AlexNet
Target Accuracy	0.55/0.60	0.80/0.93	0.984/0.993	0.45/0.74	0.73/0.865	0.978/0.9933
Time (min)	2483.4/414.6	53.1/45.5	50.5/52.1	594.3/*	36.1/172.9	127.3/65.16

Table 6: 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 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 maximum 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

layer is followed by one batch normalization layer and 2×2 max pooling. Then, after the two convolution layers, there are one flatten layer and three fully-connected layers (1568, 784, and 26 units). Since the convergence behavior of CNN on non-IID in Group A is not good, we make a simple modification of CNN-A-IID to CNN-A-non-IID. 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 3 and 4.

Explanation of Notations

The meanings of the major notations in this paper are summarized in Table 6. In particular, $S_m = \{s_{1,m}, ..., s_{|\mathcal{K}|,m}\}$ represents the frequency vector of Job m. At the beginning, i.e., Round 0, each $s_{k,m} \in S_m$ is 0. Let $s_{k,m}^r$ represent 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:

$$s_{k,m}^{r+1} = \begin{cases} s_{k,m}^r + 1, & \text{if Device } k \in \mathcal{V}_m^r \\ s_{k,m}^r, & \text{otherwise} \end{cases} \quad (16)$$

Comparison With Single-Job Federated Learning

In order to prove the effectiveness of our proposed framework, i.e., MJ-FL, over the Single-Job FL (SJ-FL) approach, we executed each group of jobs sequentially with FedAvg, which is denoted the Random method when adapted to multi-job FL. As shown in Table 5, RLDS with MJ-FL outperforms the FedAvg with SJ-FL, which executes jobs sequentially, up to 5.36 times faster in terms of the training time while achieving the same accuracy. Similarly, the advantage of BODS can be up to 4.68 times faster. In addition, Random within MJ-FL also outperforms SJ-FL up to 0.25 times faster.

Comparison With Multiple Targets and Other Methods

Figures 6 and 7 show the obvious fast convergence speed of RLDS and BODS. We conducted the experiment with different target accuracy, and the advantage is up to 7.63 times faster for Target 1 (0.845), 9.89 times faster for Target 2 (0.856), and 6.98 times faster for Target 3 (0.865), compared with the baselines. We conduct an ablation experiment and find that the data fairness improves both the convergence speed (up to 9.35 times faster) and the accuracy (up to 15.3%). In addition, other scheduling methods, e.g., simulated annealing (Van Laarhoven and Aarts 1987), corresponds to worse performance (up to 91.4% slower and 3.5% lower accuracy), according to our experiments. We carry out experiments to compare other black-box optimization methods (deep neural networks (Zang et al. 2019)), of which the

performance is worse (up to 90.5% slower and 26.3% lower accuracy) than our methods. Furthermore, we tested 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).

Design Details

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 the same accuracy. Within each round, Step ⑥ exploits FedAvg (McMahan et al. 2017b) to aggregate multiple models within each job, which can ensure the optimal convergence (Li et al. 2020b; Zhou and Cong 2018). 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 (Paillier 1999) and differential privacy (Dwork 2008), can be exploited to protect the privacy of sensitive data.

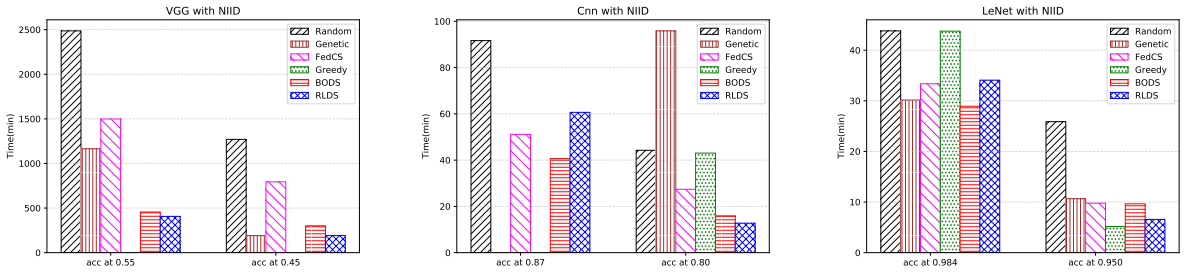
We choose the linear combination because of its convenience and good performance. In practice, we empirically set α and β based on the information from previous execution and adjust them using small epochs. We increase α for fast convergence and increase β mainly for high accuracy.

Please note that the "data fairness" is different from the "fairness" (the bias of the machine learning models concerning certain features) in machine learning (Mehrabi et al. 2021). Formula 5 is based on (Petrangeli et al. 2014), 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 (Li et al. 2020b; Zhou and Cong 2018). 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.

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.

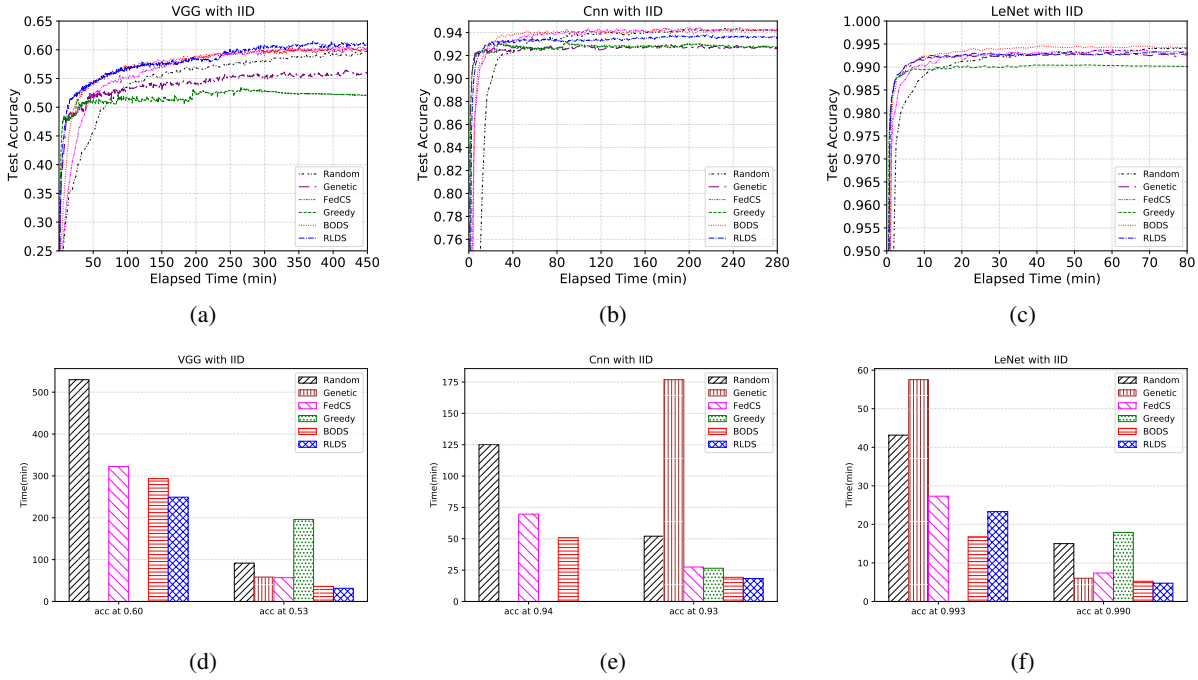
In fact, we consider the probability to release the devices in $\mathcal{V}_o$ in BODS and RLDS, and possible concurrent occupation of other devices for other jobs, which is not explained in the paper to simplify the explanation.

During the execution, we only sample 10% devices of all the devices for each job. Thus, we do not assume that all the devices are available all the time during the training process.



(a) (b) (c)

Figure 4: The time required for each job of Group A to achieve the target convergence accuracy with the non-IID distribution. As Greedy and Genetic fail to achieve the target accuracy on some jobs, the time is not shown.



(a) (b) (c)

(d) (e) (f)

Figure 5: The convergence accuracy of different jobs in Group A changes over time with the IID distribution. (d) to (f) show the time required for scheduling methods to achieve the target accuracy with non-IID, where Greedy and Genetic fail to achieve the target accuracy on some jobs.

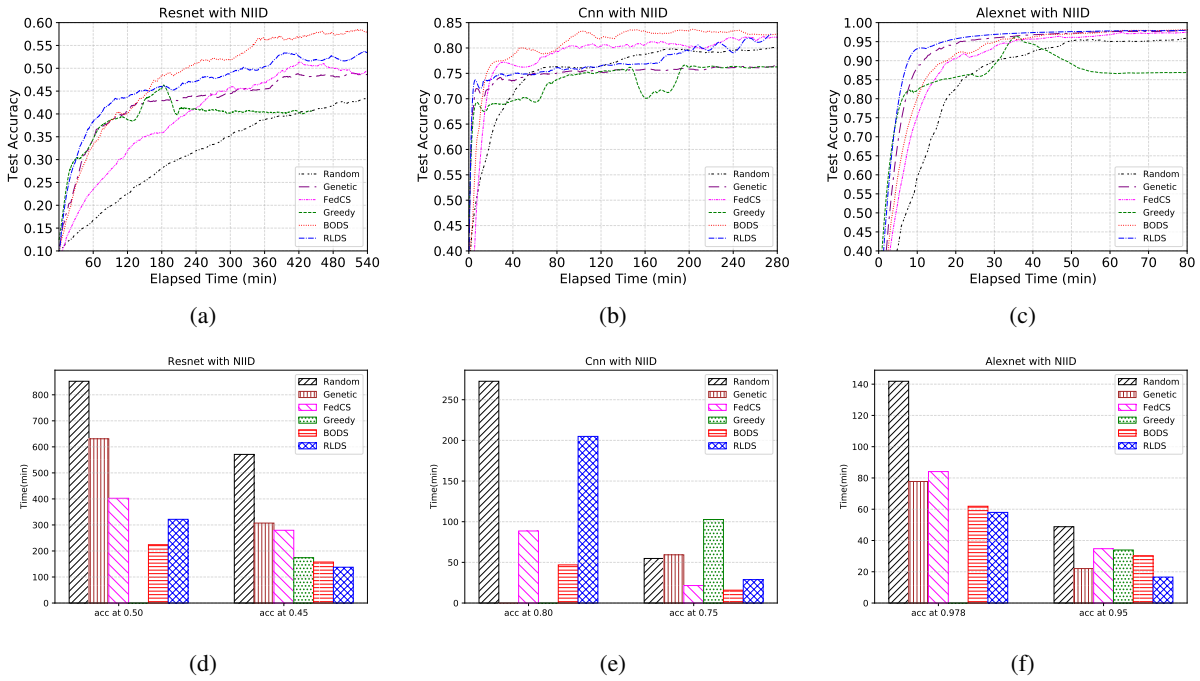


Figure 6: The convergence accuracy of different jobs in Group B changes over time with the non-IID distribution. (d) to (f) show the time required for scheduling methods to achieve the target accuracy with non-IID, where Greedy and Genetic fail to achieved the target accuracy on some jobs.

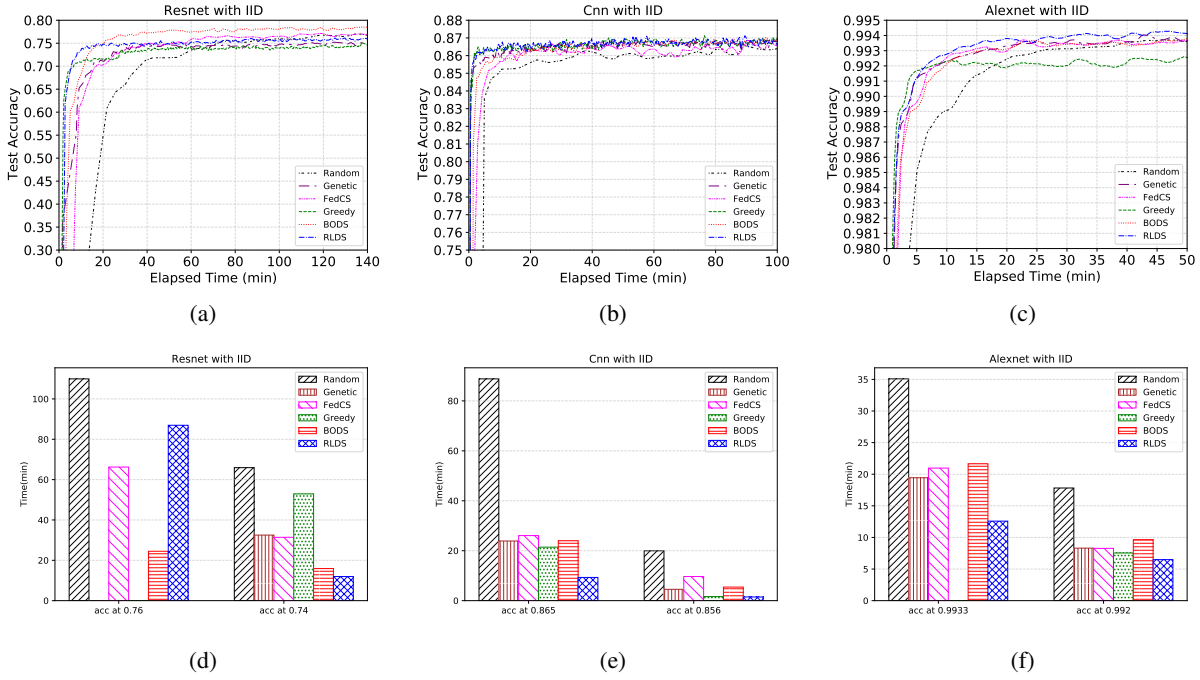


Figure 7: The convergence accuracy of different jobs in Group B changes over time with the IID distribution. (d) to (f) show the time required for scheduling methods to achieve the target accuracy with IID, where Greedy and Genetic fail to achieve the target accuracy on some jobs.