To Talk or to Work: Dynamic Batch Sizes Assisted Time Efficient Federated Learning Over Future Mobile Edge Devices

Dian Shi, *Member, IEEE*, Liang Li, *Member, IEEE*, Maoqiang Wu, *Member, IEEE*, Minglei Shu, *Member, IEEE*, Rong Yu, *Senior Member, IEEE*, Miao Pan, *Senior Member, IEEE*, and Zhu Han, *Fellow, IEEE*

***Abstract*— The coupling of federated learning (FL) and multi-access edge computing (MEC) has the potential to foster numerous applications. However, it poses great challenges to train FL fast enough with limited communication and computing resources of mobile edge devices. Motivated by recent development in ultra fast wireless transmissions and promising advances in artificial intelligence (AI) computing hardware of mobile devices, in this paper, we propose a time efficient FL over future mobile edge devices, called dynamic batch sizes assisted federated learning (DBFL) with convergence guarantee. The DBFL allows batch sizes to increase dynamically during training, which can unleash the computing potential of GPU's parallelism for on-device training and effectively leverage the fast wireless transmissions (WiFi-6, 5G, 6G, etc.) of mobile edge devices. Furthermore, based on the derived DBFL's convergence bound, we develop a batch size control scheme to minimize the total time consumption of FL over mobile edge devices, which trade-offs the "talking", i.e., communication time, and "working", i.e., computing time, by adjusting the incremental factor appropriately. Extensive simulations are conducted to validate the effectiveness of our proposed DBFL algorithm and demonstrate that our scheme outperforms existing time efficient FL approaches in terms of the total time consumption in various settings.**

***Index Terms*— Federated learning, dynamic batch sizes, future mobile edge devices, on-GPU computing.**

Manuscript received 31 October 2021; revised 5 March 2022 and 21 May 2022; accepted 28 June 2022. Date of publication 14 July 2022; date of current version 12 December 2022. The work of Dian Shi and Miao Pan was supported in part by the U.S. National Science Foundation under Grant CNS-2107057. The work of Liang Li was supported in part by the National Key Research and Development Program of China under Grant 2020YFC1511801 and in part by the National Natural Science Foundation of China under Grant U2066201. The work of Rong Yu was supported in part by the National Natural Science Foundation of China under Grant 61971148. The work of Zhu Han was supported in part by the NSF under Grant CNS-2107216 and Grant CNS-2128368. The associate editor coordinating the review of this article and approving it for publication was J. Xie. *(Corresponding author: Minglei Shu.)*

Dian Shi and Miao Pan are with the Electrical and Computer Engineering Department, University of Houston, Houston, TX 77004 USA (e-mail: dshi3@uh.edu; mpan2@uh.edu).

Liang Li is with the School of Computer Science (National Pilot Software Engineering School), Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: liliang1127@bupt.edu.cn).

Maoqiang Wu and Rong Yu are with the School of Automation, Guangdong University of Technology, Guangzhou 510006, China (e-mail: maoqiang.wu@vip.163.com; yurong@ieee.org).

Minglei Shu is with the Shandong Artificial Intelligence Institute, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250353, China (e-mail: shuml@sdas.org).

Zhu Han is with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77004 USA, and also with the Department of Computer Science and Engineering, Kyung Hee University, Seoul 446-701, South Korea (e-mail: hanzhu22@gmail.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TWC.2022.3189320.

Digital Object Identifier 10.1109/TWC.2022.3189320

I. Introduction

EMBRACING recent advances in ultra-high speed wireless transmissions (e.g., WiFi-6, 5G, 6G, etc.) and mobile hardware technologies, multi-access edge computing (MEC) has recently emerged as a promising paradigm enabling local data analysis and real-time service provisioning. With this trend, federated learning (FL) further pushes artificial intelligence (AI) functions to mobile edge devices, thereby being instrumental in spearheading the vision of intelligent mobile edge networks. As we know, through local training updates, FL enables data stakeholders (e.g., mobile edge devices) to collaboratively learn a joint global machine learning (ML) model without sharing their private raw data. Various features of MEC make it perfectly fit to support FL. First, although current mobile edge devices (e.g., 4G or 5G smartphones, tablets, etc.) can only make learning inferences, future mobile edge devices in MEC will be widely equipped with high-performance integrated processors (e.g., high-performance GPUs) and are expected to have on-device local training capability [1], given the promising research advances in mobile computer architecture designs and computing hardware development. Besides, the distributed architecture of MEC [2] is consistent with that of FL, where future mobile edge devices can serve as local FL clients, and the edge server can serve as the FL aggregator. Moreover, the widespread popularity of social networking applications will breed a wealth amount of data continually generated on future mobile edge devices, which provides data basic for on-device training. Therefore, the coupling of MEC and FL can prompt a broad range of applications, including keyboard prediction [3], cardiac event prediction [4], financial risk management, etc.

However, deploying FL over future mobile edge devices in practice is non-trivial and poses great challenges, of which

time consumption is a critical concern. In particular, the proliferation of real-time and lifelong applications with stringent requirements on low latency, such as augmented reality and voice assistant [5], has spurred a growing need for continuous training on mobile edge devices. The mismatch between the delay-sensitive property of these applications and the limited resources of mobile edge networks raises two fundamental issues of FL over future mobile edge devices. One is how to fundamentally accelerate the training process from the algorithmic perspective while guaranteeing learning convergence and model accuracy. It is also challenging for future mobile edge devices to efficiently execute the time consuming neural network training, which may cause intolerable latency for certain delay-sensitive FL applications. The other is how to determine the key training parameters adapting to the mobile edge environment in practice so that the system resources, such as GPU resources and wireless bandwidth, can be fully utilized to improve the time efficiency.

Most existing works in the machine learning community focused on improving the communication efficiency of FL since communication is usually regarded as a bottleneck in the training procedure. In [6], the local SGD (a.k.a. FedAvg) algorithm is proposed to allow every participating device to perform multiple stochastic gradient descent (SGD) iterations locally before synchronizing with others, thereby avoiding communication after every local iteration. Another notable method to reduce the necessary communication rounds is gradually enlarging the batch sizes during training, which properly increases local computations in a single round to reduce the variance [7]–[10]. Although these methods exhibit great potential in communication burden alleviation, it is questionable whether the communication is a bottleneck in itself. In fact, the rapidly expanding 5G networks can provide a high transmission rate up to 1 Gbps, and WiFi-6 claims to have a peak throughput of 9.6 Gbps. Furthermore, the forthcoming 6G networks are envisioned to open up a "Tbps" era. These advanced technologies make transmission delay no longer a dominant issue hindering FL's implementation in wireless networks [11], [12]. In this situation, wireless transmissions and local computing are comparable in time consumption. For example, performing a single-step local iteration (including data accessing and computing) of a ResNet-50 model with 100MB parameters on one GPU typically takes hundreds of milliseconds [13], which is similar to the transmission delay introduced by uploading the model via 1 Gbps wireless links (i.e., 800 ms).

Recognizing the comparability between the communication cost and computing cost, some recent works in [14]–[22] study to reduce the FL system cost over mobile edge devices via joint communication and computing resource management. Their methods just allocate system resources under a given budget and strive to acquire a proper resource allocation strategy matching the network environment. However, essentially saving the total time consumption from the learning's perspective is widely overlooked in these works, such as how to adjust the appropriate batch size. Thus, it is worthwhile to investigate the trade-off between "working" (i.e., local computing) and "talking" (i.e., wireless communication) from both the learning algorithm and resource allocation perspectives for efficient FL over future mobile edge devices.

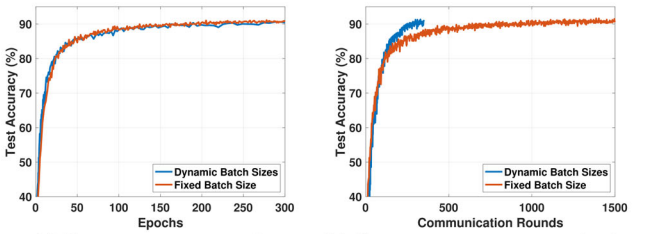
To bridge this gap, this paper targets to accelerate the FL training process by jointly considering the computing and communication conditions over future mobile edge devices. To this end, we first propose a time-efficient FL algorithm, named **d**ynamic **b**atch sizes assisted **f**ederated **l**earning (**DBFL**), with the convergence guarantee. Unlike ordinary local SGD using fixed batch sizes throughout the training, the DBFL allows users to exponentially increase the batch sizes with an incremental factor, leading to a reduction of communication rounds required to complete the training. We will explain the reduction in detail in the following section. We then employ the proposed DBFL algorithm in wireless networks and develop a batch size control scheme adapting to the specific network conditions. In particular, we determine the optimal incremental factor via an elaborate optimization problem, where "working" and "talking" at future mobile edge devices are well balanced to minimize the total training time consumption. In addition, the goal of improving time efficiency is in general consistent with that of improving energy efficiency, as the shortening of overall time consumption leads to the reduction of overall energy consumption. Our salient contributions are summarized as follows:

- Capturing the learning dynamics, we propose the DBFL, a time efficient FL algorithm allowing the batch size to increase exponentially in the training process. We also provide a theoretical analysis of the DBFL algorithm in terms of convergence rate and communication complexity.
- Guided by the theoretical results of DBFL, we further study to minimize the total time consumed for training an FL model to converge over future mobile edge devices. In particular, we develop a batch size control scheme to derive the optimal batch size incremental factor, catering to the GPU computing performances and wireless communication conditions of mobile edge devices.
- We conduct extensive simulations to evaluate the performance of the proposed scheme on various learning models and system settings. Our scheme exhibits great superiority in terms of time consumption reduction for FL over future mobile edge devices compared with the state-of-the-art FL solutions.

The rest of this paper is organized as follows. Section II reviews some preliminaries of dynamic batch sizes. Section III describes the detailed DBFL algorithm with the convergence analysis. The formulation and solution of the batch size control scheme over future mobile edge devices are presented in Section IV. Numerical simulations are provided in Section V. The related works are summarized in Section VI, and finally, Section VII concludes the paper.

II. Preliminaries of Dynamic Batch Sizes

In order to better explain the motivation of this paper, the learning schemes with the fixed batch size and dynamic batch sizes are compared through experiments in this part. Taking the ResNet20 [23] model on the CIFAR-10 dataset as an example, we consider an FL scenario with 10 participating users, and



(a) Test accuracy vs. epochs (b) Test accuracy vs. communications

Fig. 1. FL with dynamic batch sizes (ResNet20 on CIFAR-10).

TABLE I
GPU PERFORMANCE

Size	50	100	200	250	500	1000	2500
Time(s)	0.021	0.022	0.023	0.024	0.040	0.072	0.171
Ratio	6.140	3.216	1.681	1.404	1.170	1.052	1

each user sequentially takes 10 local SGD steps. Based on this, we deploy this scenario with the fixed scheme (FedAvg [6] with batch size 200) and the dynamic scheme (gradually increasing the batch sizes from 50 to 2,500), respectively, and the experiment results are shown in Fig. 1. Note that the fixed scheme needs the relatively small batch sizes (e.g., smaller than 200) in the FL to guarantee the convergence [6]. Fig. 1(a) shows that two schemes need almost the same data epochs to achieve the target accuracy, and one epoch refers to one cycle through the full training dataset. This means that the gradient calculation operations of two schemes, i.e., the computation loads during the training, are similar. The training curves with data epochs as x-axis are similar in 1(a). But for the global updates, i.e., communication cost, shown in Fig. 1(b), the dynamic scheme needs far fewer communication rounds than the fixed scheme. The reason is that the large batch size implemented in the latter training stage leads to the reduction of update frequency.

In this paper, we focus on the time consumption for the overall FL procedure, including the communication time and the computing time. From Fig. 1(b), it can be easily found that the communication cost of the dynamic scheme is much less than that of the fixed scheme due to the fewer required communication rounds. In terms of the computing cost, when using a similar number of data epochs to achieve FL convergence in the two schemes (as shown in Fig. 1(a)), the total computing time of the dynamic scheme is lower than that of the fixed scheme due to the efficient large-batch training in the later stage of training. Specifically, the local gradient calculation delay does not increase linearly with the batch size increase. For example, when the batch is doubled, the increase in time consumption will be less than twice. Such a relationship between batch size and time consumption indicates the efficiency of the large batch training. This is because the GPU pipeline is a kind of parallel processing, where it can directly process far more data simultaneously [24]. As shown in Table I, we conduct the experiments on a single "RTX 8000" GPU and record the time consumption for different batch sizes. The "Ratio" indicates the ratio of "Time for computing 2,500 samples with specified batch size" and "Time for computing 2,500 samples with batch size 2,500". This "Ratio" implies that a larger batch size can save more time when accessing the same number of stochastic gradients, and similar results can be found in [25]. Thus, compared with the fixed scheme that always uses relatively small batch sizes, the dynamic batch size scheme consumes less computing time for convergence, which is consistent with the trend of communication consumption. Moreover, to further reduce the system time consumption and balance the communication and computing in the dynamic scheme, we are going to adjust the batch size increasing rate to achieve the minimum time consumption for the training.

III. Federated Learning With Dynamic Batch Sizes

In this section, we propose a general time efficient FL framework with dynamic batch sizes in Sec. III-A, named **DBFL**, and derive the convergence analysis for it in Sec. III-B.

A. Dynamic Batch Sizes Assisted FL Algorithm Design

We consider a multi-access edge computing system for the distributed machine learning task with one edge server and a set $\mathcal{K} := \{1, \ldots, K\}$ of K participating future mobile edge devices, as shown in Fig. 2. Specifically, each device i has its own dataset $\mathcal{D}_i$ and cannot access other devices' datasets. Moreover, all devices maintain their local machine learning models with the same model structure and attempt to achieve a global goal, i.e., obtain a common model to minimize the training loss, under the coordination of the edge server. Such a scenario can be considered as a FL task, where the future mobile edge server can serve as the FL aggregator and future mobile edge devices can serve as local FL clients. In addition, FL can be also formulated as the following distributed non-convex optimization problem

$$\min_{\mathbf{w}\in\mathbb{R}^d} f(\mathbf{w}) \triangleq \frac{1}{K}\sum_{i=1}^{K} f_i(\mathbf{w}), \tag{1}$$

where $f_i(\mathbf{w})$ is the training loss for device i over the dataset $\mathcal{D}_i$, i.e., $f_i(\mathbf{w}) \triangleq \mathbb{E}_{\xi_i\sim\mathcal{D}_i}\left[F_i\left(\mathbf{w};\xi_i\right)\right]$. Here, we assume all users have the same size of dataset $\mathcal{D}_i$ for concise expression, and the size of $\mathcal{D}_i$ can be possibly different for different i. ξ denotes the randomness in the training process, e.g., different data points and different batch sizes selection. In every training iteration t, each device will take one SGD step, which means each device observes the unbiased stochastic gradients with one batch data based on the model $\mathbf{w}_i^{t-1}$ obtained at the last iteration step as $\mathbf{g}_i^t = \nabla F_i\left(\mathbf{w}_i^{t-1};\xi_i\right)$. Here, $\mathbf{g}_i^t$ is the unbiased estimation of $\nabla f_i(\mathbf{w}_i^{t-1})$, i.e., $\mathbb{E}_{\xi_i^t\sim\mathcal{D}_i}\left[\mathbf{g}_i^t \mid \boldsymbol{\xi}\right] = \nabla f_i(\mathbf{w}_i^{t-1})$.

One classical method to coordinate all devices' local model is to collect and take the average of the observed gradients among all devices in each iteration, then adopt the averaged gradients to update the global model. Furthermore, each device downloads the updated global model and continues to take the SGD step mentioned above for training. Such an optimization method is called mini-batch SGD. However, updating the global model in each iteration with only one SGD step is extremely communication inefficient, which consumes a

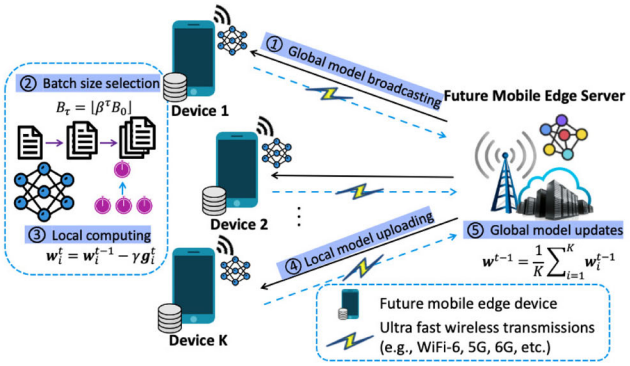


Fig. 2. The illustration of the dynamic batch sizes assisted federated learning.

tremendous amount of the limited communication resource, especially for the FL over mobile edge devices. Hence, we employ a communication efficient optimization method, called local SGD, where each device performs H sequential SGD steps and then updates the global model. Note that each data sample can be repetitively used during local updates. But it will have different effects on the model since the local model accumulates the previous gradient calculation results. Therefore, we will update the global model every H iteration. For explanation convenience, we define a virtual global model at each iteration step t as

$$\overline{\mathbf{w}}^t \triangleq \frac{1}{K}\sum_{i=1}^{K} \mathbf{w}_i^t, \tag{2}$$

and the virtual global model can be iteratively calculated as

$$\overline{\mathbf{w}}^t = \overline{\mathbf{w}}^{t-1} - \gamma \frac{1}{K}\sum_{i=1}^{K} \mathbf{g}_i^t, \tag{3}$$

where γ is the learning rate. Note that when t mod $H = 0, \mathbf{w^t} = \overline{\mathbf{w}}^{\mathbf{t}}$.

Moreover, to further reduce the communication cost and improve computing efficiency, we consider a dynamic batch size in the training process. We gradually increase the batch size in the training stage and propose a time-efficient federated learning approach, called **d**ynamic **b**atch sizes assisted **f**ederated **l**earning (**DBFL**) algorithm, which is described in Alg. 1. Particularly, in each communication round τ, the batch size is different and exponentially increased with an incremental factor β, i.e., $B_\tau = \lfloor \beta^{\tau-1} B_0 \rfloor$. In Alg. 1, E is the total number of stochastic gradient accesses. The proposed DBFL procedure over future mobile edge devices is briefly described in Fig. 2. The edge server first broadcasts a current global model to the participating mobile edge devices in FL. After receiving the global model, each mobile edge device selects the batch size for local on-device training based on the local data and its computing capability. When a mobile edge device finishes its local training, it will upload its local model via ultra fast wireless transmissions (e.g., WiFi-6, 5G, 6G, etc.) for global model updates. The above steps will repeat until the training converge. The proposed DBFL algorithm can effectively reduce the communication rounds of global updates and benefit from the time efficient large batch training, which had been discussed in detailed in Sec. II. Furthermore, the convergence analysis of the proposed DBFL algorithm will be provided in the next section.

Algorithm 1 Dynamic Batch Sizes Assisted Federated Learning Algorithm (DBFL)

Initialization: Initialize the global model $\mathbf{w}^0$ and set $\mathbf{w}_i^0 = \mathbf{w}^0, \forall i \in \mathcal{K}$; Set the learning rate γ, batch size incremental factor β, local step H, and initial batch size B_0; Initialize the communication index $\tau = 0$ and the iteration index $t = 1$; Initialize the local step count lo

while $H \cdot \sum_{\eta=0}^{\tau} B_\eta \leq E$ **do**
  Each device i identifies the batch size $B_\tau = \lfloor \beta^\tau B_0 \rfloor$
  for $lo = 1, \ldots, H$ **do**
    Each device i observes the unbiased stochastic gradients $\mathbf{g}_i^t$ of $f_i(\mathbf{w}_i^{t-1})$ with one batch data with the size B_τ from the dataset $\mathcal{D}_i$
    Each device i in parallel updates its local model

$$\mathbf{w}_i^t = \mathbf{w}_i^{t-1} - \gamma \mathbf{g}_i^t, \quad \forall i$$

    Update $t \leftarrow t + 1$
  end for
  Update the global model $\mathbf{w}^{t-1} = \frac{1}{K}\sum_{i=1}^{K} \mathbf{w}_i^{t-1}$
  Each device i in parallel updates its local model $\mathbf{w}_i^{t-1} = \mathbf{w}^{t-1}$
  Update $\tau \leftarrow \tau + 1$
end while

From the theoretical point of view, gradually increasing batch sizes is also beneficial for the training. We can interpret the SGD method as integrating a stochastic differential equation (SDE) whose "noise scale" $n \approx \gamma|\mathcal{D}|/B$ [13], [26], where $|\mathcal{D}|$ is the dataset size and B denotes the batch size. For the above non-convex optimization problem, large-scale random fluctuations help to explore the parameter space to avoid trapping in local minima in the initial stage. After that, when we locate a promising region of parameter space, small-scale fluctuations are required to fine-tune the parameters in the later stage. Therefore, exponentially gradually increasing batch sizes is helpful in the training.

B. Convergence Analysis for the DBFL Algorithm

We consider the loss function f_i, $\forall i$ in (1) satisfies the following two assumptions:

Assumption 1 (Smoothness): The objective function f_i is differentiable and L-smooth:

$$\|\nabla f_i(\mathbf{x}) - \nabla f_i(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|, \forall i. \tag{4}$$

Assumption 2 (Bounded Variances and Second Moments): The variance and the second moments of stochastic gradients evaluated with a mini-batch of size B can be bounded as

$$\mathbb{E}_{\xi_i \sim \mathcal{D}_i} \left\|\nabla F_i\left(\mathbf{w}; \xi_i\right) - \nabla f_i(\mathbf{w})\right\|^2 \leq \frac{\sigma^2}{B}, \quad \forall \mathbf{w}, \ \forall i, \tag{5}$$

$$\mathbb{E}_{\xi_i \sim \mathcal{D}_i} \left\|\nabla F_i\left(\mathbf{w}; \xi_i\right)\right\|^2 \leq \delta^2, \quad \forall \mathbf{w}, \ \forall i, \tag{6}$$

where σ and δ are the positive constants.

We set the total number of iterations as T. Under the above assumptions, the following theorem holds, which is a key property of analyzing the convergence of the DBFL algorithm.

Theorem 1: If we consider the initial batch size B_0, batch size incremental factor $\beta > 1$, local step H, the number of participating device K, and choose the learning rate $\gamma < \frac{1}{L}$, then the convergence rate for all $T > 1$ in Alg. 1 satisfies:

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\left\|\nabla f\left(\overline{\mathbf{w}}^{t-1}\right)\right\|^2\right] \leq \frac{2\left(f\left(\overline{\mathbf{w}}^{0}\right)-f^*\right)}{\gamma \log_\beta \frac{E(\beta-1)}{B_0}} + 4\gamma^2 H^2 \delta^2 L^2 + \frac{2L\gamma\sigma^2}{\log_\beta \frac{E(\beta-1)}{B_0} K B_0 H}\cdot\frac{\beta}{\beta-1}, \quad (7)$$

where f^* is the minimum value of the loss, and E is the total number of stochastic gradient accesses.

Proof: See Appendix A ■

For Theorem 1, we can derive the following corollary.

Corollary 1: Under Assumptions 1 and 2, if we choose learning rate $\gamma = \frac{\sqrt{K}}{L\sqrt{\log_\beta \frac{E(\beta-1)}{B_0}}}$, local steps $H \leq \left(\log_\beta \frac{E(\beta-1)}{B_0}\right)^{\frac{1}{4}} K^{-\frac{3}{4}}$, the number of participating devices $K < (\log E)^{\frac{5}{3}}$, then the Alg. 1 has the $O\left(\frac{1}{\sqrt{K \log E}}\right)$ convergence rate with $O\left((\log EK)^{\frac{3}{4}}\right)$ communication rounds.

Proof: See Appendix B ■

Furthermore, the communication complexity $O(T/H)$ can be considered as $O\left((K \log E)^{\frac{3}{4}}\right)$, which is lower than that of the local SGD method (FedAvg) with a fixed batch size as $O((KE)^{\frac{3}{4}})$ [27].

IV. DBFL Over Future Mobile Edge Devices

Both the experimental results obtained in Sec. II and the convergence analysis derived in Sec. III-B validate that the proposed dynamic batch sizes scheme can reduce the communication rounds of global updates, and the experimental results also show the efficiency of the large batch computing. Therefore, the total time consumption will be reduced when using the proposed DBFL algorithm. In this section, we employ the DBFL algorithm over future mobile edge devices in Sec. IV-A and develop a control scheme in Sec. IV-B to adjust the batch size incremental factor β for further minimizing the total time consumption of FL, including both the communication time and computing time.

A. Problem Formulation

We consider the multi-access edge computing environment in practice and develop the corresponding wireless communication and GPU computing model for future mobile edge devices.

Communication: For each future mobile edge device, the average transmission rate over the whole training process can be evaluated as

$$R = W\mathbb{E}_h\left[\log_2\left(1+\frac{P|h|^2}{N_0}\right)\right], \quad (8)$$

where the expectation is taken over the channel fading h, and N_0 implies the power of additive white Gaussian noise (AWGN). Here, we consider the ideal transmission condition, where we assume the interference can be well-managed [28], [29] and channel conditions are stable. W and P denote the bandwidth and the transmission power, respectively. Afterward, the transmission delay for transmitting the learning model with the model size N in one communication round can be evaluated as

$$t^{tran} = \frac{N}{R}. \quad (9)$$

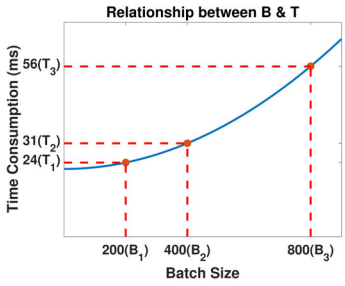


Fig. 3. The relationship between batch sizes and time consumption.

Computing: Future smart mobile edge devices will be widely armed with high-performance GPUs, which will be powerful in handling sophisticated computations of FL tasks. Consequently, we consider the GPU computing model and formulate the execution time for GPU computing the gradient of one data sample as:

$$t_0^{comp} = t_{init} + \frac{u}{f_{mem}} + \frac{v}{f_{core}}, \quad (10)$$

where f_{core} and f_{mem} denote the GPU core frequency and GPU memory frequency, and t_{init} represents the static time consumption [30], [31]. u and v are constant factors that reflect the sensitivity of the task execution to GPU memory and core frequency scaling. Therefore, the time consumption for accessing a batch size of B data can be calculated as

$$t^{comp}(B) = t_0^{comp}\cdot d(B), \quad (11)$$

where $d(B)$ is a function describing the relationship between the batch size and time consumption.

Due to the parallelism property of the GPU, the time consumption for calculating the gradients with different data sizes does not linearly increase with the batch sizes increasing, as shown in Table I. The first is that the per data sample time consumption gradually decreases with the batch size increasing. The second one is that the efficiency improvement of the large batch training gradually also decreases with the increasing batch size. Moreover, we find that when the batch size is not extremely, a part of the quadratic function can fit such a nonlinear relationship, as shown in Fig. 3, and have the assumption below. Similar observations can also be found in [25]. Therefore, we choose the quadratic function $d(B) = aB^2 + c$ in (11), where $a > 0$ and $c > 0$, to describe the nonlinear relationship, which can well reflect the efficiency of GPU computing for large batch sizes. Note that we use a segment of the quadratic function to represent the time consumption; we can select different values of a and c for different types of GPUs.

Assumption 3 (Quadratic Relationship): For the general CNN model, when the batch size is not extremely large, the relationship between the GPU time consumption and batch sizes of gradient calculation follows the quadratic form.

After establishing the communication and computing model, the next step is to identify the required communication rounds. According to Theorem 1, we have the following corollary on required communication rounds of the DBFL algorithm.

Corollary 2: The maximum number of communication rounds $M = \frac{T}{H}$ for achieving ϵ global convergence accuracy, i.e., satisfying

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\left\|\nabla f\left(\overline{\mathbf{w}}^{t-1}\right)\right\|^2\right] \leq \epsilon, \tag{13}$$

is given by

$$M_\epsilon = \frac{T}{H} = \frac{A\beta}{\beta-1} + \chi, \tag{14}$$

where both A and χ are positive constants.

Proof: According to (36), if we achieve the ϵ global convergence accuracy, i.e., $\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\left\|\nabla f\left(\overline{\mathbf{w}}^{t-1}\right)\right\|^2\right] \leq \epsilon$, we will have

$$\epsilon = \frac{2}{\gamma T}\left(f\left(\overline{\mathbf{w}}^{0}\right) - f^*\right) + 4\gamma^2 H^2\delta^2 L^2 + \frac{2L\gamma\sigma^2 H}{TKB_0}\frac{\beta}{\beta-1}.$$

Then, we have

$$\begin{aligned} M_\epsilon = \frac{T}{H} &= \frac{2bB_0K(\beta-1) + 2\gamma^2\sigma^2\beta HL}{B_0H\gamma K(\beta-1)\left(\epsilon - 4\delta^2\gamma^2H^2L^2\right)} \\ &= \frac{2b}{H\gamma\left(\epsilon - CH^2\right)} + \frac{\varrho\beta}{B_0K\left(\epsilon - CH^2\right)(\beta-1)} \qquad (15) \\ &= O(\frac{1}{H^3}) + O(\frac{\beta}{H^2K(\beta-1)}), \qquad (16) \end{aligned}$$

where we define $\varrho = 2L\gamma\sigma^2$, and $b = f\left(\overline{\mathbf{w}}^{0}\right) - f^*$. We have $C = 4\gamma^2\delta^2L^2$, and we further choose $\epsilon > CH^2$.

Therefore, the communication round M can be represented as $\frac{A\beta}{\beta-1} + \chi$, where both A and χ are positive constants. Moreover, A and χ can be determined with $\frac{1}{H^2K}$ and $\frac{1}{H^3}$ according to (16), respectively. ■

When we employ the DBFL algorithm, the total time consumption Φ can be calculated as in (12), shown at the bottom of the page, which is the summation of the total communication time and the total computing time. Hence, aiming at minimizing the overall time consumption, we determine the best incremental factor β by solving the following optimization problem:

$$\min_{\beta}\ \Phi \tag{17a}$$

$$s.t.\ \beta \geq \beta_{\min} \tag{17b}$$

$$B_0\beta^M \leq \min_i |\mathcal{D}_i|, \tag{17c}$$

where $\beta_{\min} > 1$, $|\mathcal{D}_i|$ is the size of the dataset owned by device i. The constraint in (17c) limits the maximum value of the selected batch size to be smaller than the local dataset size, which can also be simplified as,

$$\begin{aligned} \ln\beta\cdot\frac{(A+\chi)\beta-\chi}{\beta-1} &\leq \ln\frac{\min_i|\mathcal{D}_i|}{B_0}, \\ \ln\beta((A+\chi)\beta-\chi) &\leq Z(\beta-1), \end{aligned} \tag{18}$$

where $Z = \ln\frac{\min_i|\mathcal{D}_i|}{B_0}$.

We look into some critical parameters of the objective function Φ with further analysis. The total time consumption includes two parts that are communication consumption ("talking") and computing consumption ("working"). The results in (14) indicates that the incremental factor β takes impact on communication rounds M. Given a target global accuracy, a larger β results in a smaller number of communication rounds (less "talking"). However, enlarging the incremental factor leads to an increase in the calculations each round (more "working"). Therefore, there exists a trade-off between communication cost and computing cost that requires us to find an optimal β to determine "work more" or "talk more" to minimize the total time consumption. Furthermore, the overall time consumption decreases as the number of devices K increases. Still, the relationship between the total time consumption and the local step H is unknown because of the non-monotonicity.

B. Optimal Control Solutions for DBFL

For notational convenience, we define following functions,

$$\begin{aligned} P_1(\beta) &= ((A+\chi)\beta-\chi)(t^{tran} + cHt_0^{comp})(\beta+1) \\ P_2(\beta) &= aB_0^2Ht_0^{comp}(e^{\ln\beta\cdot\frac{2(A+\chi)\beta-2\chi}{\beta-1}} - 1) \\ P_3(\beta) &= \beta^2 - 1 \\ G(\beta) &= \ln\beta((A+\chi)\beta-\chi) - Z(\beta-1), \end{aligned} \tag{19}$$

The objective function Φ of the optimization problem is in a fractional form. To solve it efficiently, we employ the Dinkelbach [32] method to iteratively find the optimal results, which is widely considered in dealing with fractional programming problem. Introduce an auxiliary variable q and set q as the value of the objective function Φ, that is, $q = \frac{P_1(\beta)+P_2(\beta)}{P_3(\beta)}$. Accordingly, we have $P_1(\beta) + P_2(\beta) - qP_3(\beta) = 0$, and we further define the function $P_1(\beta) + P_2(\beta) - qP_3(\beta)$ as

$$\begin{aligned} \Phi &= M\cdot t^{tran} + H\sum_{\tau=1}^{M} t_0^{comp}(a(B_0\beta^{\tau-1})^2 + c), \\ &= \frac{(A+\chi)\beta-\chi}{\beta-1}\cdot(t^{tran} + cHt_0^{comp}) + aB_0^2Ht_0^{comp}\frac{1-\beta^{\frac{2(A+\chi)\beta-2\chi}{\beta-1}}}{1-\beta^2} \\ &= \frac{((A+\chi)\beta-\chi)(t^{tran} + cHt_0^{comp})(\beta+1) + aB_0^2Ht_0^{comp}(e^{\ln\beta\cdot\frac{2(A+\chi)\beta-2\chi}{\beta-1}} - 1)}{\beta^2-1}. \end{aligned} \tag{12}$$

$\Theta(\beta, q)$. Here, $\Theta(\beta, q)$ is strictly monotonic decreasing with the parameter q. Notice that solving the problem in (17) is essentially equivalent to finding the root of $\Theta(\beta, q)$. Therefore, we use the parametric approach to represent the objective in (17) and consider the following problem:

$$\min_{\beta} \; P_1(\beta) + P_2(\beta) - qP_3(\beta) \tag{20a}$$
$$s.t. \; \beta \geq \beta_{\min}, \tag{20b}$$
$$G(\beta) \leq 0. \tag{20c}$$

We iteratively solve the problem (20) and update the non-negative variable q in each iteration. Finally, q will be the unique root of the function (20a), which is also the optimal value of original problem (17). The updating process of variable q is described in Alg. 2. In general, the solution of the original problem (17) can be summarized as follows. We first solve problem (20) and denote the optimal solution as β^*, which is referred to as the inner loop in Alg. 2. Then, we update the corresponding q value, referred to as the outer loop in Alg. 2. Finally, the algorithm repeats the above two-loop steps until the convergence condition is reached. We further investigate the problem (20) and have the following lemma.

Lemma 1: The function $P_1(\beta)$, $P_3(\beta)$, $G(\beta)$ are convex functions in problem (20).

Proof: $P_1(\beta)$ and $P_3(\beta)$ can be easily verified to be convex. For the function $G(\beta)$, we have

$$\frac{\mathrm{d}^2 G(\beta)}{\mathrm{d}^2 \beta} = \frac{\mathrm{d}((A+\chi)(\ln\beta + 1))}{\mathrm{d}\beta} - \frac{\mathrm{d}(\chi\beta^{-1} + Z)}{\mathrm{d}\beta} = \frac{A+\chi}{\beta} + \frac{\chi}{\beta^2} > 0, \tag{21}$$

which implies that the function $G(\beta)$ is convex. ■

We now first focus on the function $P_2(\beta) = aB_0^2 H t_0^{comp} (e^{\ln\beta \cdot \frac{2(A+\chi)\beta - 2\chi}{\beta - 1}} - 1)$. Let $U_1(\beta) = h(g(\beta)) = e^{g(\beta)}$ where

$$h(x) = e^x, \quad \forall x \in \mathbb{R} \tag{22}$$
$$g(\beta) = \ln\beta \cdot \frac{2(A+\chi)\beta - 2\chi}{\beta - 1}. \tag{23}$$

Then $P_2(\beta)$ can be rewritten as $P_2(\beta) = aB_0^2 H t_0^{comp} (U_1(\beta) - 1)$. Notice that $g(\beta)$ is concave since

$$\frac{\mathrm{d}^2 g(\beta)}{\mathrm{d}^2 \beta} \leq 2 \cdot \frac{\chi(\beta-1)^3 + A\beta(\beta-1)^2}{(1-\beta)^3 \beta^2} < 0, \tag{24}$$

Thus, it is difficult to judge the convexity of the composition function $U_1(\beta)$, as well as $P_2(\beta)$. Thanks to the convexity of function $P_i(\beta), \forall i = 1, 3$ and the non-negativity of parameter q introduced in the Dinkelbach framework, the composite function $P_1(\beta) - qP_3(\beta)$ is in the Difference of Convex (DC) structure. Let $U_2(\beta) = P_1(\beta) - qP_3(\beta)$. We then re-represent the problem in (20) as:

$$\min_{\beta} \; a \; B_0^2 H t_0^{comp} (U_1(\beta) - 1) + U_2(\beta) \tag{25a}$$
$$s.t. \; \beta \geq \beta_{\min}, \tag{25b}$$
$$G(\beta) \leq 0. \tag{25c}$$

We then focus on solving the problem in (25) with non-convex objective (25a), by applying the iNner cOnVex Approximation (NOVA) [33] method and finding the stationary points iteratively. The main idea is to continuously optimize certain approximations of the non-convex objective in (25) and maintain feasibility at each iteration. This requires us to derive a strongly convex approximant of (25a) around each feasible iteration. To this end, we build an approximation for functions $U_1(\beta)$ and $U_2(\beta)$ as follows:

$$\tilde{U}_1(\beta; \beta^\nu) \triangleq h(g(\beta^\nu) + \nabla g(\beta^\nu)(\beta - \beta^\nu)) + \frac{\kappa}{2}\|\beta - \beta^\nu\|^2, \tag{26}$$
$$\tilde{U}_2(\beta, \beta^\nu) \triangleq P_1(\beta) - qP_3(\beta^\nu) - q\nabla P_3(\beta^\nu)(\beta - \beta^\nu), \tag{27}$$

where β^ν denote the current intermediate β obtained in the ν-th iteration. $\nabla g(\beta^\nu)$ and $\nabla P_3(\beta^\nu)$ are the gradients of g and P_3 at β^ν, respectively, and $\kappa > 0$. To acquire the optimal solution β^* in (25), we iteratively compute the solution $\beta(\beta^\nu)$ by taking a step from β^ν with a initial feasible point β^0. In this way, we build the approximation for $\Theta(\beta)$ in (25) as $\tilde{\Theta}(\beta) = \tilde{U}_2(\beta, \beta^\nu) + aB_0^2 H t_0^{comp}(\tilde{U}_1(\beta) - 1)$ and the approximated problem of (25) can be formulated as

$$\min_{\beta} \; \tilde{\Theta}(\beta) \tag{28a}$$
$$s.t. \; \beta \geq \beta_{\min}, \tag{28b}$$
$$G(\beta) \leq 0. \tag{28c}$$

Obviously, the problem in (28) is convex, continuously differentiable, and the Slater's condition is satisfied, implying that strong duality holds. Hence, the problem can be derived using the KKT conditions, and the stationary point β^* can be determined accordingly.

The first order derivative of the objective $\tilde{\Theta}(\beta)$ can be calculated as

$$\frac{\mathrm{d}\tilde{\Theta}(\beta)}{\mathrm{d}\beta} = I\beta + AR - 2q\beta^\nu + J(Ye^{\nabla g(\beta^\nu)\beta} + \kappa\beta - \kappa\beta^\nu) = (I + J\kappa)\beta + JYe^{\nabla g(\beta^\nu)\beta} + S, \tag{29}$$

Algorithm 2 Batch Sizes Control of DBFL

Initialization: Initialize $q = q^{(0)} > 0$, outer-loop iteration index $k = 0$, the stop criteria ε, and the maximum number of iterations K_{max}

repeat
  Initialize the step size $\zeta^0 \in (0, 1]$
  Set inner-loop iteration index $\nu = 0$ and start with β^0
  repeat
    Calculate $\beta^*(\beta^\nu)$ via (32)
    Set $\beta^{\nu+1} = \beta^\nu + \zeta^\nu(\beta^*(\beta^\nu) - \beta^\nu)$
    Set $\nu = \nu + 1$
  until β^ν is a stationary solution of problem (28)
  Set $\beta^* = \beta^\nu$ as the current solution of the primal problem (20).
  Update $q^{(k+1)} = \frac{P_1(\beta^*) + P_2(\beta^*)}{P_3(\beta^*)}$
  if $|\Theta(q^{(k)} - \Theta(q^{(k+1)})| \leq \varepsilon$ **then**
    return The current solution β^*.
  else
    Set $k = k + 1$
until $k = K_{max}$
return Optimal solution β^*.

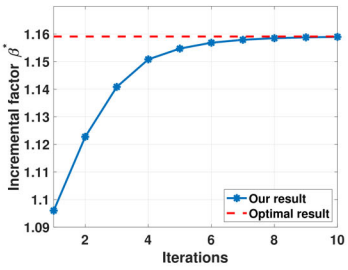


Fig. 4. The iterative convergence process of the proposed solution.

where $R = t^{tran} + cHt_0^{comp}$, $I = 2(A+\chi)R$, $J = aB_0^2Ht_0^{comp}$, $Y = e^{g(\beta^\nu)-\nabla g(\beta^\nu)\beta^\nu}\nabla g(\beta^\nu)$, $S = (AR - J\kappa\beta^\nu - 2q\beta^\nu)$.

We further calculate the first order derivative of $g(\beta)$ and obtain $\nabla g(\beta^\nu) = \frac{2\beta^\nu(A+\chi)-2\chi}{\beta^\nu(\beta^\nu-1)} - \frac{2A\ln(\beta^\nu)}{(\beta^\nu-1)^2}$. As the first order derivative of the function $G(\beta)$ has already shown in (21), we have the following equations according to the KKT conditions

$$\begin{cases} \dfrac{\partial\tilde{\Theta}(\beta)}{\partial\beta} + \mu\dfrac{\partial G(\beta)}{\partial\beta} = 0, \\ \mu G(\beta) = 0, \\ \mu \geq 0, \\ G(\beta) \leq 0, \end{cases} \tag{30}$$

where μ is the Lagrange multiplier. Then, we have

$$\begin{aligned} \beta_1 &= -\frac{(I+J\kappa)\mathrm{W}\left(\frac{JY\nabla g(\beta^\nu)\mathrm{e}^{-\frac{\nabla g(\beta^\nu)S}{I+J\kappa}}}{I+J\kappa}\right) + \nabla g(\beta^\nu)\mathrm{S}}{\nabla g(\beta^\nu)(I+J\kappa)}, \\ \beta_2 &= e^{\lambda^*}, \end{aligned} \tag{31}$$

where $\mathrm{W}(\cdot)$ represents the Lambert function, and λ^* satisfies $Ae^{\lambda^*}\lambda^* + e^{\lambda^*}\lambda^*\chi - Ze^{\lambda^*} - \lambda^*\chi + Z = 0$. It should note that λ^* can be easily acquired through the bisection method. We further define $\tilde{\beta} = \arg\min\{\Theta(\beta_1), \Theta(\beta_2)\}$, and the optimal solution β^* can be given as:

$$\begin{cases} \beta_{\min}, & \tilde{\beta} \leq \beta_{\min}, \\ \tilde{\beta}, & \text{otherwise}. \end{cases} \tag{32}$$

We summarize the procedure of the inner convex approximation as the inner-loop in Alg. 2. Alg. 2 describes how we obtain the optimal incremental factor β^* that can control the batch size in each training round to minimize the total time consumption in MEC networks by well balancing the communication and computing. Note that the main computational workload of the inner convex approximation lies in the calculation of $\beta^*(\beta^\nu)$, i.e., the solution of the problem in (28). We mitigate the workload by computing $\beta^*(\beta^\nu)$ in closed form, which allows us to obtain the unique solutions of the inner-loop problems directly without resorting to any iterative solver that provides approximate solutions only. Thanks to the desirable convergence properties from the Dinkelbach and NOVA algorithms, our batch size control algorithm in Alg. 2 can finish finding β^* after a finite number of steps with relatively low computational complexity. Fig. 4 shows one approximation error example of the proposed solution, where we employ the brute force approach to find the optimal results, which verifies the accuracy and efficiency of the proposed solution.

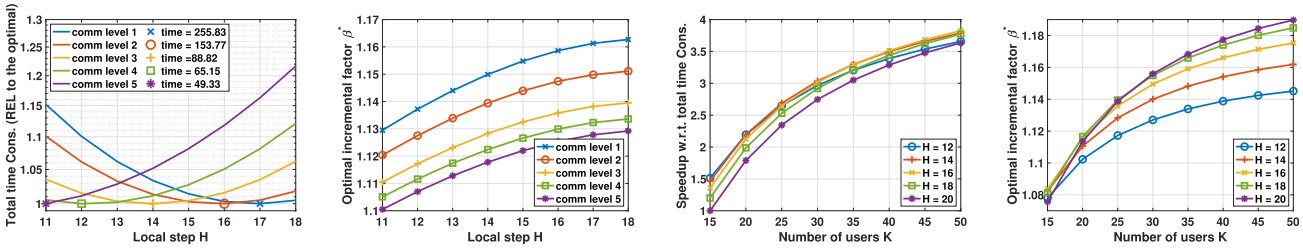
V. Performance Evaluation

In this section, we conduct extensive simulations to validate the effectiveness of our proposed DBFL algorithm and the corresponding control scheme over future mobile edge devices. To evaluate the performance of our proposed scheme over future mobile edge devices in practice, we need to find the proper model parameters to represent the computing and communication of MEC. First of all, we take some experiments on the edge device to identify the GPU computing model. We use the edge devices occupied with one NVIDIA RTX 8000 GPU to model the edge computing environment of the future mobile device in practice, where the GPU core frequency f_{core} and memory frequency f_{mem} equal to 1,400 MHz and 1,750 MHz. Through running the ResNet20 model with CIFAR-10 dataset on devices multiple times, we identify that t_0^{comp} nearly equals to 0.024, and a and c can be selected as 9.49×10^{-7} and 1 in function $d(B)$, respectively. Next, for the communication model, we consider the ultra fast wireless transmissions (e.g., 5G) to simulate the edge communication environments, where the achievable upload transmission rates R is between 5 Mbps and 125 Mbps, depending on the wireless communication conditions.

A. Impacts of Communication Conditions, Local Steps, and the Number of Participants on the DBFL Algorithm

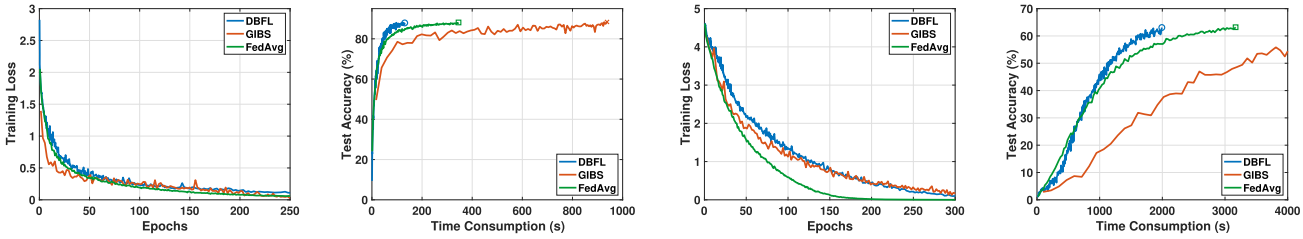
After identifying the parameters of the GPU computing model and wireless communication model, the expressions of communication round M and objective Φ still have some parameters to be estimated. We estimate these parameters with different learning settings by experiments and take $\frac{2b}{\gamma} = 250$, $C = 0.0005$, and $\varrho = 400$ as an example in Eq. (15) to identify the values of A and χ. We further take $\epsilon = 0.5$ to guarantee $\epsilon > CH^2$ for different potential H. The transmission model size N can be estimated as 1.25 MB, and the dataset size for each user equals to 5,000. The simulation results for different parameter settings of our proposed DBFL algorithm with optimal control are shown in Fig. 5.

We first consider different communication conditions and different local steps H with a fixed number of participating devices $K = 25$, and the simulation results are shown in Fig. 5(a) and Fig. 5(b). Different communication levels represent different communication conditions, where level 1 to 5 correspond to upload transmission rates $R = 5, 10, 25, 50$, and 125 Mbps, respectively. In particular, the level 1 indicates that the communication cost is dominant in total time consumption, while level 5 indicates that the computing cost is dominant. Fig. 5(a) demonstrates the total time consumption with varying local steps H under different communication conditions, where the y-axis represents the value relative to the optimal time consumption for each H. In Fig. 5(a), under the same communication condition, the local step H has an optimal value in the middle of the candidate ranges. This is because too small H will lead to inefficient



(a) Impact of comm. level on time (b) Impact of comm. level on β^* (c) Speedup over H and K (d) Impacts of H and K on β^*

Fig. 5. Numerical illustration of DBFL algorithm.



(a) Training accuracy vs. epochs (b) Test accuracy vs. time cons. (c) Training accuracy vs. epochs (d) Test accuracy vs. time cons.

Fig. 6. Simulation results on various architectures and datasets. ((a-b): ResNet20 on CIFAR-10; (c-d): VGG19-BN on CIFAR-100.)

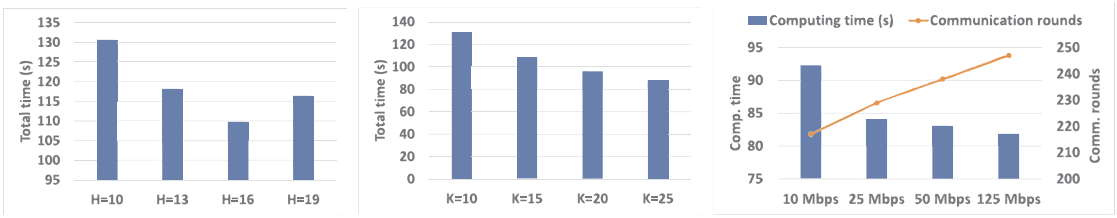
communication. In contrast, if H is too large, the overfitting problem of local computation also affects the convergence rate. Moreover, it is easy to observe that with the deterioration of communication conditions, the time consumption will increase. It's also notable that the optimal H is gradually decreasing with the rising of the communication level, as the remarked points in Fig. 5(a). The reason can be explained as the optimal solutions prefer to reduce the total time consumption by putting more communication rounds (more "talking") when the communication conditions are good, and vice versa. This reflects the trade-off between communication and computing of our proposed scheme. Fig. 5(b) further shows the relationship between the obtained optimal incremental factor β^* and different communication conditions. Here, as the communication conditions degrade, the optimal incremental factor β^* increases accordingly. This is another reflection of our algorithm regulating communication and computing. The degraded communication environment indicates that the time consumption of each transmission will increase. In this situation, the incremental factor should be increased for taking more local computing tasks (more "working") in each round to reduce the global updates, so as to minimize the total time.

Next, we investigate the impact of different numbers of participating devices K on the total time consumption, and the results are shown in Fig. 5(c) and Fig. 5(d). We simulate the upload transmission rates $R = 25$ Mbps, which indicates the communication cost and computing cost are comparable. Fig. 5(c) shows that the speedup of the total time consumption grows with the increase of the participating users under all H situations, but the growth rate gradually slows down. This results from the fact that increasing the number of participating users can help speed up the convergence, and thereby reduce the time consumption. However, when there are enough devices to capture the characteristics of the database $\bigcup_i \mathcal{D}_i$ among all users, adding extra users will only have a slightly positive impact on the training convergence. Fig. 5(d) depicts the increase of the optimal incremental factor β^* when increasing the number of participating FL users. More devices participating in the training will accelerate the convergence, so the optimal incremental factor should be increased to match the change in the convergence rate, thus further minimizing the total time consumption.

B. Convergence Analysis and Comparisons

To demonstrate the effectiveness of our proposed DBFL scheme with optimal incremental factor β^* under different learning architectures and varied datasets, we compare it with a typical fixed batch size FL scheme "FedAvg [6]," and a FL scheme with gradually increased batch size "GIBS". In GIBS, all devices communicate with the edge server after every local iteration, and the batch size exponentially increases in each iteration. We consider the scenarios with $K = 10$ participating devices and take local steps $H = 10$ for FedAvg and our DBFL scheme. The experiment results are shown in Fig. 6. All schemes have the same initial batch size $B_0 = 100$ and learning rate 0.2 to ensure comparability. Moreover, We guarantee that our proposed DBFL scheme and GIBS scheme have the same batch size increasing rate in terms of the training epochs, which is also consistent with the decay of the learning rate for the FedAvg scheme.

Figs. 6(a) and 6(b) show the learning results of the ResNet20 model on the CIFAR-10 dataset with 1 MB parameters, where we consider the training with NVIDIA RTX 8000 GPU and simulate the current 5G network with 50 Mbps upload transmission rate. Fig. 6(a) demonstrates that the training loss curves of all schemes are nearly identical in terms of the number of gradient calculation operations, i.e., data epochs.



(a) Impact of H. (b) Impact of K. (c) Impact of R.

Fig. 7. Simulation results on different values of local steps H, the number of participants K, and transmission rates R.

Fig. 6(b) demonstrates the total time consumption to the same target accuracy (e.g., 88%) for the corresponding schemes, and we find that the overall time consumption of our proposed DBFL scheme with optimal β^* can be reduced to one-third of that of the FedAvg. In particular, our proposed scheme requires fewer communication rounds than FedAvg, thus reducing the communication time. At the same time, our scheme consumes less computing time due to the large batch training efficiency. Moreover, the optimal β^* can trade-off the relationship between communication and computing by adapting to the current mobile edge environments to further reduce the overall time consumption. For the GIBS scheme, although the GIBS scheme saves some computing time than the FedAvg, its huge communication overhead makes its convergence speed very slow. We also conduct the experiments on a relatively large model, i.e., the VGG19 model with batch normalization, on the CIFAR-100 dataset, which contains 549 MB parameters. Here, we simulate the 5G network with 1 Gbps upload transmission rate. The results are shown in Figs. 6(c) and 6(d), which reach similar conclusions as those of the "ResNet" one. It is noteworthy that the performance of the FedAvg scheme is slightly better than the others. The reason is that the relatively small batch size of the FedAvg on the initial learning stage speeds up the convergence in this model set.

From Fig. 6(a) and Fig. 6(c), we also find that the proposed DBFL approach has a similar convergence performance as the typical FL approach, e.g., FedAvg. In other words, both the DBFL and the FedAvg can approach the same target accuracy with similar training epochs. For this reason, there are some potential benefits of our DBFL algorithm when compared with other time-efficient FL algorithms integrating model compression techniques (e.g., the quantization, sparsification, etc.). On the one hand, unlike the model compression based approaches that usually result in a considerable sacrifice of training accuracy, the DBFL algorithm ensures all the gradient information is aggregated in every global round without distortion, and thereby hardly impairs the learning outcome. On the other hand, our DBFL method can be considered an add-on module integrated with any model compression method for further FL time reduction.

We further investigate the impacts of the number of local steps H, the number of participants K, and transmission rate R on the DBFL approach. We still utilize the ResNet20 model on the CIFAR-10 dataset with varying numbers of H, K, and R, where simulation settings are the same as the configurations of Fig. 6(a) and Fig. 6(b). Fig. 7(a) shows that the number of local steps H has a modest effect on the time consumption, where either too large or too small H will degrade the performance. Fig. 7(b) exhibits the performance under different numbers of participating devices K, from which we observe that a larger K leads to less time consumption. However, as K increases, the improvement of time-saving gradually gets slight. Moreover, Fig. 7(c) illustrates that when the communication conditions get worse, the proposed DBFL approach will enable more local computing to reduce the required communication rounds, thus minimizing the overall time consumption. These observations are also consistent with our analysis taken in Sec. V-A.

VI. Related Works

With the prevalence of mobile edge devices and the upgrading of hardware performance, FL over mobile edge devices, as a distributed machine learning framework integrated with MEC, has enormous potential and attracts a lot of attention from researchers and scholars. Tran *et al.* [16], [17] developed an optimization approach to explore the trade-offs of FL in mobile edge networks, e.g., between the computing and the communication, between the time and the energy. By considering the time-varying edge environments during training, Wang *et al.* [18] and Zhan *et al.* [19] proposed the dynamical control schemes to determine the optimal model aggregation frequency and the optimal CPU-cycle frequency under a given resource budget, respectively. Furthermore, Chen *et al.* [20], [21] established a joint user selection and resource block allocation scheme to minimize the convergence time and optimize the FL performance. Zeng *et al.* [34] implemented FL algorithms within a UAV swarm and proposed a joint power allocation and scheduling design to optimize the convergence rate. These works indeed save the FL's system cost in mobile edge networks, but they don't consider the intrinsic mechanism of the learning to accelerate the training process. More specifically, they allocate the existing resources appropriately to adapt to the FL approach, and do not essentially consider this time efficiency issue from the algorithmic perspective.

Nowadays, there is a growing interest in the learning structure with dynamic batch sizes, which can accelerate the training process by capturing the inherent nature of the learning process. It was a common practice to decrease the learning rate during training, but Smith and Devarakonda *et al.* [7], [8] experimentally found that in centralized training, one can obtain the same learning curve by instead gradually increasing

the batch size. Meanwhile, the latter required a decreased amount of parameter updates and could enjoy the time-efficient advantages of large batch training, and a similar rigorous theoretical analysis was shown in some pioneer research works [35], [36]. Along with the increasing complexity of the machine learning models and the growing demand for the needed data, some works focused on exploring the dynamic batch size structures in a distributed training manner from both the theoretical [9] and experimental [10] point of view, where the communication cost can be significantly reduced. However, there are two main differences between our work and the above work. On the one hand, these distributed frameworks did not integrate local SGD, a communication-efficient training method, while we integrated the local SGD to improve the training efficiency. On the other hand, they only consider the FL problem from the learning algorithm perspective itself and widely ignore the coupling between the resource consumption in the training process and the environment of the mobile edge devices. Besides, finding the proper batch size increasing policy to make the trade-off between computing and communication to minimize the total time consumption is still unexplored.

VII. Conclusion

In this paper, we have focused on minimizing the overall time consumption of FL over future mobile edge devices in terms of both communication and computing time. We first developed a time efficient FL algorithm, named DBFL, with the convergence guarantee, where the batch size exponentially increases during FL training. In particular, gradually increasing the batch sizes in DBFL can fully adapt to the GPUs' parallelism property and reduce the required communication rounds for global updates. To minimize the total time consumption of deploying DBFL in MEC networks, we have further designed a batch size control scheme, which helps balance the time consumption of wireless communication and that of local computing in practice. Extensive simulations have shown the impacts of different communication conditions, local steps, and the number of participants on the proposed DBFL algorithm. Furthermore, the results have validated the effectiveness of our proposed control scheme, which saves nearly $3\times$ time than the state-of-the-art approaches.

Appendix

A. Proof of Theorem 1

We set the batch size in each communication round τ, $\tau \in \{0, 1, \ldots, T/H-1\}$ as $\lfloor \beta^{\tau} B_0 \rfloor$ and assume T/H is an integer. Therefore, in Alg. 1, we must have $H\sum_{\tau=0}^{T/H-1} \lfloor \beta^{\tau} B_0 \rfloor \geq E$, which future implies $H\sum_{\tau=0}^{T/H-1} \beta^{\tau} B_0 \geq E$. Through the summation formula of the geometric series, We can observe that $\beta^{T/H} \geq \frac{E(\beta-1)}{HB_0} + 1$, which yields,

$$\begin{aligned} T &\geq H log_{\beta}\left(\frac{E(\beta-1)}{HB_0}+1\right) \\ &\overset{(a)}{\geq} log_{\beta}\left(\frac{E(\beta-1)}{B_0}+1\right) \\ &\geq log_{\beta}\left(\frac{E(\beta-1)}{B_0}\right), \end{aligned} \tag{33}$$

where (a) follows by $(1+x)^r \geq 1+rx$ for $x \geq -1, r \in \mathbb{R}\backslash(0,1)$.

Under Assumption 1, we have

$$\begin{aligned} \mathbb{E}\left[f\left(\overline{\mathbf{w}}^{t}\right)\right] \leq \mathbb{E}\left[f\left(\overline{\mathbf{w}}^{t-1}\right)\right] &+ \mathbb{E}\left[\left\langle \nabla f\left(\overline{\mathbf{w}}^{t-1}\right), \overline{\mathbf{w}}^{t} - \overline{\mathbf{w}}^{t-1}\right\rangle\right] \\ &+ \frac{L}{2}\mathbb{E}\left[\left\|\overline{\mathbf{w}}^{t} - \overline{\mathbf{w}}^{t-1}\right\|^2\right]. \end{aligned} \tag{34}$$

Through the convergence analysis provided for *Theorem 1* in [27], we can bound the terms in the above inequality as

$$\begin{aligned} &\mathbb{E}\left[\left\|\nabla f\left(\overline{\mathbf{w}}^{t-1}\right)\right\|^2\right] \\ &\leq \frac{2}{\gamma}\left(\mathbb{E}\left[f\left(\overline{\mathbf{w}}^{t-1}\right)\right] - \mathbb{E}\left[f\left(\overline{\mathbf{w}}^{t}\right)\right]\right) + 4\gamma^2 H^2 \delta^2 L^2 + \frac{L\gamma\sigma^2}{KB_t} \\ &\leq \frac{2}{\gamma}\left(\mathbb{E}\left[f\left(\overline{\mathbf{w}}^{t-1}\right)\right] - \mathbb{E}\left[f\left(\overline{\mathbf{w}}^{t}\right)\right]\right) + CH^2 + \frac{L\gamma\sigma^2}{K\left\lfloor B_0\beta^{\lfloor t/H \rfloor}\right\rfloor} \\ &\overset{(a)}{\leq} \frac{2}{\gamma}\left(\mathbb{E}\left[f\left(\overline{\mathbf{w}}^{t-1}\right)\right] - \mathbb{E}\left[f\left(\overline{\mathbf{w}}^{t}\right)\right]\right) + CH^2 + \frac{2L\gamma\sigma^2}{KB_0\beta^{\lfloor t/H \rfloor}}, \end{aligned} \tag{35}$$

where we set $C = 4\gamma^2\delta^2L^2$, and (a) follows by $\lfloor x \rfloor > \frac{1}{2}x$ if $x > 2$. After summing the above inequality (10) over $t \in \{1, 2, \ldots, T\}$ iterations and dividing it by T, we have

$$\begin{aligned} &\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\left\|\nabla f\left(\overline{\mathbf{w}}^{t-1}\right)\right\|^2\right] \\ &\quad\leq \frac{2}{\gamma T}\left(f\left(\overline{\mathbf{w}}^{0}\right) - f^*\right) + CH^2 \\ &\qquad + \frac{2L\gamma\sigma^2}{TKB_0}\left(H\sum_{\tau=1}^{T/H}\frac{1}{\beta^{\tau-1}} - 1 + \frac{1}{\beta^{T/H}}\right) \\ &\quad\leq \frac{2}{\gamma T}\left(f\left(\overline{\mathbf{w}}^{0}\right) - f^*\right) + CH^2 + \frac{2L\gamma\sigma^2 H}{TKB_0}\sum_{\tau=1}^{T/H}\frac{1}{\beta^{\tau-1}} \\ &\quad\overset{(a)}{\leq} \frac{2}{\gamma T}\left(f\left(\overline{\mathbf{w}}^{0}\right) - f^*\right) + CH^2 + \frac{2L\gamma\sigma^2 H}{TKB_0}\frac{\beta}{\beta-1}, \end{aligned} \tag{36}$$

where (a) follows by simplifying the sum of the proportional sequence and $\frac{1}{\beta} < 1$. Next, Substituting (33) into (36) yields

$$\begin{aligned} \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\left\|\nabla f\left(\overline{\mathbf{w}}^{t-1}\right)\right\|^2\right] &\leq \frac{2\left(f\left(\overline{\mathbf{w}}^{0}\right) - f^*\right)}{\gamma \log_{\beta}\frac{E(\beta-1)}{B_0}} \\ &\quad + 4\gamma^2H^2\delta^2L^2 + \frac{2L\gamma\sigma^2H}{\log_{\beta}\frac{E(\beta-1)}{B_0}KB_0}\cdot\frac{\beta}{\beta-1}. \end{aligned} \tag{37}$$

B. Proof of Corollary 1

Similar to the settings in a fixed batch size scenario [27], if we choose $\gamma = \frac{\sqrt{K}}{L\sqrt{\log_{\beta}\frac{E(\beta-1)}{B_0}}}$, $H \leq \left(\log_{\beta}\frac{E(\beta-1)}{B_0}\right)^{\frac{1}{4}} K^{-\frac{3}{4}}$, $K < (\log E)^{\frac{5}{3}}$, and substitute them into the (37), we have

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\left\|\nabla f\left(\overline{\mathbf{w}}^{t-1}\right)\right\|^2\right]$$

$$\leq \frac{2L}{\sqrt{K\log_\beta \frac{E(\beta-1)}{B_0}}}\left(f\left(\overline{\mathbf{w}}^0\right)-f^*\right) + \frac{4\delta^2}{\sqrt{K\log_\beta \frac{E(\beta-1)}{B_0}}} + \frac{\sigma^2}{K^{\frac{5}{4}}(\log_\beta \frac{E(\beta-1)}{B_0})^{\frac{5}{4}}B_0}\cdot\frac{\beta}{\beta-1}$$
$$= O\left(\frac{1}{\sqrt{K\log E}}\right)+O\left(\frac{1}{\sqrt{K\log E}}\right)+O\left(\frac{1}{(K\log E)^{\frac{5}{4}}}\right)$$
$$\overset{(a)}{=} O\left(\frac{1}{\sqrt{K\log E}}\right). \tag{38}$$

Since the term $\frac{1}{(K\log E)^{\frac{5}{4}}}$ is dominated by the term $\frac{1}{\sqrt{K\log E}}$, $O\left(\frac{1}{(K\log E)^{\frac{5}{4}}}\right)$ decays faster than $O\left(\frac{1}{\sqrt{K\log E}}\right)$. In this way, (a) will hold and Alg. 1 has the convergence rate of order $O\left(\frac{1}{\sqrt{K\log E}}\right)$.

REFERENCES

[1] Qualcomm. *5G+AI: The Ingredients Fueling Tomorrow's Tech Innovations*. Accessed: Jan. 2021. [Online]. Available: https://www.qualcomm.com/news/onq/2020/02/04/5gai-ingredients-fueling-tomorrows-tech-innovations

[2] D. Shi, H. Gao, L. Wang, M. Pan, Z. Han, and H. V. Poor, "Mean field game guided deep reinforcement learning for task placement in cooperative multiaccess edge computing," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9330–9340, Oct. 2020.

[3] A. Hard *et al.*, "Federated learning for mobile keyboard prediction," 2018, *arXiv:1811.03604*.

[4] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, "Federated learning of predictive models from federated electronic health records," *Int. J. Med. Informat.*, vol. 112, pp. 59–67, Apr. 2018.

[5] S. T. Apple. *Hey Siri: An on-Device DNN-Powered Voice Trigger for Apple's Personal Assistant*. Accessed: Jun. 2020. [Online]. Available: https://machinelearning.apple.com/research/hey-siri

[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Statist. (AISTATS)*, Fort Lauderdale, FL, USA, Apr. 2017, pp. 1273–1282.

[7] S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le, "Don't decay the learning rate, increase the batch size," in *Proc. Int. Conf. Learn. Represent.*, Vancouver, BC, Canada, Apr. 2018, pp. 1–11.

[8] A. Devarakonda, M. Naumov, and M. Garland, "AdaBatch: Adaptive batch sizes for training deep neural networks," 2017, *arXiv:1712.02029*.

[9] H. Yu and R. Jin, "On the computation and communication complexity of parallel SGD with dynamic batch sizes for stochastic non-convex optimization," in *Proc. Int. Conf. Artif. Intell. Statist. (AISTATS)*, Long Beach, CA, USA, Jun. 2019, pp. 7174–7183.

[10] H. Lin *et al.*, "Dynamic mini-batch SGD for elastic distributed training: Learning in the Limbo of resources," 2019, *arXiv:1904.12043*.

[11] M. Shafi *et al.*, "5G: A tutorial overview of standards, trials, challenges, deployment, and practice," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 6, pp. 1201–1221, Jun. 2017.

[12] W. Saad, M. Bennis, and M. Chen, "A vision of 6G wireless systems: Applications, trends, technologies, and open research problems," *IEEE Netw.*, vol. 34, no. 3, pp. 134–142, May 2020.

[13] P. Goyal *et al.*, "Accurate, large minibatch SGD: Training ImageNet in 1 hour," 2018, *arXiv:1706.02677*.

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

[15] D. Shi, L. Li, R. Chen, P. Prakash, M. Pan, and Y. Fang, "Towards energy efficient federated learning over 5G+ mobile devices," *IEEE Wireless Commun.*, early access, May 9, 2022, doi: 10.1109/MWC.003.2100028.

[16] N. H. Tran, W. Bao, A. Zomaya, M. N. H. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Paris, France, Apr. 2019.

[17] C. T. Dinh *et al.*, "Federated learning over wireless networks: Convergence analysis and resource allocation," 2019, *arXiv:1910.13067*.

[18] S. Wang *et al.*, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 1205–1221, Jun. 2019.

[19] Y. Zhan, P. Li, and S. Guo, "Experience-driven computational resource allocation of federated learning by deep reinforcement learning," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, New Orleans, LA, USA, May 2020, pp. 234–243.

[20] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Convergence time optimization for federated learning over wireless networks," 2020, *arXiv:2001.07845*.

[21] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Convergence time minimization of federated learning over wireless networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Dublin, Ireland, Jun. 2020, pp. 1–6.

[22] P. Prakash *et al.*, "IoT device friendly and communication efficient federated learning via joint model pruning and quantization," *IEEE Internet Things J.*, early access, Jan. 3, 2022, doi: 10.1109/JIOT.2022.3145865.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778.

[24] N. Zhang, Y. Chen, and J. Wang, "Proc. Image parallel processing based on GPU," in *Proc. 2nd Int. Conf. Adv. Comput. Control*, vol. 3, Mar. 2010, pp. 367–370.

[25] T. Lin, S. U. Stich, K. K. Patel, and M. Jaggi, "Don't use large mini-batches, use local SGD," in *Proc. Int. Conf. Learn. Represent.*, Addis Ababa, Ethiopia, Apr. 2020, pp. 1–40.

[26] S. L. Smith and Q. V. Le, "A Bayesian perspective on generalization and stochastic gradient descent," in *Proc. Int. Conf. Learn. Represent.*, Vancouver, BC, Canada, Apr. 2018, pp. 1–13.

[27] H. Yu, S. Yang, and S. Zhu, "Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning," in *Proc. AAAI Conf. Artif. Intell.*, Honolulu, HI, USA, Jan. 2019, pp. 5693–5700.

[28] W. Nam, D. Bai, J. Lee, and I. Kang, "Advanced interference management for 5G cellular networks," *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 52–60, May 2014.

[29] V. V. Veeravalli and A. El Gamal, *Interference Management in Wireless Networks: Fundamental Bounds and the Role of Cooperation*. Cambridge, U.K.: Cambridge Univ. Press, 2018.

[30] X. Mei, X. Chu, H. Liu, Y.-W. Leung, and Z. Li, "Energy efficient real-time task scheduling on CPU-GPU hybrid clusters," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Atlanta, GA, USA, May 2017, pp. 1–9.

[31] Y. Abe, H. Sasaki, S. Kato, K. Inoue, M. Edahiro, and M. Peres, "Power and performance characterization and modeling of GPU-accelerated systems," in *Proc. IEEE 28th Int. Parallel Distrib. Process. Symp.*, Phoenix, AZ, USA, Aug. 2014, pp. 113–122.

[32] S. Schaible, "Fractional programming. II, on Dinkelbach's algorithm," *Manage. Sci.*, vol. 22, no. 8, pp. 868–873, Apr. 1976.

[33] G. Scutari, F. Facchinei, and L. Lampariello, "Parallel and distributed methods for constrained nonconvex optimization—Part I: Theory," *IEEE Trans. Signal Process.*, vol. 65, no. 8, pp. 1929–1944, Dec. 2016.

[34] T. Zeng, O. Semiari, M. Mozaffari, M. Chen, W. Saad, and M. Bennis, "Federated learning in the sky: Joint power allocation and scheduling with UAV swarms," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Dublin, Ireland, Jun. 2020, pp. 1–6.

[35] S. De, A. Yadav, D. Jacobs, and T. Goldstein, "Automated inference with adaptive batches," in *Proc. Int. Conf. Artif. Intell. Statist. (AISTATS)*, Fort Lauderdale, FL, USA, Apr. 2017, pp. 1504–1513.

[36] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Rev.*, vol. 60, no. 2, pp. 223–311, Feb. 2018.

Dian Shi (Member, IEEE) received the B.S. degree in electronic information engineering from the University of Electronic Science and Technology of China in 2017 and the Ph.D. degree in electrical engineering from the University of Houston in 2022. His research interests include deep reinforcement learning, federated learning, and edge intelligence.

Liang Li (Member, IEEE) received the Ph.D. degree from the School of Telecommunications Engineering, Xidian University, China, in 2021. She was a Visiting Ph.D. Student with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX, USA, from 2018 to 2020. She is currently a Post-Doctoral Faculty Member with the School of Computer Science (National Pilot Software Engineering School), Beijing University of Posts and Telecommunications. Her research interests include edge computing, federated learning, data-driven robust optimization, and differential privacy.

Maoqiang Wu (Member, IEEE) received the Ph.D. degree from the Guangdong University of Technology, China, in 2021. He is currently a Post-Doctoral Faculty Member with the Guangdong University of Technology. His research interests include security, privacy, and resource management in edge intelligence and wireless networks.

Minglei Shu (Member, IEEE) received the B.S. degree in automation, the M.S. degree in power electronics and power transmission, and the Ph.D. degree in communication and information systems from Shandong University in 2003, 2006, and 2016, respectively. He is currently working with the Shandong Artificial Intelligence Institute, Qilu University of Technology (ShandongAcademy of Sciences). His research interests include computer vision, the Internet of Things, and wireless sensor networks.

Rong Yu (Senior Member, IEEE) received the B.S. degree in communication engineering from the Beijing University of Posts and Telecommunications, China, in 2002, and the Ph.D. degree in electronic engineering from Tsinghua University, China, in 2007. After that, he was with the School of Electronic and Information Engineering, South China University of Technology. In 2010, he joined the School of Automation, Guangdong University of Technology, where he is currently a Professor. He is the author or coauthor of over 120 international journals and conference papers. He is the co-inventor of over 80 patents in China. His research interests include wireless networking and mobile computing, such as edge computing, deep learning, blockchain, digital twin, connected vehicles, smart grid, and the Internet of Things. He was a member of the Home Networking Standard Committee, China, where he led the standardization work of three standards.

Miao Pan (Senior Member, IEEE) received the B.Sc. degree in electrical engineering from the Dalian University of Technology, China, in 2004, the M.A.Sc. degree in electrical and computer engineering from the Beijing University of Posts and Telecommunications, China, in 2007, and the Ph.D. degree in electrical and computer engineering from the University of Florida in 2012. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of Houston. His research interests include wireless/AI for AI/wireless, deep learning privacy, cybersecurity, and underwater communications and networking. He is a member of AAAI and ACM. He was a recipient of the NSF CAREER Award in 2014. His work won IEEE Technical Committee on Green Communications and Computing (TCGCC) Best Conference Paper Awards 2019 and the Best Paper Awards in ICC 2019, VTC 2018, GLOBECOM 2017, and GLOBECOM 2015. He has also been serving as a Technical Organizing Committee for several conferences, such as a TPC Co-Chair for Mobiquitous 2019 and ACM WUWNet 2019. He is an Editor of the IEEE OPEN JOURNAL OF VEHICULAR TECHNOLOGY, an Associate Editor of *ACM Computing Surveys*, and an Associate Editor of the IEEE INTERNET OF THINGS JOURNAL (Area 5: Artificial Intelligence for IoT), and used to be an Associate Editor of IEEE INTERNET OF THINGS JOURNAL (Area 4: Services, Applications, and Other Topics for IoT) from 2015 to 2018.

Zhu Han (Fellow, IEEE) received the B.S. degree in electronic engineering from Tsinghua University in 1997, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Maryland, College Park, MD, USA, in 1999 and 2003, respectively.

From 2000 to 2002, he was a Research and Development Engineer of JDSU, Germantown, MD, USA. From 2003 to 2006, he was a Research Associate at the University of Maryland. From 2006 to 2008, he was an Assistant Professor at Boise State University, Boise, ID, USA. He is currently a John and Rebecca Moores Professor with the Electrical and Computer Engineering Department as well as with the Computer Science Department, University of Houston, Houston, TX, USA. His research interests include wireless resource allocation and management, wireless communications and networking, game theory, big data analysis, security, and smart grid. He is an AAAS Fellow since 2019 and an ACM Distinguished Member since 2019. He received an NSF Career Award in 2010, the Fred W. Ellersick Prize of the IEEE Communication Society in 2011, the EURASIP Best Paper Award for the *Journal on Advances in Signal Processing* in 2015, the IEEE Leonard G. Abraham Prize in the field of Communications Systems (Best Paper Award in IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS) in 2016, and several best paper awards in IEEE conferences. He is also the Winner of the 2021 IEEE Kiyo Tomiyasu Award, for outstanding early to mid-career contributions to technologies holding the promise of innovative applications, with the following citation: for contributions to game theory and distributed management of autonomous communication networks. He was an IEEE Communications Society Distinguished Lecturer from 2015 to 2018. He is a 1% Highly Cited Researcher since 2017 according to Web of Science.