Computation Offloading for Edge-Assisted Federated Learning

Zhongming Ji, Li Chen, Nan Zhao, *Senior Member, IEEE,* Yunfei Chen, *Senior Member, IEEE,* Guo Wei and F. Richard Yu, *Fellow, IEEE*

***Abstract*—When applying machine learning techniques to the Internet of things, aggregating massive amount of data seriously reduce the system efficiency. To tackle this challenge, a distributed learning framework called federated learning has been proposed. Due to the parallel training structure, the performance of federated learning suffers from the straggler effect. In this paper, to mitigate the straggler effect, we propose a novel learning scheme, edge-assisted federated learning (EAFL), which utilizes edge computing to reduce the computational burdens for stragglers in federated learning. It enables stragglers to offload partial computation to the edge server, and leverages the server's idle computing power to assist clients in model training. The offloading data size is optimized to minimize the learning delay of the system. Based on the optimized data size, a threshold-based offloading strategy for EAFL is proposed. Moreover, we extend EAFL to a dynamic scenario where clients may be offline after several update rounds. By grouping clients into different sets, we formulate the new EAFL delay optimization problem and derive the corresponding offloading strategy for the dynamic scenario. Simulation results are presented to show that EAFL has lower system delay than the original federated learning scheme.**

***Index Terms*—Computation offloading, delay analysis, edge computing, federated learning.**

I. Introduction

To analyze data and extract useful information for intelligent applications, machine learning techniques have been widely applied in the Internet of things (IoT) scenarios [1], [2]. However, the data distributed at the network edge is exploding exponentially. It has been predicted that there will be 30 billion networked devices by 2023, generating nearly 100 zettabytes of distributed data [3]. This will incur a huge system delay for centralized machine learning schemes, which aggregate data from a massive number of distributed devices for training.

To cope with the delay problem, a novel distributed learning framework called *federated learning* has been proposed in [4]. In federated learning, each participating device (referred to as client) performs model training based on its local dataset, and only the model parameters are uploaded, which avoids the raw data aggregation. Many research efforts have been made in federated learning. In [5], the authors provided an effective strategy to improve the learning performance on non-IID data by distributing a small subset of globally shared data among clients. The convergence bound of federated learning was analyzed in [6], based on which an adaptive parameter aggregation frequency control algorithm was proposed. The work of [7] presented a scalable production system for federated learning to tackle the practical problems, such as client availability and unreliable client connectivity. In [8], to address statistical challenges (*e.g.*, unbalanced data) and system challenges (*e.g.*, fault tolerance) of federated learning, the authors proposed a system-aware optimization method based on the multi-task learning framework. In [9], the authors applied federated learning to vehicular networks, where a selective model aggregation approach was proposed to abandon vehicular clients with low data quality and power capability. To further improve the federated learning efficiency, various methods for reducing the communication overhead of parameter aggregation were provided in [10]–[12]. To optimize the learning performance given a fixed resource budget, different client scheduling policies for federated learning were proposed in [13]–[15]. Since the client dataset information can still be revealed through analyzing the distributed model, some privacy preservation strategies for federated learning like differential privacy were investigated in [16]–[18].

Although the parallel training structure greatly improves the system efficiency, the performance of federated learning is constrained by slow clients, *i.e.*, the straggler effect [19], [20]. To mitigate the straggler effect in federated learning, researchers have proposed two approaches: asynchronous update [21], [22] and coded computation [23], [24]. For the asynchronous update approach, each client uploads the model parameter to the server immediately after training without waiting for other clients. In [21], the authors proposed an asynchronous online federated learning framework, where clients performed learning with continuous streaming local data in an asynchronous way. In [22], the authors designed a novel asynchronous federated optimization algorithm and provided some strategies to reduce the error caused by the asynchronous update. For the coded computation approach, the core idea is to embed computation redundancy to compensate the performance degradation caused by stragglers. The work in [23] applied coded computation to federated learning and

This research was supported by National Natural Science Foundation of China (Grant No. 62071445), and the Fundamental Research Funds for the Central Universities (Grant No. WK3500000007 and Grant No. YD3500002001) (*Corresponding author: Li Chen*)

Z. Ji, L. Chen and G. Wei are with CAS Key Laboratory of Wireless-Optical Communications, University of Science and Technology of China. (e-mail: jzhongm@mail.ustc.edu.cn, {chenli87, wei}@ustc.edu.cn).

N. Zhao is with the School of Information and Communication Engineering, Dalian University of Technology, Dalian 116024, China, and also with the National Mobile Communications Research Laboratory, Southeast University, Nanjing 210096, China (e-mail: zhaonan@dlut.edu.cn).

Y. Chen is with the School of Engineering, University of Warwick, Coventry CV4 7AL, U.K. (e-mail: Yunfei.Chen@warwick.ac.uk).

F.R. Yu is with the Department of Systems and Computer Engineering, Carleton University, Ottawa, ON, K1S 5B6, Canada (email: richard.yu@carleton.ca).

proposed to share the coded training data with the server to compensate the delayed straggler updates, while also preserving the data privacy well. To against straggling communication links for the gradient aggregation in distributed learning, a hierarchical aggregation structure was proposed in [24], where the authors also applied two coding approaches: repetition coding and minimum distance separable coding.

On the other hand, *multi-access edge computing* (MEC), formerly mobile edge computing, has attracted great attention from both academia and industry. MEC is defined as a platform that harvests the idle edge computation power to perform computing tasks for mobile devices [25]. With the MEC technology, stragglers in federated learning can offload computation to the edge server to relieve their computational burdens and reduce their update delay. Various works of MEC have discussed the computation offloading strategy for different application scenarios [26]–[30]. In [26], the authors proposed a one-dimensional computation offloading policy searching algorithm based on the task buffer queueing state and the channel characteristic. In [27], the authors extended the offloading problem to a multi-user scenario and adopted a game theoretic approach for achieving efficient offloading. The partial offloading problem was investigated in [28], where the authors provided both energy-optimal and delay-optimal offloading strategies using dynamic voltage scaling. The partial offloading problem was also extended to a multi-user scenario in [29], where the authors presented energy-efficient resource allocation policies for TDMA and OFDMA systems. The offloading for practical asynchronous MEC systems with non-identical task-arrivals and deadlines was investigated in [30], where the authors derived the optimal data-partitioning and time-division policies for energy minimization.

Many existing works have applied federated learning in different aspects of the MEC system, such as guiding the network resource allocation [31], optimizing the network routing [32], or reducing the communication overhead of learning tasks [33]. However, in existing works, applying MEC offloading to federated learning for straggler effect mitigation is largely ignored. This will bring up two challenges. Firstly, an edge-assisted federated learning design in which the server also performs model training is missing. Related works on federated learning usually take the edge server as a parameter aggregator or a scheduler. The powerful computing ability of the server has not been applied in the model training. Secondly, the existing offloading strategy is not suitable for edge-assisted federated learning, as they are mainly derived based on the general computing task model, which is different from federated learning due to the additional iterative process.

Motivated by the above observations, we propose a novel edge-assisted federated learning (EAFL) scheme. In EAFL, clients offload partial computation to an edge server, and the server then participates in the model training together with all clients. By offloading, the straggler effect caused by the parallel training structure is mitigated, which improves the overall efficiency of federated learning. The performance of EAFL is analyzed in terms of system delay. We formulate an EAFL delay minimization problem to optimize the offloading data size for each client. According to the solution of the problem, we derive a threshold-based offloading strategy for EAFL. The strategy determines non-offloading and optimal partial offloading for clients with offloading decision indicator below and above a given threshold, respectively. Furthermore, considering the unstable nature of clients in federated learning, we also extend EAFL to a dynamic scenario where some clients may be offline after several update rounds. Using grouping, an offloading strategy for the dynamic scenario is also derived by solving the corresponding delay minimization problem. The main contributions of this paper are summarized as follows:

- The EAFL design and delay analysis: A new EAFL design is proposed from the perspectives of training procedure and loss function. For the new design, we analyze its system delay.
- Offloading strategy for EAFL: A delay minimization problem is formulated under the EAFL framework. By solving the minimization problem, a threshold-based offloading strategy is proposed for EAFL.
- EAFL for dynamic scenarios: We extend EAFL to a dynamic scenario. By grouping dynamic clients to different client sets, we formulate the delay minimization problem for dynamic EAFL and derive the corresponding offloading strategy.

The rest of the paper is organized as follows. Section II will introduce the system design of EAFL. Section III will present the EAFL delay optimization problem and the offloading strategy for EAFL. The extension to dynamic scenarios will be analyzed in Section IV. Simulation results will be given in Section V, followed by the conclusion in Section VI.

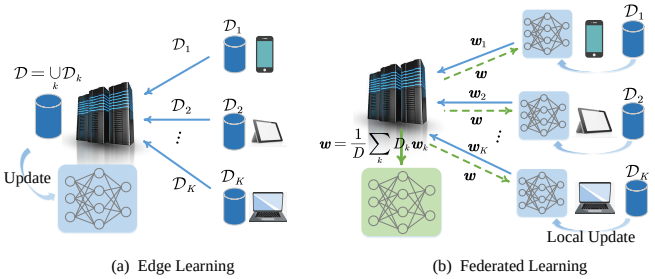
II. Edge-Assisted Federated Learning

Consider a distributed machine learning scenario with a server at the network edge and K clients, denoted by a set $\mathcal{K} = \{1, 2, \cdots, K\}$. Each client k stores a local dataset $\mathcal{D}_k$ and its size is denoted by D_k. In the dataset, a data sample i usually consists of the input vector $\boldsymbol{x}_i$ (*e.g.*, the pixels of an image) and the output scalar y_i (*e.g.*, the label of the image). In a typical machine learning problem, for sample i, the task is to find the model parameter $\boldsymbol{w}$ that characterizes the relationship between $\boldsymbol{x}_i$ and y_i by minimizing the loss function $f_i(\boldsymbol{w})$. The loss function captures the error of the model on the training data, for example, $\frac{1}{2}||y_i - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i||^2$ for linear regression and $\max\{0, 1 - y_i\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i\}$ for support vector machine.

Due to the inherent complexity of most machine learning models, minimizing the loss function is usually performed by using gradient-descent techniques. In this paper, following the method in [4], we take the mini-batch gradient descent algorithm as our model training algorithm. In the distributed scenario, to get a final trained model, there are two existing learning schemes: edge learning and federated learning. In this section, we will briefly discuss these two learning schemes before proposing edge-assisted federated learning design.

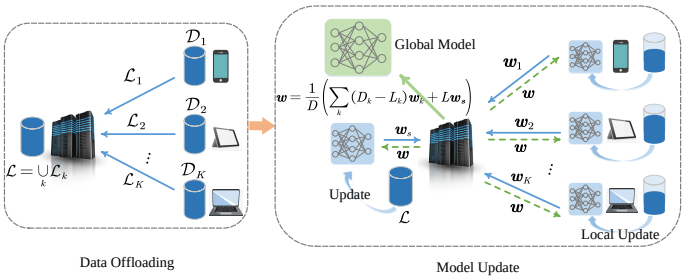
A. Edge Learning

As illustrated in Fig. 1(a), edge learning requires all clients to offload their local datasets to the edge server, who will



(a) Edge Learning (b) Federated Learning

Fig. 1. Two learning schemes in distributed scenarios.



Data Offloading Model Update

Fig. 2. Illustration of the EAFL design.

then perform model training based on the aggregated dataset. We define $\mathcal{D} = \cup_k \mathcal{D}_k$ and $D = \sum_{k\in\mathcal{K}} D_k$. The global loss function at the server is defined as

$$F(\boldsymbol{w}) = \frac{1}{D}\sum_{i\in\mathcal{D}} f_i(\boldsymbol{w}). \tag{1}$$

The learning problem in this scheme is to find

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} F(\boldsymbol{w}). \tag{2}$$

From the procedure of edge learning, its system delay consists of the data offloading delay and the model training delay. The communication links from K clients to the server are assumed to be orthogonal. Then the system delay of edge learning can be denoted as

$$T_{EL} = \max_{k\in\mathcal{K}} \frac{D_k}{B\log_2(1+p_k g_k)} + \frac{W}{e_s}, \tag{3}$$

where the first expression on the right side of the equation is the data offloading delay and the second expression on the right side denotes the model training delay in the server. In the data offloading delay, B denotes the bandwidth for each link, p_k denotes the transmission power allocated to client k, $g_k = \tilde{g}_k/N_0$, N_0 is the variance of complex white Gaussian channel noise, $\tilde{g}_k = \nu|h_k|^2$ is client k's channel gain, ν is large-scale fading path loss parameter and h_k is the channel response that is assumed to be constant during the data offloading. In the model delay, W denotes the total computation (*i.e.*, the number of CPU cycles) of the server during the model training phase and e_s is the CPU's frequency of the sever. According to the mini-batch gradient descent algorithm used in [4], the computation W is proportional to the dataset size D and is given as

$$W = N_e B_n C B_s = N_e \left(\frac{D}{B_s}\right) C B_s = N_e C D, \tag{4}$$

where N_e is the total epoch number, B_n denotes the iteration number (mini-batch number) in one epoch, B_s is the mini-batch size (training data size of one iteration) and the constant C denotes the number of CPU cycles required for training 1-bit data.

B. Federated Learning

In federated learning, as illustrated in Fig. 1(b), clients perform local updates based on their local datasets and only their model parameters are uploaded to the server. The server aggregates these updated parameters to obtain a global model and then broadcasts it back to clients for the next local update. Before the parameter aggregation, each client may perform one or multiple epochs of model training during the local update phase and we define this duration as a round. For client k, the loss function is

$$F_k(\boldsymbol{w}_k) = \frac{1}{D_k}\sum_{i\in\mathcal{D}_k} f_i(\boldsymbol{w}_k), \tag{5}$$

where $\boldsymbol{w}_k$ denotes client k's local model parameter. According to the FedAvg algorithm proposed in [4], the global model parameter is defined as

$$\boldsymbol{w} = \frac{1}{D}\sum_{k\in\mathcal{K}} D_k \boldsymbol{w}_k. \tag{6}$$

For federated learning, the system delay is composed of the local update delay, the parameter aggregation delay and the model broadcast delay. After each round of local update, we reserve a fixed time slot t_w for parameter aggregation and model broadcast. The time slot can be adjusted according to the chosen model size. Since each client independently performs local update, the system delay of federated learning can be denoted as:

$$T_{FL} = N \max_{k\in\mathcal{K}} \frac{W_k}{e_k} + N t_w, \tag{7}$$

where N denotes the total rounds in federated learning, W_k denotes the computation of client k per round, e_k is the CPU's frequency of client k and the first expression on the right side of the equation denotes client k's total update delay. The computation of client k in a round is proportional to its dataset size and is given as

$$W_k = \tau B_n^k C B_s = \tau \left(\frac{D_k}{B_s}\right) C B_s = \tau C D_k, \tag{8}$$

where τ denotes the epoch number per round and B_n^k is the batch number of client k.

C. The Proposed Scheme: EAFL

From the perspective of system delay, the performance of the above two schemes is mainly limited by communication or computation. For edge learning, uploading the whole datasets requires huge communication resources and relies on the communication environment. For federated learning, the client with low computational capability or large dataset size will

TABLE I
SUMMARY OF MAIN NOTATIONS IN EAFL.

Notation	Description
$\mathcal{K}$, K	Total client set, size of $\mathcal{K}$
$\mathcal{K}_i$, K_i	Client set in the i-th update segment of dynamic EAFL, size of $\mathcal{K}_i$
D_k, L_k	Dataset size of client k, offloaded data size of client k
$D_{k,i}$, $L_{k,i}$	Dataset size of client k in set $\mathcal{K}_i$, offloaded data size of client k in set $\mathcal{K}_i$
$\boldsymbol{w}$, $\boldsymbol{w}_k$, $\boldsymbol{w}_s$	Global model parameter, model parameter of client k, model parameter of the server
$f(i)$, F_k, F	Loss function of sample i, loss function in client k, loss function in the server
P, C, B	Total power constraint, number of CPU cycles required for training 1-bit data, transmission bandwidth for each link
τ, ϵ	epoch number per round, target training loss
t_k^o, t_k^u, t_s, t_w	Offloading delay of client k, update delay of client k, server update delay, time slot for parameter aggregation and model broadcast
N, N_i	Total training rounds, offline round of segment i in dynamic EAFL
p_k^o, p_k^u	Transmission power, computational power allocated to client k
$p_{k,i}^o$, $p_{k,i}^u$	Transmission power, computational power allocated to client k in set $\mathcal{K}_i$
g_k, ζ_k	Equivalent channel gain, effective capacitance coefficient of client k
$g_{k,i}$, $\zeta_{k,i}$	Equivalent channel gain, effective capacitance coefficient of client k in set $\mathcal{K}_i$ of dynamic EAFL
e_k, e_s	CPU frequency of client k, CPU frequency of the server
α, β	Parameters related to the federated learning configurations and data distribution
a, b, d	Variables related to the offloading data size
$\widehat{a}$, $\widehat{b}$, $\widehat{d}_j$	Variables related to the offloading data size in dynamic EAFL
φ_k, Π	Offloading decision indicator of client k, offloading decision threshold
$\varphi_{k,i}$, Π_i	Offloading decision indicator of client k, offloading decision threshold in set $\mathcal{K}_i$ in dynamic EAFL

slow down the overall performance, *i.e.*, there exists the straggler effect. These observations motivate us to propose edge-assisted federated learning (EAFL), which achieves a tradeoff between edge learning and federated learning.

As shown in Fig. 2, EAFL consists of two phases: data offloading and model update. Each client k first offloads partial data $\mathcal{L}_k$ (L_k denotes its size) to the server for processing. During the model update phase, the remaining data in each client will be used to update the local model. Meanwhile, the server will utilize the aggregated data to update the model in itself. Note that the model update in the server needs to wait until the data offloading phase is completed. Hence, the model update phase is set to be performed after the data offloading in the proposed EAFL. Although the client model update in the first round can be performed in parallel with the data offloading, we ignore the slight delay changes caused by the parallel processing and assume that the local update is performed after the computation offloading.

For the model in the server, the loss function defined on the offloaded data is given as

$$F(\boldsymbol{w}_s) = \frac{1}{L}\sum_{i\in\mathcal{L}} f_i(\boldsymbol{w}_s), \tag{9}$$

where $\boldsymbol{w}_s$ denotes model parameter in the server, $\mathcal{L} = \cup_k \mathcal{L}_k$ denotes the set of all offloaded data and $L = \sum_{k\in\mathcal{K}} L_k$ is its size. For the local model in client k, the loss function defined on its remaining data is

$$F_k(\boldsymbol{w}_k) = \frac{1}{D_k - L_k}\sum_{i\in\mathcal{D}_k\setminus\mathcal{L}_k} f_i(\boldsymbol{w}_k). \tag{10}$$

After the local update in clients and the sever, there also exists a global aggregation process, where the server aggregates all client parameters and its own model parameters to obtain a global model and broadcast it to all clients. We define the global model parameter as

$$\boldsymbol{w} = \frac{1}{D}\left(\sum_{k\in\mathcal{K}}(D_k - L_k)\boldsymbol{w}_k + L\boldsymbol{w}_s\right). \tag{11}$$

Under the EAFL framework, the server and all clients are required to perform the model training during the model update phase. Compared with the original federated learning proposal, EAFL adds the training data offloading and the model training in the server.

Remark 1 (The Convergence Performance of EAFL)**.** *In our EAFL design, the server assists clients to update model together, which is similar to adding a new client on the basis of the original federated learning (ignoring communication aspects such as data offloading). Hence, the convergence performance of EAFL is close to federated learning. In particular, for the case of $\tau = 1$ and full-batch size, the convergence performance of EAFL is equivalent to the centralized gradient descent algorithm. After the parameter broadcast in each global aggregation phase, we have $\boldsymbol{w}_k(t) = \boldsymbol{w}_s(t) = \boldsymbol{w}(t)$, where $t = 0, 1, 2\cdots$ denotes the local update index. According to (11) and the linearity of the gradient operator, we have*

$$\begin{aligned}
&\boldsymbol{w}(t) = \frac{1}{D}\left[\sum_{k\in\mathcal{K}}(D_k - L_k)\boldsymbol{w}_k(t) + L\boldsymbol{w}_s(t)\right]\\
=&\frac{\sum_{k\in\mathcal{K}}(D_k - L_k)(\boldsymbol{w}_k(t-1) - \eta\nabla F_k(\boldsymbol{w}_k(t-1)))}{D}\\
&+ \frac{L(\boldsymbol{w}_s(t-1) - \eta\nabla F_s(\boldsymbol{w}_s(t-1)))}{D}\\
=&\frac{\sum_{k\in\mathcal{K}}(D_k - L_k)\boldsymbol{w}(t-1) + L\boldsymbol{w}(t-1)}{D}\\
&- \frac{\eta\left[\sum_{k\in\mathcal{K}}(D_k - L_k)\nabla F_k(\boldsymbol{w}(t-1)) + L\nabla F_s(\boldsymbol{w}(t-1))\right]}{D}\\
=&\frac{\sum_{k\in\mathcal{K}} D_k\boldsymbol{w}(t-1)}{D}\\
&- \eta\nabla\left[\frac{\sum_{k\in\mathcal{K}}(D_k - L_k)F_k(\boldsymbol{w}(t-1)) + LF_s(\boldsymbol{w}(t-1))}{D}\right]\\
=&\boldsymbol{w}(t-1) - \eta\nabla F(\boldsymbol{w}(t-1)).
\end{aligned} \tag{12}$$

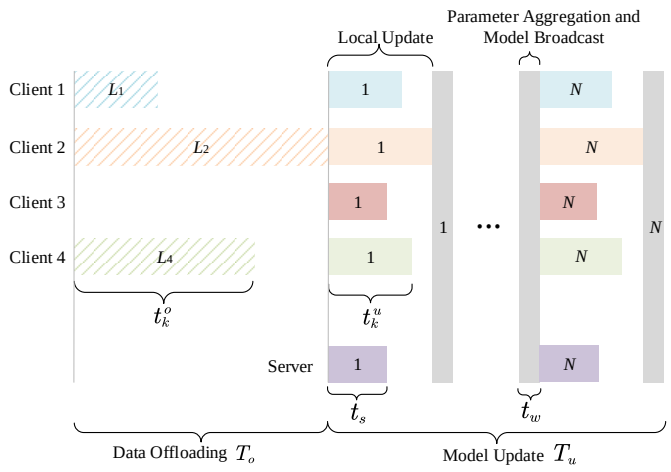


Fig. 3. Illustration of EAFL delay.

One can observe that the above recurrence relation is same as that in the centralized gradient descent algorithm.

Remark 2 (The Advantages of EAFL)**.** *On the one hand, by offloading partial data to the server, EAFL can effectively relieve the computational burdens at clients and mitigate the straggler effect in federated learning. On the other hand, unlike edge learning, EAFL does not need to offload the whole dataset, avoiding the large communication overhead. Therefore, EAFL can achieve a tradeoff between edge learning and federated learning to take advantage of their respective benefits.*

III. Offloading Strategy for EAFL

In this section, we will first analyze the system delay of EAFL. Based on the analysis, we will formulate a power-constrained delay minimization problem. By solving the optimization problem, we will derive an offloading strategy for EAFL. Note that the analysis of EAFL delay is based on the assumptions in edge learning and federated learning discussed in Section II.

A. Problem Formulation

As illustrated in Fig. 3, EAFL delay T_{EAFL} consists of two kinds of delay, *i.e.*, data offloading delay T_o and model update delay T_u. We have

$$T_{EAFL} = T_o + T_u. \tag{13}$$

The offloading delay is denoted as

$$T_o = \max_{k\in\mathcal{K}} \ t_k^o, \tag{14}$$

where t_k^o is the offloading delay of client k. Referring to (3), we have

$$t_k^o = \frac{L_k}{R_k} = \frac{L_k}{B\log_2(1+p_k^o g_k)}, \tag{15}$$

where p_k^o is the transmission power allocated to client k. The model update delay in EAFL is given as

$$T_u = N\max\left\{\max_{k\in\mathcal{K}} t_k^u,\ t_s\right\} + Nt_w, \tag{16}$$

where t_k^u is client k's update delay, t_s is the server update delay and t_w denotes the fixed time slot for parameter aggregation and model broadcast. Referring to (8), we have

$$t_k^u = \frac{\tau C(D_k - L_k)}{e_k} \tag{17}$$

and

$$t_s = \frac{\tau C\sum_{k\in\mathcal{K}} L_k}{e_s}, \tag{18}$$

where e_k denotes client k's CPU frequency and e_s denotes the server's CPU frequency.

For client k, as in [34], the relationship between computational power p_k^u and frequency e_k can be calculated as

$$e_k = \sqrt{\frac{p_k^u}{\zeta_k}}, \tag{19}$$

where $\zeta_k > 0$ is client k's effective capacitance coefficient depending on chip architecture. As shown in [35], the number of rounds N required to attain a certain training loss ϵ in federated learning can be calculated as $N = r(\epsilon)$, where $r(\epsilon)$ is defined as

$$r(\epsilon) = \left[\left(1+\frac{1}{K}\right)\tau\alpha + \frac{\beta}{\tau}\right]\frac{1}{\epsilon}. \tag{20}$$

α and β are parameters related to the federated learning configurations and data distribution. According to [35], the magnitude of β reflects the heterogeneity of the data distribution. A large β means a large degree of non-IID. Since the values of α and β can not be derived during the EAFL process, we refer to [36] and obtain the parameters by extrapolation. More specifically, based on the historical data of federated learning, we establish the relationship between the learning rounds and data characteristics, and then perform curve fitting to derive the values of α and β.

Assume the total transmission power and the total computational power of clients in EAFL are both constrained by P. According to (13)-(20), we can formulate the EAFL delay minimization problem as follows.

$$\begin{aligned}
\text{P1}: \min_{L_k,p_k^o,p_k^u} & \left\{\max_{k\in\mathcal{K}} \frac{L_k}{B\log_2(1+p_k^o g_k)} + r(\epsilon)t_w \right.\\
& \left. + r(\epsilon)\tau C\max\left\{\max_{k\in\mathcal{K}}(D_k-L_k)\sqrt{\frac{\zeta_k}{p_k^u}},\ \frac{\sum_{k\in\mathcal{K}} L_k}{e_s}\right\}\right\}\\
\text{s.t.}\quad & \text{C1}: \sum_{k\in\mathcal{K}} p_k^o \le P,\\
& \text{C2}: \sum_{k\in\mathcal{K}} p_k^u \le P,\\
& \text{C3}: p_k^o \ge 0,\ \forall k\in\mathcal{K},\\
& \text{C4}: p_k^u \ge 0,\ \forall k\in\mathcal{K},\\
& \text{C5}: 0 \le L_k \le D_k,\ \forall k\in\mathcal{K}.
\end{aligned} \tag{21}$$

It can be verified that P1 is a non-convex optimization problem since the objective function in P1 is not jointly convex with respect to the optimization variables. For tractability, we decouple P1 into a master problem and two subproblems. We take L_k as the coupling variable and derive two subproblems

by fixing L_k. The subproblem $\mathrm{P}_{sub}^{(1)}$ optimizes the transmission power p_k^o as

$$\mathrm{P}_{sub}^{(1)}: \min_{p_k^o} \max_{k\in\mathcal{K}} \frac{L_k}{B\log_2(1+p_k^o g_k)} \quad \text{s.t.} \quad \mathrm{C1}, \mathrm{C3}. \tag{22}$$

The subproblem $\mathrm{P}_{sub}^{(2)}$ optimizes the computational power p_k^u as

$$\mathrm{P}_{sub}^{(2)}: \min_{p_k^u} \max \left\{ \max_{k\in\mathcal{K}} (D_k - L_k)\sqrt{\frac{\zeta_k}{p_k^u}}, \frac{\sum_{k\in\mathcal{K}} L_k}{e_s} \right\} \quad \text{s.t.} \quad \mathrm{C2}, \mathrm{C4}. \tag{23}$$

The master problem only needs to optimize L_k and is described as

$$\mathrm{P}_{master}: \min_{L_k} \{T_1^*(L_k) + r(\epsilon)t_w + r(\epsilon)\tau C T_2^*(L_k)\} \quad \text{s.t.} \quad \mathrm{C5}, \tag{24}$$

where $T_1^*(L_k)$ and $T_2^*(L_k)$ are calculated using the derived p_k^o and p_k^u from $\mathrm{P}_{sub}^{(1)}$ and $\mathrm{P}_{sub}^{(2)}$, respectively.

B. Solution to the Subproblems

In what follows, we will solve the two above subproblems.

1) : The objective function of $\mathrm{P}_{sub}^{(1)}$ is the pointwise maximum of convex functions. Combining it with linear constraints, $\mathrm{P}_{sub}^{(1)}$ is a convex optimization problem. To solve it, we first convert it into a epigraph form as

$$\min_{p_k^o, T} \ T \quad \text{s.t.} \quad \frac{L_k}{B\log_2(1+p_k^o g_k)} \le T, \ \forall k \in \mathcal{K}, \quad \mathrm{C1}, \ \mathrm{C3}, \tag{25}$$

where T is an auxiliary variable and $T_1^*(L_k) = T^*$. Since the converted problem is convex, we define the Lagrange function as

$$\mathcal{F} = T + \sum_{k\in\mathcal{K}} \lambda_k \left(\frac{L_k}{B\log_2(1+p_k^o g_k)} - T \right) + \mu \left(\sum_{k\in\mathcal{K}} p_k^o - P \right), \tag{26}$$

where λ_k and μ are the non-negative Lagrange multiplier. Then applying the Karush-Kuhn-Tucker (KKT) conditions leads to the following equations:

$$\frac{\partial \mathcal{F}}{\partial p_k^{o*}} = -\frac{\lambda_k^* L_k g_k}{B(1+p_k^{o*}g_k)\log^2(1+p_k^{o*}g_k)} + \mu^* = 0, \ \forall k \in \mathcal{K}, \tag{27}$$

$$\frac{\partial \mathcal{F}}{\partial T^*} = 1 - \sum_{k\in\mathcal{K}} \lambda_k^* = 0, \tag{28}$$

$$\sum_{k\in\mathcal{K}} p_k^{o*} \le P, \ \frac{L_k}{B\log_2(1+p_k^{o*}g_k)} \le T^*, \tag{29}$$

$$\mu^* \left(\sum_{k\in\mathcal{K}} p_k^{o*} - P \right) = 0, \tag{30}$$

$$\lambda_k^* \left(\frac{L_k}{B\log_2(1+p_k^{o*}g_k)} - T^* \right) = 0, \ \forall k \in \mathcal{K}. \tag{31}$$

From (27) and (28), we can derive that $\mu^* > 0$ and $\lambda_k^* > 0, \forall k \in \mathcal{K}$. Then, according to (30) and (31), the optimal power allocation satisfies

$$\sum_{k\in\mathcal{K}} p_k^{o*} = P, \ p_k^{o*} = \frac{1}{g_k}\left(2^{\frac{L_k}{BT^*}} - 1\right), \forall k \in \mathcal{K}. \tag{32}$$

Then we have

$$\sum_{k\in\mathcal{K}} \frac{1}{g_k}\left(2^{\frac{L_k}{BT^*}} - 1\right) = P. \tag{33}$$

Note that the relationship between T^* and L_k does not have an explicit expression. Therefore, we perform a second-order Taylor approximation to the exponential term in the optimal power p_k^{o*} as

$$2^{\frac{L_k}{BT^*}} - 1 \approx \frac{L_k}{BT^*}\ln 2 + \frac{1}{2}\left(\frac{L_k}{BT^*}\ln 2\right)^2. \tag{34}$$

We define $\frac{\widehat{R}_k}{B} = \frac{L_k}{BT^*}$ as the spectral efficiency, where $\widehat{R}_k$ denotes the average transmission rate of client k. Then, we assume that the spectral efficiency in EAFL is small and insert (34) into (33):

$$\frac{b}{T^*} + \frac{a}{T^{*2}} = P, \tag{35}$$

where

$$a = \sum_{k\in\mathcal{K}} \left(\frac{L_k \ln 2}{B\sqrt{2g_k}}\right)^2, \ b = \sum_{k\in\mathcal{K}} \frac{L_k \ln 2}{Bg_k}. \tag{36}$$

Solving the quadratic equation above and taking its positive root as the approximated solution of $\mathrm{P}_{sub}^{(1)}$, we have

$$T_1^*(L_k) = T^* = \frac{\sqrt{b^2 + 4aP} + b}{2P}. \tag{37}$$

2) : It can also be verified that $\mathrm{P}_{sub}^{(2)}$ is a convex optimization problem due to its convex objective function and linear constraints. Since the optimization variable of $\mathrm{P}_{sub}^{(2)}$ is p_k^u, we only need to solve the problem:

$$\min_{p_k^u} \max_{k\in\mathcal{K}} (D_k - L_k)\sqrt{\frac{\zeta_k}{p_k^u}} \quad \text{s.t.} \quad \mathrm{C2}, \mathrm{C4}. \tag{38}$$

We introduce the variable T and convert this problem into its epigraph form.

$$\min_{p_k^u, T} \ T \quad \text{s.t.} \quad (D_k - L_k)\sqrt{\frac{\zeta_k}{p_k^u}} \le T, \ \forall k \in \mathcal{K}, \quad \mathrm{C2}, \ \mathrm{C4}. \tag{39}$$

This problem is convex. We define the Lagrange function as

$$\mathcal{F} = T + \sum_{k\in\mathcal{K}} \theta_k \left((D_k - L_k)\sqrt{\frac{\zeta_k}{p_k^u}} - T \right) + \eta \left(\sum_{k\in\mathcal{K}} p_k^u - P \right), \tag{40}$$

where θ_k and η are the non-negative Lagrange multiplier. Applying KKT conditions, we can get the following equations:

$$\sum_{k\in\mathcal{K}} p_k^{u*} = P, \ p_k^{u*} = \frac{\zeta_k (D_k - L_k)^2}{T^{*2}}, \forall k \in \mathcal{K}. \tag{41}$$

Based on (41), we derive the solution:

$$T^* = \sqrt{\frac{d}{P}}, \tag{42}$$

where

$$d = \sum_{k\in\mathcal{K}} \zeta_k (D_k - L_k)^2. \tag{43}$$

Based on (42), we derive the solution to $\mathrm{P}_{sub}^{(2)}$, *i.e.*,

$$T_2^*(L_k) = \max\left\{\sqrt{\frac{d}{P}},\ \frac{\sum_{k\in\mathcal{K}} L_k}{e_s}\right\}. \tag{44}$$

C. Solution to the Master Problem

Inserting (37) and (44) into the objective function in P_{master}, we derive

$$\begin{aligned}\mathrm{P2}: \min_{L_k} \Bigg\{ &\frac{\sqrt{b^2+4aP}+b}{2P} + r(\epsilon)t_w \\ &+ r(\epsilon)\tau C \max\left\{\sqrt{\frac{d}{P}},\ \frac{\sum_{k\in\mathcal{K}} L_k}{e_s}\right\}\Bigg\} \\ \text{s.t.}\quad &\mathrm{C5}.\end{aligned} \tag{45}$$

To solve P2, we introduce an auxiliary variable $T > 0$ and convert P2 into an equivalent problem P3 as

$$\begin{aligned}\mathrm{P3}: \min_{L_k, T} \Bigg\{ &\frac{\sqrt{b^2+4aP}+b}{2P} + r(\epsilon)\tau C T + r(\epsilon)t_w \Bigg\} \\ \text{s.t.}\quad &\mathrm{C6}: \sqrt{\frac{d}{P}} \le T, \\ &\mathrm{C7}: \frac{\sum_{k\in\mathcal{K}} L_k}{e_s} \le T, \\ &\mathrm{C5}.\end{aligned} \tag{46}$$

Since the objective function in P_3 is the summation of a convex function (37) and a linear function, the objective function is convex. Combining it with the convex constraint C6 and the linear constraint C7, P_3 is a convex optimization problem. Therefore, the KKT conditions are necessary and sufficient for the optimal value of this problem. To obtain the KKT conditions, we first obtain the Lagrangian function of P_3, which is written as

$$\begin{aligned}\mathcal{F} = &\frac{\sqrt{b^2+4aP}+b}{2P} + r(\epsilon)\tau C T + r(\epsilon)t_w \\ &+ \rho\left(\sqrt{\frac{d}{P}} - T\right) + \delta\left(\frac{\sum_{k\in\mathcal{K}} L_k}{e_s} - T\right),\end{aligned} \tag{47}$$

where $\rho \ge 0$ and $\delta \ge 0$ are the Lagrange multipliers associated with the constraint C6 and C7, respectively.

For simplicity, we define

$$\begin{aligned} f_1(L_k) &= \frac{\partial}{\partial L_k}\left(\frac{\sqrt{b^2+4aP}+b}{2P}\right) \\ &= \frac{B(b+\sqrt{b^2+4aP})\ln 2 + 2PL_k \ln^2 2}{2PB^2 g_k \sqrt{b^2+4aP}} \end{aligned} \tag{48}$$

$$f_2(L_k) = \frac{\partial}{\partial L_k}\left(\sqrt{\frac{d}{P}}\right) = -\frac{\zeta_k(D_k - L_k)}{\sqrt{Pd}}. \tag{49}$$

Then applying KKT conditions, we have

$$\frac{\partial \mathcal{F}}{\partial L_k^*} = f_1(L_k^*) + \rho^* f_2(L_k^*) + \frac{\delta^*}{e_s}\begin{cases} >0,\ L_k^* = 0 \\ =0,\ L_k^* \in (0, D_k) \\ <0,\ L_k^* = D_k, \forall k \in \mathcal{K} \end{cases}, \tag{50}$$

$$\frac{\partial \mathcal{F}}{\partial T^*} = r(\epsilon)\tau C - \rho^* - \delta^* = 0, \tag{51}$$

$$\sqrt{\frac{d}{P}} \le T^*,\ \frac{\sum_{k\in\mathcal{K}} L_k^*}{e_s} \le T^*, \tag{52}$$

$$\rho^*\left(\sqrt{\frac{d}{P}} - T^*\right) = 0,\ \delta^*\left(\frac{\sum_{k\in\mathcal{K}} L_k^*}{e_s} - T^*\right) = 0. \tag{53}$$

We define the computation delay difference of each iteration between the server and clients as

$$T_{s-c} = \frac{\sum_{k\in\mathcal{K}} L_k}{e_s} - \sqrt{\frac{d}{P}}. \tag{54}$$

Inserting $f_2(D_k) = 0$ into (50), we note that $L_k^* \ne D_k$. According to (51), we have $\delta^* = r(\epsilon)\tau C - \rho^*$. Combining it with (52) and (53) leads to

$$\rho^*\begin{cases} =0, & \text{if}\quad T_{s-c} > 0 \\ \in (0,\ r(\epsilon)\tau C), & \text{if}\quad T_{s-c} = 0 \\ = r(\epsilon)\tau C, & \text{if}\quad T_{s-c} < 0 \end{cases}. \tag{55}$$

According to the value of T_{s-c}, the optimal solution to P3 can be characterized as follows.

1) Case 1: If $T_{s-c} > 0$, *i.e.*, $\rho^* = 0$, we have

$$\frac{\partial \mathcal{F}}{\partial L_k^*} = f_1(L_k^*) + \frac{r(\epsilon)\tau C}{e_s} > 0. \tag{56}$$

According to (50), we have $L_k^* = 0$.

2) Case 2: If $T_{s-c} < 0$, *i.e.*, $\rho^* = r(\epsilon)\tau C > 0$, we have

$$\frac{\partial \mathcal{F}}{\partial L_k^*} = f_1(L_k^*) + r(\epsilon)\tau C f_2(L_k^*). \tag{57}$$

According to (50), we have

$$L_k^* = \begin{cases} 0, & \zeta_k g_k D_k < \dfrac{c_2\sqrt{d}}{c_1 r(\epsilon)\tau C} \\ L_k^{case2}, & \zeta_k g_k D_k \ge \dfrac{c_2\sqrt{d}}{c_1 r(\epsilon)\tau C} \end{cases}, \tag{58}$$

where

$$L_k^{case2} = \frac{c_1 r(\epsilon)\tau C \zeta_k g_k D_k - c_2\sqrt{d}}{c_1 r(\epsilon)\tau C \zeta_k g_k + 2P\sqrt{d}\ln^2 2} \tag{59}$$

and

$$c_1 = 2B^2\sqrt{P}\sqrt{b^2+4aP},\ c_2 = B(b+\sqrt{b^2+4aP})\ln 2. \tag{60}$$

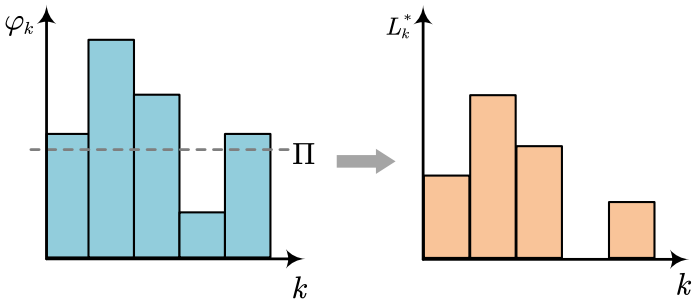


Fig. 4. Illustration of the offloading strategy for EAFL.

3) Case 3: If $T_{s-c}=0$, *i.e.*, $0<\rho^*<r(\epsilon)\tau C$, we have

$$\frac{\partial \mathcal{F}}{\partial L_k^*}=f_1(L_k^*)+\rho^* f_2(L_k^*)+\frac{r(\epsilon)\tau C-\rho^*}{e_s} \tag{61}$$

According to (50), we have

$$L_k^*=\begin{cases}0, & \left(\zeta_k D_k-\frac{\sqrt{Pd}(r(\epsilon)\tau C-\rho^*)}{e_s\rho^*}\right)g_k<\frac{c_2\sqrt{d}}{c_1\rho^*}\\ L_k^{case3}, & \left(\zeta_k D_k-\frac{\sqrt{Pd}(r(\epsilon)\tau C-\rho^*)}{e_s\rho^*}\right)g_k\geq\frac{c_2\sqrt{d}}{c_1\rho^*}\end{cases}, \tag{62}$$

where

$$L_k^{case3}=\frac{c_1\rho^*\left(\zeta_k D_k-\frac{\sqrt{Pd}(r(\epsilon)\tau C-\rho^*)}{e_s\rho^*}\right)g_k-c_2\sqrt{d}}{c_1\rho^*\zeta_k g_k+2P\sqrt{d}\ln^2 2}. \tag{63}$$

D. Offloading Strategy

Based on the analysis of the above three cases, we propose a corresponding offloading strategy for EAFL. To facilitate the description, we first define two function $f_3(\rho^*)$ and $f_4(\rho^*)$ as

$$\begin{aligned}f_3(\rho^*)&=\begin{cases}\frac{\sqrt{Pd}(r(\epsilon)\tau C-\rho^*)}{e_s\rho^*}, & \rho^*\neq 0\\ +\infty, & \rho^*=0\end{cases},\\ f_4(\rho^*)&=\begin{cases}\frac{c_2\sqrt{d}}{c_1\rho^*}, & \rho^*\neq 0\\ +\infty, & \rho^*=0\end{cases}.\end{aligned} \tag{64}$$

Proposition 1 (The Offloading Strategy for EAFL)**.** *The optimal offloading data size of client k is denoted as L_k^*. For each client, we define an offloading decision indicator φ_k as:*

$$\varphi_k=(\zeta_k D_k-f_3(\rho^*))g_k, \tag{65}$$

where the optimal Lagrange multiplier ρ^* is defined in (55). We also define an offloading decision threshold Π as

$$\Pi=f_4(\rho^*). \tag{66}$$

Then the offloading strategy for EAFL has the following structure, which is illustrated in Fig. 4.

1) If $\varphi_k<\Pi$, client k should not offload any data to the server, i.e.*, $L_k^*=0$.*

2) If $\varphi_k\geq\Pi$, client k requires to offload partial data to the server and $L_k^*=L_k^o$, where

$$L_k^o=\frac{c_1\rho^*(\zeta_k D_k-f_3(\rho^*))g_k-c_2\sqrt{d}}{c_1\rho^*\zeta_k g_k+2P\sqrt{d}\ln^2 2}. \tag{67}$$

Proof: According to (56), (58) and (62), we have the proposition. ■

Remark 3 (Offloading or Not)**.** *We find that there always exists $\varphi_k<\Pi$ and $L_k^*=0$ for each client when $T_{s-c}>0$ ($\rho^*=0$). In this case, offloading can not bring any delay reduction but to increase the computation burden of the server and total system delay. Therefore, we should give up offloading when the update delay of the server is larger than that of all clients. When $T_{s-c}\leq 0$, the offloading decision indicator φ_k determines whether to offload data for client k. Offloading is not necessary except for two cases. First, ζ_k of client k is so large that local computing fails to meet the local power constraint. Second, some client has a large dataset size D_k or a high channel gain g_k, indicating that latency shortening can be achieved by offloading its partial data to the server.*

Remark 4 (The Offloading Data Size)**.** *Note that L_k^o in (67) is a monotonous increasing function with respect to ζ_k and g_k and can be expressed as*

$$L_k^o=\psi_1(\zeta_k,g_k)D_k-\psi_2(\zeta_k,g_k), \tag{68}$$

where ψ_1 is a monotonous increasing function and ψ_2 is a monotonous decreasing function. For the client with a larger ζ_k or g_k, it requires to offload larger proportion of data to the server. Further, for the clients having almost equal D_k, the clients with a larger ζ_k or g_k require to offload larger data to the server.

Remark 5 (The Case of $T_{s-c}<0$)**.** *When $T_{s-c}<0$, we note that it actually corresponds to the case where the server has a high computing capability. In this case, the optimal Lagrangian multiplier $\rho^*=r(\epsilon)\tau C$, which makes the offloading related parameter φ_k and Π determined values. Therefore, if the edge server in EAFL is equipped with multiple high performance GPUs and the model training delay in the server is negligible, Proposition 2 will degenerate into a more simple offloading strategy with determined offloading parameters.*

Based on the offloading strategy in Proposition 1, the corresponding algorithm is proposed in Algorithm 1. Each client's offloading data size is first given as an initial value. Then multiple rounds of size update are performed according to Proposition 2 until convergence.

Referring to [37], we take a learning-based autonomous driving system as a use case of EAFL. In this system, each vehicular client is configured with a learning model and the roadside server has powerful computing ability. The learning model takes on-board sensor data as input, and decisions like braking and steering as output. To fuse the information of multiple vehicular clients for better trained model while avoiding large communication overhead, federated learning is applied. Due to the existence of vehicular clients with weak computing ability, we utilize EAFL to mitigate the

Algorithm 1: Iterative algorithm for the offloading strategy

Input: K, C, B, P, τ, ϵ, α, β, ζ_k, g_k, D_k
Output: L_k^*

Initialize iteration round $n = 0$.
Initialize $L_k^{(0)} = 0.5 * D_k$ for all k.
Initialize $\rho_l = 0$, $\rho_h = r(\epsilon)\tau C$.
repeat
  Calculate $a^{(n)}$, $b^{(n)}$, $d^{(n)}$ and $T_{s-c}^{(n)}$.
  for $k = 1$ *to* K **do**
    if $T_{s-c}^{(n)} > 0$ **then**
      $L_k^{(n+1)} = 0$
    else if $T_{s-c}^{(n)} < 0$ **then**
      $\rho^* = r(\epsilon)\tau C$.
      Calculate φ_k and $\Pi^{(n)}$.
      if $\varphi_k < \Pi^{(n)}$ **then** $L_k^{(n+1)} = 0$;
      else $L_k^{(n+1)} = L_k^o$;
    else
      $\rho_m = (\rho_l + \rho_h)/2$.
      $\rho^* = \rho_m$.
      Calculate φ_k and $\Pi^{(n)}$.
      if $\varphi_k < \Pi^{(n)}$ **then** $L_k^{(n+1)} = 0$;
      else $L_k^{(n+1)} = L_k^o$;
      if $L_k^{(n+1)} > L_k^{(n)}$ **then** $\rho_l = \rho_m$;
      if $L_k^{(n+1)} < L_k^{(n)}$ **then** $\rho_h = \rho_m$;
    $n = n + 1$.
  end
until $L_k^{(n+1)} = L_k^{(n)}$ *for all* k;
Result: $L_k^* = L_k^{(n+1)}$.

straggler effect. Assume that Algorithm 1 is executed on the roadside server and the server has perfect knowledge of related parameters, which can be obtained by feedback. After running Algorithm 1, the server will send the optimal offloading data size to clients. During the model update phase, each vehicular client updates its local model and the roadside server utilizes the aggregated data to update the model in itself. After multiple rounds of training, vehicular clients can finally obtain the model that guides their decision-making.

IV. EAFL with Dynamic Clients

In practice, we also need to consider the device disconnection during the model training. Due to the preemption of other tasks or equipment network failures, some clients can only participate in the model training within a certain round. For example, in the federated learning of the internet of vehicles scenarios, vehicular clients are likely to exceed the range that the server can cover during the training process. In this section, we investigate a dynamic scenario for EAFL, where some clients may disconnect from the system after several rounds of updates. For simplicity, we refer to such clients as dynamic clients and the EAFL with dynamic clients as dynamic EAFL. Compared with the original EAFL, the introduction of dynamic clients changes the delay modeling and the delay minimization problem, which requires a new offloading strategy.

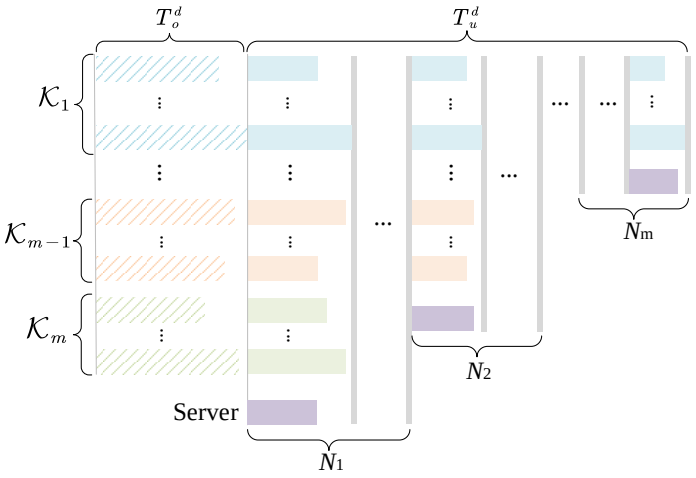


Fig. 5. Delay analysis for dynamic EAFL.

A. Problem Formulation

As depicted in Fig. 5, we assume that all clients can be grouped into m sets according to their offline round. The local update process can be divided into m update segments and each segment i contains N_i rounds of updates. The clients belonging to set $\mathcal{K}_1$ keep online until the update process is finished and the clients in set $\mathcal{K}_{i,\ 2\le i\le m}$ will be offline after $\sum_{j=1}^{m-i+1} N_j$ rounds. The size of client sets in the i-th update segment is denoted as $K_i = \sum_{j=1}^{m-i+1} |\mathcal{K}_j|$.

The data offloading delay of dynamic EAFL is defined as

$$T_o^d = \max_{\{k\in\mathcal{K}_i, 1\le i\le m\}} \frac{L_{k,i}}{B\log_2(1+p_{k,i}^o g_{k,i})}, \tag{69}$$

where $L_{k,i}$, $p_{k,i}^o$, $g_{k,i}$ denote the offloading data size, the allocated transmission power and the channel gain of client k in set $\mathcal{K}_i$, respectively. The model update delay of dynamic EAFL consists of m segments of update delay and is defined as

$$T_u^d = \sum_{j=1}^{m} N_j \tau C \max\left\{ \frac{\sum_{i=1}^{m-j+1}\sum_{k\in\mathcal{K}_i} L_{k,i}}{e_s}, \max_{k\in\mathcal{K}_i, 1\le i\le m-j+1} (D_{k,i} - L_{k,i})\sqrt{\frac{\zeta_{k,i}}{p_{k,i,j}^u}} \right\} + \sum_{j=1}^{m} N_j t_w, \tag{70}$$

where $p_{k,i,j}^u$ is the allocated computational power of client k in set $\mathcal{K}_i$ during the j-th segment of update, $D_{k,i}$ is the client's dataset size and $\zeta_{k,i}$ denotes the client's effective capacitance coefficient.

We assume that the proportion of stable clients is large in dynamic EAFL, which is reasonable in the federated learning settings. Therefore, the dynamic change of the client number has little effect on the parameter α and β in (20) and we ignore the change of them in this paper. Based on (20), the relationship among the number of clients, the number of

rounds and the learning accuracy in dynamic EAFL can be determined as

$$N_i = \begin{cases} \left[\left(1+\frac{1}{K_1}\right)\tau\alpha+\frac{\beta}{\tau}\right]\frac{1}{\epsilon_1}, & i=1 \\ \left[\left(1+\frac{1}{K_i}\right)\tau\alpha+\frac{\beta}{\tau}\right]\left(\frac{1}{\epsilon_i}-\frac{1}{\epsilon_{i-1}}\right), & 2\le i\le m \end{cases}, \tag{71}$$

where ϵ_i denotes the learning accuracy after i segments of updates and $\epsilon_m = \epsilon$. We assume that dynamic clients can give their expected offline rounds based on the current task carrying capacity or device network status. Then, we can derive the last segment update rounds N_m based on (71):

$$N_m = \left[\left(1+\frac{1}{K_m}\right)\tau\alpha+\frac{\beta}{\tau}\right]\left[\frac{1}{\epsilon}-\sum_{i=1}^{m-1}\frac{N_i}{\left(1+\frac{1}{K_i}\right)\tau\alpha+\frac{\beta}{\tau}}\right]. \tag{72}$$

The delay minimization problem with dynamic client sets can be formulated as follows.

$$\begin{aligned} \mathrm{P}_4: \min_{\{L_{k,i},p^o_{k,i},p^u_{k,i,j}\}} & \left\{T_o^d+T_u^d\right\}, \\ \text{s.t.}\quad & \mathrm{C8}: \sum_{i=1}^{m}\sum_{k\in\mathcal{K}_i} p^o_{k,i}\le P, \\ & \mathrm{C9}: \sum_{i=1}^{m-j+1}\sum_{k\in\mathcal{K}_i} p^u_{k,i,j}\le P,\ 1\le j\le m, \\ & \mathrm{C10}: p^o_{k,i}\ge 0,\ \forall k\in\mathcal{K}_i,\ 1\le i\le m, \\ & \mathrm{C11}: p^u_{k,i,j}\ge 0,\ \forall k\in\mathcal{K}_i,\ 1\le i\le m, \\ & \mathrm{C12}: 0\le L_{k,i}\le D_{k,i},\ \forall k\in\mathcal{K}_i,\ 1\le i\le m. \end{aligned} \tag{73}$$

To solve P_4, we take similar steps in solving P_1. Given $L_{k,i}$, P_3 can be decomposed into one offloading delay minimization subproblem ($\mathrm{P}^{(3)}_{sub}$) and m model update delay minimization subproblems ($\mathrm{P}^{(4,j)}_{sub}, 1\le j\le m$). $\mathrm{P}^{(3)}_{sub}$ and $\mathrm{P}^{(4,j)}_{sub}$ are described as follows.

$$\begin{aligned} \mathrm{P}^{(3)}_{sub}: \min_{p^o_{k,i}} & \max_{\{k\in\mathcal{K}_i,1\le i\le m\}} \frac{L_{k,i}}{B\log_2(1+p^o_{k,i}g_{k,i})}, \\ \text{s.t.}\quad & \mathrm{C8},\mathrm{C10}. \end{aligned} \tag{74}$$

$$\begin{aligned} \mathrm{P}^{(4,j)}_{sub}: \min_{p^u_{k,i,j}} & \max\left\{\frac{\sum_{i=1}^{m-j+1}\sum_{k\in\mathcal{K}_i}L_{k,i}}{e_s}, \right. \\ & \left. \max_{k\in\mathcal{K}_i,1\le i\le m-j+1}(D_{k,i}-L_{k,i})\sqrt{\frac{\zeta_{k,i}}{p^u_{k,i,j}}}\right\}, \\ \text{s.t.}\quad & \mathrm{C9},\mathrm{C11}. \end{aligned} \tag{75}$$

The master problem is described as

$$\begin{aligned} \mathrm{P}_{master}: \min_{L_k} & \left\{T_3^*(L_k)+\sum_{j=1}^{m}N_j\tau C T^*_{4,j}(L_k)+\sum_{j=1}^{m}N_j t_w\right\} \\ \text{s.t.}\quad & \mathrm{C12}, \end{aligned} \tag{76}$$

where $T_3^*(L_k)$ and $T^*_{4,j}(L_k)$ are optimal objective functions of $\mathrm{P}^{(3)}_{sub}$ and $\mathrm{P}^{(4,j)}_{sub}$, respectively.

B. Solution

According to the solutions in Section III B, we can easily derive the similar solutions to the $m+1$ subproblems above. Then, by inserting the optimal values in the $m+1$ subproblems into the master problem, we can formulate P4 as follows.

$$\begin{aligned} \mathrm{P5}: \min_{L_{k,i}} & \left\{\frac{\sqrt{\widehat{b}^2+4\widehat{a}P}+\widehat{b}}{2P}+\sum_{j=1}^{m}N_j t_w \right. \\ & \left. +\sum_{j=1}^{m}N_j\tau C\max\left\{\sqrt{\frac{\widehat{d}_j}{P}}, \frac{\sum_{i=1}^{m-j+1}\sum_{k\in\mathcal{K}_i}L_{k,i}}{e_s}\right\}\right\} \\ \text{s.t.}\quad & \mathrm{C12}, \end{aligned} \tag{77}$$

where

$$\begin{aligned} \widehat{a} &= \sum_{i=1}^{m}\sum_{k\in\mathcal{K}_i}\left(\frac{L_{k,i}\ln 2}{B\sqrt{2g_{k,i}}}\right)^2, \widehat{b} = \sum_{i=1}^{m}\sum_{k\in\mathcal{K}_i}\frac{L_{k,i}\ln 2}{Bg_{k,i}}, \\ \widehat{d}_j &= \sum_{i=1}^{m-j+1}\sum_{k\in\mathcal{K}_i}\zeta_{k,i}(D_{k,i}-L_{k,i})^2. \end{aligned} \tag{78}$$

To solve P5, we introduce the auxiliary variable $T_j>0$ and convert P5 into an equivalent problem P6 as

$$\begin{aligned} \mathrm{P6}: \min_{L_{k,i},T_j} & \left\{\frac{\sqrt{\widehat{b}^2+4\widehat{a}P}+\widehat{b}}{2P}+\tau C\sum_{j=1}^{m}N_jT_j+\sum_{j=1}^{m}N_jt_w\right\} \\ \text{s.t.}\quad & \mathrm{C13}: \sqrt{\frac{\widehat{d}_j}{P}}\le T_j,\ 0\le j\le m, \\ & \mathrm{C14}: \frac{\sum_{i=1}^{m-j+1}\sum_{k\in\mathcal{K}_i}L_{k,i}}{e_s}\le T_j,\ 0\le j\le m, \\ & \mathrm{C12}. \end{aligned} \tag{79}$$

Since the objective function and the constraints are both convex, P_6 is a convex optimization problem. Therefore, the KKT conditions are necessary and sufficient for the optimal value of this problem. The Lagrangian function of P_6 is written as

$$\begin{aligned} \mathcal{F} =& \frac{\sqrt{\widehat{b}^2+4\widehat{a}P}+\widehat{b}}{2P}+\tau C\sum_{j=1}^{m}N_jT_j+\sum_{j=1}^{m}N_jt_w \\ & +\sum_{j=1}^{m}\rho_j\left(\sqrt{\frac{\widehat{d}_j}{P}}-T_j\right) \\ & +\sum_{j=1}^{m}\delta_j\left(\frac{\sum_{i=1}^{m-j+1}\sum_{k\in\mathcal{K}_i}L_{k,i}}{e_s}-T_j\right), \end{aligned} \tag{80}$$

where $\rho_j\ge 0$ and $\delta_j\ge 0$ are the Lagrange multipliers associated with the constraint C13 and C14, respectively. Then we apply KKT conditions, which is similar to the procedure in Section III C. And we have

$$\rho_j^*\begin{cases} =0, & \text{if}\quad T_{s-c,j}>0 \\ \in(0,\ N_j\tau C), & \text{if}\quad T_{s-c,j}=0, \\ =N_j\tau C, & \text{if}\quad T_{s-c,j}<0 \end{cases} \tag{81}$$

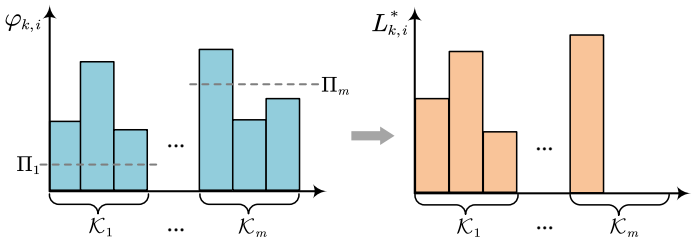


Fig. 6. Illustration of the offloading strategy for dynamic EAFL.

where $T_{s-c,j}$ is defined as

$$T_{s-c,j} = \frac{\sum_{i=1}^{m-j+1}\sum_{k\in\mathcal{K}_i} L_{k,i}}{e_s} - \sqrt{\frac{\widehat{d}_j}{P}}. \tag{82}$$

Then for all the above three cases of $T_{s-c,j}$, we have

$$L^*_{k,i} = \begin{cases} 0, & \text{if} \quad (\zeta_{k,i}D_{k,i} - f_{5,i}(\widehat{\rho}^*_i))g_{k,i} < f_{6,i}(\widehat{\rho}^*_i) \\ L^o_{k,i}, & \text{if} \quad (\zeta_{k,i}D_{k,i} - f_{5,i}(\widehat{\rho}^*_i))g_{k,i} \geq f_{6,i}(\widehat{\rho}^*_i) \end{cases}, \tag{83}$$

where

$$L^o_{k,i} = \frac{c_1\widehat{\rho}^*_i(\zeta_{k,i}D_{k,i} - f_{5,i}(\widehat{\rho}^*_i))g_{k,i} - c_2}{c_1\widehat{\rho}^*_i\zeta_{k,i}g_{k,i} + 2P\ln^2 2} \tag{84}$$

and

$$f_{5,i}(\widehat{\rho}^*_i) = \begin{cases} \dfrac{\sqrt{P}}{e_s\widehat{\rho}^*_i}\displaystyle\sum_{j=1}^{m-i+1}(N_j\tau C - \rho^*_j), & \widehat{\rho}^*_i \neq 0 \\ +\infty, & \widehat{\rho}^*_i = 0 \end{cases},$$

$$f_{6,i}(\widehat{\rho}^*_i) = \begin{cases} \dfrac{c_2}{c_1\widehat{\rho}^*_i}, & \widehat{\rho}^*_i \neq 0 \\ +\infty, & \widehat{\rho}^*_i = 0. \end{cases}, \quad \widehat{\rho}^*_i = \sum_{j=1}^{m-i+1}\frac{\rho^*_j}{\sqrt{d_j}}. \tag{85}$$

C. Offloading Strategy

Proposition 2 (The Offloading Strategy for Dynamic EAFL)**.** *For client $k \in \mathcal{K}_i$, the optimal offloading data size is denoted as $L^*_{k,i}$. For client set $\mathcal{K}_i$, we define the offloading decision indicator as*

$$\varphi_{k,i} = (\zeta_{k,i}D_{k,i} - f_{5,i}(\widehat{\rho}^*_i))g_{k,i} \tag{86}$$

and the offloading decision threshold Π_i as

$$\Pi_i = f_{6,i}(\widehat{\rho}^*_i), \tag{87}$$

where $\widehat{\rho}^*_i$, $f_{5,i}(\widehat{\rho}^*_i)$ and $f_{6,i}(\widehat{\rho}^*_i)$ are defined in (85). The offloading strategy for client $k \in \mathcal{K}_i$ has the following structure.

1) *If $\varphi_{k,i} < \Pi_i$, client k should not offload any data to the server,* i.e.*, $L^*_{k,i} = 0$.*
2) *If $\varphi_{k,i} \geq \Pi_i$, client k requires to offload partial data to the server and $L^*_{k,i} = L^o_{k,i}$.*

Proof: According to (83) and (84), we have the proposition. ∎

Remark 6 (Offloading Decision Threshold)**.** *The offloading strategy for dynamic EAFL also has a threshold-based structure that determines the optimal offloading data size in each client set, which is depicted in Fig. 6. According to (87), we note that the threshold of client set $\mathcal{K}_i$ is determined by $\widehat{\rho}^*_i$. For the client set with fewer update rounds,* i.e.*, with a smaller $\widehat{\rho}^*_i$, the offloading threshold is larger and the clients prefer to update locally instead of offloading. The explanation is that for the client set with fewer update rounds, the straggler effect they bring has a little impact on the system delay, which makes local update a better choice.*

Algorithm 2: Iterative algorithm for the offloading strategy in dynamic EAFL

Input: m, $K_{1\sim m}$, $N_{1\sim m-1}$, C, B, P, τ, ϵ, α, β, $\zeta_{k,i}$, $g_{k,i}$, $D_{k,i}$
Output: $L^*_{k,i}$

Initialize iteration round $n = 0$.
Initialize $L^{(0)}_{k,i} = 0.5 * D_{k,i}$.
Calculate N_m based on (72).
Initialize $\rho_{l,i} = 0$, $\rho_{h,i} = N_i\tau C$.
repeat
  Calculate $\widehat{a}^{(n)}$, $\widehat{b}^{(n)}$ and $\widehat{d}^{(n)}_1 \sim \widehat{d}^{(n)}_m$.
  for $i = 1$ *to* m **do**
    Calculate $T^{(n)}_{s-c,i}$.
    for $k = 1$ *to* $|\mathcal{K}_i|$ **do**
      if $T^{(n)}_{s-c,i} > 0$ **then**
        $L^{(n+1)}_{k,i} = 0$.
      else if $T^{(n)}_{s-c,i} < 0$ **then**
        $\rho^*_i = N_i\tau C$.
        Calculate $\varphi_{k,i}$ and $\Pi^{(n)}_i$.
        if $\varphi_{k,i} < \Pi^{(n)}_i$ **then** $L^{(n+1)}_{k,i} = 0$;
        else $L^{(n+1)}_{k,i} = L^o_{k,i}$;
      else
        $\rho_{m,i} = (\rho_{l,i} + \rho_{h,i})/2$.
        $\rho^*_i = \rho_{m,i}$.
        Calculate $\varphi_{k,i}$ and $\Pi^{(n)}$.
        if $\varphi_{k,i} < \Pi^{(n)}_i$ **then** $L^{(n+1)}_{k,i} = 0$;
        else $L^{(n+1)}_{k,i} = L^o_{k,i}$;
        if $L^{(n+1)}_{k,i} > L^{(n)}_{k,i}$ **then** $\rho_{l,i} = \rho_{m,i}$;
        if $L^{(n+1)}_{k,i} < L^{(n)}_{k,i}$ **then** $\rho_{h,i} = \rho_{m,i}$;
      $n = n + 1$.
    end
  end
until $L^{(n+1)}_{k,i} = L^{(n)}_{k,i}$ *for all* k *and* i;
Result: $L^*_k = L^{(n+1)}_k$.

The algorithm for the offloading strategy in Proposition 2 is summarized in Algorithm 2. The algorithm gives an iterative framework to perform the optimal offloading in dynamic EAFL. It is assumed that the server can obtain the expected offline round of each client through feedback. Compared with Algorithm 1, dynamic EAFL requires iterative update for multiple sets of dynamic clients.

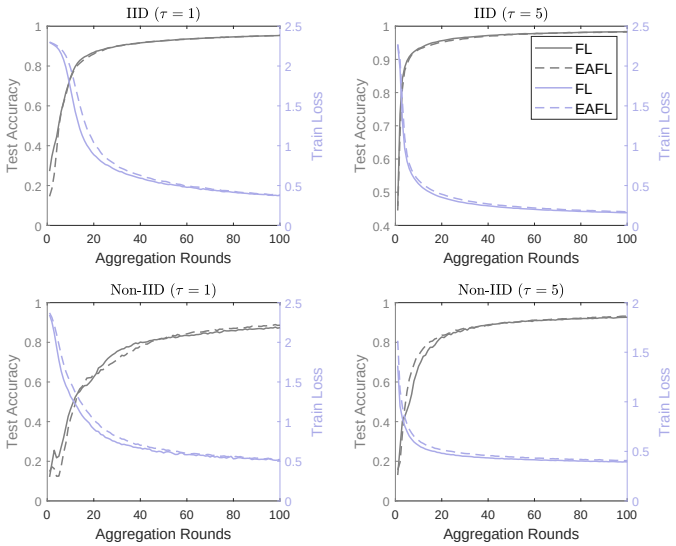


Fig. 7. Learning performance for the MNIST datasets with different data distributions (IID and non-IID) and different epoch number per round (τ).

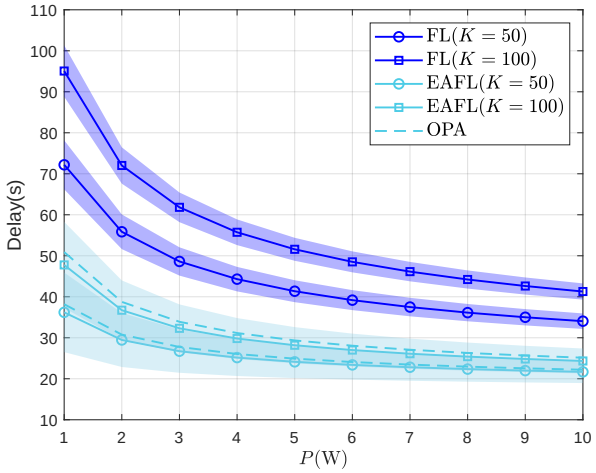


Fig. 8. System delay versus power constraint with different client numbers.

V. Simulation Results and Discussions

In this section, we provide simulation results to evaluate the performances of EAFL on both real datasets and synthetic datasets. Real datasets allow us to evaluate the convergence rate and the learning accuracy of EAFL, while synthetic datasets allow us to manipulate the heterogeneity of data more precisely and evaluate the EAFL delay performance more comprehensively. For real datasets, we choose MNIST dataset [38] due to its widely academic use. For synthetic datasets, the dataset size among clients follows the uniform distribution with $D_k \in [100, D_{max}]$Mb.

A. Learning Performance of EAFL

We distribute the MNIST dataset (400Mb total data size) and vary the number of samples among $K = 50$ clients. We follow the experimental settings in [4], where the learning model structure is a CNN with two 5×5 convolution layers, a fully connected layer with 512 units and ReLu activation, and a final softmax output layer (40Mb model size). Following the method in [4], we also perform the unbalanced partition of the MNIST data into IID and non-IID. For IID, the total data is shuffled and then partitioned into 50 clients each receiving 1200 examples. And for non-IID, we first sort the data by digit label, divide it into 200 shards of size 300, and assign each of 50 clients 4 shards. This is a pathological non-IID partition of the data, as most clients will only have examples of 4 digits. In order to explore the learning performance of the EAFL scheme, we randomly select a certain number of clients to send random-sized data to the server for training. The reference scheme is the original federated learning scheme.

Fig. 7 depicts the learning performance (*i.e.*, train loss and test accuracy) of federated learning and the proposed EAFL. It can be observed that under different data distributions and different number of epochs per round, the learning performance of these two schemes is close, which is consistent with the analysis in Remark 1. One can also observe that when the data is IID or τ is larger, the two schemes have a faster convergence rate and can achieve a higher test accuracy within 100 aggregation rounds. The explanation for this is that, the IID training data is easier to get a better fitting model than non-IID data. And reasonably selecting a larger number of epochs per round brings more local iterations, which will help improve the convergence rate.

B. Delay Performance of EAFL

The simulation settings for delay performance evaluation are as follows unless specified otherwise. The total client number is set as $K = 50$. The system power constraint is set as $P =$ 10W. All the channels are assumed to follow Rayleigh slow fading with path loss parameter ν set as 10^{-3}. The channel bandwidth is set as $B = 20$ MHz. The variance of complex white Gaussian channel noise is $N_0 = 10^{-6}$ W. The number of CPU cycles required for training 1-bit data is set as $C = 500$. The CPU frequency of the server e_s is set as 10^{10} cycles per second. The effective capacitance coefficient ζ is fixed to 10^{-18}, consistent with the measurements in [39]. We set $D_{max} = 1000$. The model parameter size is set as 40Mb, close to the CNN model size used in [4]. We reserve a fixed time slot 200ms for parameter aggregation and model broadcast. The number of iterations per round is set as $\tau = 1$. We set the parameter $\alpha = 1$ and $\beta = 40$. The target model training loss is set as $\epsilon = 0.5$. All results are averaged over 10^3 Monte-Carlo simulations and include 95% confidence intervals. In the following results, "FL" and "EAFL" represent federated learning and the proposed EAFL, respectively. Since coded computation is another method to mitigate the straggler effect, we also evaluate its delay performance by referring to [23]. The corresponding delay curve is represented as "CFL". Since we perform the Taylor approximation in the power allocation to obtain the explicit expression of $T_1^*(L_k)$, we derive the delays under the optimal power allocation and denote it as "OPA".

Fig. 8 depicts the system delay of different schemes versus the system power constraint. Several observations can be made. First, the delays of federated learning and the proposed

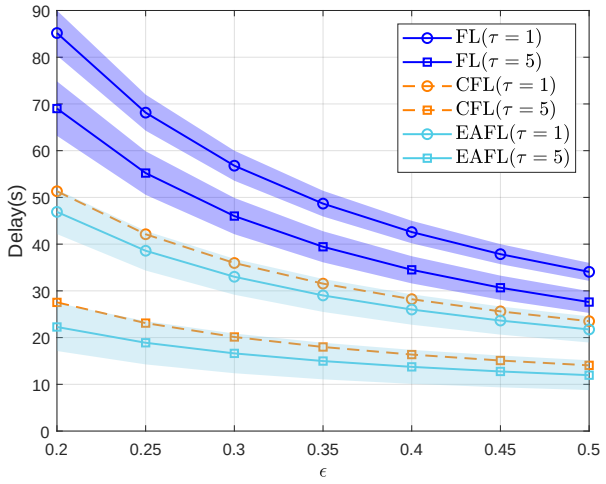


Fig. 9. System delay versus target model train loss with different numbers of epochs per round.

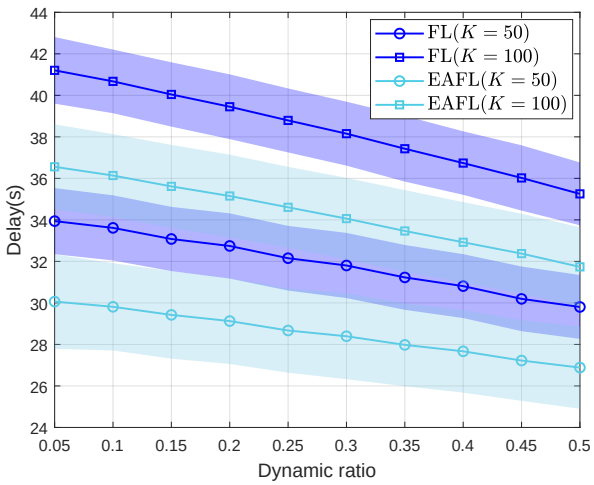


Fig. 11. System delay in the dynamic scenario versus dynamic ratio.

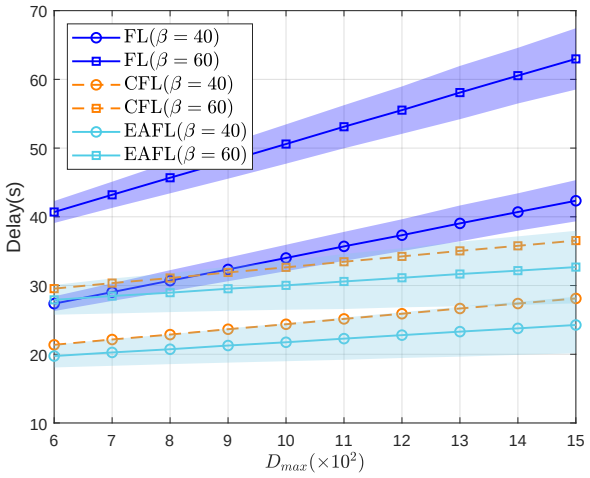


Fig. 10. System delay versus dataset size with different data distribution characteristics.

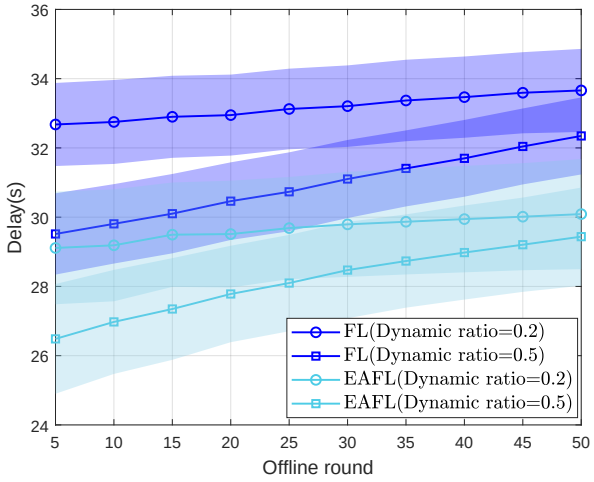


Fig. 12. System delay in the dynamic scenario versus the offline round of dynamic clients.

EAFL both decrease as the system power grows. This is because a larger power allocated for transmission or computation increases the processing efficiency of clients. Second, the delay increases as the number of clients increases. The explanation is that, although the system with more clients require less update rounds, the power allocated for transmission or computation is reduced, resulting in a larger delay. Next, the delay of EAFL is lower than federated learning with the proposed offloading strategy and is close to the EAFL performance under the optimal power allocation.

Fig. 9 shows the delay curves of three federated learning schemes versus the model train loss. One can observe that the delays of the three schemes reduce as the train loss ϵ grows. The reason is that, a larger training loss requires less update rounds for machine learning, resulting in a lower delay. It can be observed that the delay is smaller when $\tau = 5$ for the three schemes. This is because that the system with more epochs of update per round reduces the communication overhead of the aggregation and improves the convergence rate, resulting in the smaller total delay. In addition, we find that EAFL outperforms the coded computation method. The explanation is that, although the coded computation method only needs to wait for partial clients during each round, the update delay difference among these clients is larger than EAFL.

Fig. 10 illustrates the effect of dataset size and data distribution characteristics on the delay of the three schemes. We set D_{max} to vary from 600 to 1500. It can be observed that the system delay increases as the dataset size grows. The explanation is that, the growth of dataset size increases the computational or offloading burdens for clients, resulting in a larger system delay. Furthermore, in the figure, we choose two different values for β. One can observe that, a larger β (*i.e.*, larger degree of non-IID) consumes a larger delay since more training rounds are required. Under different dataset sizes and data distribution characteristics, EAFL always achieves a lower system delay compared with federated learning and the coded computation method.

For the dynamic scenario, we analyze a simple case where

the client group number is set as $m = 2$. We define the dynamic ratio as the ratio of dynamic client number to the total client number. We set the offline round of the dynamic clients as $N_1 = 30$. The curves of system delay versus the dynamic ratio for different total client numbers is plotted in Fig. 11. One can observe that, the system with high dynamic ratio has less delay for both federated learning and EAFL. The reason is that, the higher proportion of dynamic clients enables stable clients to gain more power in the subsequent update rounds. As can be seen from the figure, EAFL has smaller system delay for different dynamic ratios or client numbers.

Fig. 12 evaluates the performance of federated learning and EAFL with dynamic clients under different offline rounds. It can be observed that the earlier offline round will reduce the system delay. This is because a reasonably reduced number of clients for distributed learning will not significantly increase the number of update rounds, but gives each client more allocated power, thereby increasing the update speed.

VI. Conclusion

In this paper, we have proposed EAFL to mitigate the straggler effect in federated learning. In EAFL, stragglers offload partial computation to the edge server to reduce their computational burdens, which greatly improves the system efficiency. The performance of EAFL has been analyzed in terms of system delay. To optimize the offloading data size for each client, the delay optimization problem of EAFL has been formulated and solved. According to the problem solution, we have provided a threshold-based offloading strategy for EAFL. We have also extended EAFL to a more practical scenario with dynamic clients and have given the corresponding offloading strategy for this dynamic scenario. Extensive simulation results confirm the effectiveness of EAFL and the proposed offloading strategies.

References

[1] L. Cui, S. Yang, F. Chen, Z. Ming, N. Lu, and J. Qin, "A survey on application of machine learning for Internet of Things," *Int. J. Mach. Learn. Cybern.*, vol. 9, no. 8, pp. 1399–1417, 2018.

[2] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for IoT big data and streaming analytics: A survey," *IEEE Commun. Surv. Tutorials*, vol. 20, no. 4, pp. 2923–2960, 2018.

[3] "Cisco annual internet report (2018–2023) white paper," https://www.cisco.com/c/en/us/solutions/executive-perspectives/annual-internet-report/index.html.

[4] H. Brendan McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:1602.05629*, 2016.

[5] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[6] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, 2019.

[7] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[8] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Proc. Conf. Neural Inf. Process. Syst.*, 2017, pp. 4424–4434.

[9] D. Ye, R. Yu, M. Pan, and Z. Han, "Federated learning in vehicular edge computing: A selective model aggregation approach," *IEEE Access*, vol. 8, pp. 23 920–23 935, 2020.

[10] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[11] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE Trans. Neural Networks Learn. Syst.*, 2019.

[12] G. Zhu, Y. Wang, and K. Huang, "Broadband analog aggregation for low-latency federated edge learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 491–506, 2019.

[13] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun.* IEEE, 2019, pp. 1–7.

[14] W. Shi, S. Zhou, and Z. Niu, "Device scheduling with fast convergence for wireless federated learning," *arXiv preprint arXiv:1911.00856*, 2019.

[15] M. M. Amiri, D. Gunduz, S. R. Kulkarni, and H. V. Poor, "Update aware device scheduling for federated learning at the wireless edge," *arXiv preprint arXiv:2001.10402*, 2020.

[16] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *arXiv preprint arXiv:1712.07557*, 2017.

[17] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farhad, S. Jin, T. Q. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *arXiv*, pp. arXiv–1911, 2019.

[18] C. Ma, J. Li, M. Ding, H. H. Yang, F. Shu, T. Q. Quek, and H. V. Poor, "On safeguarding privacy and security in the framework of federated learning," *IEEE Network*, 2020.

[19] A. Harlap, H. Cui, W. Dai, J. Wei, G. R. Ganger, P. B. Gibbons, G. A. Gibson, and E. P. Xing, "Addressing the straggler problem for iterative convergent parallel ML," in *Proc. 7th ACM Symp. Cloud Comput.*, 2016, pp. 98–111.

[20] S. Li, S. M. M. Kalan, A. S. Avestimehr, and M. Soltanolkotabi, "Near-optimal straggler mitigation for distributed gradient methods," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops (IPDPSW)*. IEEE, 2018, pp. 857–866.

[21] Y. Chen, Y. Ning, and H. Rangwala, "Asynchronous online federated learning for edge devices," *arXiv preprint arXiv:1911.02134*, 2019.

[22] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," *arXiv preprint arXiv:1903.03934*, 2019.

[23] S. Dhakal, S. Prakash, Y. Yona, S. Talwar, and N. Himayat, "Coded federated learning," in *Proc. IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2019, pp. 1–6.

[24] S. Prakash, A. Reisizadeh, R. Pedarsani, and A. S. Avestimehr, "Hierarchical coded gradient aggregation for learning at the edge," in *Proc. IEEE ISIT*. IEEE, 2020, pp. 2616–2621.

[25] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, A. Neal *et al.*, "Mobile-edge computing introductory technical white paper," *White paper, mobile-edge computing (MEC) industry initiative*, pp. 1089–7801, 2014.

[26] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*. IEEE, 2016, pp. 1451–1455.

[27] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Networking*, vol. 24, no. 5, pp. 2795–2808, 2015.

[28] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, 2016.

[29] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, 2016.

[30] C. You, Y. Zeng, R. Zhang, and K. Huang, "Asynchronous mobile-edge computation offloading: Energy-efficient resource management," *IEEE Trans. Wireless Commun.*, vol. 17, no. 11, pp. 7590–7605, 2018.

[31] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Network*, vol. 33, no. 5, pp. 156–165, 2019.

[32] A. Sacco, F. Esposito, and G. Marchetto, "A federated learning approach to routing in challenged sdn-enabled edge networks," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2020, pp. 150–154.

[33] J. Ren, H. Wang, T. Hou, S. Zheng, and C. Tang, "Federated learning-based computation offloading optimization in edge computing-supported internet of things," *IEEE Access*, vol. 7, pp. 69 194–69 201, 2019.

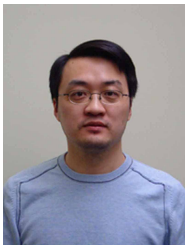
[34] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, 2013.

[35] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, 2019.
[36] S. Wang, Y.-C. Wu, M. Xia, R. Wang, and H. V. Poor, "Machine intelligence at the edge with learning centric power allocation," *IEEE Trans. Wireless Commun.*, vol. 19, no. 11, pp. 7293–7308, 2020.
[37] A. M. Elbir, B. Soner, and S. Coleri, "Federated learning in vehicular networks," *arXiv preprint arXiv:2006.01412*, 2020.
[38] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[39] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," *HotCloud*, vol. 10, pp. 4–4, 2010.

Zhongming Ji received the B.E. from the School of Computer Science and Information Engineering, HeFei University of Technology (HFUT), Hefei, China, in 2017. He is currently pursuing for his Ph.D. degree in Communication and Information Engineering at University of Science and Technology of China (USTC), Hefei, China. His current research interests include multi-access edge computing, federated learning and distributed coded computing.

Li Chen received the B.E. in electrical and information engineering from Harbin Institute of Technology, Harbin, China, in 2009 and the Ph.D. degree in electrical engineering from the University of Science and Technology of China, Hefei, China, in 2014. He is currently an Associate Professor with the Department of Electronic Engineering and Information Science, University of Science and Technology of China. His research interests include wireless IoT communications and wireless optical communications.

Nan Zhao (S'08-M'11-SM'16) is currently a Professor at Dalian University of Technology, China. He received the Ph.D. degree in information and communication engineering in 2011, from Harbin Institute of Technology, Harbin, China. Dr. Zhao is serving on the editorial boards of IEEE Wireless Communications and IEEE Wireless Communications Letters. He won the best paper awards in IEEE VTC 2017 Spring, ICNC 2018, WCSP 2018 and WCSP 2019. He also received the IEEE Communications Society Asia Pacific Board Outstanding Young Researcher Award in 2018.



Yunfei Chen (S'02-M'06-SM'10) received his B.E. and M.E. degrees in electronics engineering from Shanghai Jiaotong University, Shanghai, P.R.China, in 1998 and 2001, respectively. He received his Ph.D. degree from the University of Alberta in 2006. He is currently working as an Associate Professor at the University of Warwick, U.K. His research interests include wireless communications, cognitive radios, wireless relaying and energy harvesting.

Guo Wei received the B.S. degree in electronic engineering from the University of Science and Technology of China (USTC), Hefei, China, in 1983 and the M.S. and Ph.D. degrees in electronic engineering from the Chinese Academy of Sciences, Beijing, China, in 1986 and 1991, respectively. He is currently a Professor with the School of Information Science and Technology, USTC. His current research interests include wireless and mobile communications, wireless multimedia communications, and wireless information networks.

F. Richard Yu (S'00-M'04-SM'08-F'18) received the PhD degree in electrical engineering from the University of British Columbia (UBC) in 2003. From 2002 to 2006, he was with Ericsson (in Lund, Sweden) and a start-up in California, USA. He joined Carleton University in 2007, where he is currently a Professor. He received the IEEE Outstanding Service Award in 2016, IEEE Outstanding Leadership Award in 2013, Carleton Research Achievement Award in 2012, the Ontario Early Researcher Award (formerly Premiers Research Excellence Award) in 2011, the Excellent Contribution Award at IEEE/IFIP TrustCom 2010, the Leadership Opportunity Fund Award from Canada Foundation of Innovation in 2009 and the Best Paper Awards at IEEE ICNC 2018, VTC 2017 Spring, ICC 2014, Globecom 2012, IEEE/IFIP TrustCom 2009 and Int'l Conference on Networking 2005. His research interests include wireless cyber-physical systems, connected/autonomous vehicles, security, distributed ledger technology, and deep learning.

He serves on the editorial boards of several journals, including Co-Editor-in-Chief for Ad Hoc & Sensor Wireless Networks, Lead Series Editor for IEEE Transactions on Vehicular Technology, IEEE Transactions on Green Communications and Networking, and IEEE Communications Surveys & Tutorials. He has served as the Technical Program Committee (TPC) Co-Chair of numerous conferences. Dr. Yu is a registered Professional Engineer in the province of Ontario, Canada, a Fellow of the Institution of Engineering and Technology (IET), and a Fellow of the IEEE. He is a Distinguished Lecturer, the Vice President (Membership), and an elected member of the Board of Governors (BoG) of the IEEE Vehicular Technology Society.