Edge Resource Autoscaling for Hierarchical Federated Learning Over Public Edge Platforms

Mingliao Zhao Kongyange Zhao Zhi Zhou Xu Chen
School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, 510006, China

***Abstract*—Federated learning promises to empower ubiquitous end devices to collaboratively learn a shared model in a privacy-preserving manner. To reduce the enormous and expensive wide-area-network (WAN) traffic incurred by the traditional two-tiered cloud-device federated learning, hierarchical federated learning over cloud-edge-device has been proposed recently. With hierarchical federated learning, edge servers are leveraged as intermediaries to perform local model aggregations to reduce the model updates aggregated by the centralized cloud. Considering the emerging public edge platforms such as Aliyun Edge Node Service that rent edge servers to users in an on-demand manner, we present AutoEdge, an edge server autoscaling framework for hierarchical federated learning. The goal of AutoEdge is to autoscale edge servers against dynamical device participants in a cost-efficient manner. Achieving this goal is challenging since the underlying long-term optimization problem is NP-hard involves the future system information. To attack these challenges, AutoEdge first applies regularization technique to decompose the long-term problem into a set of solvable fractional subproblems. Then, adopting a randomized dependent rounding scheme, AutoEdge further rounds the fractional solutions to a near-optimal and feasible integral solution. AutoEdge achieves outstanding performance guarantee, as verified by both rigorous theoretical analysis and extensive trace-driven simulations.**

I. Introduction and Related Work

Federated learning, with its promise to collaboratively train a shared model by ubiquitous end devices in a decentralized manner, is emerging as a new AI paradigm and ascending to the spotlight [1]. With federated learning, each end device first learns a local model based on its own data, and then the cloud aggregates all the local model updates to generate a global model. Since it only ships the local model updates rather than the raw data samples — which contain sensitive information and have a larger size — to the cloud, federated learning improves both privacy preservation and communication efficiency for the AI model training. Due to these appealing features, federated learning has witnessed pilot adoptions in a wide spectrum of applications, ranging from healthcare and finance to online-shopping [2].

While recognizing the superiority of federated learning in protecting the data privacy, we should also note that the native two-tiered cloud-edge federated learning still incurs enormous WAN traffic which is costly in practice [3]. Specifically, for the WAN that exchanges the model updates between the cloud and the end devices, its bandwidth is typically scarce and expensive. For example, the leading public cloud platform Amazon AWS charges per GB WAN data transfer. Clearly, as the size of DNN models (the size of state-of-the-art transform models can be as large as GBs [4]) continues to grow and the scale of device participant constantly expands (the scale of federated learning system for Taobao has reached one billion mobile clients [5]), the monetary cost incurred by the WAN traffic will surge wildly [6].

To curb the escalating usage of the expensive WAN bandwidth, various communication-efficient federated learning schemes have been proposed. In particular, at the model side, gradient compression, sparsification, and quantization approaches have been proposed to reduce the size of each model update exchanged between the device and the cloud [7], [8], [9]. Additionally, methods such as participant selection, aggregation frequency control, and local update drop out can be also applied to reduce the total number of model updates exchanged [10], [11]. At the architecture side, hierarchical federated learning over the three-tiered hierarchy of cloud-edge-device has also been proposed recently [12]. The basic idea of hierarchical federated learning is to leverage edge servers as intermediaries to perform local model aggregations within the local-area-network (LAN), and thus to reduce the number of model updates aggregated by the cloud. Note that compared to those model-oriented approaches, hierarchical federated learning has better general availability to various applications since it is model-independent.

Except for the general availability, the proliferating public edge platforms such as Aliyun Edge Node Service (ENS) and AWS CloudFront Edge Locations will also provide widespread and easy access to edge computing capabilities. For example, Aliyun Edge Node Service has deployed more than 2,800 edge sites across China, promising to maintain a proximity less than 10 Kilometers for any accessed end device in the near future. Besides, Amazon CloudFront has also deployed hundreds of edge sites at major cities around the world. With these emerging public edge platforms, developers can easily recent edge servers to deploy these edge-native applications in a flexible on-demand and pay-as-you-go manner, just like renting the traditional cloud virtual machines (VM). Given such performance, cost and flexibility

*This work was supported in part by the National Science Foundation of China under Grant 62172454, and the Guangdong Basic and Applied Basic Research Foundation under Grant 2021A1515011912. The corresponding author is Zhi Zhou.

merits of the public edge platforms, we envision that in the near future, public edge platform-based hierarchical federating learning across the cloud-edge-device will become the norm rather than the rarity.

However, running hierarchical federated learning on top of the public edge platform is non-trivial. This is because that the operators typically throttle the I/O bandwidth of each rented edge server, as verified by the recent empirical measurement studies [13]. This limitation of the I/O bandwidth implies that the edge server should by dynamically autoscaled to cover the bandwidth requirement of the local model updates uploaded by the edge devices. Moreover, since the mobile devices may dynamically move between the coverage of different edge sites spanning across geo-distributed locations, the autoscaling of edge servers at various edge sites should performed in an online manner. To this end, we present AutoEdge, an edge server autoscaling framework for hierarchical federated learning on top of public edge platforms. The goal of AutoEdge is to adaptively autoscale edge servers and dispatch local model updates to multiple edge sites in a cost-efficient manner, against dynamical device participants.

Notably, autoscaling edge servers in a cost-efficient manner is rather difficult. Specifically, by formulating the underlying long-term cost minimization problem as a mixed-integer linear programming (MILP) across time slots, we observe that this time-coupling problem is NP-hard and involves future uncertain information such as the amount of dynamic model parameters. To attack these challenges, AutoEdge carefully fuses the power of a regularization method for online algorithm design and a dependent rounding technique for approximation algorithm design. AutoEdge first applies the regularization technique to temporally relax and decompose the time-coupling problem into a series of one-shot fractional subproblems, which can be solved in an online manner, i.e., do not require any future information as a priori. Then, taking the dependent rounding technique, AutoEdge further rounds the regularized and fractional solution to a feasible yet near-optimal integeral solution of the original problem. With both rigorous theoretical analysis and extensive simulations based on realistic dataset, we demonstrate the efficacy of AutoEdge.

II. System Model and Problem Formulation for Hierarchical Federated Learning

In this part, we present the system model and problem formulation for the hierarchical federated learning system. The notations used in this section are summarized in Table I.

TABLE I
Main Notations

Notation	Definition
β_j	Unit switching cost of an edge server at location j
P_j	The operational cost of an edge server at location j
S	The data size of each model update
W_{ij}	WAN bandwidth cost from edge location i to edge location j
$G(\gamma, \theta, \epsilon)$	The total number of edge iterations when the local accuracy of the model reaches ϵ, the edge accuracy reaches θ and the global accuracy reaches γ
Z_j	Whether edge location j is selected to deploy the federated learning model or not
B_j	The maximum bandwidth capacity of an edge server at location j
$A_i(t)$	The total amount of device participants at location i at time slot t
Y_j	Maximum number of edge servers that can be rented at location j
$x_{ij}(t)$	The number of local model updates dispatched from edge location i to edge location j at time slot t
$y_j(t)$	The number of edge servers rented at edge location j at time slot t

A. Overview of the Hierarchical Federated Learning System

We consider a hierarchical federated learning system running across the three-tiered cloud-edge-device. For the top-tier of the centralized cloud, it consists of a powerful cloud server instance or a tiny cloud server cluster built on top of public cloud platforms such Amazon EC2 and Aliyun Elastic Compute Service (ECS) [14]. For the middle-tier of the geo-distributed edges, it consists of multiple geo-distributed edge nodes built on top of public edge platforms such as Amazon CloudFront and Aliyun Edge Node Service (ENS) [15]. Each edge node represents a small server cluster containing one or more edge servers in an edge site that covers a specific geographical area (e.g., a small town). For the bottom-tier of the end devices, it consists of widespread and ubiquitous mobile and IoT devices at geographical areas. These devices continuously generate fresh data samples feeding to the federating learning algorithm.

When running hierarchical federated learning across the three-tiered hierarchy of cloud-edge-device, the centralized cloud, geo-distributed edge nodes and ubiquitous end devices collaboratively train the model in a hybrid manner, with on-device model training, local edge aggregation and global cloud aggregation. At each learning iteration, the learning process can be divided into the following three steps:

- On-device model training: each device participants first locally trains the machine learning model based on its private data samples, and the output of this locally training process, i.e., the model weights or model parameters, is referred to as local model update.
- Local edge aggregation: each edge node first pulls the local model updates from its associated device participants, and then computes the local edge model parameters by performing the model-averaging operation. Ideally, each edge node only pulls model updates from nearly devices via the LAN, but in practice, an edge node can also pull model updates from remote

devices via the WAN.

- Global cloud aggregation: the cloud pulls the locally aggregated model parameters from all the edge nodes, and then computes the global cloud model parameters by performing the model-averaging operation.

To model the geo-distribution of the edge locations as well as the ubiquitous end devices, we use $\mathcal{N} = \{1, 2, \cdots, N\}$ to denote the set of geo-distributed edge locations. In line with recent work on dynamical resource allocation for federated learning [16], [17], we adopt a discrete time-slotted model to fully characterize and leverage the system dynamics (e.g., time-varying device participants at each edge location), each time slot $t = (0, 1, 2, \cdots)$ matches the time granularity at which the number of device participants changes at the edge locations. At each time slot t, the number of device participants at each edge location $i \in \mathcal{N}$ is denoted by $A_i(t)$.

B. Control Decisions

Due to the mobility as well as the dynamical ON/OFF of the devices participants, the number of the devices participants at each location typically fluctuates over time. As a result of this dynamics on the number of the devices participants, the amount of model updates aggregated at each edge node also varies over time. While the computation resource requirement of model-averaging operation is very slight and can be readily handled by a single edge server, the bandwidth consumption of the model aggregating can greatly overwhelm the bandwidth capacity of a single edge server. This is because that the operator of the public edge platforms typically throttles the network bandwidth of each rented edge server, due to the concern of the network congestion control. This phenomenon is very common for both public edge and cloud platforms in practice, as verified by the recent empirical measurement studies [18]. This limitation of the network bandwidth strongly suggests that with time-varying device participant, the edge servers rented at each edge locations should be also dynamically autoscaled to cover the time-varying network bandwidth requirement. In this paper, we use $y_j(t)$ to denote the number of edge servers rented at each edge location $j \in \mathcal{N}$ and time slot t. For each public edge platform at location $j \in \mathcal{N}$, we assume that the operator poses a limitation Y_j on the maximum number of rented server for each user. Then, we have the following constraint of the control decision of resource autoscaling:

$$y_j(t) \in \{0, 1, 2, \cdots, Y_j\}, \forall j \in \mathcal{N}.$$

Ideally, we want to deploy an enough amount of edge servers at each location, and thus to ensure that the local edge aggregation only consumes the cheap LAN bandwidth when collecting local model updates from the device participant. Unfortunately however, deploying servers at all the edge locations would greatly boost the resource cost, as well as the system management complexity due to the large scale of the edge locations. Therefore, in practice, an application developer would only carefully select a small set of edge locations to deploy the federated learning model. In this paper, we assume that the set of selected edge locations that are deployed with the federated learning model is given beforehand, and we use a constant indicator $Z_j \in \{0, 1\}$ to denote whether edge location i is selected to deploy the federated learning model ($Z_j = 1$) or not ($Z_j = 0$). For locations that are not deployed with the federated learning model, the local model updates of device participant within these locations should be dispatched to edge servers of nearly locations via the WAN. Moreover, even if an edge location is deployed with edge servers and the federated learning model, we can also the dispatch the local model updates of device participant within this location to nearly locations, and thus to potentially reduce the resource cost considering the spatial price heterogeneity of edge servers. In this paper, we use $x_{ij}(t)$ (out of $A_i(t)$) to denote the number of local model updates dispatched from edge location $i \in \mathcal{N}$ to edge location $j \in \mathcal{N}$. Note that given the potentially huge amount of device participant $A_i(t)$, the decision variable $x_{ij}(t)$ can be relaxed to a continuous one and satisfies:

$$\sum_{j \in \mathcal{N}} x_{ij}(t) = A_i(t), \forall i \in \mathcal{N}.$$

For each model update, we assume it have a data size of S. For an edge server rented at edge location j, we use B_j to denote its bandwidth capacity throttled by the operator. Then, we further have the follow constraint on the bandwidth conservation:

$$\sum_{i \in \mathcal{N}} S x_{ij}(t) \leq B_j y_j(t), \forall j \in \mathcal{N}.$$

Considering the pre-defined federated learning model deployment strategy Z_j at each location $j \in \mathcal{N}$, the decision variable of $y_j(t)$ further satisfies:

$$y_j(t) \leq Z_j Y_j, \forall j \in \mathcal{N}.$$

C. Cost Model of Hierarchical Federated Learning

Based on the control decisions introduced above, we are now ready to model the different cost terms incurred by the hierarchical federated learning, which include the resource operational cost, WAN bandwidth usage cost and the edge server switching cost.

Resource operational cost: when renting servers from public edge platforms such as Aliyun ENS and Amazon CloudFront, the developer of the federated learning application should pay for the usage of the edge servers. For public edge platforms, the on-demand pay-as-you-go charging scheme is typically adopted [19]. With this scheme, we use P_j to denote the cost of renting one server at edge location $j \in \mathcal{N}$ for one time slot. Then the total cost of renting edge servers at each time slot t is given by $\sum_{j \in \mathcal{N}} P_j y_j(t)$. Note that given the fixed deployment strategy of the federated learning model, the amount of edge model updates transferred to the cloud is also fixed, making

the cost of renting cloud servers unchanged. Therefore, the constant cloud server usage cost can be ignored here.

WAN bandwidth usage cost: when dispatching the local model updates of device participants from one edge location to an another location via the WAN, the usage of the expensive WAN bandwidth would also incur the monetary cost. To model this cost, we first formulate the number of edge iterations required to reach a predefined edge training accuracy. Specifically, to achieve a local accuracy ϵ which is common to all the device participants, and an edge accuracy θ which is common to all the edge servers, the number of edge iterations required is given by

$$I(\theta, \epsilon) = \frac{\phi(\log(1/\theta))}{1-\epsilon},$$

where ϕ is a constant that depends on the learning task [12]. Similarly, for each global iteration, to achieve an edge accuracy θ and a global accuracy γ, the number of cloud iterations required is given by

$$L(\gamma, \theta) = \frac{\phi(\log(1/\gamma))}{1-\theta}.$$

Therefore, the total iteration number of edge is

$$G(\gamma, \theta, \epsilon) = L(\gamma, \theta) I(\theta, \epsilon) = \frac{\phi^2(\log(1/\gamma)\log(1/\theta))}{(1-\theta)(1-\epsilon)}.$$

Then, for total edge iterations, the WAN bandwidth usage cost incurred by the uploading of the model updates is given by $\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}} W_{ij} SG(\gamma,\theta,\epsilon) x_{ij}(t)$, here W_{ij} denotes the cost of transmitting one unit data from location i to location j. Also, since the amount of edge model updates transferred to the cloud is also fixed, we ignore the WAN bandwidth usage cost incurred by uploading the edge model updates to the cloud.

Edge server switching cost: launching a new edge server instance commonly involves transferring an edge server image containing the federated learning model from the cloud or edge storage to the hosting edge server, and booting and attaching the image to the server memory. This process would incur additional monetary cost and performance overhead which is refereed to as the server switching cost [20]. Typically, the server switching cost is considered on the order of the cost to run a server for a number of seconds or minutes. The implication of this cost term is that we should keep the amount of launched servers relatively stable over time, rather than frequently launching new edge servers or destroy launched servers. In this paper, we use β_j to denote the switching cost of launching a new edge server at location j, then, at each time slot t, the total edge server switching cost is given by $\sum_{j\in\mathcal{N}} \beta_j [y_j(t) - y_j(t-1)]^+$, where $[a]^+ = \max\{a, 0\}$.

D. The Cost Minimization Problem

By summing up the resource operational cost, WAN bandwidth usage cost and the edge server switching cost, we aim at minimizing the long-term holistic cost over the long-term time horizon T. Formally, this problem can be cast as:

$$\begin{aligned}
\min \quad & \sum_{t\in\mathcal{T}} \Big(\sum_{j\in\mathcal{N}} P_j y_j(t) + \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}} W_{ij} SG(\gamma,\theta,\epsilon) x_{ij}(t) \\
& + \sum_{j\in\mathcal{N}} \beta_j [y_j(t) - y_j(t-1)]^+ \Big) \\
s.t. \quad & y_j(t) \in \{0, 1, 2, \cdots, Y_j\}, \forall j \in \mathcal{N}, \forall t \in \mathcal{T}, && \text{(1a)} \\
& \sum_{i\in\mathcal{N}} S x_{ij}(t) \le B_j y_j(t), \forall j \in \mathcal{N}, \forall t \in \mathcal{T}, && \text{(1b)} \\
& \sum_{j\in\mathcal{N}} x_{ij}(t) = A_i(t), \forall i \in \mathcal{N}, \forall t \in \mathcal{T}, && \text{(1c)} \\
& x_{ij}(t) \ge 0. \forall i, j \in \mathcal{N}, \forall t \in \mathcal{T}, && \text{(1d)} \\
& y_j(t) \le Z_j Y_j, \forall j \in \mathcal{N}, \forall t \in \mathcal{T}. && \text{(1e)}
\end{aligned}$$

We denote the above problem (1) as $\mathsf{P_L}$ which is formulated as a mixed integer linear programming (MILP) over time slots, it is rather challenging due to the following two difficulties. Firstly, since it is a long-term optimization problem with time-coupling switching cost terms, it involves future uncertain information, i.e., the amount of model updates $A_i(t)$, which fluctuates over time. Then, how can we solve problem (1) in an online manner, without knowing the future $A_i(t)$ as a priori knowledge? Secondly, even if in the offline case, the combinatorial problem is NP-hard, since it can be reduced from the classical NP-hard minimum knapsack problem (MKP) [20]. The challenge caused by NP-hard is that the optimal solution cannot be obtained in polynomial time.

III. Online Algorithm Design

In order to overcome the challenges of the uncertain future information and NP-hardness in the long-term cost minimization problem $\mathsf{P_L}$, we first relax the integer variables $y_j(t)$ and replace the switching cost term by applying regularization technique. Then, we decouple the long-term regularization problem into a series of one-shot convex subproblems, and propose Algorithm 1 to solve them in an online manner. Finally, we propose Algorithm 2, which rounds the fractional solution to a feasible yet near-optimal integer solution of the original problem. We further use additional notations $\hat{\boldsymbol{y}}(t)$ to denote the fractional solution obtained by Algorithm 1 and $\bar{\boldsymbol{y}}(t)$ to denote the integer solution obtained by Algorithm 2. Other variables are also similarly denoted.

A. Online Regularization-based Algorithm

We first expend the original optimization problem to the continuous domain by relaxing the integer constraint (1a) to obtain a fractional problem. Directly decoupling the fractional optimization problem in an online manner may cause a large optimal gap, because greedily selecting the number of rented edge servers may lead to high switching cost when the model parameters surge in the short future. Therefore,

we apply an algorithmic technique of regularization [21] to stabilize the solution between the adjacent time slots, basing on the relative entropy function as follow:

$$\Delta\big(y_j(t)||y_j(t-1)\big) = y_j(t)\ln\frac{y_j(t)}{y_j(t-1)} + y_j(t-1) - y_j(t). \tag{2}$$

To adapt to our problem in specific, we add a positive constant ε to both $y_j(t)$ and $y_j(t-1)$ in (2) to ensure that the fraction is valid when no edge server is rented at time slot $t-1$ (i.e., $y_j(t-1) = 0$). Furthermore, we define $\eta_j = \ln(1+\frac{Y_j}{\varepsilon})$ as an approximation weight factor, and multiply $\frac{1}{\eta_j}$ to the above smooth convex function for normalization. Specially, we use $\mathsf{P}_{\mathsf{L}}^{\mathsf{r}}(t)$ to denote the decomposed regularization problem for each time slot $t \in \mathcal{T}$ as follow:

$$\begin{aligned}\min \quad P_L^r(t) = &\sum\nolimits_t \sum\nolimits_j \frac{\beta_j}{\eta_j}\Big(\big(y_j(t)+\varepsilon\big)\ln\frac{y_j(t)+\varepsilon}{y_j(t-1)+\varepsilon} \\ &- y_j(t)\Big) + \sum\nolimits_t \sum\nolimits_j P_j y_j(t) \\ &+ \sum\nolimits_t \sum\nolimits_i \sum\nolimits_j SW_{ij} x_{ij}(t) G(\gamma,\theta,\epsilon)\end{aligned}$$

$$\begin{aligned}
s.t. \quad & y_j(t) \in [0, Y_j], && \forall j, \forall t, && \text{(3a)}\\
& \sum\nolimits_i x_{ij}(t) S - B_j y_j(t) \le 0, && \forall j, \forall t, && \text{(3b)}\\
& \sum\nolimits_j x_{ij}(t) - A_i(t) = 0, && \forall i, \forall t, && \text{(3c)}\\
& x_{ij}(t) \ge 0, && \forall i, \forall j, \forall t, && \text{(3d)}\\
& y_j(t) - Z_j Y_j \le 0, && \forall j, \forall t. && \text{(3e)}
\end{aligned}$$

We design an *Online Regularization-based Algorithm* (ORA) to obtain the optimal fractional solution $\hat{\boldsymbol{x}}(t)$ and $\hat{\boldsymbol{y}}(t)$ as shown in Algorithm 1. In each time slot $t \in \mathcal{T}$, ORA observers $\boldsymbol{A}(t)$ and $\hat{\boldsymbol{y}}(t-1)$ and calculates the iteration number of the edge server $G(\gamma,\theta,\epsilon)$. By taking existing convex optimization techniques such as interior-point method [22], ORA solves the standard convex optimization problem $\mathsf{P}_{\mathsf{L}}^{\mathsf{r}}(t)$ in polynomial time. We will discuss the feasibility of the solution and derive the competitive ratio of Algorithm 1 in Section IV.

Algorithm 1: Online Regularization-based Algorithm — ORA

Input: $\mathcal{N}, \boldsymbol{B}, \boldsymbol{W}, \boldsymbol{Z}, \boldsymbol{\eta}, S, \theta, \epsilon, \varepsilon$.
Initialize $\hat{\boldsymbol{y}}(0) = \boldsymbol{0}$;
for time slot $t \in \mathcal{T}$ **do**
  Observe $\boldsymbol{A}(t), \hat{\boldsymbol{y}}(t-1)$;
  Calculate $G(\gamma,\theta,\epsilon)$;
  Solve $\mathsf{P}_{\mathsf{NL}}^{\mathsf{r}}(t)$ to obtain the solution $\hat{\boldsymbol{x}}(t), \hat{\boldsymbol{y}}(t)$;
  Return $\hat{\boldsymbol{x}}(t), \hat{\boldsymbol{y}}(t)$;
Output: $\hat{\boldsymbol{x}}(t), \hat{\boldsymbol{y}}(t)$.

B. Randomized Rounding Algorithm

To satisfy constraint (3a), i.e., $y_j(t)$ is an integer variable, we further round the fractional solution $\hat{\boldsymbol{y}}(t)$ obtained in Algorithm 1 to an integer solution $\bar{\boldsymbol{y}}(t)$ by applying the randomized rounding scheme. Intuitively, the randomized rounding scheme rounds up each fractional solution $\hat{\boldsymbol{y}}(t)$ to the nearest integer $\bar{\boldsymbol{y}}(t) = \lceil \hat{\boldsymbol{y}}(t) \rceil$ with the probability of $\hat{\boldsymbol{y}}(t) - \lfloor \hat{\boldsymbol{y}}(t) \rfloor$ and rounds down each fractional solution $\hat{\boldsymbol{y}}(t)$ to the nearest integer $\bar{\boldsymbol{y}}(t) = \lfloor \hat{\boldsymbol{y}}(t) \rfloor$ with the probability of $\lceil \hat{\boldsymbol{y}}(t) \rceil - \hat{\boldsymbol{y}}(t)$.

Algorithm 2: Randomized Rounding Algorithm — RRA

Input: $\hat{\boldsymbol{y}}(t)$.
1 $\chi_j(t) \triangleq \hat{y}_j(t), \forall j$;
2 $J' \triangleq J\{j | \chi_j(t) - \lfloor \chi_j(t) \rfloor\}$;
3 **while** $|J'| \ge 1$ **do**
4 Select j_1,j_2 and $j_1 \ne j_2$;
5 $\theta_1 \triangleq \min\{\lceil \chi_{j_1}(t) \rceil - \chi_{j_2}, \chi_{j_2}\}$;
6 $\theta_2 \triangleq \min\{\lceil \chi_{j_2}(t) \rceil - \chi_{j_1}, \chi_{j_1}\}$;
7 With the probability $\frac{\theta_2}{\theta_1+\theta_2}$,
8 set $\chi'_{j_1}(t) = \chi_{j_1}(t) + \theta_1, \chi'_{j_2}(t) = \chi_{j_2}(t) - \theta_1$;
9 With the probability $\frac{\theta_1}{\theta_1+\theta_2}$,
10 set $\chi'_{j_1}(t) = \chi_{j_1}(t) - \theta_2, \chi'_{j_2}(t) = \chi_{j_2}(t) + \theta_2$;
11 **if** $\chi'_{j_1}(t) - \lfloor \chi'_{j_1}(t) \rfloor = 0$ **then**
12 Set $\bar{y}_{j_1}(t) = \chi'_{j_1}(t), J' = J' \backslash j_1$;
13 **else**
14 Set $\bar{y}_{j_1}(t) = \chi'_{j_1}(t)$;
15 **if** $\chi'_{j_2}(t) - \lfloor \chi'_{j_2}(t) \rfloor = 0$ **then**
16 Set $\bar{y}_{j_2}(t) = \chi'_{j_2}(t), J' = J' \backslash j_2$;
17 **else**
18 Set $\bar{y}_{j_2}(t) = \chi'_{j_2}(t)$;
19 **if** $|J'| = 1$ **then**
20 Set $\bar{y}_j(t) = \lceil \hat{y}_j(t) \rceil$.
21 Return $\bar{y}_j(t)$.
Output: $\bar{\boldsymbol{y}}(t)$.

For the sake of reducing operational cost and ensuring the bandwidth within the capacity of the edge server, we further design a *Randomized Rounding Algorithm* (RRA) inspired by the pairwise rounding technique [23]. In each iteration, RRA picks a pair of fractions randomly and rounds one or two of the picked values to an integer. Based on probability, RRA executes one of the line 8 and 10 per iteration in the main loop.

In our proposed rounding algorithm, the main loop of each iteration has two properties: (1) In each iteration, at least one fraction is rounded to an integer; (2) After each iteration, there is $\chi_{j_1}(t) + \chi_{j_2}(t) = \chi'_{j_1}(t) + \chi'_{j_2}(t)$, which ensures that the solutions do not violate the constraints of $P_L^r(t)$. Based on the two properties above, we discuss the

rounding gap between the integer solution and the fractional solution in Section IV.

IV. Theoretical Analysis

In this section, we rigorously analyze the performance of the presented online algorithm, by deriving the its competitive ratio based on the well-established duality theory.

A. Basic Idea

We prove that the value of the parameterized constant times the offline optimum of $\mathsf{P_L}$ is the upper bound of the long-term cost achieved by our proposed online algorithm framework. $\mathsf{D_L}$ is the dual problem of $\mathsf{P_L}$, and π denotes a mapping that maps the fractional solutions of $\mathsf{P_L}$ to a feasible solutions of $\mathsf{D_L}$. We establish and prove the chain of inequalities:

$$\begin{aligned} & P_L(\hat{x}(t), \bar{y}(t)) && \text{(4a)}\\ \leq & r_2 P_L(\hat{x}(t), \hat{y}(t)) && \text{(4b)}\\ \leq & r_1 r_2 D_L(\pi(\hat{x}(t), \hat{y}(t)) && \text{(4c)}\\ \leq & r_1 r_2 P_{L_{OPT}}, && \text{(4d)} \end{aligned}$$

where $r = r_1 r_2$ is the overall competitive ratio. $P_L(\hat{x}(t), \bar{y}(t))$ is the long-term total cost achieved by our proposed online algorithm framework. We derive the rounding gap of our Algorithm 2 in r_2 through (4a)$\leq$(4b). We derive (4b)$\leq$(4c) with the primal-dual method. $P_{L_{OPT}}$ is the offline optimum of the original problem. Regarding (4c)$\leq$(4d), it establishes based on the weak duality.

B. Competitive Ratio

In this subsection, we derive the competitive ratio r_1 of Algorithm 1 based on the primal-dual method and subsequently prove (4b)$\leq$(4c)$\leq$(4d).

In ORA, we obtain $\hat{x}(t)$ and $\hat{y}(t)$ by solving the one-shot regularization problem $P_L^r(t)$. On account of that $P_L^r(t)$ is convex and differentiable, the optimal solutions of the primal and dual problems are equivalent when the solutions of $P_L^r(t)$ satisfy the Karush-Kuhn-Tucker (KKT) conditions, which is the necessary and sufficient conditions for a nonlinear programming problem. $\mathsf{D_L}$ is the dual problem of original problem $P_L(t)$, and π maps the fractional solutions of $P_L(t)$ to feasible solutions of $D_L(t)$. Consequently, we only need to prove that the solutions constructed by mapping π satisfy (4b)$\leq$(4c) when they satisfy the KKT conditions of $P_L^r(t)$.

Formulating Lagrange Dual. To cast the dual form of the original problem more intuitively, we define $w_j(t)$ to replace $\big[y_j(t)-y_j(t-1)\big]^+$, and the equivalent transformation form of the original problem $\mathsf{P_L}$ as follow:

$$\begin{aligned} \min \quad P_L(t) = & \sum\nolimits_t \sum\nolimits_j \beta_j w_j(t) + \sum\nolimits_t \sum\nolimits_j P_j y_j(t) \\ & + \sum\nolimits_t \sum\nolimits_i \sum\nolimits_j S W_{ij} x_{ij}(t) G(\gamma, \theta, \epsilon) \end{aligned}$$

$$s.t. \quad y_j(t) \in [0, Y_j], \qquad \forall j, \forall t, \tag{5a}$$

$$y_j(t) - y_j(t-1) - w_j(t) \leq 0, \qquad \forall j, \forall t, \tag{5b}$$

$$\sum\nolimits_i x_{ij}(t) S - B_j y_j(t) \leq 0, \qquad \forall j, \forall t, \tag{5c}$$

$$\sum\nolimits_j x_{ij}(t) - A_i(t) = 0, \qquad \forall i, \forall t, \tag{5d}$$

$$y_j(t) - Z_j Y_j \leq 0, \qquad \forall j, \forall t, \tag{5e}$$

$$x_{ij}(t) \geq 0, \qquad \forall i, \forall j, \forall t. \tag{5f}$$

To maintain consistency with the original problem, we add constraint (5b). In the subsequent derivation, we adopt this transformation to facilitate our performance analysis.

The Lagrangian dual problem is constructed as follows, where $\alpha_j(t)$, $\lambda_j(t)$, $\phi_i(t)$ and $\varphi_j(t)$ are dual variables.

$$\max \quad D_L = \sum\nolimits_t \sum\nolimits_i \phi_i(t) A_i(t) - \sum\nolimits_t \sum\nolimits_j \varphi_j(t) Z_j Y_j$$

$$\begin{aligned} s.t. \quad & \beta_j - \lambda_j(t) \geq 0 \quad \forall j, \forall t, && \text{(6a)}\\ & P_j - B_j \alpha_j(t) + \lambda_j(t) - \lambda_j(t+1) + \varphi_j(t) \geq 0 \forall j, \forall t, && \text{(6b)}\\ & S\alpha_j(t) + S W_{ij} G(\theta, \epsilon) - \phi_i(t) \geq 0 \quad \forall i, \forall j, \forall t, && \text{(6c)}\\ & \alpha_j(t), \lambda_j(t), \varphi_j(t), \phi_i(t) \geq 0 \quad \forall i, \forall j, \forall t. && \text{(6d)} \end{aligned}$$

Constructing the Mapping. We construct the mapping as follows, where $\hat{y}_j(t)$ are derived by solving $P_L^r(t)$ in Algorithm 1, and $\alpha_j'(t)$, $\lambda_j'(t)$, $\phi_i'(t)$ and $\varphi_j'(t)$ are dual variables.

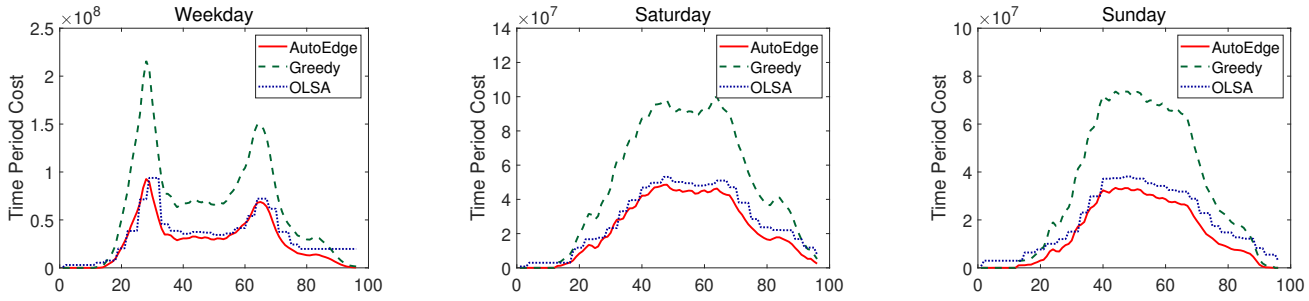
$$\begin{aligned} & \alpha_j(t) = \alpha_j'(t), && \forall j, t, \\ & \phi_i(t) = \phi_i'(t), && \forall i, t, \\ & \varphi_j(t) = \varphi_j'(t), && \forall j, t, \\ & \lambda_j(t) = \frac{\beta_j}{\eta_j} \ln \frac{Y_j + \varepsilon}{\hat{y}_j(t-1) + \varepsilon}, && \forall j, t. \end{aligned}$$

The Karush-Kuhn-Tucker (KKT) conditions of $\mathsf{P_L^r(t)}$ are as follows:

$$\begin{aligned} & S\alpha_j(t) + S W_{ij} G(\theta, \epsilon) - \phi_i(t) = 0, && \text{(7a)}\\ & P_j - B_j \alpha_j(t) + \frac{\beta_j}{\eta_j} \ln \frac{\hat{y}_j(t) + \varepsilon}{\hat{y}_j(t-1) + \varepsilon} + \varphi_j(t) = 0, && \text{(7b)}\\ & A_i(t) - \sum\nolimits_j \hat{x}_{ij}(t) = 0, && \text{(7c)}\\ & \alpha_j(t) \big(\sum\nolimits_i \hat{x}_{ij}(t) S - B_j \hat{y}_j(t) \big) = 0, && \text{(7d)}\\ & \varphi_j(t) (\hat{y}_j(t) - Z_j Y_j) = 0, && \text{(7e)}\\ & \text{all primal and dual variables} \geq 0. && \text{(7f)} \end{aligned}$$

Since $\hat{y}_j(t-1) \geq 0$, $\lambda_j(t) \leq \frac{\beta_j}{\eta_j} \ln \frac{Y_j + \varepsilon}{\varepsilon}$ establishes. As we mentioned in Section III, we define $\eta_j = \ln \frac{Y_j + \varepsilon}{\varepsilon}$, then, the solution satisfies (6a). Transform (7b) into the following form:

$$P_j - B_j \alpha_j(t) + \frac{\beta_j}{\eta_j} \ln \frac{Y_j + \varepsilon}{\hat{y}_j(t-1) + \varepsilon} - \frac{\beta_j}{\eta_j} \ln \frac{Y_j + \varepsilon}{\hat{y}_j(t) + \varepsilon} + \varphi_j(t) = 0,$$



(a) Time Period Cost of Weekday (b) Time Period Cost of Saturday (c) Time Period Cost of Sunday

Fig. 1. Time Period Cost comparison chart.

which satisfies (6b). Furthermore, the solutions obtained via π-map are able to satisfy constraints (6c) and (6d). Therefore, π is a map of feasible solutions of $D_L(t)$.

Bounding. Considering switching cost and non-switching cost separately, we can derive $P_L(\hat{x}(t), \hat{y}(t)) \leq r_1 D_L(\pi(\hat{x}(t), \hat{y}(t))$ in Theorem 1.

Theorem 1. *$P_L(\hat{x}(t), \hat{y}(t)) \leq r_1 D_L(\pi(\hat{x}(t), \hat{y}(t))$, where $r_1 = 1 + |J| max_j \beta_j Y_j$.*

Proof. The detailed proof of Theorem 1 is given in the Appendix A of our online technical report [24]. □

Since the solutions obtained via π-map are feasible solutions of the maximum optimization problem D_L, we have $D_L(\pi(\hat{x}(t), \hat{y}(t))) \leq D_{L_{OPT}}$. Due to weak duality, we have $D_{L_{OPT}} \leq P_{L_{OPT}}$. Therefore, (4c) $\leq$ (4d) establishes.

C. Rounding Gap

We derive r_2 in Theorem 2 and prove (4b)$\leq$(4c).

Theorem 2. *$P_L(\hat{x}(t), \bar{y}(t)) \leq r_2 P_L(\hat{x}(t), \hat{y}(t))$ holds since $\sum_t \sum_j P_j \bar{y}_j(t) + \sum_t \sum_i \sum_j SW_{ij} x_{ij}(t) G(\gamma, \theta, \epsilon) \leq \delta'_y P_L(\hat{x}(t), \hat{y}(t))$ and $\sum_t \sum_j \beta_j [\bar{y}_j(t) - \bar{y}_j(t-1)]^+ \leq \delta''_y P_L(\hat{x}(t), \hat{y}(t))$, where*

$\delta'_y = 1 + \frac{|J| \max_{j,t} Y_j P_j}{\min_i A_i(t) \min_{i,j,t} SW_{ij} G(\theta,\epsilon)}$,

$\delta''_y = \frac{|J| \max_j Y_j \beta_j}{\min_i A_i(t) \min_{i,j,t} SW_{ij} G(\theta,\epsilon)}$,

$r_2 = \delta'_y + \delta''_y$.

Proof. The detailed proof of Theorem 2 is given in the Appendix B of our online technical report [24]. □

V. Performance Evaluation

In this section, we conduct trace-driven simulations to empirically evaluate the performance of the presented AutoEdge.

A. Experimental Setup

Device Participants and Edge Locations. To simulate the dynamics of the device participants at edge locations, we resort to the number of dynamic passengers at London's underground stations. Specifically, out of London's 268 underground stations, we take the largest $N = 20$ stations as the set of edge locations (based on the annual passenger count). We use the dynamic numbers of passengers of each underground station as the amount of device participants arrived at each edge location at every time slot. The adopted sample trace [25] of the passenger traffic is measured for every quarter (15 minutes) for a weekday, a Saturday, and a Sunday around Nov. 16, 2016. Following such a sampling frequency, we set the length of each time slot as 15 minutes. For the federated learning model, we assume that it is deployed at 7 edge locations (i.e., $Z_j = 1$) that host colocation datacenters according to the Data Center Map [26].

System Parameters. According to the geographical locations of the edge locations, we measure the distance among them by using Google Maps, and set the unit WAN bandwidth usage cost W_{ij} in equal proportion. Referring to rental quotations of edge servers, we set the unit edge server switching cost and unit resource operational cost. Moreover, we randomly set the value of parameters as follow: the range of edge server capacity is from 2,500 to 3,000, and the range of upper bound of edge servers is from 10 to 30.

Baselines. In the results, we denote our approach as `AutoEdge`. We also implement two comparing algorithms. The first is `Greedy`, which calculates the unit cost of data transmission first, according to WAN bandwidth usage cost, the edge server switching cost, resource operational cost, etc. Based on the order of cost, the algorithm makes the decision of data routing greedily. The second algorithm is the online lazy switching algorithm (`OLSA`) [27], which decides whether to reset the number of edge servers according to the comparison between switching cost and non-switching cost.

B. Evaluation Results

We use the three segments of the passenger trace on Weekday, Saturday and Sunday respectively, to conduct simulation experiments. We use the Time Period Cost and the Total Cost to evaluate the performance of the algorithms. For the two metrics, the former is the cost obtained by the algorithms of a single time slot, and the latter is the total cost of entire time horizon. The objective function is the addition of several different costs, including the edge server switching cost, resource operational cost and WAN bandwidth cost. On

account of the inconsistent units of different costs, we adjust the weighted coefficients of each cost and consequently the total cost is unitless.

Time Period Cost. We plot the Time Period Cost of the different algorithms over three datasets in Fig.1. There is a big difference between weekday and weekend trends. The trend of resource scheduling cost obtained by three algorithms is consistent. Fig.1 confirms that the cost of `Greedy` is significantly more than cost of the other two algorithms in each time period, which is consistent with the results of the total cost. The cost trend lines of `AutoEdge` and `OLSA` are very close, which is consistent with the fact that their total cost is close.

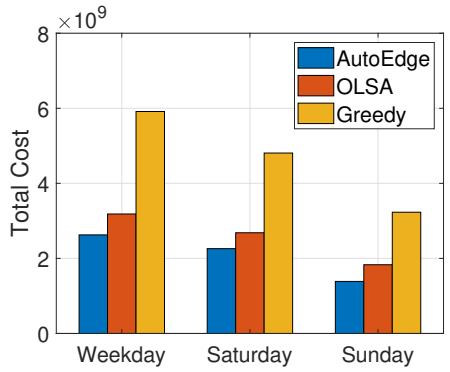


Fig. 2. Total Cost

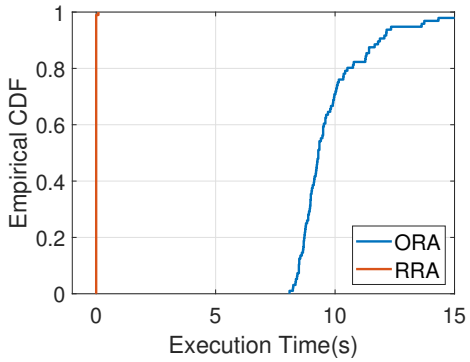


Fig. 3. Execution Time.

Total Cost. We plot the Total Cost of the different algorithms over three datasets in Fig.2. Our approach saves 55.21% and 19.16% total cost compared to `Greedy` and `OLSA`. The cost of `Greedy` is significantly more than the other two algorithms because it makes decisions greedily on edge server deployment without considering the huge edge server switching cost between adjacent time slots. It is different that `OLSA` achieves comparatively good results by considering the edge server switching cost of adjacent coupling time slots. However, due to its "laziness" characteristics, the effect is not as good as our proposed online algorithm framework.

Execution Time. Fig.3 depicts the execution time of our proposed algorithms. The CDF curve of `RRA` is almost a vertical line in Fig.3, because the execution time of `RRA` is very short, within one second. The average execution time of `ORA` is about 9.2174s, which is much less then the length of 15 minutes of a single time slot. Therefore, our proposed algorithms are efficient in an online manner.

VI. Conclusion

In this work, we presented AutoEdge, an online edge resource autoscaling framework for hierarchical federated learning over the emerging public edge platforms. It jointly optimizes the edge server autoscaling and model updates dispatching to minimize the system-wide cost. The formulated optimization problem is rather difficult since it is NP-hard and involves future uncertain information. These challenges are addressed by carefully blending the advantages of an online optimization technique and an approximate optimization method. Specifically, with a regularization technique, we first decompose the long-term problem into a series of one-shot fractional problems which can be readily solved. Adopting a randomized dependent scheme, we further round the fractional solutions to a near-optimal integral solution of the original problem. The efficacy of AutoEdge is verified by both theoretical analysis and trace-driven simulations.

References

[1] 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.

[2] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, 2019.

[3] Q. Pu, G. Ananthanarayanan, P. Bodik, S. Kandula, A. Akella, P. Bahl, and I. Stoica, "Low latency geo-distributed data analytics," in *Proc. of ACM SIGCOMM*, 2015.

[4] J. Du, J. Jiang, Y. You, D. Huang, and Y. Lu, "Handling heavy-tailed input of transformer inference on gpus," in *Proceedings of the 36th ACM International Conference on Supercomputing*, 2022, pp. 1–11.

[5] C. Niu, F. Wu, S. Tang, L. Hua, R. Jia, C. Lv, Z. Wu, and G. Chen, "Billion-scale federated learning on mobile clients: A submodel design with tunable privacy," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, 2020, pp. 1–14.

[6] 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.

[7] 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," in *Proc. of IEEE INFOCOM*, 2018.

[8] Y. Li, J. Park, M. Alian, Y. Yuan, Z. Qu, P. Pan, R. Wang, A. Schwing, H. Esmaeilzadeh, and N. S. Kim, "A network-centric hardware/algorithm co-design to accelerate distributed training of deep neural networks," in *Proc. of IEEE/ACM MICRO*, 2018.

[9] L. Wang, W. Wang, and B. Li, "Cmfl: Mitigating communication overhead for federated learning," 2019.

[10] J. Ren, G. Yu, and G. Ding, "Accelerating dnn training in wireless federated edge learning system," *arXiv preprint arXiv:1905.09712*, 2019.

[11] N. H. Tran, W. Bao, A. Zomaya, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE INFOCOM*, 2019.

[12] S. Luo, X. Chen, Q. Wu, Z. Zhou, and S. Yu, "Hfel: Joint edge association and resource allocation for cost-efficient hierarchical federated edge learning," *IEEE Transactions on Wireless Communications*, vol. 19, no. 10, pp. 6535–6548, 2020.

[13] M. Xu, Z. Fu, X. Ma, L. Zhang, Y. Li, F. Qian, S. Wang, K. Li, J. Yang, and X. Liu, "From cloud to edge: a first look at public edge platforms," in *Proceedings of the 21st ACM Internet Measurement Conference*, 2021, pp. 37–53.

[14] Z. Zhou, F. Liu, Y. Xu, R. Zou, H. Xu, J. C. Lui, and H. Jin, "Carbon-aware Load Balancing for Geo-distributed Cloud Services," in *Proc. of IEEE MASCOTS*, 2013.

[15] X. Wang, L. T. Yang, X. Xie, J. Jin, and M. J. Deen, "A cloud-edge computing framework for cyber-physical-social services," *IEEE Communications Magazine*, vol. 55, no. 11, pp. 80–85, 2017.

[16] Y. Jin, L. Jiao, Z. Qian, S. Zhang, and S. Lu, "Budget-aware online control of edge federated learning on streaming data with stochastic inputs," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3704–3722, 2021.

[17] Y. Jin, L. Jiao, Z. Qian, S. Zhang, S. Lu, and X. Wang, "Resource-efficient and convergence-preserving online participant selection in federated learning," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2020, pp. 606–616.

[18] H. Zheng, F. Xu, L. Chen, Z. Zhou, and F. Liu, "Cynthia: Cost-efficient cloud resource provisioning for predictable distributed deep neural network training," in *Proceedings of the 48th International Conference on Parallel Processing*, 2019, pp. 1–11.

[19] "Edge node service - alibaba cloud," https://www.alibabacloud.com/pricing?spm=a3c0i.26313822.6791778070.dnavpricing0.76212b3eSUM5SR, accessed on November 6, 2022.

[20] Z. Zhou, Q. Wu, and X. Chen, "Online orchestration of cross-edge service function chaining for cost-efficient edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 8, pp. 1866–1880, 2019.

[21] N. Buchbinder, S. Chen, and J. Naor, "Competitive analysis via regularization," in *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*. SIAM, 2014, pp. 436–444.

[22] Y. Nesterov and A. Nemirovskii, *Interior-point polynomial algorithms in convex programming*. SIAM, 1994.

[23] A. A. Ageev and M. I. Sviridenko, "Pipage rounding: A new method of constructing algorithms with proven performance guarantee," *Journal of Combinatorial Optimization*, vol. 8, no. 3, pp. 307–328, 2004.

[24] Technical Report for Edge Resource Autoscaling for Hierarchical Federated Learning Over Public Edge Platforms. Accessed on November 14, 2022.[Online]. [Online]. Available: https://1drv.ms/u/s!Aoh3a1diNSPfdATYpxjn1fljBXg?e=JkQYdW

[25] L. Jiao, L. Pu, L. Wang, X. Lin, and J. Li, "Multiple granularity online control of cloudlet networks for edge computing," in *2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 2018, pp. 1–9.

[26] Data Center Map. [Online]. Available: https://www.datacentermap.com/

[27] B. Gao, Z. Zhou, F. Liu, and F. Xu, "Winning at the starting line: Joint network selection and service placement for mobile edge computing," in *IEEE INFOCOM 2019-IEEE conference on computer communications*. IEEE, 2019, pp. 1459–1467.