2021 IEEE Wireless Communications and Networking Conference Workshops (WCNCW) | 978-1-7281-9507-0/20/$31.00 ©2021 IEEE | DOI: 10.1109/WCNCW49093.2021.9419985

Tradeoff between Model Accuracy and Cost for Federated Learning in the Mobile Edge Computing Systems

Xueting Luo, Zhongyuan Zhao, Mugen Peng
The State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications, Beijing, 100876, China
E-mail: {Luoxt, zyzhao, pmg}@bupt.edu.cn

***Abstract*—The combination of MEC and federated learning is a promising research direction in network intelligence. However, the performance and efficiency of federated learning cannot be guaranteed in MEC systems. In this paper, in order to balance the model accuracy and resource consumption of federated learning in MEC systems, a framework for deploying federated learning in MEC systems is proposed. Based on this framework, the model accuracy performance of federated learning is analyzed, and a tractable upper bound of accuracy loss is given. In addition, an optimization problem is proposed to balance the accuracy loss and resource consumption of the training model, and a joint optimization algorithm with high computational efficiency is designed to approach the optimal solution. Finally, numerical simulation and experimental results show that the joint optimization algorithm can not only improve the model accuracy of federated learning, but also significantly reduce the resource consumption.**

***Index Terms*—Artificial intelligence, Federated learning, Mobile edge computing, Resource management**

I. Introduction

With the development of the Internet of Everything era, the number of mobile terminals and mobile devices has increased significantly. With the significant increase of the computation capacity of the edge equipment, the vast amount of the idle computation power and storage space distributed at the network edges can provide support for the computation-intensive and latency-sensitive tasks on mobile devices. Therefore, mobile edge computing (MEC) is considered as a promising technology that integrates wireless communication and mobile computing [1]. In order to support computationally intensive applications and tasks, the optimal design of MEC systems has attracted a lot of attention.

Due to the sufficient computation resources in MEC systems, the combination of MEC and artificial intelligence (AI) will be a new trend in the future development of wireless networks. The current work of deploying AI in MEC is mainly based on the centralized learning paradigm, which cannot fully explore the potential of MEC. Therefore, in order to promote the collaborative learning of complex models between distributed devices in MEC systems, a machine learning method called federated learning has been proposed [2]. In federated learning, mobile devices do not need to aggregate raw data on a centralized server, allowing users to keep their private data locally. And users can use their locally collected data to train the model cooperatively, which can significantly reduce the communication costs of data transmission and protect users' privacy.

Although the deployment of federated learning in MEC systems is a promising technology, it still has some challenging problems: First, the communication and computing capabilities of edge devices are limited, so the performance and efficiency of deploying federated learning in MEC systems will be limited accordingly. Second, the conventional network management strategies in MEC systems mainly focus on the optimization of communication and computing performance, so the model accuracy cannot be guaranteed. Finally, how to obtain a high computational efficiency optimization algorithm which jointly considers model accuracy and training costs is also a challenge.

Motivated by providing some insights of the aforementioned key issues, the model accuracy and training cost of federated learning in MEC systems is studied in this paper, and our main contributions can be summarized as follows:

- First, considering the effective utilization of computation and communication resources, we provide a framework of federated learning in MEC systems. According to the framework, the performance of model accuracy is analyzed, and a tractable upper bound of the accuracy loss is derived.
- Second, we propose an optimization problem to jointly minimize the model accuracy loss and training costs. And a computation-efficient optimization algorithm is designed to approach the optimal solution accordingly.
- Finally, the experiment results are provided to evaluate the performance of our proposed algorithm. The experiment results show that the algorithm can improve the accuracy performance as much as possible while significantly reduce the training cost of federated learning.

II. System Model

Consider the deployment of federated learning in MEC systems, which consists of an access point F_A and M users $U_1, \ldots, U_M$. In particular, an edge computing server S_E is equipped with F_A, while a local processing unit S_m is

978-1-7281-9507-0/21/$31.00 ©2021 IEEE

equipped with U_m. As introduced in [2], federated learning is a promising technique to generate high quality models without aggregating the local private data of all the users. The key idea of federated learning is to encourage each user to train its local model individually, and then generate the global model via the model-level collaborations [3].

In the existing paradigms of federated learning, the interaction between the server and the clients are via wired links with high reliability and low cost. However, the reliability and efficiency of federated learning cannot be guaranteed in the MEC systems, due to the wireless transmission circumstances. During the k-th round of interaction, it can be divided into two phases, which can be named as local model training and federated averaging, respectively.

1) Local Model Training: Without loss of generality, we focus on a specific user U_m, and the local model can be updated based on its own data. In this paper, the stochastic gradient descent (SGD) method is employed for local model training, which is an iterative optimization algorithm to minimize the empirical risk [4]. In particular, τ different batches of data elements are randomly sampled from the local data set $\mathcal{D}_m$, which can be denoted as $\mathcal{D}_{k,1}^m, \ldots, \mathcal{D}_{k,\tau}^m$. All these batches are with the same volume of data elements, and utilized for generating the update results for the k-th round of interaction. Then, the corresponding update result based on $\mathcal{D}_{k,t}^m$ can be expressed as follows [4]:

$$\mathbf{w}_{k,t}^m = \mathbf{w}_{k,t-1}^m - \eta_{k,t} \nabla F(\mathbf{w}_{k,t-1}^m, \mathcal{D}_{k,t}^m),\ t = 1, \ldots, \tau, \quad (1)$$

where $\mathbf{w}_{k,t}^m$ and $\mathbf{w}_{k,t-1}^m$ denote the $q \times 1$ updated parameter vectors based on $\mathcal{D}_{k,t}^m$ and $\mathcal{D}_{k,t-1}^m$, respectively, $\eta_{k,t}$ denotes the learning rate, $F(\mathbf{w}_{k,t-1}^m, \mathcal{D}_{k,t}^m)$ is the loss function, and $\nabla F(\mathbf{w}_{k,t-1}^m, \mathcal{D}_{k,t}^m)$ denotes its first-order derivative with respect to $\mathbf{w}_{k,t-1}^m$.

2) Federated Averaging: The final update result for the k-th round of interaction, which is denoted as $\mathbf{w}_{k,\tau}^m$, can be obtained after local model training by employing $\mathcal{D}_{k,1}^m, \ldots, \mathcal{D}_{k,\tau}^m$, where the total volume of data elements using for each round of interaction is the same. To improved the accuracy performance of federated learning, $\mathbf{w}_{k,\tau}^m$ should be transmitted to S_E via the wireless channel between U_m and F_A, which can be used to generate a global model via federated averaging. After receiving all the update results from $U_1, \ldots, U_M$, the global model can be expressed as

$$\mathbf{w}_k = \sum_{m=1}^{M} \frac{N_m}{N_F} \mathbf{w}_{k,\tau}^m, \quad (2)$$

where N_m denotes the size of $\mathcal{D}_{k,1}^m, \ldots, \mathcal{D}_{k,\tau}^m$, and $N_F = \sum_{m=1}^{M} N_m$. After federated averaging, $\mathbf{w}_k$ is sent back to all the users, and each user updates its local model accordingly.

III. Accuracy Performance of Federated Learning

As introduced in [5] and [6], the model accuracy of federated learning can be characterized by *accuracy loss*, which can be expressed as follows:

$$\mathcal{L}_K = \mathbb{E}\{\left\|F(\mathbf{w}_K) - F(\mathbf{w}^*)\right\|^2\}, \quad (3)$$

where $\mathbf{w}_K$ denotes the global model after K rounds of federated averaging as given by (2), $\mathbf{w}^*$ denotes the optimal model that can minimize the empirical risk. $F(\mathbf{w})$ denotes the loss function with respect to $\mathcal{D}$, i.e.,

$$F(\mathbf{w}) = \frac{1}{N} \sum_{\mathbf{x}_n \in \mathcal{D}} f(\mathbf{w}, \mathbf{x}_n), \quad (4)$$

where $\mathbf{x}_n$ denotes the n-th element of the employed training data set $\mathcal{D}$, N is the size of $\mathcal{D}$, $f(\mathbf{w}, \mathbf{x}_n)$ denotes the loss value with respect to $\mathbf{x}_n$. To guarantee the accuracy performance, $f(\mathbf{w}, \mathbf{x}_n)$ should be γ-Lipschitz and λ-strongly convex. And to ensure the convergence, the gradient of loss function is finite, i.e., $\|\nabla f(\mathbf{w}, \mathbf{x}_n)\| \leqslant \mu$, and so is its standard deviation, i.e., $\sqrt{\mathbb{E}\{\|\nabla f(\mathbf{w}, \mathbf{x}_n)\|^2\} - (\mathbb{E}\{\|\nabla f(\mathbf{w}, \mathbf{x}_n)\|\})^2} \leqslant \nu$.

Based on (2) and (3), $\mathcal{L}_K$ can be derived as

$$\begin{aligned} \mathcal{L}_K \leqslant & \gamma^2 \mathbb{E}\left\{\left\|\sum_{m=1}^{M} \frac{N_m}{N_F} \mathbf{w}_{K,\tau}^m - \mathbf{w}^*\right\|^2\right\} \\ \overset{(a)}{\leqslant} & \sum_{m=1}^{M} \frac{N_m}{N_F} \gamma^2 \underbrace{\mathbb{E}\left\{\left\|\mathbf{w}_{K,\tau}^m - \mathbf{w}^*\right\|^2\right\}}_{\beta_{K,\tau}^m}, \end{aligned} \quad (5)$$

where inequality (a) can be derived due to the convexity of mean square error. To derived a tractable upper bound of $\mathcal{L}_K$, the following lemma is first provided.

Lemma 1. *When the learning rate is set as* $\eta_{k,t} = \frac{1}{\lambda((k-1)\tau + t)}$, *an upper bound of* $\beta_{k,\tau}^m$ *in* (5) *can be expressed as follows:*

$$\begin{aligned} \beta_{k,\tau}^m \leqslant & \frac{(k-1)((k-1)\tau - 1)}{k(k\tau - 1)} \beta_{k,0}^m + \\ & \frac{1}{\lambda^2 k\tau (k\tau - 1)} \sum_{t=1}^{\tau - 1} \mathbb{E}\left\{\left\|\nabla F(\mathbf{w}_{k,t-1}^m, \mathcal{D}_{k,t}^m)\right\|^2\right\} + \\ & \frac{1}{\lambda^2 k^2 \tau^2} \mathbb{E}\left\{\left\|\nabla F(\mathbf{w}_{k,\tau-1}^m, \mathcal{D}_{k,\tau}^m)\right\|^2\right\}. \end{aligned} \quad (6)$$

Proof: Based on (1), $\beta_{k,t}^m$ can be expressed as

$$\begin{aligned} \beta_{k,t}^m = & \beta_{k,t-1}^m + \eta_{k,t}^2 \mathbb{E}\left\{\left\|\nabla F(\mathbf{w}_{k,t-1}^m, \mathcal{D}_{k,t}^m)\right\|^2\right\} - \\ & 2\eta_{k,t} \mathbb{E}\left\{\left\langle \nabla F(\mathbf{w}_{k,t-1}^m, \mathcal{D}_{k,t}^m), \mathbf{w}_{k,t-1}^m - \mathbf{w}^* \right\rangle\right\}. \end{aligned} \quad (7)$$

Due to the λ-strongly convexity of $F(\mathbf{w}_{k,t-1}^m, \mathcal{D}_{k,t}^m)$, and then the following inequality can be derived:

$$\mathbb{E}\left\{\left\langle \nabla F(\mathbf{w}_{k,t-1}^m, \mathcal{D}_{k,t}^m), \mathbf{w}_{k,t-1}^m - \mathbf{w}^* \right\rangle\right\} \geqslant \lambda \beta_{k,t-1}^m. \quad (8)$$

Then, $\beta_{k,t}^m$ can be rewritten as follows based on (8):

$$\beta_{k,t}^m \leqslant (1 - 2\lambda\eta_{k,t})\, \beta_{k,t-1}^m + \eta_{k,t}^2 \mathbb{E}\left\{\left\|\nabla F(\mathbf{w}_{k,t-1}^m, \mathcal{D}_{k,t}^m)\right\|^2\right\}. \quad (9)$$

Based on the recursion relationship with respect to $\beta_{k,t}^m$ and $\beta_{k,t-1}^m$ given by (9), $\beta_{k,\tau}^m$ can be derived as

$$\begin{aligned}\beta_{k,\tau}^m \leqslant & \eta_{k,\tau}^2 \mathbb{E}\left\{\left\|\nabla F(\mathbf{w}_{k,\tau-1}^m, \mathcal{D}_{k,\tau}^m)\right\|^2\right\} + \\ & \prod_{t=1}^{\tau}(1-2\lambda\eta_{k,t})\beta_{k,0}^m \\ & \sum_{t=1}^{\tau-1}\prod_{i=t+1}^{\tau}(1-2\lambda\eta_{k,i})\eta_{k,t}^2 \mathbb{E}\left\{\left\|\nabla F(\mathbf{w}_{k,t-1}^m, \mathcal{D}_{k,t}^m)\right\|^2\right\}.\end{aligned} \tag{10}$$

Then (6) can be obtained by substituting the learning rate $\eta_{k,t} = \frac{1}{\lambda((k-1)\tau+t)}$ into (10), and the proof has been finished. ∎

As shown in (6), $\beta_{k,\tau}^m$ is determined by $\mathbb{E}\{\|\nabla F(\mathbf{w}_{k,t-1}^m, \mathcal{D}_{k,t}^m)\|^2\}$. In particular, the empirical risk $\nabla F(\mathbf{w}_{k,t-1}^m, \mathcal{D}_{k,t}^m)$ is employed to estimate the expected risk, which can be expressed as follows:

$$\mu(\mathbf{w}_{k,t-1}^m) = \mathbb{E}_{\mathbf{x}\sim P(\mathbf{X})}\left\{\nabla f(\mathbf{w}_{k,t-1}^m, \mathbf{x})\right\}, \tag{11}$$

where $P(\mathbf{X})$ denotes the distribution of data element $\mathbf{x}$. Then, an upper bound of $\mathbb{E}\{\|\nabla F(\mathbf{w}_{k,t-1}^m, \mathcal{D}_{k,t}^m)\|^2\}$ can be given as follows.

Lemma 2. *An upper bound of* $\mathbb{E}\{\|\nabla F(\mathbf{w}_{k,t-1}^m, \mathcal{D}_{k,t}^m)\|^2\}$ *can be expressed as follows when* N_m *is large enough:*

$$\mathbb{E}\{\|\nabla F(\mathbf{w}_{k,t-1}^m, \mathcal{D}_{k,t}^m)\|^2\} \leqslant G(N_m) = 2\left(\frac{\nu^2}{N_m} + \mu^2\right), \tag{12}$$

where μ *and* ν *have been given before* (5)*.*

Proof: Due to the central limit theorem [7], the gap between the empirical risk and expected risk can be modeled as a multi-dimension Gaussian distributed random vector when N_m is large enough, i.e.,

$$\begin{aligned}&\mathbf{z} = \nabla F(\mathbf{w}_{k,t-1}^m, \mathcal{D}_{k,t}^m) - \mu(\mathbf{w}_{k,t-1}^m), \\ &\mathbf{z} \sim \mathcal{N}\left(\mathbf{0}, \frac{1}{N_m}\boldsymbol{\sigma}^2(\mathbf{w}_{k,t-1}^m)\right),\end{aligned} \tag{13}$$

where $\boldsymbol{\sigma}^2(\mathbf{w}_{k,t-1}^m)$ denotes the covariance matrix of $\nabla f(\mathbf{w}_{k,t-1}^m, \mathbf{x}_n)$. Based on (13), an upper bound of $\mathbb{E}\{\|\nabla F(\mathbf{w}_{k,t-1}^m, \mathcal{D}_{k,t}^m)\|^2\}$ can be derived as

$$\begin{aligned}&\mathbb{E}\{\|\nabla F(\mathbf{w}_{k,t-1}^m, \mathcal{D}_{k,t}^m)\|^2\} = \mathbb{E}\{\|\mathbf{z} + \mu(\mathbf{w}_{k,t-1}^m)\|^2\} \\ &\leqslant 2\mathbb{E}\{\|\mathbf{z}\|^2 + \|\mu(\mathbf{w}_{k,t-1}^m)\|^2\} \leqslant 2\left(\frac{\nu^2}{N_m} + \mu^2\right).\end{aligned} \tag{14}$$

Then Lemma 2 has been finished. ∎

By substituting (14) into (6), an upper bound of $\beta_{K,\tau}^m$ can be expressed as follows by setting $k = K$:

$$\beta_{K,\tau}^m \leqslant \frac{(K-1)((K-1)\tau-1)}{K(K\tau-1)}\beta_{K,0}^m + \frac{G(N_m)}{\lambda^2 K(K\tau-1)}, \tag{15}$$

In (15), $\beta_{K,0}^m$ can be replaced by $\mathcal{L}_{K-1}$ based on the fact that $\mathbf{w}_{K,0}^m$ is set as $\mathbf{w}_{K,0}^m = \mathbf{w}_{K-1}$ after the $(k-1)$-th round of federated averaging. Therefore, based on (5) and (15), a recursion relationship of $\mathcal{L}_K$ and $\mathcal{L}_1$ can be derived as

$$\mathcal{L}_K \leqslant \frac{(\tau-1)\mathcal{L}_1}{K(K\tau-1)} + \frac{K-1}{\lambda^2 K(K\tau-1)}\sum_{m=1}^{M}\frac{N_m}{N_F}G(N_m). \tag{16}$$

And $\mathcal{L}_1$ can be expressed as follows:

$$\mathcal{L}_1 \leqslant \sum_{m=1}^{M}\frac{N_m}{N_F}\beta_{1,\tau}^m \overset{(a)}{\leqslant} \sum_{m=1}^{M}\frac{N_m}{N_F}\cdot\frac{4G(N_m)}{\lambda^2\tau}, \tag{17}$$

where inequality (a) in (17) can be obtained based on Lemma 1 in [5]. By substituting (16) and (17) into (5), an upper bound of $\mathcal{L}_K$ can be obtained, which can be expressed as follows.

Corollary 3. *When the loss function is* γ*-Lipschitz and* λ*-strongly convex, a tractable upper bound of* $\mathcal{L}_K$ *can be derived as*

$$\mathcal{L}_K \leqslant \bar{\mathcal{L}}_K = \frac{\gamma^2(K+3)}{\lambda^2 K(K\tau-1)}\sum_{m=1}^{M}\frac{N_m}{N_F}G(N_m). \tag{18}$$

IV. Optimization Design of Federated Learning in MEC Systems

In federated learning, each round of interactions has communication cost, and the computation latency and energy consumption are also generated in the local model training phase. Therefore, how to achieve a trade-off between the training cost and model accuracy is an important issue we need to consider. In this section, a joint optimization problem for this issue is designed.

A. Problem Formulation

In federated learning, the computation resource consumption is mainly determined by the local model training, where the latency and energy consumption are proportional to the size of the training data set N_m, which can be expressed as follows respectively:

$$D_m = C_m^D N_m,\ E_m = C_m^E N_m, \tag{19}$$

where C_m^D and C_m^E denote the latency and energy consumption coefficients of processing unit volume of training data in S_m, respectively. The communication resource consumption is not considered in this paper, since it is determined by the transmission of model updates in federated learning, which is relatively small.

To keep the balance of the model accuracy and training costs, the upper bound of the accuracy loss and costs should be taken into account. Note that the local model training (from S_m to S_E) of each user can be accomplished in a parallel way, and thus the total latency is determined by the maximum latency of users. Therefore, the optimization problem can be established as follows:

$$\begin{aligned}&\min_{N_m, x_m} Q = \theta_{\mathcal{L}}\bar{\mathcal{L}}_K + \theta_D K \max_m\{x_m D_m\} + \theta_E K\sum_m x_m E_m, \\ &s.t. \sum_{m=1,\ldots,M} x_m = x_{th},\ x_m \in \{0,1\}, \\ &0 \leqslant N_m \leqslant N_m^{\max}, m = 1,\ldots,M,\end{aligned} \tag{20}$$

where $\theta_{\mathcal{L}}$, θ_D and θ_E denote the normalized weight coefficients of the upper bound of the accuracy loss $\bar{\mathcal{L}}_K$, latency and the energy consumption, respectively, the first set of constraints denote the user scheduling constraints, x_m denotes an indicator of whether or not U_m is scheduled, x_{th} denotes the user number must be scheduled for each round of interaction to ensure the model accuracy, the second set of constraints denote the storage resources constraints, and $N_m^{\max}$ denotes the maximum value of N_m.

B. Joint Optimization Algorithm

The joint optimization problem defined as (20) which is nonlinear and non-convex. To obtain a tractable solution, in this section, (20) is first decouple into two sub-optimization problems, and then an iterative algorithm is designed.

1) Optimization of the Local Batch Size: When we focus on the optimization of N_m, the objective function of N_m can be expressed as follows:

$$\begin{aligned} Q_1 =& \frac{\theta_{\mathcal{L}} C_{\mathcal{L}}}{\sum_{m=1}^{M} x_m N_m} + \theta_D K \cdot \max_m \left\{ C_m^D N_m \right\} + \\ & \theta_E K \cdot \sum_{m=1}^{M} C_m^E N_m, \end{aligned} \tag{21}$$

where $C_{\mathcal{L}} = \frac{2(K+3)\gamma^2\nu^2}{\lambda^2 K(K\tau-1)} x_{th}$. Since that $\max_m \left\{ C_m^D N_m \right\} \in \left\{ C_1^D N_1, \cdots, C_M^D N_M \right\}$, its optimal solution can be obtained by solving individually as M equivalent independent problems. Without loss of generality, we focus on the case when $C_i^D N_i = \max_m \left\{ C_m^D N_m \right\}$, and then the optimization problem of the local batch size can be written as follows based on (20):

$$\begin{aligned} \min_{N_m} \; & Q_{1,i} = \frac{\theta_{\mathcal{L}} C_{\mathcal{L}}}{\sum_{m=1}^{M} x_m N_m} + \theta_E K \cdot \sum_{m \neq i} C_m^E N_m + \\ & K \left(\theta_E C_i^E + \theta_D C_i^D \right) N_i \\ s.t. \; & 0 \leqslant N_m \leqslant N_m^{\max}, \; C_m^D N_m \leqslant C_i^D N_i, \; m \neq i. \end{aligned} \tag{22}$$

To efficiently approach the optimal solution, denoting that $y_1 = \sum_{m=1}^{M} x_m N_m$, then (22) can be expressed as

$$\begin{aligned} \min_{N_m, y_1} \; & Q_{1,i} = \frac{\theta_{\mathcal{L}} C_{\mathcal{L}}}{y_1} + \theta_E K \cdot \sum_{m \neq i} C_m^E N_m + \\ & K \left(\theta_E C_i^E + \theta_D C_i^D \right) N_i \\ s.t. \; & 0 \leqslant N_m \leqslant N_m^{\max}, m = 1, \ldots, M, \\ & y_1 = \sum_{m=1}^{M} x_m N_m, \; C_m^D N_m \leq C_i^D N_i, m \neq i. \end{aligned} \tag{23}$$

The convexity of (23) can be verified.

Then the optimum solution $\mathbf{N}_i^{op} = (\left\{ N_{i,1}^{op}, \ldots, N_{i,M}^{op} \right\})$ can be obtained by using KKT-conditions. Based on (23), Algorithm 1 can be designed to obtain the solution of (22).

2) Optimization of User Scheduling: When we focus on the optimization of user scheduling, the objective function can first be obtained as follows:

$$\begin{aligned} Q_2 =& \frac{\theta_{\mathcal{L}} C_{\mathcal{L},x} x_{th}}{\sum_{m=1}^{M} x_m N_m} + \theta_D K \max_m \{x_m D_m\} + \\ & \theta_E K \sum_{m=1}^{M} x_m E_m, \end{aligned} \tag{24}$$

where $C_{\mathcal{L},x} = \frac{2\gamma^2\nu^2(K+3)}{\lambda^2 K(K\tau-1)}$. Similar to (22), the optimal solution can be obtained by solving M equivalent problems. Without loss of generality, we focus on the case that $\arg\max_m \{x_m D_m\} = i$, that is $x_i = 1$, then the optimization problem of users scheduling can be expressed as follows based on (20):

$$\begin{aligned} \min_{x_m} \; & Q_{2,i} = \frac{\theta_{\mathcal{L}} C_{\mathcal{L}} x_{th}}{\sum_{m=1}^{M} x_m N_m} + \theta_D K D_i + \theta_E K \sum_{m=1}^{M} x_m E_m, \\ s.t. \; & \sum_{m=1}^{M} x_m = x_{th}, \; x_m D_m \leqslant D_i, \; x_i = 1. \end{aligned} \tag{25}$$

As shown in (25), it can be modeled as a $0-1$ nonlinear fractional programming. Based on the Theorem 1 in [8], denoting $r = Q_{2,i}$, the optimal solution of (25) can be obtained by solving the following problem:

$$\begin{aligned} \min_{x_m} \; & g(r) = \theta_{\mathcal{L}} C_{\mathcal{L}} x_{th} + (\theta_D K D_i - r) \sum_{m=1}^{M} x_m N_m + \\ & \theta_E K \sum_{m=1}^{M} \sum_{n=1}^{M} E_m N_n x_m x_n, \\ s.t. \; & \sum_{m=1}^{M} x_m = x_{th}, \; x_m D_m \leqslant D_i, \; x_i = 1. \end{aligned} \tag{26}$$

The problem given by (26) is a 0-1 quadratic programming problem. To solve this problem, the standard linearization method of quadratic programming is adopted [9]. Denoting

Algorithm 1 Optimization of the Local Batch Size

1: **Initialization:** $N_{1,m}(0), \ldots, N_{i,m}(0), \ldots, N_{M,m}(0)$, $i, m = 1, \ldots, M$, $T_{\max}$, and ε.
2: *Step 1: Solve M independent optimization problems given by* (22)*:*
3: **for** the i-th problem, **do**
4: **Repeat:**
5: **for** the t-th iteration, $1 \leqslant t \leqslant T_{\max}$ **do**
6: Update $N_{i,m}(t)$ by solving (23), $m = 1, \ldots, M$.
7: Update $Q_{1,i}(t)$ based on (21).
8: **end for**
9: **end for**
10: **Termination:** When $t > T_{\max}$ or $|Q_{1,i}(t) - Q_{1,i}(t-1)| < \varepsilon$.
11: **Return:** $\mathbf{N}_i^{op} = (\left\{ N_{i,1}^{op}, \ldots, N_{i,M}^{op} \right\})$, and $Q_{1,i}^{op} = Q_{1,i}(t)$.
12: *Step 2: Find the optimal solution of* (22)*:*
13: **Return:** $\mathbf{N}^{op} = \arg\min_{\mathbf{N}_1^{op}, \ldots, \mathbf{N}_M^{op}} \{Q_1\}$ as the final result.

Algorithm 2 Optimization of User Scheduling

Initialization: $r(0)$, $S_{\max}$, and ε.
Step 1: Solve M independent optimization problems given by (25)*:*
for the i-th problem, **do**
  Repeat:
  for the s-th iteration, $1 \leqslant s \leqslant S_{\max}$ **do**
    Update $\mathbf{x}_i(s)$ by solving (27)
    Update $r_i^*(s) = Q_{2,i}(\mathbf{x}_i(s))$;
    Update $g(r_i^*(s))$ based on (26)
  end for
end for
Termination: When $s > S_{\max}$ or $|g(r_l^*(s))| < \varepsilon$.
Return: $\mathbf{x}_i$, and $Q_{2,i} = r_i^*(s)$.
Step 2: Find the optimal solution of (25)*:*
Return: $\mathbf{x}^* = \arg\min_{\mathbf{x}_1^*,\ldots,\mathbf{x}_M^*} Q_2$ as the final result.

$y_{m,n} = x_m x_n$, an equivalent optimization problem of (26) can be obtained as follows:

$$\begin{aligned} \min_{x_m} \overline{g}(r) &= \theta_{\mathcal{L}} C_{\mathcal{L}} x_{th} + (\theta_D K D_i - r) \sum_{m=1}^{M} x_m N_m + \\ &\quad \theta_E K \sum_{m=1}^{M} \sum_{n=1}^{M} E_m N_n y_{m,n}, \\ s.t.\ &\sum_{m=1}^{M} x_m = x_{th},\ x_m D_m \leqslant D_i,\ x_i = 1, \\ &\quad y_{m,n} \leqslant x_m,\ y_{m,n} \leqslant x_n,\ y_{m,n} \geqslant x_m + x_n - 1. \end{aligned} \tag{27}$$

Then (26) has been transformed as a 0-1 linear programming problem expressed by (27). Its optimal solution can be obtained efficiently by using various calculation softwares, such as the PuLP package in Python.

Based on (26) and (27), Algorithm 2 can be designed to obtain the optimal solution of (25).

3) A Joint Optimization Algorithm: To obtained the optimal solution of (20), a joint optimization algorithm is proposed by optimizing the aforementioned two subproblems iteratively. As shown in Algorithm 3, during each iteration, the local batch size optimization subproblem can be solved by using Algorithm 1, and the user scheduling can be updated based on Algorithm 2. As introduced previously, since the solutions of all the subproblems are optimal, the total cost Q can be decreased as the iteration index increases. Moreover, Q can be lower bounded by a fixed minimum value 0, and thus the convergence of Algorithm 3 can be guaranteed with limited iterations.

The computational complexity is mainly determined by solving studied subproblems during each iteration, which can be estimated as follows:

- The computational complexity of Algorithm 1 can be expressed as $O(M((M+1)^3 t_1 + t_2))$, where $O((M+1)^3 t_1)$ is the computational complexity of solving (23), and t_2 is the computation time of comparing $\mathbf{N}_i^{op}$ and $\mathbf{N}_j^{op}$.

Algorithm 3 The Joint Optimization Algorithm

Initialization: $x_m(0)$, $N_m(0)$, the maximum number of iterations $J_{\max}$, and the convergence threshold ε.
Repeat: For the j-th iteration, $1 \leqslant j \leqslant J_{\max}$
Update $N_m(j)$ based on Algorithm 1, $m = 1, \ldots, M$;
Update $x_m(j)$ based on Algorithm 2, $m = 1, \ldots, M$;
Update $Q(j)$ based on (20).
Termination: When $j > J_{\max}$ or $|Q(j) - Q(j-1)| \leqslant \varepsilon$.

- The computational complexity of Algorithm 2 can be expressed as $O[M(S_{\max}((3(M-1)^2 - M + 2)t_3 + t_4) + t_5)]$, where $O[(3(M-1)^2 - M + 2)t_3]$ is the computational complexity of solving (27), t_4 denotes the computation time of updating $r_i^*(s)$ and $g(r_i^*(s))$ of each iteration, and t_5 is the computation time of comparing $\mathbf{x}_i^*$ and $\mathbf{x}_j^*$.

Therefore, the computational complexity of Algorithm 3 of each iteration can be estimated as $O(M((M+1)^3 t_1 + t_2) + M(S_{\max}((3(M-1)^2 - M + 2)t_3 + t_4) + t_5))$, which is a polynomial function with respect to the number of users M.

V. Experiment Results

In this section, the experiment results of our proposed joint optimization algorithm are presented. In particular, we assume that there are 600 data elements in the local data set $\mathcal{D}_m$ of each user, each time during local model training, $\tau = 10$ different batches of data are sampled from $\mathcal{D}_m$, and the maximum time of federated averaging is set as $K = 500$.

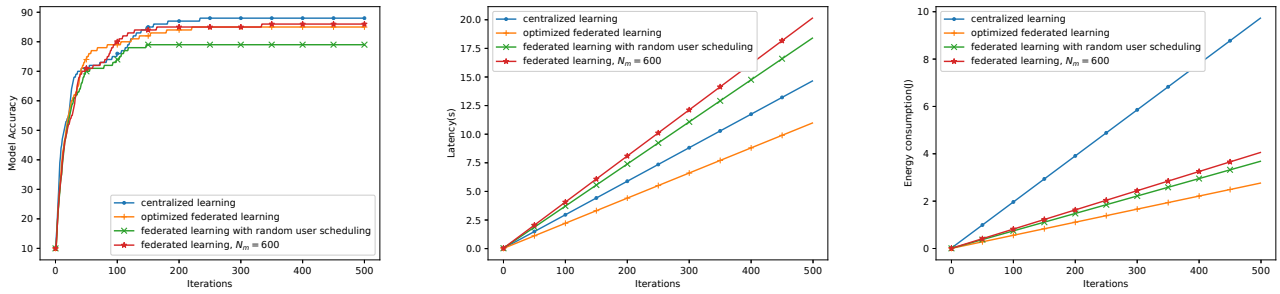
A. The Convergence Performance of Algorithm 3

To evaluate the convergence performance of Algorithm 3, the numerical simulation results are presented in this part. Recalling Algorithm 3, its convergence is determined jointly by the maximum number of iterations $J_{\max}$ and the convergence threshold ε. As shown in Fig. 1, both the accuracy loss and the cost of the federated learning keep decreasing as the iteration index increases, where the total numbers of users are set as $M = 5, 10, 15, 20, 25$, and the corresponding numbers of scheduled users are set as $x_{th} = 2, 5, 6, 10, 15$, respectively. Moreover, the simulation results show that in finite number of iterations, Algorithm 3 can converge to the stationary optimal solution.

B. Experiment Results on MNIST Data Set

In this part, the MNIST data set is employed to practically testify the performance of Algorithm 3. The experiment results are provided in Fig. 2, where the MLP model is employed for local model training.

As shown in the figure, the federated learning scheme optimized by Algorithm 3 can significantly improve the test accuracy compared to the federated learning scheme with random user scheduling. In Fig. 2(a), for the comparison between the two schemes, the test accuracy can be improved by 7.6%. Moreover, the test accuracy of the federated learning scheme optimized by Algorithm 3 is very close to that of the federated learning scheme without local batch size optimization, which



(a) MLP learning model. (b) Latency. (c) Energy consumption.

Fig. 2. The experiment results of optimized federated learning with Algorithm 3.

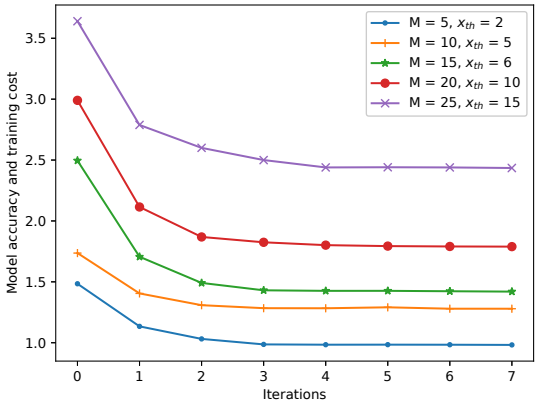


Fig. 1. Convergence of Algorithm 3.

uses all 600 data elements in the local data set for local model training. In Fig. 2(a), the difference between the two is only 1%. In addition, after being optimized by using Algorithm 3, the performance of federated learning can approach the centralized learning scheme, indicating that federated learning can generate high-quality models.

In the Fig. 2(b) and Fig. 2(c), the latency and energy consumption results of the model training process under the four learning schemes based on the MLP model is presented respectively. In Fig. 2(b) and Fig. 2(c), $\mathcal{S}_{F,1}$ reduces the latency by 25.1% and energy consumption by 71.6% respectively compared to the centralized learning scheme, reduces the latency by 40.3% and energy consumption by 25.0% respectively compared to the federated learning scheme with random user scheduling, and reduces the latency by 45.5% and energy consumption by 31.9% respectively compared to the federated learning scheme without local batch size optimization.

VI. Conclusion

In this paper, we study the design of federated learning in MEC systems. First, we derive a tractable upper bound of the accuracy loss, and formulate an optimization problem. By managing the storage resources and user scheduling, a joint optimization algorithm is provided to balance model accuracy and costs. We also provide the experiment results to evaluate the performance of the proposed optimization algorithm, which show that our proposed algorithm can improve the model accuracy of federated learning while significantly reducing the computation and communication costs.

VII. Acknowledgment

This work was supported in part by Beijing Natural Science Foundation (Grant L182039), and National Natural Science Foundation of China (Grant 61971061).

References

[1] G. Yu, J. Zhang, V. C. M. Leung, M. Kountouris, and C. Wang"IEEE access special section editorial: Mobile edge computing for wireless networks," *IEEE Access*, vol. 6, pp. 11439-11442, 2018.

[2] H. B. Mcmahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," In *Proc. 20th International Conference on Artificial Intelligence and Statistics*, Fort Lauderdale, FL, USA, pp. 1273-1282, Apr. 2017.

[3] J. Konecny, "Stochastic, Distributed and Federated Optimization for Machine Learning," arXiv preprint, [Online]. Avaliable: https://arxiv.org/pdf/1707.01155, 2017.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, Cambridge, MA, USA: MIT Press, 2016.

[5] A. Rakhlin, O. Shamir, and K. Sridharan, "Making Gradient Descent Optimal for Strongly Convex Stochastic Optimization," In *Proc. the 29th International Conference on Machine Learning*, Edinburgh, Scotland, pp. 1571-1578, Jun. 2012.

[6] O. Shamir and T. Zhang, "Stochastic gradient descent for non-smooth optimization: convergence results and optimal averaging schemes," in *Proc. the 30th International Conference on Machine Learning*, Atlanta, USA, pp. 71-79, Jun. 2013.

[7] R. G. Laha and V. K. Rohatgi, *Probability Theory*, New York, USA: Wiley, 1979.

[8] Z. Zhao, S. Bu, T. Zhao, Z. Yin, M. Peng, Z. Ding, and T. Q. S. Quek, "On the design of computation offloading in fog radio access networks," *IEEE Transaction on Vehicular Technology*, vol. 68, no. 7, pp. 7136-7149, Jul. 2019.

[9] A. Caprara, "Constrained 0-1 quadratic programming: Basic approaches and extensions, " *European Journal of Operational Research*, vol. 187, no. 3, pp. 1494-1503, Jun. 2008.