AdaCoOpt: Leverage the Interplay of Batch Size and Aggregation Frequency for Federated Learning

Weijie Liu[1], Xiaoxi Zhang[1], Jingpu Duan[2], Carlee Joe-Wong[3], Zhi Zhou[1], Xu Chen[1]
[1]Sun Yat-sen University, [2]Pengcheng Laboratory, [3]Carnegie Mellon University
Email: liuwj55@mail2.sysu.edu.cn, {zhangxx89, zhouzhi9, chenxu35}@mail.sysu.edu.cn
duanjp@pcl.ac.cn, cjoewong@andrew.cmu.edu

***Abstract*—Federated Learning (FL) is a distributed learning paradigm that can coordinate heterogeneous edge devices to perform model training without sharing private raw data. Many prior works have analyzed the FL convergence with respect to important hyperparameters, including batch size and aggregation frequency. However, adjusting the batch size and the number of local updates can affect the model performance, training time, and the cost of consuming computation and communication resources, in different and perhaps complex forms. Their joint effects have been overlooked and should be exploited to achieve accurate models with controllable operational expenditure. This paper proposes novel analytical models and optimization algorithms that leverage the interplay of batch size and aggregation frequency to navigate the trade-offs among convergence, cost, and completion time for FL. We first obtain a new convergence bound of the training error under heterogeneous training datasets across devices. Based on this bound, we derive closed-form solutions of a co-optimized batch size and aggregation frequency, a single configuration for all the devices. We then design an efficient exact algorithm for assigning different batch configurations across devices that can further improve the model accuracy to address the heterogeneity of both data and system characteristics. Further, we propose an adaptive control algorithm to dynamically adjust the solutions with estimated network states. Extensive experiments demonstrate the superiority of our offline optimal solutions and online adaptive algorithm.**

I. Introduction

Federated Learning (FL) [1]–[3] has gained much attention as it enables distributed model training through multiple collaborative devices without exposing their raw data. In the meantime, with the proliferation of edge computing technologies [4], [5], deploying FL at edge devices has become a promising computation paradigm to facilitate data-driven applications while preserving data privacy. Unlike traditional distributed machine learning (DML) [6], [7], FL allows each training device (a.k.a. worker) to perform multiple local updates before uploading their model parameters to the central server in each aggregation round and does not require partitioning a central pool of data across distributed workers.

Xiaoxi Zhang is the corresponding author.

This work was supported by NSFC grant 62102460, grant 202201011392 under the Guangzhou Science and Technology Plan Project, Young Outstanding award under the Zhujiang Talent Plan of Guangdong province, NSFC grant 61972432, Guangdong Basic and Applied Basic Research Foundation (2021B151520008), Guangdong Introducing Innovative and Entrepreneurial Teams (2017ZT07X355), and NSF grants CNS-2106891 and CNS-1751075.

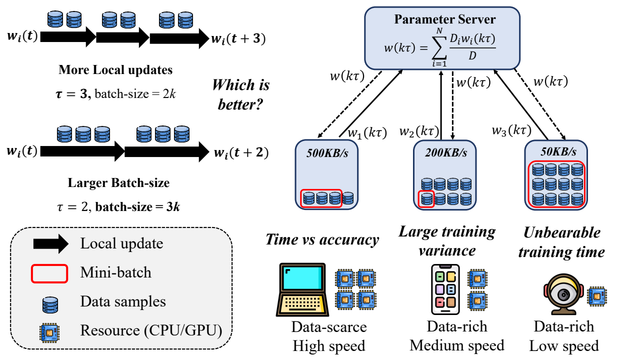


Fig. 1. Left: Interplay of mini-batch size and aggregation frequency; Right: Heterogeneous mini-batch sizes among clients

Despite its advantages, FL still faces two major challenges: 1) skewed distributions and unbalanced sizes of training data at different devices (*statistical challenge*), and 2) heterogeneous and limited edge resources (*system challenge*). The former is also referred to as non-i.i.d. data, which has been comprehensively analyzed for representative FL algorithms, especially FedAvg [3], [8]. Studies to address the system challenge have mainly focused on improving the learning efficiency by mitigating the impact of slow "straggler" devices on the wall-clock training time [5], [8]. In addition, the cost due to either the energy consumed over a long training period [9], [10] or monetary incentives paid to participating clients [11], [12] can be prohibitive for FL at the edge [13]. Thus, taking both time and cost into consideration when configuring training tasks on heterogeneous devices is of vital importance for FL algorithms. A few inspiring works have analyzed the model convergence when varying different controls for FL running at the edge, e.g., balancing the number of local updates and aggregation rounds [8], or adjusting workers' mini-batch sizes under a time budget [5], but the co-optimization of these control variables are still under-explored. In this work, we call for a full-fledged FL algorithm that can capture the three-way trade-off between convergence, training time, and cost expenditure. We jointly optimize the aggregation frequency and batch sizes, as they are the hyperparameters that determine the amount of data processed in each aggregation round and thus most affect these performance metrics.

Further, we have the following intuitions. As illustrated in Figure 1-Left, increasing either the mini-batch size or the number of local updates can lead to more training samples processed and thus improve the local model accuracy. However, doing so can also increase the consumed cost and training time. Moreover, a larger number of local updates (lower aggregation frequency) may result in a larger gap between the local and global models [3], though this effect may also depend on the batch size at each device. Therefore, we ask: *what is the best way to improve the FL model training when we can control both of these variables?*

This work reveals that strategically choosing different mini-batch sizes among clients is also crucial. A motivating example might be performing an FL task for object detection on heterogeneous edge devices using their locally captured pictures. As illustrated in Figure 1-Right, in this scenario, the "no-straggler" principle [5], which assigns the batch sizes of different FL devices for ensuring a uniform time per aggregation round [5], [14], may not be optimal. Specifically, the laptop with high training speed but relatively few data samples will have a large mini-batch, while other data-rich devices such as the smartphone can only have a small mini-batch due to the relatively slow training speed. This could severely impede the convergence rate, as a small batch size could introduce a high variance to the stochastic gradients (see Section IV). On the other hand, if we neglect the clients' heterogeneous computing capacities by simply setting a uniform batch size as FL practitioners usually do [3], [13], [15], the straggler effects can be severe. Batch sizes, however, cannot help to limit communication latency during model synchronization. Therefore, jointly choosing the aggregation frequency and batch sizes is important for balancing the energy cost, training time, and model accuracy. To achieve this, we make the following **technical contributions**:

1) *New convergence bound with respect to batch size and global aggregation frequency (Section IV).* We extend the FedAvg [3] framework by allowing different clients to use different mini-batch sizes. We capture FL clients' non-i.i.d. local datasets, based on which we then derive an upper bound of the global training error, with respect to the aggregation frequency and batch sizes. Prior theoretical works usually assume a *full-batch* training setting in analyzing the convergence rates, but practical FL deployments generally adopt the mini-batch approach. Our error bound can help bridge this inconsistency by quantifying the impacts of batch sizes considering heterogeneous data and system characteristics.

2) *Novel closed-form results and co-optimization algorithm design (Section V).* We propose an optimization model to capture the complex trade-offs among accuracy, completion (computation plus communication) time, and cost. Driven by our derived convergence bound, we provide closed-form solutions that co-optimize the batch size and aggregation frequency uniformly across clients. These results capture the interplay between these two control variables and can be easily adopted by FL developers. We also propose an efficient algorithm to optimize the assignment of heterogeneous batch sizes for different clients, further increasing the model accuracy.

3) *Online adaptive joint optimization algorithm (Sections VI and VII).* We design **AdaCoOpt**, an adaptive control algorithm to dynamically choose the number of local updates and heterogeneous batch sizes among different clients, accommodating the online estimates of the computation and communication capabilities in a fluctuating edge network. Extensive experiments under different testbed settings demonstrate the superiority of our algorithms in terms of the accuracy, cost, and training time.

II. Related Work

Convergence analysis for FL has been extensively studied in recent years. For instance, [3] analyzes the convergence of the classic *FedAvg* algorithm on non-i.i.d. data and establishes an $O(1/T)$ convergence bound for strongly convex and smooth problems. A refined FL framework *FedProx* [2] has accounted for clients' different amounts of partial work, with provable convergence guarantees. Further, [16] proves that the asynchronous *FedAvg* has near-linear convergence to the global optimum for strongly convex optimization problems. A few other works propose FL algorithms and analysis for non-convex optimizations [17]–[19]. These FL convergence analysis works mainly focus on the effect of the number of local updates or the total number of iterations.

Improving the FL efficiency has been studied in several directions, such as gradient compression [7], [20]–[22] and hyperparameter selection [5], [8], [23]. This work is orthogonal to the former (i.e., it can be combined with gradient compression), and falls in the latter regime, since we also aim to choose the best hyperparameters (i.e., batch sizes and aggregation frequency). To optimize the learning speed, most studies choose hyperparameters to mitigate the effect of "straggler" devices, such as client/device selection [11]–[13], [24] and staleness control [4], [25], [26]. Alternatively, recent works [5], [27], [28] also consider optimizing batch sizes or aggregation frequency to improve FL efficiency by equalizing the epoch time for each device to eliminate the straggler effect. However, their works either lack theoretical analysis [27] or neglect the joint impact of data and system heterogeneity across clients under resource constraints [5], [28], which are important characteristics in edge systems.

Controlling FL under resource constraints has risen as one of the major challenges in edge-enabled FL training. An increasing number of studies have been proposed to improve FL accuracy under resource budgets, accounting for either completion time [29]–[31] or operational cost [9], [10]. Luo et al. [32] propose a cost-effective FL design to choose the number of participants and local updates for total training cost minimization. Wang et al. [8] derive a tractable convergence bound with an arbitrary number of local updates and design an algorithm for dynamically adjusting the aggregation frequency. Our work additionally analyzes the interplay between the batch size and aggregation frequency in convergence, time, and cost metrics. A few recent works also consider choosing the mini-batch size. E.g, Ma et al. [5] propose a synchronous FL algorithm to adjust the batch size for edge FL, and Liu et al. [22] jointly optimize the batch size, gradient compression ratio, and spectrum allocation for wireless FL.

III. Preliminaries and Problem Formulation

A. Federated Learning

We consider a parameter-server (PS) architecture, which consists of a total of N distributed edge devices (clients), defined as a set $\mathcal{N}$, and a centralized PS for global aggregation. Each device $i \in \mathcal{N}$ has a local data set $\mathcal{D}_i$ with $D_i = |\mathcal{D}_i|$ data samples $\mathbf{x}_i = [\mathbf{x}_{i,1}, \mathbf{x}_{i,2}, ..., \mathbf{x}_{i,D_i}]$, and $\mathcal{D}_i$ is non-i.i.d. across i. We then define the loss function for each sample $\mathbf{x}_{i,j}$ as $f(\mathbf{w}, \mathbf{x}_{i,j})$ and the local loss function of device i as:

$$F_i(\mathbf{w}) = \frac{1}{D_i} \sum_{j \in \mathcal{D}_i} f(\mathbf{w}, \mathbf{x}_{i,j}). \tag{1}$$

The ultimate goal is to train a shared (global) model $\mathbf{w}$ that minimizes the global loss function, defined as:

$$F(\mathbf{w}) = \sum_{i \in \mathcal{N}} \frac{D_i}{D} F_i(\mathbf{w}), \tag{2}$$

where D is defined as $D = \sum_{i \in \mathcal{N}} D_i$.

As in the classic FedAvg [1] framework, in each round of training, clients divide their local data into mini-batches, perform multiple local updates, and upload their local models to the PS, which then broadcasts the global model updated by aggregating the local models to the clients. Prior works either assume use of the whole dataset for each round (*full-batch training*) [3], [8] or simplify the effects of batch size on the convergence and training time in their scenarios (e.g., [5]). Here we propose a more general FL setting by enabling customized batch sizes and the number of local updates.

B. FL with arbitrary batch size and aggregation frequency

To capture different batch sizes across clients, we define the loss function $F_{i,\mathcal{S}_i}(\mathbf{w})$ under a mini-batch instead of the original local loss function $F_i(\mathbf{w})$ for each end device i:

$$F_{i,\mathcal{S}_i}(\mathbf{w}) = \frac{1}{s_i} \sum_{j \in \mathcal{S}_i} f(\mathbf{w}, \mathbf{x}_{i,j}), \tag{3}$$

where $\mathcal{S}_i$ denotes a mini-batch randomly selected from $\mathcal{D}_i$ and s_i is the size of $\mathcal{S}_i$, i.e., the batch size that we will optimize in later Sections. The special case of full-batch training yields $s_i = D_i$ and thus $F_{i,\mathcal{S}_i}(\mathbf{w}) = F_i(\mathbf{w})$. With a learning rate $\eta > 0$, the rule of updating each local model at step t is:

$$\mathbf{w}_i(t) = \mathbf{w}_i(t-1) - \eta g_i(\mathbf{w}_i(t-1)), t \neq k\tau, \tag{4}$$

where $g_i(\mathbf{w}_i(t-1)) \triangleq \nabla F_{i,\mathcal{S}_i}(\mathbf{w}_i(t-1))$. We consider that a total of K aggregation rounds are performed in the training. The update of the global model at each aggregation step is:

$$\mathbf{w}(t) = \frac{\sum_{i=1}^{N} D_i \mathbf{w}_i(t)}{D}, \; t = k\tau, \tag{5}$$

where τ is the number of local updates in each aggregation round, meaning that the PS performs (5) and sends the global model $\mathbf{w}(t)$ to the clients only at steps $t = k\tau, k = 1, 2, ..., K$.

C. Accuracy-time-and-cost joint optimization model

Compared to data centers, edge devices usually have limited computing resources such as CPUs and GPUs. Their limited battery lives also restrict their energy utilization. Moreover, edge devices in FL training often establish the connection with the PS through the Wide Area Network [33], which could also incur high bandwidth costs. It is therefore necessary to consider both computation and communication costs.

Limited budget of the total cost expenditure. Formally, we suppose that the training consumes a units of computation cost (e.g., the cost of energy consumption) for processing a single sample and b units of bandwidth cost for each round of model synchronization. Let $s_{tot} = \sum_{i \in \mathcal{N}} s_i$ represent the sum of batch sizes per iteration over all the devices. We consider that the total cost incurred by the entire training process should not exceed R, i.e., $K(a\tau s_{tot} + b) \leq R$, which conforms to the conventional definition of model training cost [34]. Here, R can represent a cost budget of the energy consumption if the devices are owned by the FL owner, or the budget of total monetary reward sent to participating clients, e.g., for compensating their battery consumption or privacy losses [35].

Heterogeneous system capabilities. In practice, different edge devices can have heterogeneous computation and communication capacities, and the training time in each round is determined by the slowest device (straggler). Let p_i denote the computation speed (number of samples processed per time) of device i. We then define t_{ci} as the computation time of i for a single local update and assume that it is proportional to the batch size, i.e., $t_{ci} = s_i/p_i$. Further, t_{ui} is the communication time of each device i incurred by synchronizing her local model with the PS. These definitions are consistent with practical system modelings for FL training [5], [12]. Suppose that the FL task has a completion time deadline θ. We then have the following constraint on the training time:

$$\max_{i \in \mathcal{N}} K(\tau t_{ci} + t_{ui}) \leq \theta. \tag{6}$$

Our goal is to find the optimal solutions of batch sizes $\mathbf{s} = [s_1, s_2, ..., s_N]$ and the number of local update steps τ for N devices, to minimize the gap between the expected global loss function $\mathbb{E}[F(\mathbf{w}(K\tau))]$ and the optimum F^* after performing K communication rounds, while satisfying the cost and completion time constraints. We define $[X] \triangleq \{1, \cdots, X\}$. The optimization problem is formulated as follows:

$$\begin{aligned}
&\mathop{\textbf{Minimize}}_{\mathbf{s},\tau} && \mathbb{E}[F(\mathbf{w}(K\tau))] - F^* && \textbf{(Training error)} && (7)\\
&\textbf{S.t.} && \max_{i \in [N]} K(\tau t_{ci} + t_{ui}) \leq \theta && \textbf{(Completion time)} && (8)\\
& && K(a\tau s_{tot} + b) \leq R && \textbf{(Cost)} && (9)\\
& && s_i \in [D_i], \forall i, \tau \in [\tau_{max}] && \textbf{(Feasibility)} && (10)
\end{aligned}$$

To solve the above optimization problem, we need to first navigate the complex trade-offs among the expected error, completion time, and total cost incurred by the training process, via controlling our decision variables $\mathbf{s}$ (batch sizes) and τ (the number of local updates). We emphasize that, in addition to $\tau > 1$, unlike centralized DML, edge FL faces heterogeneous distributions and sizes of local datasets (D_i), and thus may yield heterogeneous optimal batch sizes s_i across workers, which we shall show in Section V. In contrast, the number of local updates τ is assumed uniform across clients, as unequal aggregation frequencies for different clients can cause objective inconsistency, i.e., the model converges to a mismatched objective function [36]. Existing works [24], [36] have addressed exogenously determined heterogeneous τ across different clients through new aggregation methods. Combining these works to *control* τ is complex however, and we leave this to our future work. To solve (7)-(10), our first challenge is to simultaneously quantify the effects of $\mathbf{s}$ and τ in the training error, formalized in our next section.

IV. Training Error Bound Analysis

In this section, we derive a new convergence bound to approximate (7), considering the joint effects of mini-batch sizes s_i and the number of local update steps τ. We first list our assumptions posed on the training model, which are generally adopted in pioneering FL works [37], [38].

Assumption 1. *ρ-quadratic-continuous: There exists a $\rho > 0$ that, for each device $i \in \mathcal{N}$, the batch loss function $F_{i,\mathcal{S}_i}$ satisfies: $\|F_{i,\mathcal{S}_i}(\mathbf{w}_1) - F_{i,\mathcal{S}_i}(\mathbf{w}_2)\| \leq \rho \|\mathbf{w}_1 - \mathbf{w}_2\|_2^2$, $\forall \mathbf{w}_1, \mathbf{w}_2$.*

Assumption 2. *β-smooth: For each client $i \in \mathcal{N}$, and some $\beta > 0$, $F_{i,\mathcal{S}_i}$ satisfies: $\|\nabla F_{i,\mathcal{S}_i}(\mathbf{w}_1) - \nabla F_{i,\mathcal{S}_i}(\mathbf{w}_2)\| \leq \beta \|\mathbf{w}_1 - \mathbf{w}_2\|$ for all $\mathbf{w}_1, \mathbf{w}_2$.*

The local and global loss functions satisfy Assumptions 1 and 2 straightforwardly due to the definition of $F_i(\cdot)$ and $F(\cdot)$.

Assumption 3. *Polyak-Łojasiewicz condition [39]: There exists some constant c that $0 < c \leq \beta$ and $c \leq 2\rho$, and for each device $i \in \mathcal{N}$, the global loss function $F(\mathbf{w})$ satisfies: $\|\nabla F(\mathbf{w})\|_2^2 \geq 2c(F(\mathbf{w}) - F^*)$, $\forall \mathbf{w}$.*

Assumption 4. *(First and Second Moment Limits) For some scalars $\mu_G \geq \mu > 0$ and $M_i > 0$, under any given model $\mathbf{w}$ and batch of data samples ξ_t randomly selected from $\cup_i \mathcal{D}_i$ at step t, the global batch-gradient $g(\mathbf{w}, \xi_t)$ and the variance of the gradient under any single data $\mathbf{x}_{i,j} \in \mathcal{D}_i$ of each client i, denoted as $\mathbb{V}_{\mathbf{x}_{i,j}}[\nabla f(\mathbf{w}, \mathbf{x}_{i,j})]$, satisfy:*

$$\nabla F(\mathbf{w})^T \mathbb{E}_{\xi_t}[g(\mathbf{w}, \xi_t)] \geq \mu \|\nabla F(\mathbf{w})\|_2^2,$$
$$\|\mathbb{E}_{\xi_t}[g(\mathbf{w}, \xi_t)]\|_2 \leq \mu_G \|\nabla F(\mathbf{w})\|_2,$$
$$\mathbb{V}_{\mathbf{x}_{i,j}}[\nabla f(\mathbf{w}, \mathbf{x}_{i,j})] \leq M_i, \ \forall i \in \mathcal{N}.$$

Assumption 5. *(Bounded Gradient Divergence, a.k.a. Non-i.i.d. Degree) Let $g(\mathbf{w})$ denote the global gradient under the dataset $\cup_i \mathcal{D}_i$. For some bounded scalar $\delta_i > 0$, the local gradient $g_i(\mathbf{w})$ of each client i under her dataset $\mathcal{D}_i$ satisfies:*

$$\|g_i(\mathbf{w}) - g(\mathbf{w})\| \leq \delta_i, \ \forall \mathbf{w}, i.$$

Theorem 1 (Error bound with heterogeneous batch sizes s_i)**.** *Suppose that the loss functions satisfy Assumptions 1–5. Assuming $F^* \geq 0$, given a fixed learning rate $0 \leq \eta \leq \frac{\mu}{\beta \mu_G^2}$ and the initial global parameter $\mathbf{w}(0)$, the expected error after K aggregation rounds with τ local updates per round is:*

$$\mathbb{E}[F(\mathbf{w}(K\tau))] - F^* \leq q^{K\tau}[F(\mathbf{w}(0)) - F^*] + \frac{1-q^K}{1-q}\left(\frac{\beta\eta^2(1-q^\tau)}{2D^2(1-q)}\sum_{i\in\mathcal{N}}\frac{M_i D_i^2}{s_i} + \rho h(\tau)^2\right), \quad (11)$$

where $q = 1 - \eta c \mu$, $h(\tau) = \frac{\delta}{\beta}((\eta\beta + 1)^\tau - 1) - \eta\delta\tau$, and $\delta = \sum_{i\in\mathcal{N}} \frac{D_i \delta_i}{D}$. Especially, when $\tau = 1$ and $s_i = s_{i'}$, $\forall i \neq i'$, the above theorem is consistent with the DML convergence rate in prior work [38].

We provide the basic idea of proving Theorem 1 below and defer the full proof to our technical report [40].

Proof sketch. We first extend the convergence bound of DML [3] with i.i.d. datasets and a uniform batch size to heterogeneous batch sizes in the non-i.i.d. scenario. In particular, we analyze the variance in model gradient computation among clients. Then, motivated by [5], [8], we upper bound the gap of global loss between DML and FL, allowing clients to use different batch sizes. Formally, we capture the local bias resulted from local updates τ, i.e., $F(\mathbf{w}(t)) - F(\mathbf{v}_{[k]}(t)) \leq \rho h(\tau)^2$, where $\mathbf{v}_{[k]}(t)$ is an auxiliary parameter vector that follows a centralized gradient descent. Finally, we combine these two bounds and apply them recursively over K rounds to derive the bound w.r.t. both aggregation frequency and batch sizes. □

Our bound (11) has a richer structure than those in [5], [8], [22] to show the effects of s_i, τ, and the data distributions. The first term is determined by the initial global loss, which continuously decreases during training. The term associated with M_i can be interpreted as the "gradient variance loss" resulting from the error of using a randomly selected batch to estimate the loss gradient under the entire local dataset. The last term $\rho h(\tau)^2$ can be regarded as the "local bias" which monotonically increases with τ, since a larger τ means less frequent communications between the clients and server and thus a larger gap between the global and local models.

V. Offline Optimization: Theory and Algorithm

In this section, we provide optimal solutions for co-optimized stationary batch sizes and the number of local updates in two cases. The total number of aggregation rounds K is pre-determined in our problem [15], [24], [36]. We assume they are both *offline* settings, where the parameters related to the model (in (11)) and the system (in the optimization constraints) can be obtained, e.g., through pre-run tests [6], [13]. We will design an adaptive control algorithm with parameters estimated online in Section VI.

A. Case 1: Co-optimizing uniform s and τ

We first consider the most common FL scenario in practice [1], [2] where every device has the same batch size s and number of local updates per round τ. Based on our bound (11), we derive closed-form solutions of s and τ in Theorem 2, by solving (7)–(10) with $s_i = s_{i'}, \ \forall i \neq i'$.

Theorem 2 (Interplay of uniform s and τ)**.** *Given the number of aggregation rounds K, and a feasible deadline ($\theta > Kt_{ui}$) and cost budget ($R > Kb$), the optimal uniform batch size s^* and the number of local updates τ^* satisfy:*

$$s^*(\tau) = \mathbf{min}\left\{\frac{R-Kb}{a\tau n}, \min_{i\in\mathcal{N}}\left\{\frac{p_i(\theta-Kt_{ui})}{K\tau}\right\}\right\}, \tag{12}$$

$$\tau_1 = \lfloor\hat{\tau}\rfloor, \, \tau_2 = \lceil\hat{\tau}\rceil, \, \frac{\partial f(\hat{\tau})}{\partial \tau} = 0, \tag{13}$$

$$\tau^* = \arg\min_{\tau\in\{\tau_1,\tau_2\}} f(\tau), \, s^* = \lfloor s^*(\tau^*)\rfloor, \tag{14}$$

where $h(\tau) = \frac{\delta}{\beta}\left((\eta\beta+1)^\tau - 1\right) - \eta\delta\tau$, $f(\tau) = q^{K\tau}G(0) + \frac{1-q^K}{1-q}\left(\frac{\beta\eta^2(1-q^\tau)}{2D^2(1-q)}\sum_{i\in\mathcal{N}}\frac{M_iD_i^2}{s^*(\tau)} + \rho h(\tau)^2\right)$, $G(0) = [F(\mathbf{w}(0)) - F^*]$, *and* $q = 1-\eta c\mu$.

Proof. Given that the objective function is monotonically decreasing with the client batch size, we can obtain the optimal uniform batch size $s^*(\tau)$ by finding its maximum value under both completion time and training cost constraints. Then we can substitute $s^*(\tau)$ into the objective function, i.e., $f(\tau) = q^{K\tau}G(0) + \frac{1-q^K}{1-q}\left(\frac{\beta\eta^2(1-q^\tau)}{2D^2(1-q)}\sum_{i\in\mathcal{N}}\frac{M_iD_i^2}{s^*(\tau)} + \rho h(\tau)^2\right)$, which can be easily proved to be a convex function when $\tau < 2/log(1/q)$. Since the value of $q = 1-\eta c\mu$ is only slightly less than 1 , this interval of τ can be large enough. Thus, we can solve $\hat{\tau}$ to minimize the expected error bound by letting the derivative of $f(\tau)$ to be zero. However, $\hat{\tau}$ can be fractional, so we need to compare the values of $f(\lfloor\hat{\tau}\rfloor)$ and $f(\lceil\hat{\tau}\rceil)$ to find the optimal τ^*. □

Our result quantitatively verifies an intuitive common practice that communicating with the PS every iteration ($\tau = 1$) is the optimum, if the number of aggregation rounds K and thus the total number of training iterations are sufficiently large (shown in Remark 1).

Remark 1. As the number of aggregation rounds K increases, the optimal solution τ^*, which is expressed in (14), will decrease to 1, i.e., $\lim_{K\to\infty}\tau^* = 1$.

Proof. When K is small, the first term $q^{K\tau}G(0)$ dominates $f(\tau)$, and it monotonically decreases with τ. As K grows larger, the second term dominates $f(\tau)$ and monotonically increases with τ. Thus the optimal number of local update steps τ^* decreases with the increase of K. □

Remark 1 can be intuitively explained as follows. When K is small, e.g., due to the high communication cost or limited bandwidth, a bigger τ leads to a larger total number of model updates and thus higher accuracy. In contrast, one should reduce τ if K is sufficiently large especially in later training iterations, since a larger τ may increase the gap between the global and local models and thus lead to a large final error. This implication is consistent with the intuition in [8].

B. Case 2: Co-optimizing τ and heterogeneous s_i

In this case, we generalize Case 1 by enabling different batch sizes assigned for different clients. Since edge devices can have different and potentially limited computation and communication capacities, increasing the batch size leads to longer computation time. Following [5], [12], the computation time of each step of local update can be modeled by $t_{ci} = s_i/p_i$. For the clarity of the following analysis, we first fix the value of τ, and then our optimization problem becomes:

$$\underset{\substack{\mathbf{s}=[s_1,s_2,\ldots s_N]\\ \tau\in[\tau_{max}]}}{\textbf{Minimize}} \quad \sum_{i\in\mathcal{N}}\frac{M_iD_i^2}{s_i} \tag{15}$$

$$\textbf{S.t.} \quad s_i \leq p_i\left(\frac{\theta}{K\tau} - \frac{t_{ui}}{\tau}\right), s_i \in [D_i], \ \forall i \tag{16}$$

$$s_{tot} = \sum_{i\in\mathcal{N}} s_i \leq (R-Kb)\,/\,(a\tau) \tag{17}$$

Directly applying an integer programming optimizer such as Gurobi [41] to solve (15)–(17) or using a brute force algorithm may incur a high time complexity with at least $O(\kappa^N\tau_{max})$, where $\kappa = \frac{s_{tot}}{N} >> 1$. Instead, we design a more efficient exact algorithm, as we state in the following theorem.

Theorem 3. *Given the number of aggregation rounds K and the number of local updates per round τ, Algorithm 1 outputs the optimal batch sizes $\mathbf{s}^* = [s_1, s_2, ..., s_i]$ and τ^* for FL training with at most $O(N^2\tau_{max})$ time complexity.*

The detailed proof is deferred to our technical report [40].

Intuition of Algorithm 1. Since the objective function (15) decreases with s_i, the time constraint (16) defines the largest batch size allowed for any device i under deadline θ, i.e., $s_i(\theta) = p_i\left(\frac{\theta}{K\tau} - \frac{t_{ui}}{\tau}\right)$. Similarly, the cost constraint (17) is equivalent to defining a total batch size under the cost budget R, i.e., $\sum_{i\in\mathcal{N}} s_i = s_{tot}(R) = \frac{R-Kb}{a\tau}$. If neglecting $s_i(\theta)$ firstly, the Cauchy–Schwarz inequality yields:

$$\sum_{i\in\mathcal{N}}\frac{M_iD_i^2}{s_i}\cdot\sum_{i\in\mathcal{N}}s_i \geq \sum_{i\in\mathcal{N}}\left(\sqrt{M_i}D_i\right)^2. \tag{18}$$

Since $\sum_{i\in\mathcal{N}}\left(\sqrt{M_i}D_i\right)^2$ and $s_{tot}(R)$ are both constants, we can minimize the objective function $\sum_{i\in\mathcal{N}}\frac{M_iD_i^2}{s_i}$ when the equality holds with $\frac{\sqrt{M_1}D_1}{s_1} = \frac{\sqrt{M_2}D_2}{s_2} = \cdots = \frac{\sqrt{M_N}D_N}{s_N}$, i.e., $s_i \propto \sqrt{M_i}D_i$. Then, considering $s_i(\theta)$, we need to reduce s_i to $s_i(\theta)$ for *time-constrained devices* which have $s_i > s_i(\theta)$ (Lines 9-10). The s_i of those devices will not be revised again (Line 11) since they reach the maximum allowed batch size. In addition, we will re-assign (increase) the s_i of other devices while keeping $s_i \leq s_i(\theta)$ satisfied to make the best use of the extra data samples due to the reduced s_i of those *time-constrained devices*, which is in fact a sub-problem of our original optimization problem. We can get the final solution by repeating the previous procedure recursively (Lines 6-11), which can be proved optimal by using the Cauchy inequality again for $\sum_{i\in\mathcal{C}} s_i = s_{tot}(R) - \sum_{i\in\mathcal{N}\setminus\mathcal{C}} s_i(\theta)$, where $\mathcal{N}\setminus\mathcal{C}$ denotes the set of clients whose s_i have been regulated to be equal to $s_i(\theta)$. Since we always round down s_i (line 8), we may still have some remaining resource budget. due to the round down operation in the previous steps. We then increase the batch size of the device in the decreasing order of $\frac{M_i D_i^2}{s_i(s_i+1)} = \frac{M_i D_i^2}{s_i} - \frac{M_i D_i^2}{s_i+1}$ one at a time until the total batch size of all devices equals $s_{tot}(R)$ or $C = \emptyset$ (Lines 14-17). Finally, we can find τ^* that yields the smallest error bound according to (11), by enumerating each feasible τ under which s^* is optimized using the above method (Line 18).

Implication I. If the time constraint is not the bottleneck, s_i^* is proportional to $\sqrt{M_i} D_i$ (Line 8 of Algorithm 1). This insight is intuitive, as devices with larger data sizes (D_i) have the potential to contribute more samples each step, while datasets with a higher diversity (larger M_i) need larger batch sizes to reduce the local variances of their computed gradients. **This result reveals that using either full-batch ($s_i = D_i$) training [3] or uniform batch size [8], as FL practitioners usually adopt, can be ineffective under non-i.i.d. data.**

Implication II. Other batch size assignment schemes (e.g., [5], [27]), on the other hand, focus on eliminating straggler effects. They choose clients' batch sizes according to their computational capacity in order to minimize the average waiting time. **However, this "no-straggler" strategy is sub-optimal when cost constraints are present, a common scenario in edge systems [32]. Using their strategies [5], [27], devices with higher computation capacities but possibly a smaller $D_i\sqrt{M_i}$ of data always have bigger batch sizes, which could significantly undermine the model accuracy.** Our Algorithm 1 instead captures both the data heterogeneity ($D_i\sqrt{M_i}$) and system heterogeneity (mitigated straggler effects), as well as navigating the trade-off between the completion time and resource consumption.

Algorithm 1: An exact offline algorithm to Co-Optimize batch sizes and the number of local updates for FL training (**CoOptFL**)

```
Input  : $\mathbf{G}, M_i, D_i, K, \tau_{max}, a, b, R, \theta, p_i, t_{ui}, \forall i$
Output: $\tau^*, \mathbf{s}^* = [s_1, s_2, ..., s_N]$
1  foreach $\tau \in [1, \tau_{max}]$ do
2      Set $C = \mathcal{N}$, $s_{tot} = \frac{R-Kb}{a\tau}$, $s_r = s_{tot}$;
3      foreach node $i \in \mathcal{N}$ do
4          $s_i(\theta) = \lfloor p_i \left(\frac{\theta}{K\tau} - \frac{t_{ui}}{\tau}\right) \rfloor$
5      repeat
6          $flag = 0$;
7          foreach node $i \in C$ do
8              $s_i = \lfloor \frac{s_r \sqrt{M_i} D_i}{\sum_{i \in C} \sqrt{M_i} D_i} \rfloor$;
9              if $s_i \geq s_i(\theta)$ then
10                 $s_i = s_i(\theta), s_r = s_r - s_i(\theta)$;
11                 Remove node $i$ from set $C$, $flag = 1$;
12     until $flag = 0$ or $C = \emptyset$;
13     repeat
14         Find $i' = \text{argmax}_{i \in C} \frac{D_i^2}{s_i(s_i+1)}$, $s_{i'} = s_{i'} + 1$ ;
15         if $s_{i'} = s_{i'}(\theta)$ then
16             Remove node $i$ from set $C$;
17     until $\sum_{i \in \mathcal{N}} s_i = s_{tot}$ or $C = \emptyset$;
18 Find the optimum $(\tau^*, \mathbf{s}^*) = \text{argmin}_{(\tau, \mathbf{s})} \mathbf{G}$;
   /* Offline:$\mathbf{G} \triangleq$ (11) Online:$\mathbf{G} \triangleq$ (21)    */
```

VI. Online Adaptive Control Algorithm

Section V provides optimal solutions for devices' batch sizes and the number of local updates, but it does not consider how to adapt them online as we update estimates of potentially unknown parameters, including the computation speed c_i, communication time t_{ui}, and the parameters associated with the model. Therefore, in this section, we propose an adaptive algorithm to adjust s and τ at the beginning of each aggregation round, based on our online parameter estimation, realizing a more practical FL training by considering fluctuating network characteristics at the edge.

A. Marginal Error bound

Revisiting our offline optimization problem (7)–(10), the objective function derived in (11) with static parameters and decision variables is no longer suitable for our online setting. Therefore, we use a marginal upper bound, which is defined as the gap between the optimum F^* and the expected global loss that will be improved **in aggregation round** k, formalized as $\mathbb{E}[F(\mathbf{w}^{(k)})] - F^*$. We derive this in Lemma 1 of which the detailed proof is deferred to the technical report [40].

Lemma 1 (Marginal bound with heterogeneous batch size s_{ik}**).** *Suppose that the loss function satisfies Assumptions 1-5 and* $F^* \geq 0$. *For a fixed learning rate* $0 \leq \eta \leq \frac{\mu}{\beta \mu_G^2}$, *the expected error of the empirical loss after* k *global communication rounds with the number of local updates* τ_k *and batch sizes* s_{ik} *for round* k, *defined as* $\mathbb{E}[F(\mathbf{w}^{(k)})] - F^*$, *is at most*

$$q^{\tau_k}[\mathbb{E}[F(\mathbf{w}^{(k-1)})] - F^*] + \frac{\beta\eta^2(1-q^{\tau_k})}{2D^2(1-q)} \sum_{i \in \mathcal{N}} \frac{M_i D_i^2}{s_{ik}} + \rho h(\tau_k)^2, \tag{19}$$

where $q = 1 - \eta c \mu$, $h(\tau) = \frac{\delta}{\beta}((\eta\beta+1)^\tau - 1) - \eta\delta\tau$, *and* $F(\mathbf{w}^{(k-1)}) \triangleq F(\mathbf{w}(\sum_{i=1}^{k-1} \tau_i))$.

Compared to Theorem 1, Lemma 1 is defined for a more practical setting where the unknown parameters, system characteristics and decision variables need to be estimated online. It upper-bounds the error $\mathbb{E}[F(\mathbf{w}^{(k)})] - F^*$ incurred until round k. Our optimization problem (7)–(10) can be adapted to the following to solve for τ_k and batch size assignments $\mathbf{s}_k = [s_{1k}, s_{2k}, ...s_{ik}]$ for each client $i \in \mathcal{N}$ at round $k \in [K]$.

$$\underset{\mathbf{s}_k, \tau_k}{\textbf{Minimize}} \quad \mathbb{E}[F(\mathbf{w}^{(k)})] - F^* \quad \text{(Approximated by (19))}$$

$$\textbf{S.t.} \max_{i \in \mathcal{N}} \sum_{k=1}^{K} (\tau_k t_{ci} + t_{ui}) \leq \theta, \; t_{ci} = s_{ik}/p_i \tag{20}$$

$$\sum_{k=1}^{K} (a\tau_k \sum_{i \in \mathcal{N}} s_{ik} + b) \leq R, s_{ik} \leq D_i, \tau_k > 0$$

To solve (20), the remaining work is to estimate the unknown

Algorithm 2: AdaCoOpt (Procedure at the PS)

```
Input  : $\theta, R, K, \tau_{max}, \eta$
Output: $\mathbf{w}(R)$
Initialize: $\theta_c \leftarrow \theta, R_c \leftarrow R, t \leftarrow 0, k \leftarrow 0$
            $\tau_1 \leftarrow 1, \mathbf{w}(0) \leftarrow \mathbf{0}, \mathbf{s_1}$
1  Receive $D_i, M_i,\ \forall i \in \mathcal{N}$;
2  repeat
3      $k \leftarrow k+1,\ K \leftarrow K-1$;
4      Send $\mathbf{w}(t), \tau_k, s_{ik}$ to each node $i$;
5      $t_0 \leftarrow t, t \leftarrow t+\tau$;
6      Receive $\mathbf{w}_i(t), p_i$ from all the working nodes;
7      Execute global update according to (5) ;
8      if $t_0 > 0$ then
9          Receive $\rho_i, \beta_i, c_i, F_{i,\mathcal{S}_i}(\mathbf{w}(t_0)), \nabla F_{i,\mathcal{S}_i}(\mathbf{w}(t_0))$;
10         Calculate $g(\mathbf{w}(t_0)) \leftarrow \frac{\sum_{i=1}^N D_i g_i(\mathbf{w}_i(t_0))}{D}$
             $\delta_i \leftarrow \|g_i(\mathbf{w}_i(t_0)) - g(\mathbf{w}(t_0))\|$ ;
11         Estimate $\rho \leftarrow \frac{\sum_{i=1}^N D_i \rho_i}{D}, \beta \leftarrow \frac{\sum_{i=1}^N D_i \beta_i}{D}$;
12         Estimate $c \leftarrow \frac{\sum_{i=1}^N D_i c_i}{D}, \delta \leftarrow \frac{\sum_{i=1}^N D_i \delta_i}{D}$;
13         Estimate remaining resources $\theta_c, R_c$ and
             communication time of each device $t_{ui}$ ;
14         Define function $\mathbf{G}$ to be (21);
15         $\tau_{k+1}, \mathbf{s}_{k+1} =$
             CoOptFL$(\mathbf{G}, M_i, D_i, K, \tau_{max}, a, b, R_c, \theta_c, p_i, t_{ui})$
16 until $K = 0$ or $\theta_c < 0$ or $R_c < 0$;
17 Send $STOP$ flag to all devices;
```

Algorithm 3: AdaCoOpt (Procedure at client i)

```
Initialize: $D_i, t \leftarrow 0$
1  Estimate $M_i$ using the local dataset;
2  Send the local dataset size $D_i$ and $M_i$ to the server;
3  repeat
4      Receive $\mathbf{w}(t), \tau, s_i$ from the server;
5      $t_0 \leftarrow t$;
6      if $t_0 > 0$ then
7          Compute $F_{i,\mathcal{S}_i}(\mathbf{w}(t))$ and $\nabla F_{i,\mathcal{S}_i}(\mathbf{w}(t))$;
8          $c_i \leftarrow \|\nabla F_{i,\mathcal{S}_i}(\mathbf{w}(t))\|^2 / 2F_{i,\mathcal{S}_i}(\mathbf{w}(t))$;
9          $\rho_i \leftarrow$
             $\|F_{i,\mathcal{S}_i}(\mathbf{w}_i(t)) - F_{i,\mathcal{S}_i}(\mathbf{w}(t))\| / \|\mathbf{w}_i(t) - \mathbf{w}(t)\|^2$ ;
10         $\beta_i \leftarrow$
             $\|\nabla F_{i,\mathcal{S}_i}(\mathbf{w}_i(t)) - \nabla F_{i,\mathcal{S}_i}(\mathbf{w}(t))\| / \|\mathbf{w}_i(t) - \mathbf{w}(t)\|$ ;
11     $\mathbf{w}_i(t) \leftarrow \mathbf{w}(t)$;
12     for $r = 1, 2, ..., \tau$ do
13         $t \leftarrow t+1$;
14         Execute local update according to (4);
15     Record the average computation time per iteration $t_{ci}$
         and estimate the computing capacity by $p_i = s_i / t_{ci}$;
16     Send $\mathbf{w}_i(t), p_i$ to the parameter server;
17     if $t_0 > 0$ then
18         Send $\rho_i, \beta_i, c_i, F_{i,\mathcal{S}_i}(\mathbf{w}(t_0))$, and $\nabla F_{i,\mathcal{S}_i}(\mathbf{w}(t_0))$
             to the PS ;
19 until $STOP$ flag is received;
```

parameters c_i, t_{ui} and those in (19) and (20) on the fly, as elaborated in Section VI-B.

B. Online Parameter Estimation

To simplify the problem (20), we first set $F^* = 0$ as it is impossible to accurately evaluate it for model training. We then approximate the first term in (19), i.e. $F(\mathbf{w}^{(k-1)}) = \frac{\sum_{i=1}^N D_i F_i(\mathbf{w}^{(k-1)})}{D} \approx \frac{\sum_{i=1}^N D_i F_{i,\mathcal{S}_i}(\mathbf{w}^{(k-1)})}{D} \triangleq \hat{F}(\mathbf{w}^{(k-1)})$ by replacing the local loss $F_i(\cdot)$ with the batch loss $F_{i,\mathcal{S}_i}(\cdot)$, since it can be quite time-consuming to calculate the exact value of $F_i(\mathbf{w}^{(k-1)})$, especially for a large number of data samples.

For ρ, β, c, and δ, we evaluate them in two steps. First, each client estimates these parameters ρ_i, β_i, c_i, and $F_{i,\mathcal{S}_i}(\mathbf{w}(t))$ using the global model $\mathbf{w}(t)$ just received at the beginning of every round k before synchronizing their local model $\mathbf{w}_i(t)$ with the global model. Since the training can take a large number of iterations, e.g., 10^5, the estimates based on taking the average of empirical measurements will be accurate, at least in probability converging to their true expectations, according to the law of large numbers. One can also pick a good online estimation approach, such as OMD, FTRL, and bandits methods [42], which are not the focus of this work and thus left for future work. Then the clients will send these results back to the PS to calculate ρ, β, c, and δ as a weighted average of ρ_i, β_i, c_i, and δ_i. Note that these parameter estimates do not expose extra information of clients' raw data compared to sending the computed gradients. Finally, we set $\mu = \mu_G = 1$ for simplicity in experiment, since $\nabla F_{i,\mathcal{S}_i}(\mathbf{w}, \xi_t)$ can be seen as an unbiased estimate of $\nabla F_i(\mathbf{w})$ and our objective function (19) can be approximated by the following error bound:

$$q^{\tau_k} \hat{F}(\mathbf{w}^{(k-1)}) + \frac{\beta \eta^2 (1 - q^{\tau_k})}{2D^2(1-q)} \sum_{i \in \mathcal{N}} \frac{M_i D_i^2}{s_{ik}} + \rho h(\tau_k)^2. \quad (21)$$

C. The workflow of our adaptive control algorithm

In this subsection, we present our Online Co-Optimization based FL algorithm, named **AdaCoOpt**, for the PS (Algorithm 2) and clients (Algorithm 3) to solve our refined batch size and aggregation frequency co-optimization problem shown in (20).

At the beginning of the model training, the PS initializes the allowed remaining time to finish the task (denoted by θ_c) to be the deadline θ, the remaining cost budget R_c to be the total budget R, the current time step t to be zero, the number of local updates per round τ_1 to be one, and the model weights to be $\mathbf{w}(0) = \mathbf{0}$. The batch size configuration $\mathbf{s_1}$ is initialized to be a uniform value for each client. In each aggregation round k, the server sends the global model $\mathbf{w}(t)$, number of local updates τ_k, and batch sizes s_{ik} of round k to the corresponding clients. Besides, it estimates the unknown parameters of the FL model (e.g., ρ, β, and c) and the edge network (e.g., θ_c and R_c) (Lines 9–13 of Algorithm 2) after receiving estimated parameters from all the clients.

On the client side, each device first estimates M_i through pre-run tests over its local dataset. Then it performs local updates and uploads the local model $\mathbf{w}_i(t)$ along with the estimated $c_i, \rho_i, \beta_i, p_i, F_{i,\mathcal{S}_i}(\mathbf{w}(t_0))$, and $\nabla F_{i,\mathcal{S}_i}(\mathbf{w}(t_0))$ to the PS (Lines 7–18 of Algorithm 3). Finally, the server will perform an aggregation step to update the global model and adopt our **CoOptFL** (Algorithm 1) with the estimated parameters to compute τ_{k+1} and $\mathbf{s}_{k+1}$ for all clients in the next round using the remaining budget R_c and θ_c (Line 15 in Algorithm 2). The key is to utilize the marginal error bound (21) instead of the cumulative error bound (11) when using our subroutine algorithm **CoOptFL**. It finally outputs the optimal solution of $(\tau_{k+1}, \mathbf{s}_{k+1})$, which is the combination of τ and s that minimizes the value of (21) for the next aggregation round.

VII. Experimental Validation

In this section, we validate our theories and proposed algorithms in three parts: 1) Offline optimal local update step τ and uniform batch size s; 2) Optimal batch size assignment in CoOptFL (Algorithm 1); 3) Online adaptive control algorithm AdaCoOpt (Algorithms 2 and 3) presented in Section VII-B.

A. Experiment setup

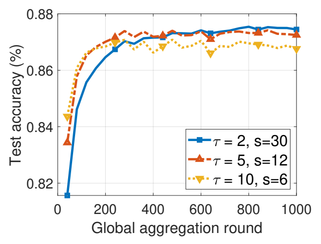
1) Testbed: To simulate the system heterogeneity, we first conduct our experiments in a *small-scale testbed* with various types of edge devices, including 1 laptop PC (CPU: Intel i5-7300HQ 4-core @2.50GHz), 1 desktop PC (CPU: Intel i5-1135G7 8-core @2.40GHz), and 3 docker containers [43] launched from a workstation. We manually assign different numbers of CPU cores (3, 6, 12) to each container. The PS instance is deployed on the container with the most CPU cores, while the rest of the containers and devices are used as clients.

To further evaluate our proposed algorithms **CoOptFL** and **AdaCoOpt**, we conduct two *larger scale experiments*: 1) 100 clients simulated in a lab server cluster; and 2) a 20-client testbed deployed at 20 geo-distributed VM instances rented from Hetzner [44], including six 1-vCPU instance (2GB RAM, 20GB storage), seven 2-vCPU instances (4GB RAM, 40GB storage), and seven 4-vCPU instances (8GB RAM, 80GB storage) for reflecting computational heterogeneity among clients. We deploy our PS on one of the 4-core instances.

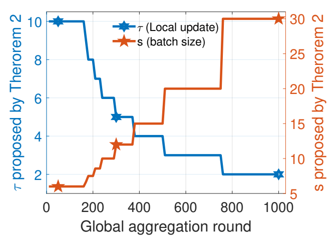
2) Models and datasets: We implement all the FL training models with TensorFlow. We use MNIST and CIFAR-10 datasets to train a convex SVM model and a non-convex CNN model. We adopt a similar non-i.i.d. data distribution setting in [8] to simulate data heterogeneity among clients. To evaluate our (offline) algorithm **CoOptFL**, we initialize its input parameters ($t_{ui}, M_i, \rho, c, \beta, \delta, p_i$, and t_{ci}) using the same estimation method as in our online algorithm **AdaCoOpt**.

3) Baselines: To demonstrate the effectiveness of our carefully chosen batch size configurations for different clients using **CoOptFL**, we compare with **Uniform**, a widely-adopted method with uniform batch size [8] for all the clients, and with **No-straggler**, a time-efficient strategy proposed by [5].

To evaluate the performance of our online algorithm **AdaCoOpt** under imperfect estimation of the model and system



(a) Test Accuracy



(b) τ, s proposed in Theorem 2

Fig. 2. Optimal τ and s (squared-SVM, MNIST)

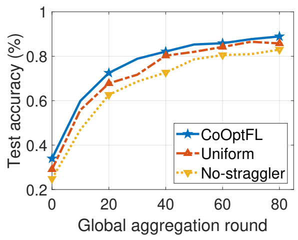
parameters, we compare with **FedAvg**, which maintains τ and batch size unchanged after their initialization, an adaptive algorithm **Dynamic-τ** proposed by [8], and the **No-straggler** algorithm provided in [5].

4) Parameters and run-time traces of the FL training: We initialize models with $\mathbf{w}(0) = \mathbf{0}$ and set the default local update step $\tau = 2$, mini-batch size $s = 60$, and step size $\eta = 0.005$ unless otherwise specified. To evaluate the training cost and the completion time, we set the computation cost per sample $a = 0.0005$ and the communication cost per round $b = |\mathcal{N}|/10$. On each of our testbeds, the clients are training the same FL model, but they have different resource configurations and run-times. In particular, the run-time logs of the 5-client and 20-client experiments are real; but in the 100-client simulation, we sample the run-time of each client from the run-time trace collected at the 20 VM instances located in different edge clusters to simulate the real-world communication and computation overhead.

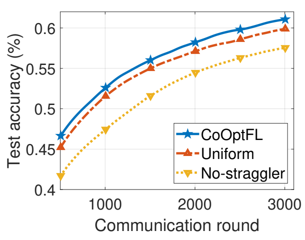
B. Experimental results and interpretation

1) **Optimal number of local updates per round τ and uniform batch size s:** We find the optimal combination τ^* and s^* using our Theorem 2 for a squared-SVM model training and testing under the MNIST dataset. We compare three different combinations: $\tau = 10, s = 6$; $\tau = 5, s = 12$; $\tau = 2, s = 30$. Fig. 2a shows the optimal τ and s combination varying K, e.g., $(\tau = 10, s = 6)$ for $K < 50$, $(\tau = 5, s = 12)$ for $K \approx 300$, and $(\tau = 2, s = 30)$ for $K > 800$ achieve the highest accuracy respectively. We also mark the optimal τ and batch size proposed in our Theorem 2 at the corresponding rounds in Fig. 2b for better visualization. Besides, Fig. 2b shows that the optimal τ decreases with the increase of K, supporting our theoretical result in Remark 1.

2) **Optimal heterogeneous batch sizes s across clients:** We compare our offline algorithm **CoOptFL** to **No-straggler** [5], which configures batch sizes s_i according to clients' computing capacities ($s_i \propto p_i$) so as to eliminate the straggler effect across clients, and **Uniform** ($s = 60$). We set the communication round $K = 80$ and $K = 3000$ for the MNIST (SVM) and CIFAR-10 datasets (CNN), respectively. For fairness, we set the total batch size $s_{tot} = \sum_{i \in \mathcal{N}} s_i$ as a constant to ensure that clients will process the same amount of data samples in total while using different batch size assignment strategies. Fig. 3 shows that **CoOptFL** can converge faster

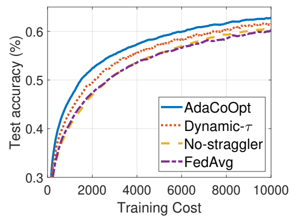


(a) Accuracy (MNIST) 5-client

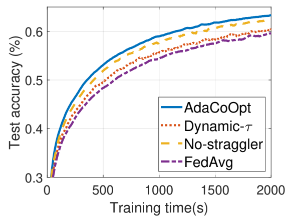


(b) Accuracy (CIFAR) 20-client

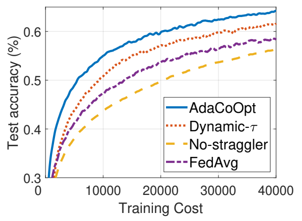
Fig. 3. Our batch size assignment in offline algorithm **CoOptFL** achieves the highest accuracy for both datasets



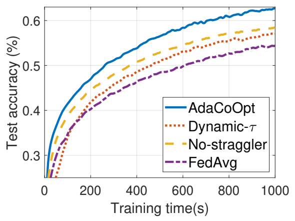
(a) Cost-dominant (20-client) $R = 10000, \theta = 5000s$



(b) Time-dominant (20-client) $R = 20000, \theta = 2000s$

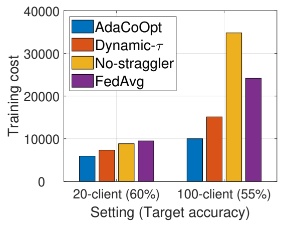


(c) Cost-dominant (100-client) $R = 40000, \theta = 5000s$

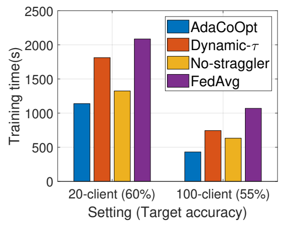


(d) Time-dominant (100-client) $R = 80000, \theta = 1000s$

Fig. 4. **AdaCoOpt** achieves the highest accuracy under CIFAR-10 in both cost-sensitive and time-sensitive scenarios

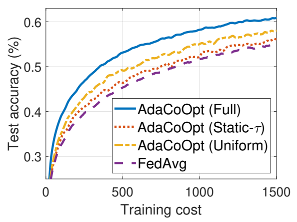


(a) Cost-dominant scenario

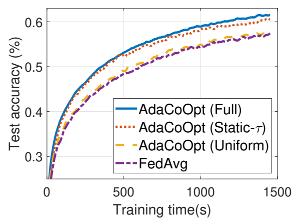


(b) Time-dominant scenario

Fig. 5. Our online algorithm **AdaCoOpt** consumes minimal cost and time to achieve the target accuracy under CIFAR-10 in both cost-sensitive and time-sensitive scenarios



(a) Cost-dominant (20-client)



(b) Time-dominant (100-client)

Fig. 6. Optimizing the aggregation frequency (resp. batch size) has the greatest effect in cost- (resp. time-) dominant scenarios

and achieve better final testing accuracy compared to the two baselines in both 5-client and 20-client settings. Note that **No-straggler** always tends to assign bigger batch sizes to devices with higher computing capacities regardless of their non-i.i.d. data properties, which leads to lower model accuracy and resource utilization than **Uniform**, especially when devices with higher computing capacity have fewer and similar data samples (Theorem 3). Similar results can be found in Fig. 4c and Fig. 5a in the following online experiments as well.

3) **Adaptive control for co-optimized aggregation frequency and heterogeneous batch sizes:** We further compare our **AdaCoOpt** with three benchmarks for CIFAR-10 FL training: the first two are vanilla FedAvg [3] and the time-efficient **No-straggler** [5]; the third one is **Dynamic-τ** [8] which dynamically adjusts τ_k for each round k. We compare the strategies in two different scenarios of our optimization problem, where the cost constraint and time constraint dominates, respectively. We set different values of R and θ to simulate these two different scenarios. Fig.4 and Fig.5 together show that **AdaCoOpt** can outperform the baselines in both scenarios under different settings. For instance, in the cost-dominant scenario, **AdaCoOpt** can achieve a 2.7%–7.9% higher final test accuracy than **FedAvg** and reduce the cost by 37.6%–58% when achieving the same accuracy. In the time-dominant scenario, it achieves a 3.8%–8.4% higher final test accuracy and 45.4%–59.6% less completion time if achieving the same accuracy. These results indicate the great adaptability of **AdaCoOpt**.

Moreover, we conduct ablation experiments on both 20-client and 100-client settings to test the value of co-optimizing τ_k and s_{ik} of our **AdaCoOpt** in both cost-dominant and time-dominant scenarios. We compare our **AdaCoOpt** with **AdaCoOpt (Static-τ)**, which only optimizes the batch sizes using a fixed aggregation frequency and **AdaCoOpt (Uniform)**, which uses a uniform batch size among clients, only adjusting the local update steps adaptively. Fig. 6a shows that a timely adjusted global aggregation frequency (**AdaCoOpt (Uniform)**) can effectively reduce the training (communication) cost and thus is more critical in a cost-dominant training scenario. On the other hand, Fig. 6b shows that a careful batch size assignment (**AdaCoOpt (Static-τ)**) can well capture the system and data heterogeneity so as to achieve a better model accuracy in a time-dominant scenario. These results also match our experiments in Fig. 4, where **Dynamic-τ** performs better than **No-straggler** in the cost-sensitive scenario, but worse than **No-straggler** in the time-sensitive scenario.

VIII. Conclusion

This work proposes a novel framework to quantify and leverage the interplay of the number of local update steps and heterogeneous batch sizes across clients for federated learning performed at distributed edge devices. Technically, we derive a novel convergence bound with respect to those control variables and analyze the performance metrics of cost and training time as well. We then provide closed-form solutions for our joint optimization in a typical case and propose an efficient exact algorithm for the general case. Our strategies consider both heterogeneous system characteristics and non-i.i.d. data, which can improve the common strategies that FL practitioners adopt. Moreover, we adapt our offline strategy to dynamically adjust the decisions on the fly, with superiority of several performances demonstrated in extensive experiments.

REFERENCES

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. of Artificial intelligence and statistics*, 2017.

[2] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine Learning and Systems*, vol. 2, pp. 429–450, 2020.

[3] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *Proc. of International Conference on Learning Representations*, 2019.

[4] J. Liu, H. Xu, L. Wang, Y. Xu, C. Qian, J. Huang, and H. Huang, "Adaptive asynchronous federated learning in resource-constrained edge computing," *IEEE Transactions on Mobile Computing*, early access, 2021.

[5] Z. Ma, Y. Xu, H. Xu, Z. Meng, L. Huang, and Y. Xue, "Adaptive batch size for federated learning in resource-constrained edge computing," *IEEE Transactions on Mobile Computing*, early access, 2021.

[6] X. Zhang, J. Wang, G. Joshi, and C. Joe-Wong, "Machine learning on volatile instances," in *Proc. of IEEE INFOCOM*, 2020.

[7] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu, "Gaia:{Geo-Distributed} machine learning approaching {LAN} speeds," in *Proc. of USENIX NSDI*, 2017.

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

[9] X. Mo and J. Xu, "Energy-efficient federated edge learning with joint communication and computation design," *Journal of Communications and Information Networks*, vol. 6, no. 2, pp. 110–124, 2021.

[10] Q. Zeng, Y. Du, K. Huang, and K. K. Leung, "Energy-efficient resource management for federated edge learning with cpu-gpu heterogeneous computing," *IEEE Transactions on Wireless Communications*, vol. 20, no. 12, pp. 7947–7962, 2021.

[11] Y. Ruan, X. Zhang, and C. Joe-Wong, "How valuable is your data? optimizing client recruitment in federated learning," in *Proc. of WiOpt*, 2021.

[12] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "Oort: Efficient federated learning via guided participant selection," in *Proc. of USENIX OSDI*, 2021.

[13] B. Luo, W. Xiao, S. Wang, J. Huang, and L. Tassiulas, "Tackling system and statistical heterogeneity for federated learning with adaptive client sampling," *arXiv preprint arXiv:2112.11256*, 2021.

[14] S. Tyagi and P. Sharma, "Taming resource heterogeneity in distributed ml training with dynamic batching," in *Proc. of 2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*, 2020.

[15] Z. Charles, Z. Garrett, Z. Huo, S. Shmulyian, and V. Smith, "On large-cohort training for federated learning," in *Proc. of NeurIPS*, 2021.

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

[17] L. Zhu, H. Lin, Y. Lu, Y. Lin, and S. Han, "Delayed gradient averaging: Tolerate the communication latency for federated learning," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[18] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization," in *Proc. of AISTATS*, 2020.

[19] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," *Advances in Neural Information Processing Systems*, vol. 33, pp. 3557–3568, 2020.

[20] L. Wang, W. Wang, and B. LI, "Cmfl: Mitigating communication overhead for federated learning," in *Proc. of IEEE ICDCS*, 2019.

[21] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Communication-efficient sgd via gradient quantization and encoding," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[22] S. Liu, G. Yu, R. Yin, J. Yuan, and F. Qu, "Adaptive batchsize selection and gradient compression for wireless federated learning," in *Proc. of IEEE GLOBECOM*, 2020.

[23] J. Zhang, S. Guo, Z. Qu, D. Zeng, Y. Zhan, Q. Liu, and R. A. Akerkar, "Adaptive federated learning on non-iid data with resource constraint," *IEEE Transactions on Computers*, 2021.

[24] Y. Ruan, X. Zhang, S.-C. Liang, and C. Joe-Wong, "Towards flexible device participation in federated learning," in *Proc. of AISTATS*, 2021.

[25] J. Cipar, Q. Ho, J. K. Kim, S. Lee, G. R. Ganger, G. Gibson, K. Keeton, and E. Xing, "Solving the straggler problem with bounded staleness," in *Proc. of 14th Workshop on Hot Topics in Operating Systems (HotOS XIV)*, 2013.

[26] Q. Ma, Y. Xu, H. Xu, Z. Jiang, L. Huang, and H. Huang, "Fedsa: A semi-asynchronous federated learning mechanism in heterogeneous edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3654–3672, 2021.

[27] J. Park, D. Yoon, S. Yeo, and S. Oh, "Amble: Adjusting mini-batch and local epoch for federated learning with heterogeneous devices," *Journal of Parallel and Distributed Computing*, vol. 170, pp. 13–23, 2022.

[28] D. Shi, L. Li, M. Wu, M. Shu, R. Yu, M. Pan, and Z. Han, "To talk or to work: Dynamic batch sizes assisted time efficient federated learning over future mobile edge devices," *IEEE Transactions on Wireless Communications*, vol. 21, no. 12, pp. 11 038–11 050, 2022.

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

[30] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *Proc. of IEEE INFOCOM*, 2020.

[31] W. Shi, S. Zhou, and Z. Niu, "Device scheduling with fast convergence for wireless federated learning," in *Proc. of IEEE ICC*, 2020.

[32] B. Luo, X. Li, S. Wang, J. Huang, and L. Tassiulas, "Cost-effective federated learning design," in *Proc. of IEEE INFOCOM*, 2021.

[33] J. Yuan, M. Xu, X. Ma, A. Zhou, X. Liu, and S. Wang, "Hierarchical federated learning through lan-wan orchestration," *arXiv preprint arXiv:2010.11612*, 2020.

[34] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green ai," *Communications of the ACM*, vol. 63, no. 12, pp. 54–63, 2020.

[35] C. Li, D. Y. Li, G. Miklau, and D. Suciu, "A theory of pricing private data," *Communications of the ACM*, vol. 60, no. 12, 2017.

[36] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," in *Proc. of NeurIPS*, 2020.

[37] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

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

[39] H. Karimi, J. Nutini, and M. Schmidt, "Linear convergence of gradient and proximal-gradient methods under the polyak-lojasiewicz condition," 2020. [Online]. Available: https://arxiv.org/pdf/1608.04636.pdf

[40] "Adacoopt: Leverage the interplay of batch-size and aggregation frequency in federated learning," https://www.dropbox.com/s/myo3j1vet9yy6sk/AdaCoOpt-IWQOS-TR.pdf?dl=0.

[41] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2022. [Online]. Available: https://www.gurobi.com

[42] E. Hazan *et al.*, "Introduction to online convex optimization," *Foundations and Trends® in Optimization*, vol. 2, no. 3-4, pp. 157–325, 2016.

[43] D. Merkel *et al.*, "Docker: lightweight linux containers for consistent development and deployment," *Linux journal*, vol. 2014, no. 239, p. 2, 2014.

[44] Hetzner Online GmbH. [Online]. Available: https://www.hetzner.com/cloud