Learning for Learning: Predictive Online Control of Federated Learning with Edge Provisioning

Yibo Jin[1], Lei Jiao[2§], Zhuzhong Qian[1§], Sheng Zhang[1§], Sanglu Lu[1]
[1]Nanjing University, China [2]University of Oregon, USA
Email: {yibo.jin@smail.nju.edu.cn, jiao@cs.uoregon.edu, {qzz, sheng, sanglu}@nju.edu.cn}

***Abstract*—Operating federated learning optimally over distributed cloud-edge networks is a non-trivial task, which requires to manage data transference from user devices to edges, resource provisioning at edges, and federated learning between edges and the cloud. We formulate a non-linear mixed integer program, minimizing the long-term cumulative cost of such a federated learning system while guaranteeing the desired convergence of the machine learning models being trained. We then design a set of novel polynomial-time online algorithms to make adaptive decisions by solving continuous solutions and converting them to integers to control the system on the fly, based only on the predicted inputs about the dynamic and uncertain cloud-edge environments via online learning. We rigorously prove the competitive ratio, capturing the multiplicative gap between our approach using predicted inputs and the offline optimum using actual inputs. Extensive evaluations with real-world training datasets and system parameters confirm the empirical superiority of our approach over multiple state-of-the-art algorithms.**

I. Introduction

The emerging distributed cloud-edge infrastructures provide computing resources in closer proximity to end users, and are thus well-positioned to support *federated learning*. Federated learning is a novel machine learning paradigm which trains machine learning models through iterative local model updates and global aggregations of model parameters [1] while keeping users' data local, in contrast to uploading them to a central location like the cloud as in other traditional machine learning approaches. By data localization, this approach protects user privacy [2], and respects the regulatory and ownership requirements [3]. As artificial intelligence finds increasing adoptions in Internet services and applications (e.g., web browsing [4] and GPS positioning [5]), operating federated learning across cloud-edge networks becomes essential. Fig. 1 illustrates such a system in a multi-carrier cloud-edge environment.

It is, however, non-trivial for a service provider to optimally operating federated learning as a service across user devices, edges, and the cloud, since it often requires the end-to-end, cross-layer management of many factors, such as data transference from devices to edges, provisioning of edge resources, local updates at edges, model transference between edges and the cloud, as well as global aggregations in the cloud. In fact, the service provider faces multiple critical challenges:

§ These are corresponding authors.

This work was supported in part by the National Key R&D Program of China (2017YFB1001801), the National Science Foundation of China (61832005, 61872175), the Ripple Faculty Fellowship, the Natural Science Foundation of Jiangsu Province (BK20181252), the Collaborative Innovation Center of Novel Software Technology and Industrialization, and the Nanjing University Innovation and Creative Program for PhD (CXCY19-25).

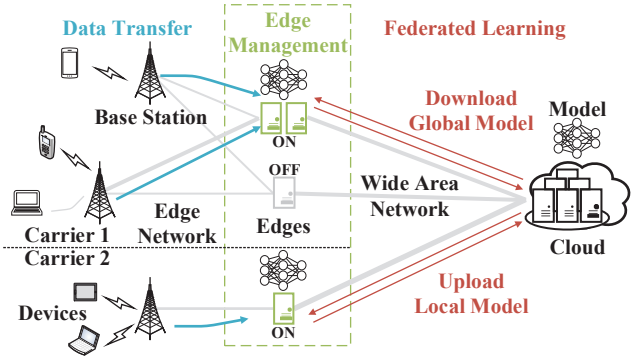


Fig. 1: Federated learning over cloud-edge networks

First of all, it is crucial to balance the resources used and the quality (e.g., convergence accuracy [6, 7]) of the models trained by federated learning. The local model updates at edges consume computation, and the transference of model parameters between edges and the cloud for aggregation consumes network bandwidth, which all contribute to the model convergence. Preserving the specified convergence at the minimum computation and communication cost is a challenging task, since the trained models are often non-linear and how the local computation and global aggregation impacts convergence [8] is complicated. This challenge escalates when managing large-scale federated learning that consumes excessive resources across geographically distributed edges and the cloud.

Second, real-world cloud-edge platforms are intrinsically dynamic and uncertain, and to train multiple models over time, the system control decisions often need to be made before knowing the dynamic inputs regarding the environments. Such inputs include time-varying volumes of training data, fluctuating available network bandwidths [9], and edge operational cost such as energy consumption. One may utilize a certain prediction mechanism to predict the inputs and control the system based on such predictions; however, it is difficult to ensure the quality of predictions and optimize the cumulative system performance through *predicted* inputs, compared to the offline optimum in hindsight (i.e., an oracle's perspective where all the *actual* inputs are known at once in advance).

Third, no matter the inputs are revealed and known in real time or not, it remains hard to manage the resources online [10, 11]. To strategically reduce the overall resource usage [12, 13], one may desire to switch on/off the edges dynamically. Such dynamic edge management, however, incurs switching cost, (e.g., time needed for edge initialization or any cost related to system oscillation, reliability risk, and hardware wear-and-tear

[14], and determining the on/off status on the fly for long-term optimization is uneasy: having an edge on may unnecessarily incurs operational cost; but having it off may incur excessive switching cost if it needs to be on in the next epoch). Such decisions are time-coupled, and often not straightforward.

Existing research falls insufficient for addressing the aforementioned challenges. Some [6, 8, 15–17] have considered various local model optimizers and the convergence of the global aggregated models, but they do not capture the resource consumption of the federated learning process. Others [18–22] optimize federated learning systems with ensured convergence, but they overlook the cloud-edge environments that have dedicated cost management complexities and consider no predictions. A substantial body of research [23–27] studies online cloud/edge resource provisioning, but cannot be applied as they are not for federated learning, failing to capture the corresponding data and resource patterns. The rest [28–34] study predictive control in various scenarios, but do not involve federated learning convergence over cloud-edge networks.

In this paper, we firstly model and formulate the problem of the joint control of federated learning and edge provisioning in distributed cloud-edge networks. Our problem is an NP-hard, non-linear mixed integer program, minimizing the long-term total cost, including data transference cost from devices to edges, computation cost of local model updates at edges, model transference cost from edges to the cloud, computation cost of global model aggregations in the cloud, as well as edge provisioning and switching cost. Our formulation features federated learning by leveraging the dependency of the numbers of model updates and aggregations upon model convergence, and allows arbitrary dynamic inputs and predictions.

Then, we propose and design a group of polynomial-time algorithms to solve our problem in an online manner. Our first algorithm proposed, while solving the federated learning control decisions, keeps the existing on/off status of edges and only changes them until a carefully-designed condition is met as the cumulative non-switching cost exceeds a pre-specified constant times the current switching cost. Our second algorithm, invoked by our first algorithm, converts the fractional edge status into integral on/off decisions in a randomized manner without violating any constraints. Our third algorithm predicts the inputs in real time and feeds them to the first two algorithms, using our novel, rectified online learning approach through a series of well-designed functions based only on the existing, observable inputs and previous control decisions.

Further, we perform rigorous formal analysis for our algorithms. We exhibit a parameterized competitive ratio as our entire online approach's worst-case performance guarantee. Unlike the standard definitions, our version of the competitive ratio not only compares the cost of our online approach against the offline optimum, but also does so using the predicted inputs for the former and the actual inputs for the latter, which makes more sense in our scenario. We highlight that this is a non-trivial, interesting theoretical result on its own, requiring all our algorithms to work together and differing our work from most existing online algorithms and competitive analysis.

TABLE I: Summary of Notations

Inputs	Descriptions[1]
d_{ikt}	Volume of data generated at device i
$a_{ijkt}, \widehat{a}_{ijkt}$	Transference cost[2] per unit data from device i to edge j
$b_{jkt}, \widehat{b}_{jkt}$	Transference cost per unit data from edge j to the cloud
m_t	Size of the model trained by federated learning
p_{jkt}	Processing cost per single local update at edge j
c_t	Processing cost per unit data in the cloud
s_{jkt}	Unit switching cost for activating edge j
o_{jkt}	Operational cost of edge j
Decisions	**Descriptions**
x_{ijkt}	Ratio of data transferred from device i to edge j
y_{jkt}	Whether to activate edge j
η_t	Local "convergence accuracy" of federated learning

1. The subscript k means "in the edge network k"; t means "in the epoch t".
2. The mark of $\widehat{\ }$ refers to the corresponding predicted values.

Finally, we conduct extensive evaluations using real-world traces, including the MNIST training data [35], Google Device Participants for federated learning [1], and the cellular [9] and wide area network (WAN) [36] bandwidths, to train the models of support vector machines and convolutional neural networks. We compare the practical performance of our approach to several alternatives combining different control and prediction algorithms. We find that our approach performs the best in a variety of settings, saving 37% or more cost cumulatively; the models trained by the federated learning process under our control have desired inference accuracy of around 0.8, aligned with existing literatures [7, 23]; our approach scales well, and finishes execution within only several seconds on average.

II. System Model and Problem Formulation

A. System Settings and Models

We summarize the major notations used in Table I.

Cloud-Edge Networks: We consider a set of edge networks $\mathcal{K}$, each of which is owned and operated by its corresponding carrier. We consider a service provider that deploys a set of distributed edges $\mathcal{E}_k$ within the edge network $k \in \mathcal{K}$, where an "edge" refers to a micro data center or server cluster [37, 38], locating in the neighborhood, regional office, or metro center. Such edges are connected to end users via cellular or wireline access, and to a remote cloud via wide area networks (WANs). We then consider a service deployed over such a cloud-edge infrastructure, which also trains machine learning models by federated learning. We study the system over a series of epochs $\mathcal{T} = \{1, .., T\}$, and denote the set of user devices served by the edge network k in the epoch $t \in \mathcal{T}$ as $\mathcal{U}_{kt}$.

Federated Learning: The service collects users' data at the edges and conducts federated learning between edges and the remote cloud. In each epoch t, the federated learning process can be divided into a number of "training iterations", and each training iteration r consists of three steps: (1) every participating edge $j \in \mathcal{E}_k$, $\forall k$ downloads the latest aggregated model of the previous training iteration (i.e., $\boldsymbol{w_t^{(r-1)}}$), along with the corresponding gradients downloaded from the cloud, (2) then iteratively updates the model[1] locally through an iterative

[1] $\mathcal{G}_{jkt}^{(r)}(\boldsymbol{\rho}) \triangleq \mathcal{F}_{jkt}(\boldsymbol{w_t^{(r-1)}} + \boldsymbol{\rho}) - (\nabla \mathcal{F}_{jkt}(\boldsymbol{w_t^{(r-1)}}) - \xi_1 \mathcal{J}_t(\boldsymbol{w_t^{(r-1)}}))^\top \boldsymbol{\rho} + \frac{\xi_2}{2}||\boldsymbol{\rho}||^2$, where ξ_1 and ξ_2 are non-negative constants, and $\mathcal{J}_t(\cdot)$ is the sum of the gradients among all activated edges, $\mathcal{J}_t(\cdot) = \sum_{k,j} \nabla \mathcal{F}_{jkt}(\cdot) / \sum_{k,j} y_{jkt}$.

gradient-based process $\boldsymbol{\rho}_{\boldsymbol{jkt}}^{(\boldsymbol{q})} = \boldsymbol{\rho}_{\boldsymbol{jkt}}^{(\boldsymbol{q-1})} - \delta \nabla \mathcal{G}_{jkt}^{(\boldsymbol{r})}(\boldsymbol{\rho}_{\boldsymbol{jkt}}^{(\boldsymbol{q-1})})$ by using a well-designed function $\mathcal{G}_{jkt}^{(\boldsymbol{r})}$ at each participating edge upon the loss functions $\{\mathcal{F}_{jkt}(\cdot), \forall j, k, t\}$, where $\boldsymbol{\rho}_{\boldsymbol{jkt}}^{(\boldsymbol{q})}$ is the model trained at the edge j after the q-th local update and δ is the step size of the local update, and (3) finally uploads the updated local model $\boldsymbol{w}_{\boldsymbol{t}}^{(\boldsymbol{r-1})} + \boldsymbol{\rho}_{\boldsymbol{jkt}}^{(\boldsymbol{q})}$, as well as the gradients $\{\nabla \mathcal{F}_{jkt}(\boldsymbol{w}_{\boldsymbol{t}}^{(\boldsymbol{r})}), \forall j, k\}$ to the cloud for the global aggregation. Here, $\boldsymbol{w}_{\boldsymbol{t}}^{(\boldsymbol{r})}$ refers to the aggregated model for next iteration.

We consider the convergence of the local model at each edge and the aggregated model in the cloud. Assuming $F_{jkt}(\cdot)$ is L-Lipschitz continuous and γ-strongly convex [39, 40], we write, $\forall t, k, j, q, r$, the following inequalities to capture convergence:

$$\mathcal{G}_{jkt}^{(\boldsymbol{r})}(\boldsymbol{\rho}_{\boldsymbol{jkt}}^{(\boldsymbol{q})}) - \mathcal{G}_{jkt}^{(\boldsymbol{r})*} \leq \eta_t [\mathcal{G}_{jkt}^{(\boldsymbol{r})}(\boldsymbol{\rho}_{\boldsymbol{jkt}}^{(\boldsymbol{0})}) - \mathcal{G}_{jkt}^{(\boldsymbol{r})*}],$$
$$\mathcal{F}_t(\boldsymbol{w}_{\boldsymbol{t}}^{(\boldsymbol{r})}) - \mathcal{F}_t^* \leq \varepsilon_0 [\mathcal{F}_t(\boldsymbol{w}_{\boldsymbol{t}}^{(\boldsymbol{0})}) - \mathcal{F}_t^*],$$

where η_t is the "convergence accuracy" of the local model at the edge j; ε_0 is the "convergence accuracy" of the global model in the global aggregation r; $\mathcal{G}_{jkt}^{(\boldsymbol{r})}(\cdot)$ and $\mathcal{F}_t(\cdot)$ are the local function for updates and the global loss function for all the data, respectively; $\mathcal{G}_{jkt}^{(\boldsymbol{r})*}$ and $\mathcal{F}_t^*$ are the local optimum and the global optimum, respectively. That is, in each epoch t, to achieve the desired η_t and ε_0, we need to conduct q local updates and r global aggregations [39, 41] which satisfy

$$q \geq \frac{2}{(2-L\delta)\delta\gamma} log_2(\tfrac{1}{\eta_t}) \triangleq q_0 log_2(1/\eta_t),$$
$$r \geq \frac{2L^2}{\gamma^2 \xi_1} ln(\tfrac{1}{\varepsilon_0}) \tfrac{1}{1-\eta_t} \triangleq r_0 ln(\tfrac{1}{\varepsilon_0}) \tfrac{1}{1-\eta_t},$$

where ξ_1 is the constant contained in $\mathcal{G}_{jkt}^{(\boldsymbol{r})}(\cdot)$; q_0 and r_0 are defined for simplicity (i.e., $q_0 = \frac{2}{(2-L\delta)\delta\gamma}$, and $r_0 = \frac{2L^2}{\gamma^2\xi_1}$).

Control Decisions: We introduce our control decisions. We use $x_{ijkt} \in [0,1]$, $\forall i \in \mathcal{U}_{kt}$, $\forall j \in \mathcal{E}_k$, $\forall k \in \mathcal{K}$, $\forall t \in \mathcal{T}$ to denote the ratio of the data transferred from user device i to edge j of network k in epoch t. We use $y_{jkt} \in \{1, 0\}$, $\forall j \in \mathcal{E}_k$, $\forall k \in \mathcal{K}$, $\forall t \in \mathcal{T}$ to denote edge provisioning (i.e., whether to activate edge j of network k in epoch t). We use $\eta_t \in (0,1)$, $\forall t \in \mathcal{T}$ to denote the local accuracy of federated learning in epoch t. Note that by controlling η_t, we can control the numbers of both local updates and global aggregations.

Cost of Federated Learning: Federated learning consists of data transference from devices to edges and model training between edges and the cloud. Using d_{ikt} to denote the total volume of data generated at device i of network k in epoch t and a_{ijkt} to denote the cost of transferring a single unit of data from device i to edge j of network k in epoch t, the cost of data transference is then $\sum_{t,k,i,j} x_{ijkt} d_{ikt} a_{ijkt}$. The cost of model training consists of three components: transmission cost between edges and the cloud, computation cost in the cloud, and computation cost at edges. The transmission cost of models between edges and the cloud is $\frac{r_0 ln(1/\varepsilon_0)}{1-\eta_t} \sum_j y_{jkt} m_t b_{jkt}$, where m_t is the size of the model trained by federated learning in epoch t and b_{jkt} is the cost of transferring a single unit of data from edge j of network k to the cloud in epoch t. Only the activated edges participate in the federated learning process. The computation cost for model aggregation in the cloud is $c_t \frac{r_0 ln(1/\varepsilon_0)}{1-\eta_t} \sum_j y_{jkt} m_t$, where c_t is the cost of processing a single unit of data in the cloud in epoch t. The computational cost for model updates at edges is $\sum_{t,k,j} z_{jkt}$, where

$$\forall j, t : z_{jkt} \triangleq p_{jkt} \cdot \{\textstyle\sum_i x_{ijkt}\} \cdot q_0 log_2(\tfrac{1}{\eta_t}) \cdot \tfrac{r_0 ln(\frac{1}{\varepsilon_0})}{1-\eta_t};$$

z_{jkt} is the cost for model updates at edge j of network k in epoch t; and, p_{jkt} is the cost of performing a single update to the local model at edge j of network k in epoch t. That is, the cost of model updates equals to the cost of a single update times the total number of model updates.

Cost of Edge Provisioning: Maintaining the running edges incurs operational cost, (e.g., the electricity consumption, the carbon footprint, and various license fees for hardware and/or software). We denote by o_{jkt} such operational cost of edge j of network k in epoch t. Then, the operational cost of edge j in epoch t is $o_{jkt} y_{jkt}$. The operational cost is only incurred when an edge is activated. Meanwhile, in this paper, we allow edges to be switched on and off dynamically—toggling edges incurs the "switching cost", (e.g., the initialization time of booting resources and loading profiles/configurations, and the hardware wear-and-tear). We represent by s_{jkt} the switching cost regarding edge j of network k in epoch t, and define the corresponding switching cost as $s_{jkt}[y_{jkt} - y_{jkt-1}]^+$, where $[\cdot]^+ = \max\{\cdot, 0\}$, via linking two consecutive epoches.

B. Problem Formulations and Challenges

Control Problem $\mathbb{P}$: With the above system models, we formulate the following optimization problem to control federated learning upon the cloud-edge infrastructure:

$$\min \ \mathcal{P} = \sum_{t,k} \{ \frac{r_0 ln(1/\varepsilon_0)}{1-\eta_t} \sum_j y_{jkt} m_t \{b_{jkt} + c_t\} + \sum_j z_{jkt}$$
$$+ \sum_j \{ \sum_i x_{ijkt} d_{ikt} a_{ijkt} + s_{jkt}[y_{jkt} - y_{jkt-1}]^+ + o_{jkt} y_{jkt} \} \}$$
$$s.t. \sum_{i \in \mathcal{U}_{kt}} x_{ijkt} \leq y_{jkt} \sum_{i \in \mathcal{U}_{kt}} d_{ikt}, \forall j \in \mathcal{E}_k, k \in \mathcal{K}, t \in \mathcal{T}, \quad (1)$$
$$\sum_{j \in \mathcal{E}_k} x_{ijkt} \geq 1, \forall i \in \mathcal{U}_{kt}, k \in \mathcal{K}, t \in \mathcal{T}, \quad (2)$$
$$var. \ x_{ijkt} \in [0,1], y_{jkt} \in \{0,1\}, \eta_t \in (0,1). \quad (3)$$

The objective is to minimize the long-term total cost of federated learning and edge provisioning. Constraint (1) ensures that the data generated on user devices can only be transferred to activated edges in the same edge network. Constraint (2) ensures that all of the data on each device are transferred. Constraint (3) specifies the variables' domains.

Control Problem $\widehat{\mathbb{P}}$ with Predicted Inputs: We highlight the fact that the dynamic inputs to the problem in each epoch are often only revealed after the control decisions for the epoch are made. That is, we have to solve the problem by using the "predicted" inputs, rather than the "actual" inputs (and in this paper, we will design a dedicated, learning-based algorithm to produce such predictions). Denoting by $\widehat{a}_{ijkt}$ and $\widehat{b}_{jkt}$ the predicted cost for transferring a single unit of data between device i and edge j, and the predicted cost for transferring a

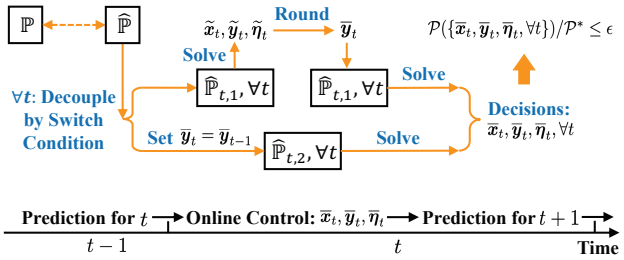


Fig. 2: Design of our predictive online schema

single unit of data between edge j and the cloud, respectively, we formulate the problem $\widehat{\mathbb{P}}$ with the predicted inputs:

$$\min \ \widehat{\mathcal{P}} = \sum_{t,k} \{ \frac{r_0 ln(1/\varepsilon_0)}{1-\eta_t} \sum_j y_{jkt} m_t \{\widehat{b}_{jkt} + c_t\} + \sum_j z_{jkt} + \sum_j \{ \sum_i x_{ijkt} d_{ikt} \widehat{a}_{ijkt} + s_{jkt}[y_{jkt} - y_{jkt-1}]^+ + o_{jkt} y_{jkt} \} \}$$
$$s.t. \ \ Constraints \ (1) \ to \ (3).$$

Algorithmic Goal: Having two problems formulated above, we now describe the goal of our algorithm design. Towards this end, we firstly introduce some concise notations: $\boldsymbol{I}$ refers to the aggregation[2] of all the variables (i.e., $\{\boldsymbol{x}_t, \boldsymbol{y}_t, \eta_t, \forall t\}$); $\boldsymbol{I}^*$ refers to the optimal solution of $\mathbb{P}$ solved in an offline manner (i.e., assuming an oracle's perspective where all the actual inputs are known at once in advance), and we also write $\mathcal{P}^* = \mathcal{P}(\boldsymbol{I}^*)$; and, $\bar{\boldsymbol{I}}_\wedge$ refers to a feasible solution solved from $\widehat{\mathbb{P}}$ in an online manner. *Our goal is to design algorithms which, in an online manner, produce predicted inputs and also produce a solution $\bar{\boldsymbol{I}}_\wedge$, while upper-bounding the competitive ratio as the performance metric defined as follows: $r = \mathcal{P}(\bar{\boldsymbol{I}}_\wedge)/\mathcal{P}^*$.*

Problem Challenges: It is yet non-trivial to design algorithms to achieve our goal above, due to multiple challenges.

First and foremost, the solution solved from the problem with the predicted inputs $\widehat{\mathbb{P}}$ needs to have bounded performance when evaluated in the problem with the actual inputs $\mathbb{P}$ and compared to the offline optimum of the latter. This poses harsh requirements for the design of both the online prediction algorithm (for producing the predicted inputs) and the online control algorithm (for making the control decisions), which need to work together to achieve the desired guarantee.

The second challenge is the online uncertainty. Even if the predicted inputs are produced, they are given on the fly and the problem $\widehat{\mathbb{P}}$ needs to be solved online, where the switching cost couples the decision of the current epoch and that of the succeeding one. That is, without knowing the decision of the next epoch, it is hard to make a good decision for the current epoch to minimize the switching cost; yet, the decision for the next epoch will only be made as the next epoch arrives.

The third challenge lies in the intractability. The proposed problems are mixed-integer programs with multiple non-linear terms, (e.g., the transference cost between edge and cloud, and the computation cost at edge, besides the switching cost). Even without these non-linear terms, our problems can be proved to be NP-hard in an offline setting (as they contain the minimum knapsack problem as a special case), not to mention that we desire to solve them in an online manner.

[2]Bold symbols denote column vectors (e.g., $\boldsymbol{y}_t^\top = [..., y_{jkt}, ...]$).

III. Algorithm Design

We design three polynomial-time online algorithms. Our Algorithm 1 keeps the on/off status of the edges unchanged until necessary, controlled by a carefully-designed switch condition, overcoming the second challenge described above. Algorithm 1 determines the possible new status of the edges by relaxing and solving the one-shot problem for each current epoch and then invoking our Algorithm 2 to round the fractional decisions into integers without violating any contraints, overcoming the third challenge. Based on the current inputs and the current decisions made by both Algorithms 1 and 2, our Algorithm 3 predicts the inputs for the next epoch (which will used by Algorithms 1 and 2), via a novel online learning approach, overcoming the first challenge. Our design is shown in Fig. 2, where $\widehat{\mathbb{P}}_{t,1}$ and $\widehat{\mathbb{P}}_{t,2}$ will be defined next.

A. Online Control Algorithm

To facilitate the design of our online algorithms, we introduce some additional notations:

$$\widehat{C}^t_{\neg S} \triangleq \textstyle\sum_{k,j} \{ \sum_i x_{ijkt} \alpha_{ijkt} + y_{jkt} \{ \frac{\beta_{jkt}}{1-\eta_t} + o_{jkt} \} + z_{jkt} \},$$
$$C^t_S \triangleq \textstyle\sum_{k,j} s_{jkt} [y_{jkt} - y_{jkt-1}]^+,$$

where $\alpha_{ijkt} = d_{ikt}\widehat{a}_{ijkt}$ and $\beta_{jkt} = r_0 ln(\frac{1}{\varepsilon_0}) m_t \{\widehat{b}_{jkt} + c_t\}$. Then, we can have two auxiliary problems[3]:

$$\begin{aligned} \min \ & \widehat{\mathbb{P}}_{t,1} = \widehat{C}^t_{\neg S}(\boldsymbol{x}_t, \boldsymbol{y}_t, \eta_t) \\ s.t. \ & C^t_S(\boldsymbol{y}_t, \bar{\boldsymbol{y}}_{t-1}) \leq \zeta_1 \widehat{C}^t_{\neg S}(\boldsymbol{x}_t, \boldsymbol{y}_t, \eta_t), \\ & Constraints \ (1), \ (2), \\ var. \ & x_{ijkt} \in [0,1], \eta_t \in (0,1), y_{jkt} \in [0,1], \end{aligned} \tag{4}$$

where Constraint (4) enforces a specified relationship between the switching cost and the non-switching cost; We also have

$$\begin{aligned} \min \ & \widehat{\mathbb{P}}_{t,2} = \widehat{C}^t_{\neg S}(\boldsymbol{x}_t, \bar{\boldsymbol{y}}_t, \eta_t) \\ s.t. \ & Constraints \ (1), \ (2), \\ var. \ & x_{ijkt} \in [0,1], \eta_t \in (0,1), \end{aligned}$$

where $\bar{\boldsymbol{y}}_t$ is required as inputs for $\widehat{\mathbb{P}}_{t,2}$. And for solving $\widehat{\mathbb{P}}_{t,1}$, the following substitution can be used:

$$s_{jkt}[u - y_{jkt-1}]^+ \Leftrightarrow s_{jkt} v_{jkt}, \ s.t. \begin{cases} v_{jkt} \geq u - y_{jkt-1} \\ v_{jkt} \geq 0, \end{cases}.$$

Given $\bar{\boldsymbol{y}}_{t-1}$ for $\widehat{\mathbb{P}}_{t,1}$ and given $\bar{\boldsymbol{y}}_t, \bar{\boldsymbol{y}}_{t-1}$ for $\widehat{\mathbb{P}}_{t,2}$, solving $\widehat{\mathbb{P}}_{t,1}$ and $\widehat{\mathbb{P}}_{t,2}$ for the other real-valued variables is efficient with the help of existing standard optimization solvers [42]. Note that the objective function is twice differentiable.

Algorithm 1 postpones changing the current on/off status of the edges even if the one-shot optimum indicates so, until the

[3]The mark of $\widehat{\ }$ refers to predicted inputs, or problems taking predicted inputs. The mark of $\bar{\ }$ refers to the outputs of our proposed online approach. The mark of $\widetilde{\ }$ refers to the intermediate outputs that are all fractions.

Algorithm 1 Online Control Algorithm

1: Initialize $t = t' = 1$; $\bar{\boldsymbol{x}}_1, \bar{\eta}_1 \leftarrow \widehat{\mathbb{P}}_{t,2}$, given $\bar{\boldsymbol{y}}_0 = \bar{\boldsymbol{y}}_1 = \mathbf{0}$;
2: Initialize $\{\widehat{a}_{ijk1}, \widehat{b}_{jk1}\}$;
3: **while** $t \leq T$ **do**
4: **if** $\widehat{C}_S^{t'}(\bar{\boldsymbol{y}}_{t'}, \bar{\boldsymbol{y}}_{t'-1}) \leq \frac{1}{\zeta_2} \sum_{v=t'}^{t-1} \widehat{C}_{\neg S}^{v}(\bar{\boldsymbol{x}}_v, \bar{\boldsymbol{y}}_v, \bar{\eta}_v)$ **then**
5: Obtain $\widetilde{\boldsymbol{y}}_t$ from $\widehat{\mathbb{P}}_{t,1}$;
6: Invoke **Algorithm 2** for rounding $\widetilde{\boldsymbol{y}}_t$ to obtain $\bar{\boldsymbol{y}}_t$;
7: Given $\bar{\boldsymbol{y}}_t$, obtain $\bar{\boldsymbol{x}}_t, \bar{\eta}_t \leftarrow \widehat{\mathbb{P}}_{t,1}$ if feasible;
Otherwise set $\bar{\boldsymbol{y}}_t = \bar{\boldsymbol{y}}_{t-1}$;
8: **if** $\bar{\boldsymbol{y}}_t \neq \bar{\boldsymbol{y}}_{t-1}$ **then**
9: $t' = t$;
10: **end if**
11: **end if**
12: **if** $t' < t$ **then**
13: $\bar{\boldsymbol{x}}_t, \bar{\eta}_t \leftarrow \widehat{\mathbb{P}}_{t,2}$, given $\bar{\boldsymbol{y}}_t = \bar{\boldsymbol{y}}_{t-1}$;
14: **end if**
15: $t = t + 1$;
16: Invoke **Algorithm 3** for predicting $\{\widehat{a}_{ijkt+1}, \widehat{b}_{jkt+1}\}$;
17: **end while**

Algorithm 2 Randomized Pairwise Rounding Algorithm

1: **for** $k \in \mathcal{K}$ **do**
2: $\psi_k = \{j \mid \widetilde{y}_{jkt} \in (0,1)\}$;
3: **while** $|\psi_k| \geq 2$ **do**
4: Select $u, v \in \psi$, where $u \neq v$;
5: $\theta_1 = \min\{1 - \widetilde{y}_{ukt}, \widetilde{y}_{vkt}\}, \theta_2 = \min\{\widetilde{y}_{ukt}, 1 - \widetilde{y}_{vkt}\}$;
6: With the probability of $\frac{\theta_2}{\theta_1 + \theta_2}$,
set $\widetilde{y}_{ukt} = \widetilde{y}_{ukt} + \theta_1$ and $\widetilde{y}_{vkt} = \widetilde{y}_{vkt} - \theta_1$;
Otherwise, set $\widetilde{y}_{ukt} = \widetilde{y}_{ukt} - \theta_2$ and $\widetilde{y}_{vkt} = \widetilde{y}_{vkt} + \theta_2$;
7: $\psi_k = \psi_k \backslash \{u\}$ and $\bar{y}_{ukt} = \widetilde{y}_{ukt}$, if $\widetilde{y}_{ukt} \in \{0, 1\}$;
8: $\psi_k = \psi_k \backslash \{v\}$ and $\bar{y}_{vkt} = \widetilde{y}_{vkt}$, if $\widetilde{y}_{vkt} \in \{0, 1\}$;
9: **end while**
10: **if** $|\psi_k| = 1$ **then**
11: $\bar{y}_{ukt} = 1, u \in \psi_k$;
12: **end if**
13: **end for**

cumulative non-switching cost has significantly exceeded the potential switching cost. Specifically, the switch condition, as in Line 4, is controlled by the "laziness" parameter ζ_2. Note that ζ_1 is in contrast used in the one-shot subproblem $\widehat{\mathbb{P}}_{t,1}$. When the switch condition is satisfied, fractions solved from $\widehat{\mathbb{P}}_{t,1}$ in Line 5 need to be rounded by our proposed Algorithm 2, as in Line 6. Afterwards, the rounded, integral decision $\bar{\boldsymbol{y}}_t$ is used as part of inputs for solving $\widehat{\mathbb{P}}_{t,1}$ (if feasible) to obtain the fractional data transference and local accuracy. Note that applying the rounded decisions obtained from our rounding algorithm may violate the constraints of $\widehat{\mathbb{P}}_{t,1}$. Therefore, only when $\widehat{\mathbb{P}}_{t,1}$ is feasible under such rounded decisions, they are actually adopted by Algorithm 1, as in Line 7 to 11; otherwise, we continue to postpone switching edges and solve the data transfer and the local accuracy decisions from $\widehat{\mathbb{P}}_{t,2}$ in Line 13.

Algorithm 2 converts the fractional edge control decisions to the integers of either 0 (i.e., edge off) or 1 (i.e., edge on). Algorithm 2 maintains the sets $\{\psi_k\}$, tracking the indices for the edge control decisions in network k, as in Line 2. Algorithm 2 iteratively chooses a pair of fractions to round at least one of them into an integer in a randomized manner, while ensuring that the sum of the two values stay unchanged after rounding, and that the expectation of each randomized integer equals its corresponding fractional value before rounding (i.e., Line 3 through 9). The complexity of the "while" loop reaches $O(\sum_k |\mathcal{E}_k|^2)$ [43]. Since the sum of all the fractions before rounding may not be an integer, there may be one element left in ψ_k after the "while" loop. If so, we have to round it to 1 (i.e., Line 10 to 12) in order not to violate any constraint.

Algorithm 3 Learning-Based Prediction Algorithm

// To predict $\widehat{a}_{ijkt+1}$, let $\sigma_t = a_{ijkt}$, $g_t = d_{ikt}\bar{x}_{ijkt}$;
// To predict $\widehat{b}_{jkt+1}$, let $\sigma_t = b_{jkt}$, $g_t = \frac{r_0 ln(1/\varepsilon_0) m_t \bar{y}_{jkt}}{1 - \bar{\eta}_t}$;
1: Initialize a proper step size λ;
2: **for** $t = 1, 2, ..., T$ **do**
3: σ_t and g_t are revealed; Construct $f_t(\sigma) = (\sigma_t - \sigma)^2 g_t^2$;
4: $\widehat{\sigma}_{t+1} = arg\min_\sigma \{\nabla f_t(\widehat{\sigma}_t)(\sigma - \widehat{\sigma}_t) + \frac{||\sigma - \widehat{\sigma}_t||^2}{2\lambda}\}$;
5: **end for**

B. Learning-Based Prediction Algorithm

Algorithm 3 is invoked at the end of each current epoch to predict the inputs of the next epoch. Line 4 is an approximation linking the inputs of the two consecutive epochs based on an alternating "primal-dual" approach, where the function f_t for epoch t is to facilitate our performance analysis later.

Specifically, to measure the "distance" between the predicted value and the actual value in the epoch $t+1$, we adopt the L_2-norm (i.e., $||\sigma_{t+1} - \widehat{\sigma}_{t+1}||^2$). Then, we minimize such norm (i.e., $\widehat{\sigma}_{t+1} = arg\min_\sigma ||\sigma_{t+1} - \sigma||^2 g_{t+1}^2$). However, in the online scenario, the actual inputs for the current epoch cannot be revealed before the decisions for the current epoch are made. Traditionally, in order to minimize the cumulative distance between the predicted and actual values, solving the convex problem of $min \sum_t f_t(\sigma_t), s.t.\ h_t(\sigma_t) \leq 0$, is exactly equivalent to solving the convex-concave problem of $min_{\sigma_t} max_{\varpi_t} \sum_t \big(f_t(\sigma_t) + \varpi_t h_t(\sigma_t)\big)$, where $h_t(\sigma_t)$ is the constraint respect to σ_t and ϖ_t is the Lagrange multiplier. To solve it in an online manner, intuitively, the gradient incurred from the epoch t can be used as a guidance to rectify the predicted value for the epoch $t+1$. Therefore, we can alternate between minimizing the objective with respect to the primal variable σ_{t+1} via a modified descent step and maximizing the objective with respect to the Lagrange multiplier via a dual ascent step. That is, the modified primal step is as follows:

$$min_{\sigma_{t+1}} \nabla f_t(\sigma_t)(\sigma_{t+1} - \sigma_t) + \varpi_{t+1} h_t(\sigma_{t+1}) + \frac{||\sigma_{t+1} - \sigma_t||^2}{2\lambda},$$

while the Lagrange multiplier ϖ_{t+1} is updated as follows:

$$\varpi_{t+1} = [\varpi_t + \lambda' h_t(\sigma_t)]^+,$$

where λ and λ' are the parameters. Since the prediction has no constraint with respect to σ_t, we can adopt the primal step with $\varpi_t = \lambda' = 0, \forall t$. Actually, the gradient-based approach

in the primal step is the approximation of f_{t+1} by linking two consecutive epochs. Thus, we predict $\widehat{\sigma}_{t+1}$ via

$$arg\min_{\sigma}\{\nabla\{||\sigma_t-\sigma'||^2 g_t^2\}|_{\sigma'=\widehat{\sigma}_t}\cdot(\sigma-\widehat{\sigma}_t)+\frac{||\sigma-\widehat{\sigma}_t||^2}{2\lambda}\},$$

where λ is the parameter for approximation.

IV. Performance Analysis

Our goal is to bound $E[\mathcal{P}(\bar{\boldsymbol{I}}_\wedge)]/\mathcal{P}^*$, where $\mathcal{P}^*=\mathcal{P}(\boldsymbol{I}^*)$, as we have introduced randomized rounding into our algorithms. In fact, we can prove the following chain of derivations:

$$\frac{E[\mathcal{P}(\bar{\boldsymbol{I}}_\wedge)]}{\mathcal{P}^*}=\frac{E[\mathcal{P}(\bar{\boldsymbol{I}}_\wedge)]}{\widehat{\mathcal{P}}(\widetilde{\boldsymbol{I}}_\wedge)}\cdot(1+\frac{\widehat{\mathcal{P}}(\widetilde{\boldsymbol{I}}_\wedge)-\mathcal{P}(\widetilde{\boldsymbol{I}}^*)+\mathcal{P}(\widetilde{\boldsymbol{I}}^*)-\mathcal{P}^*}{\mathcal{P}^*}) \quad (5)$$

$$\leq\frac{E[\widehat{\mathcal{P}}(\bar{\boldsymbol{I}}_\wedge)]+\mathcal{O}\{T^{\epsilon_1}\}}{\widehat{\mathcal{P}}(\widetilde{\boldsymbol{I}}_\wedge)}\cdot(1+\frac{\epsilon_2\widehat{\mathcal{P}}(\widetilde{\boldsymbol{I}}^*_\wedge)-\mathcal{P}(\widetilde{\boldsymbol{I}}^*)}{\mathcal{P}^*}) \quad (6)$$

$$\leq\frac{\epsilon_3\widehat{\mathcal{P}}(\widetilde{\boldsymbol{I}}_\wedge)+\mathcal{O}\{T^{\epsilon_1}\}}{\widehat{\mathcal{P}}(\widetilde{\boldsymbol{I}}_\wedge)}\cdot(1+\frac{\epsilon_2\mathcal{P}(\widetilde{\boldsymbol{I}}^*)+\mathcal{O}\{T^{\epsilon_1}\}-\mathcal{P}(\widetilde{\boldsymbol{I}}^*)}{\mathcal{P}^*}) \quad (7)$$

$$\leq(\epsilon_3+\frac{\mathcal{O}\{T^{\epsilon_1}\}}{\widehat{\mathcal{P}}^*})(\epsilon_2+\frac{\mathcal{O}\{T^{\epsilon_1}\}}{\mathcal{P}^*})=\epsilon_2\epsilon_3+\mathcal{O}\{T^{2(\epsilon_1-1)}\}. \quad (8)$$

In order to compare the cost of our online approach using the predicted inputs against the offline optimum using the actual inputs, we split the competitive ratio as the product of two parts. The first part links the cost of our online approach with the cost using predicted inputs and the fractional decisions (i.e., $E[\mathcal{P}(\bar{\boldsymbol{I}}_\wedge)]$ and $\widehat{\mathcal{P}}(\widetilde{\boldsymbol{I}}_\wedge)$). The second part further links the cost using predicted inputs and the fractional decisions with the offline optimum (i.e., $\widehat{\mathcal{P}}(\widetilde{\boldsymbol{I}}_\wedge)$ and $\mathcal{P}^*$).

In the above, from (5) to (6), we leverage Lemmas 1 and 2, and $\mathcal{P}(\widetilde{\boldsymbol{I}}^*)\leq\mathcal{P}^*$. From (6) to (7), we leverage Lemmas 1 and 3. From (7) to (8), we leverage $\widehat{\mathcal{P}}(\widetilde{\boldsymbol{I}}_\wedge)\geq\widehat{\mathcal{P}}^*$, $\widehat{\mathcal{P}}^*\geq\mathcal{O}\{T\}$, and $\mathcal{P}^*\geq\mathcal{O}\{T\}$, where $\mathcal{O}\{T\}$ serves as the lower bound. $\boldsymbol{I}^*$ refers to the optimal solution of $\mathbb{P}$ solved in an offline manner; $\widetilde{\boldsymbol{I}}^*$ refers to to the optimal solution of $\mathbb{P}$ solved offline when $\boldsymbol{y}_t,\forall t$ are in the real domain. Analogously, $\widetilde{\boldsymbol{I}}^*_\wedge$ is the optimal solution of $\widehat{\mathbb{P}}$ (i.e., with the predicted inputs), solved offline in the real domain. $\widetilde{\boldsymbol{I}}_\wedge$ and $\bar{\boldsymbol{I}}_\wedge$ are the solutions produced by our proposed algorithms before and after rounding, respectively. $\epsilon_1<1$, ϵ_2, ϵ_3 are constants, described as follows.

Before we present our lemmas and theorem, we make the following assumptions to facilitate our theoretical analysis, which are very common and easy to be satisfied:

Assumption 1: $\forall t$, $f_t(\cdot)$ in Algorithm 3 has bounded gradient (i.e., $||\nabla f_t(\cdot)||\leq F$), where F is a constant.

Assumption 2: The costs for transference a_{ijkt} and b_{jkt} are both bounded (i.e., $a_{ijkt}\in[0,a_{max}], b_{jkt}\in[0,b_{max}]$).

We now present our Lemmas 1, 2, and 3, and our Theorem 1. Lemma 1 quantifies the cumulative prediction error (i.e., the cumulative "distance" between the predicted inputs and the actual inputs). Then, Lemma 2 characterizes the competitive ratio of the fractional solutions solved based on the predicted inputs. Lemma 3 characterizes the integrality gap incurred when rounding fractional solutions into integers, based on the predicted inputs. Finally, Theorem 1 combines all the lemmas and demonstrates the competitive ratio of our entire approach composed of the three algorithms.

Lemma 1. *The cumulative prediction error incurred by Algorithm 3 grows only sub-linearly when the step size is chosen as* $\lambda=\mathcal{O}\{\sqrt{\frac{V(\{\sigma_t\})}{T}}\}$, $\sum_{t=1}^{T}(\sigma_t-\widehat{\sigma}_t)g_t=\mathcal{O}(\sqrt[4]{V(\{\sigma_t\})}T^{\frac{3}{4}})=\mathcal{O}(T^{\epsilon_1})$, *where* $\epsilon_1<1$ *and* $V(\{\sigma_t\})=\sum_{t=1}^{T}||\sigma_{t+1}-\sigma_t||$.

Proof. See Appendix A. □

Lemma 2. *Using predicted inputs, without the rounding part, Algorithm 1 is* ϵ_2*-competitive (i.e.,* $\widehat{\mathcal{P}}(\widetilde{\boldsymbol{I}}_\wedge)\leq\epsilon_2\widehat{\mathcal{P}}(\widetilde{\boldsymbol{I}}^*_\wedge)$*), where* $\epsilon_2\triangleq\epsilon_2'(1+\max\{\zeta_1,1/\zeta_2\})$ *and* ϵ_2' *are constants.*

Proof. See Appendix B. □

Lemma 3. *Using predicted inputs, Algorithm 1, by invoking Algorithm 2, produces the integral solution and the fractional solution that satisfy* $E[\widehat{\mathcal{P}}(\bar{\boldsymbol{I}}_\wedge)]\leq\epsilon_3\widehat{\mathcal{P}}(\widetilde{\boldsymbol{I}}_\wedge)$, *where* $\epsilon_3=(1+1/\zeta_1)\max\{\kappa_1,\kappa_2,\kappa_3,\kappa_4\}$ *is a constant, and* $\kappa_1\sim\kappa_4$ *are the corresponding constants for the four non-switching terms.*

Proof. See Appendix C. □

Theorem 1. *The competitive ratio of our entire predictive online approach is* $E[\mathcal{P}(\bar{\boldsymbol{I}}_\wedge)]/\mathcal{P}(\boldsymbol{I}^*)\leq\epsilon_2\epsilon_3+\mathcal{O}\{T^{2(\epsilon_1-1)}\}$.

Proof. See details in Appendix D. The proof is based on joining Lemmas 1, 2, and 3. Note that $2(\epsilon_1-1)<0$. □

V. Experimental Study

A. Data and Settings

We summarize the real-world traces used in Table II.

Federated Learning Workload and Models: We use the MNIST [35] dataset, which contains 70k gray-scale images of handwritten digits (60k for training and 10k for testing), to conduct real-world federated learning. Our implementation contains 4k lines of python codes. More specifically, we train the least-squares Support-Vector Machine (SVM) model with the loss of $\varrho||\boldsymbol{w}||^2/2+(\max\{0,1-\boldsymbol{w}^\top\boldsymbol{D}\})^2/2$, where $\boldsymbol{D}$ is the feature vector and the step size $\varrho=0.01$, outputting a binary label to imply the digit is even or odd. We also train the Convolutional Neural Network (CNN) model [18] with 9 layers, including convolutional, max pooling, local response normalization, fully connected, and softmax layers. We use CNN as we are interested in knowing the performance of our proposed algorithms for non-convex loss functions. All our evaluation results are produced using 15 virtual machines on 4 servers equipped with Geforce RTX 2080Ti, including Dell PowerEdge R740 and Inspur SN5160M4.

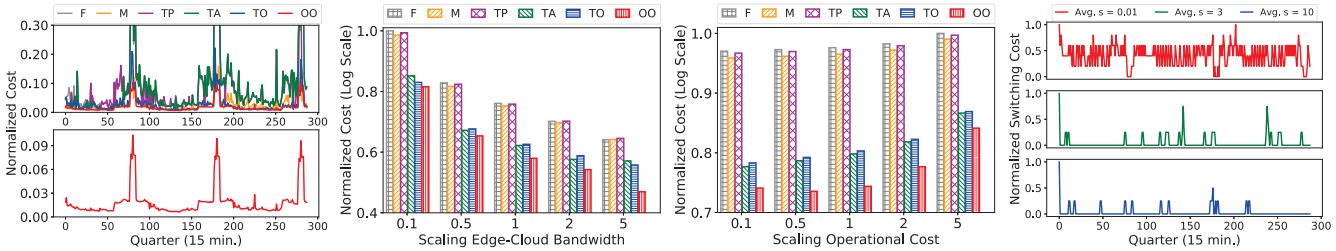
User Devices, Cloud, and Edges: The time-varying numbers of available user devices are from Google [1], which are

TABLE II: Input Traces

Content	Description
Training Dataset	MNIST, 70k images for training and testing [35]
Device Number	Available devices from Google [1]
Device-Edge Bandwidth	Six patterns of cellular bandwidth traces [9]
Edge-Cloud Bandwidth	Bandwidth of over 80 edges from 4 ISPs [36]
Electricity Price	Three-day market data from EPEX SPOT [44]

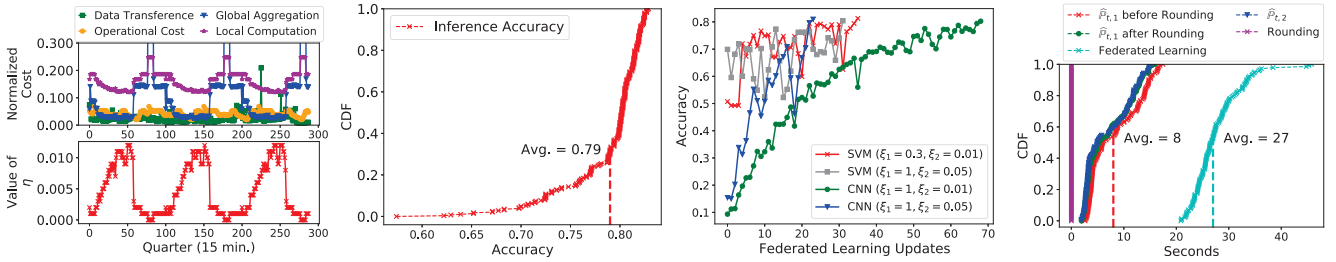
TABLE III: Costs of Different Algorithms

Algorithm	F	TP	TA	TO	M	OP
Normalized Cost	0.719	0.667	0.704	0.209	0.218	0.174
Algorithm	F-2	TP-2	TA-2	TO-2	M-2	OA
Normalized Cost	0.798	0.832	0.882	0.378	0.387	0.176
Algorithm	F-3	TP-3	TA-3	TO-3	M-3	OO
Normalized Cost	0.896	0.957	1.000	0.539	0.534	0.144



(a) Total cost per epoch (b) Impact of bandwidth (c) Impact of operational cost (d) Impact of switching cost

Fig. 3: Cost comparison for different approaches



(a) Cost components per epoch (b) Inference accuracy (c) Convergence speed (d) Execution time

Fig. 4: Further results of our proposed approach

for 3 days in 2018 with a peak number of nearly 6000 devices, measured for every quarter (15 minutes). The dynamic WAN bandwidth data are from Aliyun [36], which contain 87 edges within 4 Internet Service Providers (ISPs), and the bandwidth is measured by downloading files from the cloud. The dynamic device-edge bandwidth data [9] include 6 patterns of cellular bandwidth changes. The dynamic operational cost is from EPEX SPOT [44], which contains wholesale electricity prices from July 17 through 19, 2020. Regarding the unit switching cost, we vary it as $0 \sim 5$, in order to demonstrate a spectrum of evaluation results. The parameters used for the convergence of federated learning are derived from [40] and [18]: $\varepsilon_0 = 10^{-3}$, $\xi_1 = 0.3$, $\xi_2 = 0.01$, $r_0 = 15$, $q_0 = 4$ and $\delta = 1/6$. Finally, the parameters $\zeta_1 = 0.5$ and $\zeta_2 = 2$ are derived from [23].

Algorithms and Metrics: We experiment with different combinations of algorithms for comparison.[6] The online control algorithms we consider are as follows:

- $F*$ uses a fixed set of randomly chosen edges as the active ones in each edge network over the entire time horizon;
- $T*$ uses a fixed number of edges with the least cost of data transference and model aggregation per epoch;
- $M*$ uses a fixed number of edges with the least non-switching cost in each edge network for each epoch;
- $O*$ refers to our proposed online control algorithm.

The online prediction algorithms we consider are as follows:

- $*P$ refers to prediction based on inputs of previous epoch;
- $*A$ refers to prediction based on the average inputs so far;
- $*O$ refers to our proposed online prediction algorithm.

All these algorithms are executed and compared in the online scenario, where they have no access to the actual inputs of each epoch before making the control decisions for each epoch. Our primary performance metric is the total cumulative cost for federated learning and edge provisioning; we also investigate the inference accuracy of the trained models, the convergence speed of the training process, and the execution time of our proposed algorithms. Solving $\widehat{\mathbb{P}}_{t,1}$ and $\widehat{\mathbb{P}}_{t,2}$ is conducted using standard optimization tools (i.e., AMPL [45] and IPOPT [42]), whose results may contain a certain amount of error for non-convex problems. Note that, even with state-of-the-art integer program solvers, it will take unacceptably long time to obtain the offline optimums, so we do not consider them here.

B. Evaluation Results

Table III lists the costs of different algorithms. Our proposed approach OO performs the best, and incurs only 14.4% total cost compared to the maximum which is incurred by TA-3. Since a larger number of switched-on edges results in much more cost on the global aggregation of federated learning and also more operational expense, the results with larger numbers of edges (i.e., more than 3) for the algorithms except OO are omitted. In general, OO reduces at least 37% cost compared with other strategies for federated learning.

Fig. 3(a) plots the time-varying costs of the different algorithms that have the least costs in Table III. Our OO approach always keeps the low cost on the fly. Fig. 3(b) and Fig. 3(c) exhibit how the edge-cloud bandwidth and the operational cost (i.e., the electricity expense in this paper) impact the total cost, respectively, where the horizontal axis reflects the scale with respect to the original corresponding trace. As in Fig. 3(b), with the growth of the edge-cloud bandwidth, the difference incurred by the global model aggregation becomes higher across all algorithms. OO optimizes the global model aggregation over WAN, and elongates the reduction on the total cost, especially when the global aggregation takes higher proportion of the total cost. The average reduction on the

[6]The mark of $*$ is a wildcard to match different online control algorithms with different online prediction algorithms. The notation "$-n$" in Table III refers to the number of edges in an edge network for those algorithms apart from "$O*$", and the default value of n is 1.

total cost is 46%, while the maximum reduction is 1.18 $\times$. In Fig. 3(c), the average reduction on the total cost is 37.4% while the maximum reduction is 45%. Fig. 3(d) depicts how the unit switching cost impacts the result of our approach. When the unit cost for switching an edge is small, our approach prefers to frequently switch on/off edges for lower total cost; as it grows, the frequency of switching on/off edges drops.

Fig. 4(a) illustrates the details of the time-varying costs of the different components of our approach, as well as η_t. With the growth of the cost in terms of the global model aggregation, η_t becomes larger for a smaller number of global aggregations, balancing the number of global aggregations and that of local updates. Fig. 4(b) illustrates the cumulative distribution of the inference accuracy of our trained models on the testing dataset of MNIST (i.e., the 10k images). The average inference accuracy is 0.79, which is acceptable and aligned with existing literatures [7, 23]. Fig. 4(c) visualizes the relationship between the model's global accuracy and the training parameters defined in $\mathcal{G}_{jkt}^{(r)}(\cdot)$. The aggregated model of SVM converges to the desired accuracy stably when the parameters are small. Those CNN models are also converging to the desired accuracy despite with non-convex loss functions. Fig. 4(d) is on the execution time of our approach. It only takes several seconds on average for our algorithm to complete in each epoch, solving the problems with up to thousands of devices and tens of edges in our evaluations.

VI. Related Work

We summarize prior research in three categories, and highlight their drawbacks compared to our work, respectively.

Federated Learning Optimization: Besides the works [6, 8, 15–17] that only focused on the various local optimizers and the convergence of federated learning, Wang *et al.* [18] controlled the frequency of global model aggregations under a given resource budget. Tran *et al.* [19] optimized the training time and the energy consumption for mobile devices. Yang *et al.* [20] studied the scheduling of federated learning over wireless networks. Tu *et al.* [21] designed the network-aware optimization of federated learning for fog computing. Zhou *et al.* [22] controlled the throughout of data training for cost-efficient federated learning at the edge of the network.

These works have considered resource usage and optimization of federated learning. But they largely ignore the flexibility of managing distributed edges regarding the switching cost, and the unavailability of the online dynamic inputs.

Online Edge Provisioning: Zhang *et al.* [23] designed an online cost-minimizing approach for data migration. Xu *et al.* [24] proposed an online service caching and offloading schema in dense networks. Jiao *et al.* [25] explored related resource provisioning at edges under multi-granularity settings. Meng *et al.* [26] investigated online deadline-aware task dispatching and scheduling in edge computing. Zhou *et al.* [27] provisioned cloud-edge resources online for IoT.

These works focus on online service provisioning as well as edge management, but rarely consider the data and resource patterns and the convergence-preserving training of federated learning; they do not often adopt prediction algorithms and use predicted inputs to achieve overall performance guarantees.

Online Predictive Control: A substantial body of research focused on the predictive control in various scenarios. Wang *et al* [28] mapped the edge computing services onto the underlying physical network through a reinforcement learning approach. Tao *et al* [30] proposed a novel adaptive user-managed service placement mechanism by using online learning tools. Liao *et al* [31] used the upper confidence bound algorithm to dynamically select the channels for task delivery. Zhang *et al* [32] investigated the task offloading and resource allocation problems at edges via Lyapunov optimization.

These works use learning-based approaches to conduct predictive control for various scenarios and problems. However, none of them focuses on federated learning and its convergence over cloud-edge infrastructures, whose characteristics and features make such existing research inapplicable.

VII. Conclusion

Managing geo-distributed cloud-edge networks to support convergence-preserving federated learning is a challenging problem. In this paper, we formulate this problem by considering data transference, edge management, and the federated learning process. We build a non-linear integer program for long-term total cost minimization, and design algorithms that use the predicted inputs of the dynamic environments to make control decisions, consisting of three components: a lazy switch-control component, a randomized rounding component, and an online-learning-based prediction component. We rigorously prove the competitive ratio towards the offline optimum that take the actual inputs. Our trace-driven simulations confirm the advantages of our approach over multiple alternatives.

Appendix

A. Proof of Lemma 1

Proof. The minimization in Algorithm 3 implies it is $1/\lambda$-strongly convex, denoted by $J_t(\cdot)$. $\forall u, v$, we have

$$J_t(v) \geq J_t(u) + \nabla J_t(u)(v-u) + \frac{||v-u||^2}{2}. \tag{9}$$

Since $\widehat{\sigma}_{t+1}$ is the optimum, $\nabla J_t(\widehat{\sigma}_{t+1})(\sigma_t - \widehat{\sigma}_{t+1}) \geq 0$. By plugging the previous inequality to Inequality (9), we have $J_t(\sigma_t) \geq J_t(\widehat{\sigma}_{t+1}) + \frac{1}{2\lambda}||\sigma_t - \widehat{\sigma}_{t+1}||$. Adding $f_t(\widehat{\sigma}_t)$ on both sides, expanding $J_t(\cdot)$ as well as using the property of a convex function (i.e., $f_t(\sigma_t) \geq f_t(\widehat{\sigma}_t) + \nabla f_t(\widehat{\sigma}_t)(\sigma_t - \widehat{\sigma}_t)$), we have

$$\begin{aligned} &f_t(\widehat{\sigma}_t) + \nabla f_t(\widehat{\sigma}_t)(\widehat{\sigma}_{t+1} - \widehat{\sigma}_t) + \frac{||\widehat{\sigma}_{t+1} - \widehat{\sigma}_t||^2}{2\lambda} \\ &\leq f_t(\sigma_t) + \frac{||\sigma_t - \widehat{\sigma}_t||^2}{2\lambda} - \frac{||\sigma_t - \widehat{\sigma}_{t+1}||^2}{2\lambda}. \end{aligned} \tag{10}$$

Then, we analyze the gradient term, $-\nabla f_t(\widehat{\sigma}_t)(\widehat{\sigma}_{t+1} - \widehat{\sigma}_t)$

$$\leq \frac{||\nabla f_t(\widehat{\sigma}_t)||^2}{2u} + \frac{u}{2}||\widehat{\sigma}_{t+1} - \widehat{\sigma}_t||^2 \leq \frac{F^2}{2u} + \frac{u}{2}||\widehat{\sigma}_{t+1} - \widehat{\sigma}_t||^2,$$

where u is a positive constant. The first inequality sign holds because of the property of norms and $u^2 + v^2 \geq 2uv$; and the second inequality sign holds due to the bounded gradient. After that, we plug the previous inequality into Inequality (10):

$$f_t(\widehat{\sigma}_t) \leq f_t(\sigma_t) + \frac{\lambda F^2}{2} + \frac{1}{2\lambda}(||\sigma_t - \widehat{\sigma}_t||^2 - ||\sigma_t - \widehat{\sigma}_{t+1}||^2),$$

where the inequality holds because u could be chosen (i.e., $u = 1/\lambda$), such that $(\frac{u}{2} - \frac{1}{2\lambda}) = 0$. Next, we consider

$$||\sigma_t - \widehat{\sigma}_t||^2 \leq 2\sigma_{max}||\sigma_t - \sigma_{t-1}|| + ||\sigma_{t-1} - \widehat{\sigma}_t||^2, \quad (11)$$

where the inequality sign holds because we add two complementary terms $\mp||\sigma_{t-1} - \widehat{\sigma}_t||^2$ and we apply the difference of two squares and the triangle inequality. Then, we have

$$\sum_{t=1}^{T} f_t(\widehat{\sigma}_t) \leq \sum_{t=1}^{T} f_t(\sigma_t) + \frac{\lambda F^2 T}{2} + \frac{\sigma_{max} V(\{\sigma_t\})}{\lambda} + \frac{\sigma_{max}^2}{2\lambda},$$

where $V(\{\sigma_t\}) = \sum_t ||\sigma_{t+1} - \sigma_t||$. Since $f_t(\sigma_t) = 0$, we have

$$\sum_{t=1}^{T} (\sigma_t - \widehat{\sigma}_t)^2 g_t^2 \leq \frac{\lambda F^2 T}{2} + \frac{\sigma_{max} V(\{\sigma_t\})}{\lambda} + \frac{\sigma_{max}^2}{2\lambda}.$$

Using Cauchy–Schwarz [46], $\lambda = \mathcal{O}\{\sqrt{V(\{\sigma_t\})/T}\}$, we have

$$\sum_{t=1}^{T} (\sigma_t - \widehat{\sigma}_t) g_t \leq \sqrt{(\sum_{t=1}^{T} (\sigma_t - \widehat{\sigma}_t)^2 g_t^2)(\sum_{t=1}^{T} 1)} \triangleq \Phi(\boldsymbol{\sigma}).$$

The regret is then bounded as $\mathcal{O}(\sqrt[4]{V(\{\sigma_t\})} T^{\frac{3}{4}})$. Note that $V(\{\sigma_t\})$ is the inherent dynamics of the system. □

B. Proof of Lemma 2

Proof. We prove this lemma by linking $C_S^v(\widetilde{\boldsymbol{I}}_\wedge)$ with $\widehat{C}_{\neg S}^v(\widetilde{\boldsymbol{I}}_\wedge)$ first, and then linking $\widehat{C}_{\neg S}^v(\widetilde{\boldsymbol{I}}_\wedge)$ with its optimum under the predicted inputs without rounding (i.e., $\widehat{C}^v(\widetilde{\boldsymbol{I}}_\wedge^*)$).

For the switching cost incurred in previous epoches (i.e., $C_S^{t'_u}(\widetilde{\boldsymbol{I}}_\wedge)$), where t'_u is the timestamp of switching edges, $\forall u : 1 \leq u \leq u'$ with maximum record u', the non-switching cost from the beginning of this switch to the next one (i.e., in $[t'_u, t'_{u+1} - 1]$), is at least ζ_2 times the switching cost. Furthermore, the potential switching cost in $[t'_{u'}, t]$ is at most ζ_1 times the non-switching costs. Thus, $\forall t \leq T$, we have

$$\begin{aligned}
&\sum_{v=1}^{t} C_S^v(\widetilde{\boldsymbol{I}}_\wedge) = \sum_{u \leq u'} \sum_{v=t'_u}^{t'_{u+1}-1} C_S^v(\widetilde{\boldsymbol{I}}_\wedge) + \sum_{v=t'_{u'}}^{t} C_S^v(\widetilde{\boldsymbol{I}}_\wedge) \\
&= \sum_{u \leq u'} \{C_S^{t'_u}(\widetilde{\boldsymbol{I}}_\wedge) + \sum_{v=t'_u+1}^{t'_{u+1}-1} 0\} + \{C_S^{t'_{u'}}(\widetilde{\boldsymbol{I}}_\wedge) + \sum_{t'_{u'}+1}^{t} 0\} \\
&\leq \sum_{u \leq u'} \{\tfrac{1}{\zeta_2} \sum_{v=t'_u}^{t'_{u+1}-1} \widehat{C}_{\neg S}^v(\widetilde{\boldsymbol{I}}_\wedge) + 0\} + \{\zeta_1 \widehat{C}_{\neg S}^{t'_{u'}}(\widetilde{\boldsymbol{I}}_\wedge) + 0\} \\
&\leq \max\{\zeta_1, 1/\zeta_2\} \sum_{v=1}^{t} \widehat{C}_{\neg S}^v(\widetilde{\boldsymbol{I}}_\wedge). \quad (12)
\end{aligned}$$

Since $\epsilon'_2 \triangleq \max_v \frac{\max_{\widetilde{\boldsymbol{I}}} \widehat{C}_{\neg S}^v(\widetilde{\boldsymbol{I}})}{\min_{\widetilde{\boldsymbol{I}}} \widehat{C}_{\neg S}^v(\widetilde{\boldsymbol{I}})}$, we have $\widehat{C}_{\neg S}^v(\widetilde{\boldsymbol{I}}_\wedge) \leq \epsilon'_2 \widehat{C}_{\neg S}^v(\widetilde{\boldsymbol{I}}_\wedge^*)$. After that, we have the following inequality (i.e., $\forall t \leq T$):

$$\sum_{v=1}^{t} \widehat{C}_{\neg S}^v(\widetilde{\boldsymbol{I}}_\wedge) \leq \epsilon'_2 \sum_{v=1}^{t} \widehat{C}_{\neg S}^v(\widetilde{\boldsymbol{I}}_\wedge^*) \leq \epsilon'_2 \sum_{v=1}^{t} \widehat{C}^v(\widetilde{\boldsymbol{I}}_\wedge^*).$$

Therefore, the overall cost of the online algorithm without rounding by using predicted inputs could be bounded as

$$\begin{aligned}
&\widehat{\mathcal{P}}(\widetilde{\boldsymbol{I}}_\wedge) \leq (1 + \max\{\zeta_1, \tfrac{1}{\zeta_2}\}) \sum_{v=1}^{T} \widehat{C}_{\neg S}^v(\widetilde{\boldsymbol{I}}_\wedge) \\
&\leq \epsilon'_2 (1 + \max\{\zeta_1, \tfrac{1}{\zeta_2}\}) \sum_{v=1}^{T} \widehat{C}^v(\widetilde{\boldsymbol{I}}_\wedge^*) \triangleq \epsilon_2 \widehat{\mathcal{P}}(\widetilde{\boldsymbol{I}}_\wedge^*). \quad \square
\end{aligned}$$

C. Proof of Lemma 3

Proof. The rounding part is triggered and actually used only when $\widehat{\mathbb{P}}_{t,1}$ is feasible. For $\widehat{\mathbb{P}}_{t,1}$ invoked in line 5 of Algorithm 1, the results are $\widetilde{x}_{ijkt}, \widetilde{y}_{ijkt}, \widetilde{\eta}_t$. When $\widehat{\mathbb{P}}_{t,1}$ is invoked in line 7 of Algorithm 1, $\bar{x}_{ijkt}, \bar{\eta}_t$ are the feasible results. Therefore, we first try to link $\{\bar{y}_{jkt}\}$ and $\{\widetilde{y}_{jkt}\}$. We use index $u : 1 \leq u \leq u'$ to record the timestamp in terms of the edge switches. Then, we have the following relationship:

$$\begin{aligned}
&\sum_{t,k,j} \bar{y}_{jkt} = \sum_{u \leq u'} \{(t'_{u+1} - t'_u)(\sum_{k,j} \bar{y}_{jkt'_u})\} \quad (13) \\
&\overset{(13a)}{\leq} \sum_{u \leq u'} \{(t'_{u+1} - t'_u)(\sum_{k,j} \widetilde{y}_{jkt'_u} + \frac{|\mathcal{K}| \sum_k \sum_i \sum_j \widetilde{x}_{ijkt'_u}}{\sum_k |\mathcal{U}_{kt'_u}|})\},
\end{aligned}$$

where Inequality (13a) holds due to proposed pairwise rounding $1 + \sum_j \widetilde{y}_{jkt} \geq \sum_j \bar{y}_{jkt}$ and Constraint (1) in $\widehat{\mathbb{P}}_{t,1}$; and the right term is less than $(1 + \max_t\{\frac{\sum_i d_{ikt}}{\sum_k |\mathcal{U}_{kt}|}\}) \sum_{t,k,j} \widetilde{y}_{jkt}$ due to Constraint (1). Then, we consider all of the terms in $\widehat{C}_{\neg S}^t(\cdot)$ by using $\bar{\boldsymbol{I}}_\wedge$. For data transference, we have

$$\begin{aligned}
&\sum_{t,k,j,i} \bar{x}_{ijkt} d_{ikt} a_{ijkt} \leq \max_{t,k,j,i}\{d_{ikt} a_{ijkt}\} \sum_{t,k,j,i} \bar{x}_{ijkt} \\
&\overset{(14a)}{\leq} \kappa'_1 \sum_{t,k,j} \bar{y}_{jkt} \sum_i d_{ikt} \overset{(14b)}{\leq} \kappa_1 \sum_{t,k,j} \widetilde{y}_{jkt}, \quad (14)
\end{aligned}$$

where Inequality (14a) holds due to Constraint (1) in $\widehat{\mathbb{P}}_{t,1}$ and $\kappa'_1 \triangleq \max_{t,k,j,i}\{d_{ikt} a_{ijkt}\}$; Inequality (14b) holds due to Inequality (13). For the global aggregation term, we have

$$\sum_{t,k,j} \frac{r_0 ln(\frac{1}{\varepsilon_0})}{1 - \widetilde{\eta}_t} \bar{y}_{jkt} m_t (b_{jkt} + c_t) \leq \kappa_2 \sum_{t,k,j} \widetilde{y}_{jkt},$$

where the Inequality holds also due to Inequality (13). At last, $\sum_{t,k,j} \bar{z}_{jkt}$ can be bounded by $\kappa_3 \sum_{t,k,j} \widetilde{y}_{jkt}$ similarly according to Inequality (14) and the term $\sum_{t,k,j} o_{jkt} \bar{y}_{jkt}$ is directly bounded by $\kappa_4 \sum_{t,k,j} \widetilde{y}_{jkt}$ according to Inequality (13), too. We have $\sum_t \widehat{C}^t(\bar{\boldsymbol{I}}_\wedge) \leq (1 + 1/\zeta_1) \sum_t \widehat{C}_{\neg S}^t(\bar{\boldsymbol{I}}_\wedge)$ by using Constraint (4). Note that all of the terms in $\widehat{C}_{\neg S}^t(\bar{\boldsymbol{I}}_\wedge)$ are linked with $\sum_{t,k,j} \widetilde{y}_{jkt}$. Since the term $r_0 ln(\frac{1}{\varepsilon_0})/\{1 - \widetilde{\eta}_t\}$ is larger than 1 as well as all of the terms in the objective are non-negative, $\sum_{t,k,j} \widetilde{y}_{jkt}$ is less than $\widehat{\mathcal{P}}(\widetilde{\boldsymbol{I}}_\wedge)$. □

D. Proof of Theorem 1

Proof. Theorem 1 contains a product term. For the first part, by using Lemma 1, we have $\mathcal{P}(\bar{\boldsymbol{I}}_\wedge) \leq \widehat{\mathcal{P}}(\bar{\boldsymbol{I}}_\wedge) + \Phi(\boldsymbol{a}) + \Phi(\boldsymbol{b})$. By letting $\Phi = \Phi(\boldsymbol{a}) + \Phi(\boldsymbol{b})$ and using Lemma 3, we have

$$\frac{E[\mathcal{P}(\bar{\boldsymbol{I}}_\wedge)]}{\widehat{\mathcal{P}}(\bar{\boldsymbol{I}}_\wedge)} \leq \frac{E[\widehat{\mathcal{P}}(\bar{\boldsymbol{I}}_\wedge) + \Phi]}{\widehat{\mathcal{P}}(\bar{\boldsymbol{I}}_\wedge)} \leq \epsilon_3 + \frac{E[\Phi]}{\widehat{\mathcal{P}}(\bar{\boldsymbol{I}}_\wedge)} \leq \epsilon_3 + \frac{\Phi}{\widehat{\mathcal{P}}^*}.$$

Since Φ is only sub-linearly growth with T, $E[\Phi] = \Phi$ and the denominator is further scaled to its optimum, where $\widehat{\mathcal{P}}^* = \widehat{\mathcal{P}}(\widetilde{\boldsymbol{I}}_\wedge^*)$. Since the optimum result in the real domain is better than that in the integer domain, we have $\mathcal{P}(\widetilde{\boldsymbol{I}}^*) - \mathcal{P}^* \leq 0$, and

$$\begin{aligned}
&\widehat{\mathcal{P}}(\widetilde{\boldsymbol{I}}_\wedge) \leq \epsilon_2 \widehat{\mathcal{P}}(\widetilde{\boldsymbol{I}}_\wedge^*) = \epsilon_2 \min_{\widetilde{\boldsymbol{I}}}\{\widehat{\mathcal{P}}(\widetilde{\boldsymbol{I}})\} \overset{(15a)}{\leq} \epsilon_2 \min_{\widetilde{\boldsymbol{I}}}\{\mathcal{P}(\widetilde{\boldsymbol{I}}) + \Phi'\} \\
&\leq \epsilon_2 \min_{\widetilde{\boldsymbol{I}}}\{\mathcal{P}(\widetilde{\boldsymbol{I}})\} + \epsilon_2 \Phi' \overset{(15b)}{=} \epsilon_2 \mathcal{P}(\widetilde{\boldsymbol{I}}^*) + \epsilon_2 \Phi', \quad (15)
\end{aligned}$$

where Inequality (15a) holds by similarly applying Lemma 2 twice on the predicted terms under decision $\widetilde{\boldsymbol{I}}_\wedge^*$, whose upper bound is Φ'; Inequality (15b) holds due to $\widetilde{\boldsymbol{I}}^*$. Then, we have

$$\frac{\widehat{\mathcal{P}}(\widetilde{\boldsymbol{I}}_\wedge) - \mathcal{P}(\widetilde{\boldsymbol{I}}^*)}{\mathcal{P}^*} \leq 1 + \frac{(\epsilon_2 - 1)\mathcal{P}(\widetilde{\boldsymbol{I}}^*) + \Phi'}{\mathcal{P}^*} \overset{(16a)}{\leq} \epsilon_2 + \frac{\epsilon_2 \Phi'}{\mathcal{P}^*}, \quad (16)$$

where Inequality (16a) holds using $\mathcal{P}(\widetilde{\boldsymbol{I}}^*) - \mathcal{P}^* \leq 0$. The lower bound for both $\widehat{\mathcal{P}}^*$ and $\mathcal{P}^*$ is $T \sum_{t,j} \min\{m_t c_t + o_{jkt}\}$, considering that only one switched-on edge and one global aggregation are involved. Thus, for the terms $\frac{\Phi}{\widehat{\mathcal{P}}^*}$ and $\frac{\epsilon_2 \Phi'}{\mathcal{P}^*}$, either of them is $\mathcal{O}\{T^{\epsilon_1 - 1}\}$, where $\epsilon_1 - 1 < 0$. □

REFERENCES

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

[2] W. House, "Consumer data privacy in a networked world: A framework for protecting privacy and promoting innovation in the global digital economy," in *White House, Washington, DC*, 2012, pp. 1–62.

[3] A. Vulimiri, C. Curino, P. B. Godfrey, T. Jungblut, J. Padhye, and G. Varghese, "Global analytics in the face of bandwidth and regulatory constraints," in *USENIX NSDI*, 2015, pp. 323–336.

[4] C.-C. Hung, G. Ananthanarayanan, L. Golubchik, M. Yu, and M. Zhang, "Wide-area analytics with multiple resources," in *ACM EuroSys*, 2018, pp. 1–16.

[5] K. Chen and G. Tan, "Bikegps: Localizing shared bikes in street canyons with low-level GPS cooperation," in *ACM TOSN*, vol. 15, no. 4, 2019, pp. 1–28.

[6] C. Ma, J. Konečnỳ, M. Jaggi, V. Smith, M. I. Jordan, P. Richtárik, and M. Takáč, "Distributed optimization with arbitrary local solvers," in *Taylor & Francis Optimization Methods and Software*, vol. 32, no. 4, 2017, pp. 813–848.

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

[8] V. Smith, S. Forte, C. Ma, M. Takáč, M. I. Jordan, and M. Jaggi, "CoCoA: A general framework for communication-efficient distributed optimization," in *JMLR*, vol. 18, no. 1, 2017, pp. 8590–8638.

[9] E. Kurdoglu, Y. Liu, Y. Wang, Y. Shi, C. Gu, and J. Lyu, "Real-time bandwidth prediction and rate adaptation for video calls over cellular networks," in *ACM MMSys*, 2016, pp. 1–11.

[10] N. Chen, A. Agarwal, A. Wierman, S. Barman, and L. L. Andrew, "Online convex optimization using predictions," in *ACM SIGMETRICS*, 2015, pp. 191–204.

[11] N. Chen, J. Comden, Z. Liu, A. Gandhi, and A. Wierman, "Using predictions in online optimization: Looking forward with an eye on the past," *ACM SIGMETRICS*, vol. 44, no. 1, pp. 193–206, 2016.

[12] L. Lu, J. Tu, C.-K. Chau, M. Chen, and X. Lin, "Online energy generation scheduling for microgrids with intermittent energy sources and co-generation," in *ACM SIGMETRICS*, vol. 41, no. 1, 2013, pp. 53–66.

[13] Y. Jin, L. Jiao, Z. Qian, S. Zhang, N. Chen, S. Lu, and X. Wang, "Provisioning edge inference as a service via online learning," in *IEEE SECON*, 2020, pp. 1–9.

[14] M. Lin, A. Wierman, L. L. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," in *IEEE/ACM ToN*, vol. 21, no. 5, 2012, pp. 1378–1391.

[15] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," in *IMLS ICML*, 2019, pp. 8114–8124.

[16] F. Haddadpour and M. Mahdavi, "On the convergence of local descent methods in federated learning," in *arXiv:1910.14425*, 2019.

[17] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *ICLR*, 2020.

[18] 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 *IEEE JSAC*, vol. 37, no. 6, 2019, pp. 1205–1221.

[19] N. H. Tran, W. Bao, A. Zomaya, N. M. NH, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE INFOCOM*, 2019, pp. 1387–1395.

[20] H. H. Yang, Z. Liu, T. Q. Quek, and H. V. Poor, "Scheduling policies for federated learning in wireless networks," in *IEEE TCOMM*, vol. 68, no. 1, 2019, pp. 317–333.

[21] Y. Tu, Y. Ruan, S. Wang, S. Wagle, C. G. Brinton, and C. Joe-Wang, "Network-aware optimization of distributed learning for fog computing," in *IEEE INFOCOM*, 2020.

[22] Z. Zhou, S. Yang, L. J. Pu, and S. Yu, "Cefl: Online admission control, data scheduling and accuracy tuning for cost-efficient federated learning across edge nodes," in *IEEE IoTJ*, 2020.

[23] L. Zhang, C. Wu, Z. Li, C. Guo, M. Chen, and F. C. Lau, "Moving big data to the cloud: An online cost-minimizing approach," in *IEEE JSAC*, vol. 31, no. 12, 2013, pp. 2710–2721.

[24] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *IEEE INFOCOM*, 2018, pp. 207–215.

[25] L. Jiao, L. Pu, L. Wang, X. Lin, and J. Li, "Multiple granularity online control of cloudlet networks for edge computing," in *IEEE SECON*, 2018, pp. 1–9.

[26] J. Meng, H. Tan, C. Xu, W. Cao, L. Liu, and B. Li, "Dedas: Online task dispatching and scheduling with bandwidth constraint in edge computing," in *IEEE INFOCOM*, 2019, pp. 2287–2295.

[27] Z. Zhou, S. Yu, W. Chen, and X. Chen, "Ce-iot: Cost-effective cloud-edge resource provisioning for heterogeneous iot applications," in *IEEE IoTJ*, 2020.

[28] Y. Wang, Y. Li, T. Lan, and N. Choi, "A reinforcement learning approach for online service tree placement in edge computing," in *IEEE ICNP*, 2019, pp. 1–6.

[29] X. Xiong, K. Zheng, L. Lei, and L. Hou, "Resource allocation based on deep reinforcement learning in iot edge computing," in *IEEE JSAC*, vol. 38, no. 6, 2020, pp. 1133–1146.

[30] T. Ouyang, R. Li, X. Chen, Z. Zhou, and X. Tang, "Adaptive user-managed service placement for mobile edge computing: An online learning approach," in *IEEE INFOCOM*, 2019, pp. 1468–1476.

[31] H. Liao, Z. Zhou, X. Zhao, L. Zhang, S. Mumtaz, A. Jolfaei, S. H. Ahmed, and A. K. Bashir, "Learning-based context-aware resource allocation for edge-computing-empowered industrial iot," in *IEEE IoTJ*, vol. 7, no. 5, 2020, pp. 4260–4277.

[32] Q. Zhang, L. Gui, F. Hou, J. Chen, S. Zhu, and F. Tian, "Dynamic task offloading and resource allocation for mobile-edge computing in dense cloud ran," in *IEEE IoTJ*, vol. 7, no. 4, 2020, pp. 3282–3299.

[33] O. Dekel, N. Haghtalab, P. Jaillet *et al.*, "Online learning with a hint," in *NeurIPS*, 2017, pp. 5299–5308.

[34] L. Huang, M. Chen, and Y. Liu, "Learning-aided stochastic network optimization with state prediction," *IEEE/ACM ToN*, vol. 26, no. 4, pp. 1810–1820, 2018.

[35] "MNIST Database," http://yann.lecun.com/exdb/mnist/, 2020.

[36] "Bandwith from Edges to Aliyun," www.aliyun.com/, 2020.

[37] E. Nygren, R. K. Sitaraman, and J. Sun, "The akamai network: a platform for high-performance internet applications," in *ACM SIGOPS*, vol. 44, no. 3, 2010, pp. 2–19.

[38] D. Kumar, J. Li, A. Chandra, and R. Sitaraman, "A TTL-based approach for data aggregation in geo-distributed streaming analytics," in *ACM POMACS*, vol. 3, no. 2, 2019, pp. 1–27.

[39] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE TWC*, 2020.

[40] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE TWC*, 2020.

[41] 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 *IEEE ICDCS*, 2020.

[42] "IPOPT," https://github.com/coin-or/Ipopt/, 2020.

[43] R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan, "Dependent rounding and its applications to approximation algorithms," in *ACM JACM*, vol. 53, no. 3, 2006, pp. 324–360.

[44] "EPEX SPOT Market Data," https://www.epexspot.com/en/market-data/, 2020.

[45] "AMPL," https://ampl.com/, 2020.

[46] S. G. Walker, "A self-improvement to the cauchy–schwarz inequality," *Elsevier Statistics & Probability Letters*, vol. 122, pp. 86–89, 2017.