FedCau: A Proactive Stop Policy for Communication and Computation Efficient Federated Learning

Afsaneh Mahmoudi[1], Hossein S. Ghadikolaei[2], José Mairton Barros Da Silva Júnior[1,3], and Carlo Fischione[1]

Abstract

This paper investigates efficient distributed training of a Federated Learning (FL) model over a wireless network of wireless devices. The communication iterations of the distributed training algorithm may be substantially deteriorated or even blocked by the effects of the devices' background traffic, packet losses, congestion, or latency. We abstract the communication-computation impacts as an 'iteration cost' and propose a cost-aware causal FL algorithm (FedCau) to tackle this problem. We propose an iteration-termination method that trade-offs the training performance and networking costs. We apply our approach when workers use the slotted-ALOHA, carrier-sense multiple access with collision avoidance (CSMA/CA), and orthogonal frequency-division multiple access (OFDMA) protocols. We show that, given a total cost budget, the training performance degrades as either the background communication traffic or the dimension of the training problem increases. Our results demonstrate the importance of proactively designing optimal cost-efficient stopping criteria to avoid unnecessary communication-computation costs to achieve a marginal FL training improvement. We validate our method by training and testing FL over the MNIST and CIFAR-10 dataset. Finally, we apply our approach to existing communication efficient FL methods from the literature, achieving further efficiency. We conclude that cost-efficient stopping criteria are essential for the success of practical FL over wireless networks.

Index Terms

Federated learning, communication protocols, cost-efficient algorithm, latency, unfolding federated learning.

I. Introduction

The recent success of artificial intelligence and large-scale machine learning heavily relies on the advancements of distributed optimization algorithms [1]. The main objective of such algorithms is better training/test performance for prediction and inference tasks, such as image

[1]The authors are with the School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden (Email: {afmb, jmbdsj, carlofi}@kth.se).

[2]The author is with Ericsson, Stockholm, Sweden (Email: hossein.shokri.ghadikolaei@ericsson.com).

[3]The author is with Department of Electrical Engineering, Princeton University, Princeton, NJ 08544 USA.

recognition [2]. However, the costs of running the algorithms over a wireless network may hinder achieving the desired training accuracy due to the communication and computation costs. The state-of-the-art of such algorithms requires powerful computing platforms with vast amounts of computational and communication resources. Although such resources are available in modern data centers that use wired networks, they are not easily available at wireless devices due to communication and energy resource constraints. Yet, there is a need to extend machine learning tasks to wireless communication scenarios. Use cases as machine leaning over IoT, edge computing, or public wireless networks serving many classes of traffic [3].

One of these prominent algorithms is Federated Learning (FL), which is a new machine learning paradigm where each individual worker has to contribute in the learning process without sharing its own data to other workers and the master node. Specifically, FL methods refer to a class of privacy-preserving distributed learning algorithms in which individual worker nodes $[M]$ execute some local computations and share only their parameters, with a central controller for global model aggregation [4]. The FL problem consists in optimizing a finite sum of M differentiable functions f_j, $j \in [M]$, which take inputs from $\mathbb{R}^d$ for some positive d and give their outputs in $\mathbb{R}$, i.e., $\{f_j : \mathbb{R}^d \mapsto \mathbb{R}\}_{j \in [M]}$ with corresponding local parameters $\{\boldsymbol{w}^j \in \mathbb{R}^d\}_{j \in [M]}$. The common solution to such a problem involves an iterative procedure wherein at each global communication iteration k, worker nodes have to find the local parameter $\{\boldsymbol{w}_k^j\}_{j \in [M]}$ and upload them to a central controller. Then, the master node updates the model parameters as $\boldsymbol{w}_{k+1}$ and broadcasts it to all the nodes to start the next iteration [5].

The FL algorithm alleviates computation and privacy by parallel computations at worker nodes using their local private data [5]. However, such algorithm introduces a communication cost: parameter vectors, such as weight and bias, must be communicated between the master and the worker nodes to run a new iteration. The weights can be vector of huge sizes whose frequent transmissions and reception may deplete the battery of wireless devices. Therefore, every communication iteration of these algorithms suffers some costs [1], e.g., computation, latency, communication resource utilization, energy. As we argue in this paper, the communication cost can be orders of magnitude larger than the computation costs, thus making the iterative procedure over wireless networks potentially very inefficient. Moreover, due to the *diminishing return* rule [6], the accuracy improvement of the final model gets smaller with every new iteration. Yet, it is necessary to pay an expensive communication cost to run every new communication iteration of marginal importance for the training purpose.

In this paper, we investigate the problem of FL over wireless networks to ensure an efficient communication-computation cost. Specifically, we define our FL over wireless networks as follows. We consider a star network topology, and focus on avoiding the extra communication-

[1]Throughout the paper, we use "communication-computation cost" and "iteration cost" interchangeably.

computation cost paid in FL training to attain a marginal improvement. We show that a negligible improvement in training spends valuable resources and hardly results in tests accuracy progress. We propose novel and causal cost-efficient FL algorithms (FedCau) for both convex and non-convex loss functions. We show the significant performance improvements introduced by FedCau through experimental results, where we train the FL model over the wireless networks with slotted-ALOHA, CSMA/CA and OFDMA protocols. We apply FedCau on top of two well-known communication-efficient methods, Top-q [7], and LAQ [8] and the results show that FedCau algorithms further improve the communication efficiency of other communication-efficient methods from the literature. Our extensive results show that the FedCau methods can save the valuable resources one would spend through unnecessary iterations of FL, even when applied on top of existing methods from literature focusing on resource allocation problems [5], [9]–[11].

A. Literature Survey

Cost-efficient distributed training is addressed in the literature through communication-efficiency [4], [12]–[20] or tradeoff between computation and communication primarily by resource allocation [11], [21]. Mainly, we have two classes of approaches for communication-efficiency in the literature focusing on: 1) data compression, like quantization and sparsification of the local parameters in every iteration, and 2) communication iteration reduction.

The first class of approaches focuses on data compression, which reduces the amount of information exchanged in bits among nodes, thereby saving communication resources. However, we may need more iterations to compensate for quantization errors than the unquantized version. Recent studies have shown that proper quantization approaches, together with some error feedback, can maintain the convergence of the training algorithm and the asymptotic convergence rate [12]–[14]. However, the improved convergence rates depend on the number of iterations, thus, requiring more computation resources to perform those iterations. Sparsification is an alternative approach to quantization to reduce the amount of exchanged data for running every iteration [15]. A prominent example of this approach is top-q sparsification, where a node sends only the q most significant entries, such as the ones with the highest modulus, of the stochastic gradient [12], [16].

The second class of approaches focuses on reduction of the communication iterations by eliminating the communication between some of the workers and the master node in some iterations [17]. The work [17] has proposed lazily aggregated gradient (LAG) for communication-efficient distributed learning in master-worker architectures. In LAG, each worker reports its gradient vector to the master node only if the gradient changes from the last communication iteration are large enough. Hence, some nodes may skip sending their gradients at some iterations, which saves communication resources. LAG has been extended in [18] by sending quantized versions of the gradient vectors. In [20], local SGD techniques reduce the number of communication rounds needed to solve an optimization problem. In a generic FL setting, adding more local computations may reduce the need for frequent global aggregation, leading to a lower communication overhead [4]. Moreover, it allows the master node to update the global model with only a (randomly chosen) subset of the nodes at every iteration, which may further reduce the communication overhead and increase the robustness. The work in [19] has improved the random selection of the nodes and proposed the notion of significance filter, where each worker updates its local model and transmits it to the master node only when there is a significant change in the local parameters. In [22], the notion of local drift has been introduced as a measure to trigger an upload to the master node. Moreover, [19] has showed that adding a memory unit at the master node and using ideas from SAGA [23] reduce the upload frequency of each worker node, thus improve the communication efficiency.

The aforementioned two classes still leave significant space to further reduce the actual cost of running a distributed training algorithm and its adaptation to the underlying wireless communication protocol. The same compression or model averaging algorithm may exhibit completely different values for latency or energy consumption in different communication settings. The main problem is that the two classes mentioned above assume that the complexity of an iterative algorithm is given by the number of bits per communication round, or the number of communication rounds [9]. However, they neglect other very important costs involved in solving an FL problem: overall latency and communication resource consumption. These costs become of paramount importance when we implement FL on bandwidth or battery-limited wireless networks, where latency (e.g., wireless automation) [3] and energy consumption (e.g., common smartphones or low power Internet-of-Things) [21] may render useless the distributed algorithm and the ultimate solution.

There have been some recent works on the co-design of optimization problems and communication networks, mainly for the task of computational offloading. Reference [10] has considered the problem of task offloading using nearby edge devices to reduce the service delay, considering the loads and energy of the cloudlets. Yang *et. al.* [11] focused on optimizing the resource allocation for running FL. Similarly, reference [24] considers the joint learning of wireless resource allocation and the user selection problem as an optimization problem to minimize an FL loss function that captures the performance of the FL algorithm. Different from the literature, we propose to proactively design the stopping criteria to optimize such tradeoff. Our approach is original because it contrasts with the state-of-the-art algorithms where stopping criteria are treated as a hyper-parameter and manually set via heavy cross-validations.

In our preliminary works, we have characterized the overall communication-computation of solving a distributed gradient descent problem where the worker nodes had background traffic and followed a channel from medium access control (MAC) protocols using random access, such as slotted-ALOHA [25] or CSMA/CA [26] in the uplink. Going beyond such papers, to achieve a cost-aware training workflow, we need to consider the diminishing return rule of the optimization algorithms, which reveals that as the number of iterations increases, the improvement in training accuracy decreases. Then, we need to balance iteration-cost and the achievable accuracy before the algorithm's design phase. This paper constitutes a major step to address this important research gap. Previously in [25], [26], we proposed a cost-efficient framework considering the cost of each iterations of gradient descent algorithms along with minimizing a convex loss function. However, the theory of these papers was only limited to convex loss functions, the iteration costs did not consider FedAvg algorithm and the computation latency, and there was no adequate study between the achievable test accuracy and the iteration costs. Hence, this paper proposes a new and original study compared to our preliminary works by

1) Considering FedAvg algorithm;
2) Assuming both convex and non-convex loss functions;
3) Developing a novel theoretical framework for FedAvg that includes the communication-computation costs.

We apply the proposed framework to several wireless communication protocols, and other communication-efficient algorithms for which we show original training and testing results.

B. Contributions

We investigate the trade-off between achievable FL loss and the overall communication-computation cost of running the FL over wireless networks as an optimization problem. This work focuses on training a cost-efficient FedAvg algorithm in a "causal way", meaning that our approach does not require the future information of the training to decide how much total cost, e.g., computation, latency, or communication energy, the training algorithm needs to spend before terminating the iterations. Different than our approach, most papers in the literature aim to train the FedAvg algorithm in resource-constraint conditions and propose the best resource allocation policies "before" performing the training [11], [21]. These approaches rely mainly on approximating the future training information by using some lower and upper bounds of that information. In this work, we propose to train the FedAvg algorithm in a causal, communication, and computation efficient way. To this end, we utilize the well-known multi-objective optimization approach according to the scalarization procedure in [27]. Therefore, we propose FedCau to improve the FedAvg algorithm by training in a cost-efficient manner without any need to know the future training information or any upper and lower bounds on them. To the best of our knowledge, this is the first work that considers such causal approaches to train the FedAvg algorithm on a communication and computation efficient manner. The main contributions of this work are summarized as:

- We propose a new multi-objective cost-efficient optimization that trades off model performance and communication costs for an FL training problem over wireless networks;
- We develop three novel causal solution algorithms, named FedCau, to the multi-objective optimization above, one with focus on original FL and the others with focus on stochastic FL. We establish the convergence of these algorithms for FL training problems using both convex and non-convex loss functions;
- We investigate the training and test performance of the proposed algorithms using real-world datasets, such as MNIST and CIFAR-10, over the communication protocols: slotted-ALOHA, CSMA/CA and OFDMA. We consider these protocols because they are the dominant communication protocols in most wireless local area networks, such as IEEE 802.11-based products [28], or fixed assignment access protocol like OFDMA [29];
- We apply our proposed FedCau on top of the top-q sparsification and lazily aggregated quantized gradient (LAQ) methods showing the vast application of our approach [7], [8].
- The experimental results highlight the ability of our proposed FedCau to achieve an efficient and accurate training and we conclude that a co-design of distributed optimization algorithms and communication protocols are essential for the success of cost-efficient FL over wireless networks, including its applications to edge computing and IoT.

The rest of this paper is organized as follows. Section II describes the general system model and problem formulation. In Section III, we derive some useful results and propose our non-causal and causal FL algorithms (FedCau), which are by design intended to run over communication networks. In the analysis, we consider both convex and non-convex loss functions. In Section IV, we apply our algorithms to slotted-ALOHA, CSMA/CA, and OFDMA. In Section V, we analyze the performance of the FedCau algorithms. We then conclude the paper in Section VI. We moved all the proofs and extra materials to the Appendix.

Notation: Normal font w, bold font small-case $\boldsymbol{w}$, bold-font capital letter $\boldsymbol{W}$, and calligraphic font $\mathcal{W}$ denote scalar, vector, matrix, and set, respectively. We define the index set $[N] = \{1, 2, \dots, N\}$ for any integer N. We denote by $\|\cdot\|$ the l_2-norm, by $|\mathcal{A}|$ the cardinality of set $\mathcal{A}$, by $[\boldsymbol{w}]_i$ the entry i of vector $\boldsymbol{w}$, and by $\boldsymbol{w}^{\mathrm{T}}$ the transpose of $\boldsymbol{w}$.

II. System Model and Problem Formulation

In this section, we represent the system model and the problem formulation. First, we discuss the FedAvg algorithm, and afterwards, we propose the main approach of this paper.

A. Federated Learning

Consider a star network of M worker nodes that cooperatively solve a distributed training problem involving a loss function $f(\boldsymbol{w})$. Consider D as the whole dataset distributed among each worker $j \in [M]$ with D_j data samples. Let tuple $(\boldsymbol{x}_{ij}, y_{ij})$ denote data sample i of $|D_j|$ samples of worker node j and $\boldsymbol{w} \in \mathbb{R}^d$ denote the model parameter at the master node. Considering $\sum_{j=1}^{M} |D_j| = |D|$, and $j, j' \in [M]$, $j \neq j'$, we assume $D_j \cap D_{j'} = \emptyset$, and defining $\rho_j := |D_j|/|D|$, we formulate the following training problem

$$\boldsymbol{w}^* \in \arg \min_{\boldsymbol{w} \in \mathbb{R}^d} f(\boldsymbol{w}) = \sum_{j=1}^{M} \rho_j f_j(\boldsymbol{w}), \tag{1}$$

where $f_j(\boldsymbol{w}) := \sum_{i=1}^{|D_j|} f(\boldsymbol{w}; \boldsymbol{x}_{ij}, y_{ij})/|D_j|$. Optimization problem (1) applies to a large group of functions as convex and non-convex (such as a deep neural networks).

The standard iterative procedure to solve problem (1) with the initial vector $\boldsymbol{w}_0$ is

$$\boldsymbol{w}_k = \sum_{j=1}^{M} \rho_j \boldsymbol{w}_k^j, \quad k = 1, \ldots, K. \tag{2}$$

For a differentiable loss function $f(\boldsymbol{w})$, we choose to perform (2) by the Federated Averaging (FedAvg) algorithm.

Initializing the training process with $\boldsymbol{w}_0$, Federated Averaging (FedAvg) is a distributed learning algorithm in which the master node sends $\boldsymbol{w}_{k-1}$ to the workers at the beginning of each iteration $k \geq 1$. Every worker $j \in [M]$ performs a number E of local iterations, $i = 1, \ldots, E$, of stochastic gradient descent [7] with data subset of $\xi_k^j \leq |D_j|$, and computes its local parameter $\boldsymbol{w}_{i,k}^j$, considering the initial point of $\boldsymbol{w}_{0,k}^j = \boldsymbol{w}_{k-1}$, [30]

$$\boldsymbol{w}_{i,k}^j \leftarrow \boldsymbol{w}_{i-1,k}^j - \frac{\alpha_k}{\xi_k^j} \sum_{n=1}^{\xi_k^j} \nabla_{\boldsymbol{w}} f(\boldsymbol{w}_{i-1,k}^j; \boldsymbol{x}_{nj}, y_{nj}), k = 1, \ldots, K, \tag{3}$$

where $\boldsymbol{w}_k^j = \boldsymbol{w}_{E,k}^j$. Then each worker transmits $\boldsymbol{w}_k^j$ to the master node for updating $\boldsymbol{w}_k$ according to (2). Note that in FedAvg, when $E = 1$ and we use the exact gradient vector on the place of the stochastic gradient, we achieve the basic FL algorithm. Considering the FedAvg solver (3) for the updating process in (2), and without enforcing convexity for $f(\boldsymbol{w})$, we use the following Remark throughout the paper.

Remark 1. *[Theorem 11.7 of [31]] Consider any differentiable loss function $f(\boldsymbol{w}) : \mathbb{R}^d \mapsto \mathbb{R}$ with Lipschitz continuous gradient $\nabla_{\boldsymbol{w}} f(\boldsymbol{w})$, i.e., $\|\nabla_{\boldsymbol{w}} f(\boldsymbol{w}_1) - \nabla_{\boldsymbol{w}} f(\boldsymbol{w}_2)\| \leq L\|\boldsymbol{w}_1 - \boldsymbol{w}_2\|$, for some constant $0 < L < \infty$, and let $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k$ be the sequence obtained from the FL algorithm updates in Eq. (2). Then, by $\alpha_k \geq \gamma \|\nabla_w f(\boldsymbol{w}_k)\|^2$ and for an appropriate constant $\gamma > 0$, the following inequality holds: $f(\boldsymbol{w}_1) \geq \ldots \geq f(\boldsymbol{w}_k)$.*

The workers use the FedAvg algorithm (3) to compute their local parameters $\boldsymbol{w}_k^j$, while the master node performs the iterations of (2) until a convergence criteria for $\|f(\boldsymbol{w}_k) - f(\boldsymbol{w}^*)\|$

is met [32]. We denote by K the very first communication iteration at which the stopping criteria of FedAvg algorithm is met, namely

$$K := \text{the first value of } k \mid \|f(\boldsymbol{w}_k) - f(\boldsymbol{w}^*)\| < \epsilon\,, \tag{4}$$

where $\epsilon > 0$ is the decision threshold for terminating the algorithm at communication iteration K and $f(\boldsymbol{w}^*)$ is the optimum of the loss function at the optimal shared parameter $\boldsymbol{w}^*$. At the state-of-the-art literature, the threshold ϵ is independently defined before the training process. However, when we are taking into account the communication-computation costs to solve (1), the threshold must be optimally designed to train the FL model without wasting the available limited communication-computation resources. According to Eq. (4), designing such threshold would require to know $f(\boldsymbol{w}^*)$ before starting the training procedure, which is not realistic. Thus, in this paper we propose an alternative approach that focuses on finding K in (4) we argue in the paper that our approach does not require to know $f(\boldsymbol{w}^*)$ beforehand. Therefore, the central result of this paper consists in finding such K as a function of the communication-computation cost along with the loss function of the FedAvg algorithm (3). We will substantiate this significant result in Section II-B.

Let $c_k > 0$, $k = 1, 2, \ldots$, denote the cost of performing a complete communication iteration k. Accordingly, when we run FedAvg, namely an execution of (2) and (3), the complete training process will cost $\sum_{k=1}^{K} c_k$. Some examples of c_k in real-world applications are:

- *Communication cost*: c_k is the number of bits transmitted in every communication iteration k with or without a fixed packet structure;
- *Energy consumption*: c_k is the energy needed for performing a global iteration to receive $\boldsymbol{w}_k$ at a worker and send back $\{\boldsymbol{w}_k^j\}_{j\in[M]}$ to the master node over the wireless channel;
- *Latency*: c_k is the overall delay to compute and send parameters from and to the workers and the master node over the wireless channel [11].

Considering latency as the iteration cost, the term c_k for running every training iteration of the FedAvg algorithm (3) is generally given by the sum of four components:

1) $\ell_{1,k}$: communication latency in broadcasting parameters by master node;
2) $\ell_{2,k}$: the computation latency in computing $\boldsymbol{w}_k^j$ for every worker node j;
3) $\ell_{3,k}$: communication latency in sending $\boldsymbol{w}_k^j$ to master node;
4) $\ell_{4,k}$: computation latency in updating parameters at the master node.

See Section IV for more detailed modelling of the components of c_k for the slotted-ALOHA, CSMA/CA, and OFDMA protocols.

B. Problem Formulation

To solve optimization problem (1) over a wireless network, the FedAvg algorithm (3) faces two major challenges:

1) *Computation-communication cost*: It does not consider the cost associated with the computation and communication of the local parameters and the reception of the updated model sent by the master node. Clearly, such a cost must be a function of the computation power and resources of the local device, communication protocols, the energy for transmission, and the communication resources in general;
2) *Total number of iterations*: The number of iterations K in (4) for the termination of the training algorithm (3) has a significant role in determining the communication-computation cost of the model training. Therefore, it can be that the number of iterations K in (4) results in huge communication-computation costs without resulting in better training performance. Thus, a lower K would have consumed few resources while obtaining a negligible training optimality degradation.

Thus, the termination iteration K in (4) heavily impacts the overall training costs to solve the optimization problem (1). This behavior indicates that the performance of FedAvg (3) over wireless networks could be catastrophic in terms of communication-computation resource utilization without a good choice of K.

To address the two challenges aforementioned, we propose an original optimization of the termination iteration K in (4) in the FedAvg algorithm (3), where we explicitly consider the cost associated to running training iterations. This is possible by appropriately finding an optimal stopping iteration. To do so, we propose the following optimization problem

$$\begin{aligned} \underset{K}{\text{minimize}} \quad & \left[f(\boldsymbol{w}_K), \sum_{k=1}^{K} c_k \right] & \text{(5a)} \\ \text{subject to} \quad & \boldsymbol{w}_k = \sum_{j=1}^{M} \rho_j \boldsymbol{w}_k^j, \quad k = 1, \dots, K & \text{(5b)} \\ & \boldsymbol{w}_{0,k}^j = \boldsymbol{w}_{k-1}, \quad k = 1, \dots, K & \text{(5c)} \\ & \boldsymbol{w}_{i,k}^j \leftarrow \boldsymbol{w}_{i-1,k}^j - \frac{\alpha_k}{\xi_k^j} \sum_{n=1}^{\xi_k^j} \nabla_{\boldsymbol{w}} f(\boldsymbol{w}_{i-1,k}^j; \boldsymbol{x}_{nj}, y_{nj}), \quad k = 1, \dots, K, & \text{(5d)} \end{aligned}$$

where $\sum_{k=1}^{K} c_k$ quantifies the overall iteration-cost expenditure for the training of loss function $f(\boldsymbol{w})$ when transmitting in a particular wireless channel in uplink. Note that (5a) represents a multi-objective function, which aims at minimizing the training loss function $f(\boldsymbol{w})$ and the overall iteration cost $\sum_{k=1}^{K} c_k$. Note that the values of c_k, for $k \leq K$, can be, in general, a function of the parameter $\boldsymbol{w}_k$, but neither c_k nor $\boldsymbol{w}_k$ are optimization variables of problem (5a). Optimization problem (5) states to devote communication-computation resources as efficiently as possible while performing FedAvg algorithms (3) to achieve an accurate training result for loss function $f(\boldsymbol{w})$. To summarize, by solving optimization problem (5a), we can obtain the optimal number of iterations for FedAvg algorithm (3) which minimize the communication-computation costs while also minimizing the loss function of FedAvg.

Remark 2. *We have formulated optimization problem* (5) *according to the "unfolding method"* *of iterative algorithms [33], where it is ideally assumed that the optimizer knows beforehand (before iterations* (2) *and* (3) *occur) what the cost of each communication iteration in* (2) *would be and when they would terminate. Such an ideal formulation cannot occur in the real world since it assumes knowledge of the future, thus being called* **"non-causal setting"***. However, this formulation is useful because its solution gives the best optimal value of the stopping iteration* k^**. In this paper, we show that we can convert such a non-causal solution of problem* (5) *into a practical algorithm in a so-called* **"causal setting"***. We will show that the solution to the causal setting given by the practical algorithm is very close to* k^**.*

Optimization problem (5) is hard to solve for several reasons: it is multi-objective; it is an integer problem since the variable is K; it has objective and constraint functions that are not analytical functions at K, in other words, K depends on the objective and constrain functions in a non-explicit manner; and most importantly, it is a non-causal problem formulation meaning that the derivation of the optimal K would need to know $\boldsymbol{w}_k$'s before they are available. Generally, it is very challenging to solve these non-explicit and non-causal optimization problems [27]. In the next section, we propose a practical solution to the problem (5).

III. Solution Algorithms

In this section, we first give some preliminary technical results, then propose an iterative solution to optimization problem (5). Subsequently, we show that the proposed methods achieve the optimal or sub-optimal solution and converge in a finite number of steps.

A. Preliminary Solution Steps

In this section, we develop some preliminary results to arrive at a solution to optimization problem (5). We start by transforming (5) according to the scalarization procedure of multi-objective optimization [27]. Specifically, we define the joint communication-computation cost and the loss function of FedAvg algorithm (3) as a scalarization of the overall iteration-cost function $\sum_{k=1}^{K} c_k$ and the loss function $f(\boldsymbol{w}_K)$. Note that such a joint cost is general in the sense that, depending on the values of c_k, it can naturally model many communication-computation costs including constant charge per computation and mission-critical applications.

We transform the multi-objective optimization problem (5) into its scalarized version as

$$k^* \in \arg\min_{K} \quad G(K) \tag{6a}$$

$$\text{subject to} \quad \boldsymbol{w}_k = \sum_{j=1}^{M} \rho_j \boldsymbol{w}_k^j, \quad k = 1, \ldots, K \tag{6b}$$

$$\boldsymbol{w}_{0,k}^j = \boldsymbol{w}_{k-1}, \quad k = 1, \ldots, K \tag{6c}$$

$$\boldsymbol{w}_{i,k}^j \leftarrow \boldsymbol{w}_{i-1,k}^j - \frac{\alpha_k}{\xi_k^j} \sum_{n=1}^{\xi_k^j} \nabla_{\boldsymbol{w}} f(\boldsymbol{w}_{i-1,k}^j; \boldsymbol{x}_{nj}, y_{nj}), \quad k = 1, \ldots, K, \tag{6d}$$

where $G(K)$ and $C(K)$ are defined as

$$G(K) := (\beta C(K) + (1-\beta) f(\boldsymbol{w}_K)), \tag{7}$$

$$C(K) := \sum_{k=1}^{K} c_k. \tag{8}$$

Note that $C(K)$ is the iteration-cost function that represents all the costs the network spends from the beginning of the training until the termination iteration K, and $\beta \in (0,1)$ is the scalarization factor of the multi-objective scalarization method [27].

In the following lemma, we formalize that if $G(K)$ is a monotonically decreasing function of K, we can obtain k^* at which $G(K)$ is minimized.

Lemma 1. *Consider optimization problem* (6). *Let $G(K)$ be a non-increasing function of all $K \leq k^*$. Then, k^* indicates the index at which the sign of discrete derivation [34] of $G(K)$ changes for the first time, i.e.*

$$k^* \in \min\{K | G(K+1) - G(K) > 0\} \tag{9}$$

Proof: See Appendix A-A. □

Now that we have given these preliminary technical results, we are ready to propose in the next section, two algorithms to solve optimization problem (6). First, we describe the non-causal setting to characterize the minimizer. Then, we offer a causal setting to construct algorithms that obtain practical minimizers with convex and non-convex loss functions. Finally, we establish the optimality and convergence of these algorithms.

B. Non-causal Setting

One of the simplest but purely ideal approaches to solve problem (6) is an exhaustive search over the discrete set of $K \in [0, +\infty)$. To this end, we should have in advance, namely at time $k = 0$, the sequence of $(f(\boldsymbol{w}_k))_k$ and $(c_k)_k$ for all $k \in [0, +\infty)$. Although $(c_k)_k$ may be known a priori in some use cases, the sequence of parameters $(\boldsymbol{w}_k)_k$, and consequently the sequence of $(f(\boldsymbol{w}_k))_k$, are not available in advance. For analytical reasons, our non-causal setting assumes that all these values are available at $k = 0$. Although this approach is not practical, we investigate it for the sake of analytical purposes because it gives the ultimate minimizer k^*, which we can use later in the numerical results (see Section V) as a benchmark to assess the performance of the following causal solution algorithms.

C. FedCau for Convex Loss Functions

Here, we propose an approximation of the optimal stopping iteration k^*, called k_c. We will show that k_c can be computed in practice by a causal setting scenario, which is implementable in practice and computationally simple. We will show that, under some conditions, k_c is either

k^* or k^*+1. This means that either $k_c = k^*$ when $k^* = K^{\max}$, where $K^{\max}$ is the maximum allowable number of iterations (if there is any), or $k_c = k^*+1$ (see Section III-E).

To this end, here we develop two algorithms for solving (6). Algorithms 1 and 2 are two variations of the implementation of FedAvg algorithm (3) with our causal termination approach, FedCau, for batch and mini-batch implementations using convex loss functions.

In the batch update of Algorithm 1, workers compute $\{\boldsymbol{w}_k^j, f_k^j\}_{j\in[M]}$ and transmit them to the master node (see lines 6-12). We assume that the local parameter of each worker consists of the value of local FL model $\boldsymbol{w}_k^j$ and the local loss function f_k^j [2]. Then, the master node updates $\boldsymbol{w}_k$, and $f(\boldsymbol{w}_k)$, only after receiving all the local parameters $\{\boldsymbol{w}_k^j, f_k^j\}_{j\in[M]}$ from all workers at each iteration k (see lines 14-15). Subsequently, the master node calculates the amount of available resources, denoted as iteration cost c_k, spent for computation and transmission of local parameters. Initializing $G(0) = +\infty$ to prevent stopping in the first iteration, the master node updates the value of multi-objective cost function $G(k)$ (see line 16).

[2]We assumed that $f_k^j \in \mathbb{R}$ and $\boldsymbol{w}_k^j \in \mathbb{R}^d$, then the communication overhead, in term of number of bits, for transmission of f_k^j is negligible compared to the local FL model $\boldsymbol{w}_k^j$. Thus, we consider the local parameters consist of both local FL model and local loss function value.

Algorithm 1: Cost-efficient batch FedCau.

```
1:  **Inputs:** $\boldsymbol{w}_0$, $(\boldsymbol{x}_{ij}, y_{ij})_{i,j}$, $\alpha_k$, $M$, $\{|D_j|\}_{j\in[M]}$, $\rho_j$.
2:  **Initialize:** $k_c = +\infty$, $G(0) = +\infty$
3:  Master node broadcasts $\boldsymbol{w}_0$ to all nodes
4:  **for** $k \leq k_c$ **do**
5:      **for** $j \in [M]$ **do**
6:          Calculate $f_k^j := \sum_{i=1}^{|D_j|} f(\boldsymbol{w}_k; \boldsymbol{x}_{ij}, y_{ij})/|D_j|$
7:          Set $\boldsymbol{w}_{0,k+1}^j = \boldsymbol{w}_k$
8:          **for** $h \in [E]$ **do**
9:              Compute $\boldsymbol{w}_{h,k+1}^j \leftarrow \boldsymbol{w}_{h-1,k+1} - \alpha_k \nabla_w f_k^j$
10:         **end for**
11:         Set $\boldsymbol{w}_{k+1}^j = \boldsymbol{w}_{E,k+1}^j$
12:         Send $\boldsymbol{w}_{k+1}^j$ and $f_k^j$ to the master node
13:     **end for**
14:     Wait until master node collects all $\{\boldsymbol{w}_{k+1}^j\}_j$ and set $\boldsymbol{w}_{k+1} \leftarrow \sum_{j=1}^M \rho_j \boldsymbol{w}_{k+1}^j$
15:     Calculate $f(\boldsymbol{w}_k) := \sum_{j=1}^M \rho_j f_k^j$,
16:     Calculate $c_k$ and $G(k)$
17:     **if** $G(k) < G(k-1)$ **then**
18:         Master node broadcasts $\boldsymbol{w}_{k+1}$ to the worker nodes
19:     **else**
20:         Set $k_c = k$, Break and go to line 24
21:     **end if**
22:     Set $k \leftarrow k+1$
23: **end for**
24: **Return** $\boldsymbol{w}_{k_c}$, $k_c$, $G(k_c)$
```

Algorithm 2: Stochastic cost-efficient mini-batch FedCau.

```
1:  **Inputs:** $\boldsymbol{w}_0$, $(\boldsymbol{x}_{ij}, y_{ij})_{i,j}$, $\rho_j$, $F_f$, $\alpha_k$, $M$, $\{|D_j|\}_{j\in[M]}$
2:  **Initialize:** $k_c = +\infty$, $T_s = +\infty$, $j_s = 0$, $\hat{G}(0) = +\infty$, $\mathcal{M}_n^1 = \{[M]\}$, and Fair-Count $= \mathbf{1}_{M\times 1}$, $t_1^s = 1$
3:  Master node broadcast $\boldsymbol{w}_0$ to all nodes
4:  **for** $k \leq k_c$ **do**
5:      $\mathcal{M}_k^a = \{\}$
6:      **for** $j \in [M]$ **do**
7:          Calculate $f_k^j := \sum_{i=1}^{|D_j|} f(\boldsymbol{w}_k; \boldsymbol{x}_{ij}, y_{ij})/|D_j|$
8:          Set $\boldsymbol{w}_{0,k+1}^j = \boldsymbol{w}_k$
9:          **for** $h \in [E]$ **do**
10:             Compute $\boldsymbol{w}_{h,k+1}^j \leftarrow \boldsymbol{w}_{h-1,k+1} - \alpha_k \nabla_w f_k^j$
11:         **end for**
12:         Set $\boldsymbol{w}_{k+1}^j = \boldsymbol{w}_{E,k+1}^j$
13:         **if** $j \in \mathcal{M}_k^n$ **then**
14:             Send $\boldsymbol{w}_{k+1}^j$ and $f_k^j$ to the master node
15:         **end if**
16:     **end for**
17:     **if** $k = 1$ **then** master node:
18:         **for** $t_1^s \leq T_s$ **do** until $\mathcal{M}_k^n = \{\}$
19:             $\mathcal{M}_k^a \leftarrow \mathcal{M}_k^a \cup \{j_s\}$
20:             $\mathcal{M}_k^n \leftarrow \mathcal{M}_k^n \setminus \mathcal{M}_k^a$
21:         **end for**
22:         Set $T_s = t_1^s$
23:         Set $\boldsymbol{w}_{k+1} \leftarrow \sum_{j=1}^M \rho_j \boldsymbol{w}_{k+1}^j$
24:         Calculate $f(\boldsymbol{w}_k) := \sum_{j=1}^M \rho_j f_k^j$ and $G(k)$
25:         Set a time budget $T \leq T_s$, and $t_k^s = 0$
26:         Set $\mathcal{M}_{k+1}^n = \{[M]\}$
27:     **else**
28:         **for** $t_k^s \leq T$ **do**
29:             Every node $j \in \mathcal{M}_k^n$ send $\boldsymbol{w}_{k+1}^j$
30:             **if** Successful node $j_s \in \mathcal{M}_n$ **then**
31:                 $\mathcal{M}_k^a \leftarrow \mathcal{M}_k^a \cup \{j_s\}$
32:                 Fair-Count$[j_s]$ = Fair-Count$[j_s] + 1$
33:             **end if**
34:         **end for**
35:         Master node set
            $\boldsymbol{w}_{k+1} \leftarrow \sum_{j\in\mathcal{M}_k^a} \rho_j \boldsymbol{w}_{k+1}^j + \sum_{j'\notin\mathcal{M}_k^a} \rho_{j'} \boldsymbol{w}_k^{j'}$
36:         Master node calculate
            $\hat{f}(\boldsymbol{w}_k) := \sum_{j\in\mathcal{M}_k^a} \rho_j f_k^j + \sum_{j'\notin\mathcal{M}_k^a} \rho_{j'} f_{k-1}^{j'}$ and $\hat{G}(k)$
37:         Master node update
            $\mathcal{M}_k^n = \{j | \text{Fair-Count}[j] < F_f\}$
38:         **if** $\mathcal{M}_k^n = \{\}$ **then**
39:             $\mathcal{M}_{k+1}^n = \{[M]\}$
40:             Fair-Count $= \mathbf{0}_{M\times 1}$
41:         **end if**
42:     **end if**
43:     **if** $\hat{G}(k) < \hat{G}(k-1)$ **then**
44:         Master node broadcast $\boldsymbol{w}_{k+1}$ to the worker nodes
45:     **else**
46:         Set $k_c = k$, and $\hat{\boldsymbol{w}}_{k_c} \leftarrow \boldsymbol{w}_{k+1}$
47:         Break and go to line 51
48:     **end if**
49:     Set $k \leftarrow k+1$, and $t_{k+1}^s = 1$
50: **end for**
51: **Return** $\hat{\boldsymbol{w}}_{k_c}$, $k_c$, $\hat{G}(k_c)$
```

Thereafter, the master node compares $G(k)$ with its previous value $G(k-1)$ (see line 17), thus deciding when to terminate the iterations (see lines 19-24).

In FedAvg, there are many schemes where only some of the workers can upload their local parameters to the master node. The implicit sub-sampling (from set $\{\boldsymbol{w}_k^j\}_{j\in[M]}$), which can be interpreted as stochastic noise [35], leads to approximations of $f(\boldsymbol{w}_k)$, denoted by $\hat{f}(\boldsymbol{w}_k)$. Therefore, it leads to an approximation of the joint communication-computation and federated learning cost function, denoted by $\hat{G}(K)$. Thus, differently from Algorithm 1, Algorithm 2 uses the mini-batch update to avoid spending extra resources for achieving a very small and negligible test accuracy improvement. Algorithm 2 utilizes the descent property of FedAvg algorithm (3) for a monotonic decreasing loss function $f(\boldsymbol{w})$ explained in Remark 1. The fundamental idea behind the mini-batch update of Algorithm 2 is to achieve a decreasing sequence of $f(\boldsymbol{w}_k)_k$ and thus achieve a non-increasing behavior for $G(k)_{k\leq k^*}$.

Algorithm 2 considers partial worker participation and fairness while training FedCau. We define $\mathcal{M}_k^n$ as the node selection subset at each communication iteration k, and Fair-count$[j]$ as the counter for the number of successfully-sent local parameters by each worker $j \in \mathcal{M}_k^n$. We also define $F_f \leq K^{\max}$ as the "Fairness-Factor", which limits the worker nodes from transmitting more than F_f local parameters until all workers satisfy F_f local parameter transmission. Here, our proposal is that at the first communication iteration $k = 1$, when a worker node $j \in \mathcal{M}_1^n$ successfully transmits its local parameter $\boldsymbol{w}_1^j$, the master node removes j from the selected node subset $\mathcal{M}_1^n$ (see lines 7-21). Thus, the worker j will not transmit any packets until all the worker nodes send their local parameters. Then, the master node computes the resource used to perform the first communication iteration as T_s. Afterward, the master node considers the latency T_s as a benchmark, and determines $T \leq T_s$ as the maximum allowable time slots for performing the future iterations of $k = 2, \ldots, K$ (see line 25) [3]. Note that in $k = 1$, the low-power workers have a better chance to transmit their first local parameter, while the latency T becomes smaller than the case of full node participation at each time slot of $k = 1$. At the final step of completing communication iteration $k = 1$, the master node updates $\mathcal{M}_{k+1}^2$ (see line 26), thus the partial worker participation starts from $k \geq 2$.

For $k \geq 2$, each selected worker $j \in \mathcal{M}_k^n$ has to compute and transmit its local parameter during at most the time budget T. This time budget enforces the competition between the selected workers to transmit to the master node. However, some of the selected workers may fail to send their local parameters to the master node. We introduce the set of $\mathcal{M}_k^a$, which contains the indexes of the successful workers $j_s \in \mathcal{M}_k^n$ in transmission at each iteration k (see line 31). Then, the fairness counter of each successful node, Fair-count$[j_s]$ is increased (see

[3] Here, we allocate an equal portion of resource to each iteration. However, one can assign different portion of resources to each iteration, which is out of the scope of this paper.

line 32) to further decide on the selecting that j_s for the future communication iterations. Afterward, the master node updates the global parameter by the local parameters it has received, $\boldsymbol{w}_k^j, j \in \mathcal{M}_k^a$, and then replaces the missing local parameters by the values of the previous iteration, for the local parameters $\boldsymbol{w}_k^{j'} = \boldsymbol{w}_{k-1}^{j'}, j' \notin \mathcal{M}_k^a$ [4] and local functions $f_k^{j'} = f_{k-1}^{j'}, j' \notin \mathcal{M}_k^a$ (see lines 30-31). This replacement is important to guarantee the decreasing behavior of the loss function sequences $f(\boldsymbol{w}_k)$, see the following Lemma 2. Finally, the master node updates the selected nodes for the upcoming communication iterations, according to the fairness factor (see line 37). Once all of the node satisfy the fairness factor, the master node repeats the worker participation policy by setting $\mathcal{M}_{k+1}^n = \{[M]\}$ (see lines 38-40). This procedure requires that the master node keeps a memory of all the previous local parameters. The remaining of Algorithm 2, lines 43-51, updates the parameters and checks for the potential stopping iteration k_c, similar to lines 12-20 of Algorithm 1.

Lemma 2. *Let f_k^j be the local loss function at the communication iteration k for each worker $j \in [M]$. Suppose that $f_j(\boldsymbol{w})$ be a convex function w.r.t. $\boldsymbol{w}$. Then, Algorithm 2 guarantees the decreasing behavior of $\hat{f}(\boldsymbol{w}_k), \forall k$.*

Proof: See Appendix A-B. $\square$

D. FedCau for Non-convex Loss Functions

In this part, we explain how to extend Algorithms 1 and 2 to scenarios with non-convex loss functions. We consider FedAvg [5], in which every node j performs $E \geq 1$ local iterations over its local subset of data, $\xi_k^j \leq |D_j|$ using mini-batches, before sending its local parameters to the master node. The master node starts updating the global parameter $\boldsymbol{w}_{k+1}$ by averaging the local parameters and broadcasts it to the workers. Furthermore, the master node calculates $\tilde{F}(\boldsymbol{w}_k)$ by averaging the local loss functions of the worker nodes [5].

Different than Algorithms 1 and 2, we design a cost-efficient algorithm which optimizes $\tilde{G}(K)$, an estimate of the multi-objective cost function $G(K)$ defined as $\tilde{G}(K) := \beta C(K) + (1-\beta)\tilde{F}(\boldsymbol{w}_K)$, where recall that $C(K)$ is the iteration-cost function at K. We design such estimate since the stochastic nature of the sequences of $(\tilde{F}(\boldsymbol{w}_k))_k$, arises from the local updates by $\xi_k^j \leq |D_j|$ using mini-batches, results in a stochastic sequence of $(\tilde{G}(k))_k$. This sequence $(\tilde{G}(k))_k$ hinders the application of Algorithms 1, 2 and might lead to their early stopping at a communication iteration. Thus, we need to develop an alternative algorithm.

We propose a causal approach by establishing an upper bound, $G_u(K)$, and a lower bound, $G_l(K)$, for $(\tilde{G}(k))_k$. In this scenario, $(\tilde{G}(k))_k$ is not necessarily non-increasing and may have

[4]For simplicity, we use the notation $f_k^j := f_j(\boldsymbol{w}_k)$.

Algorithm 3: Stochastic non-convex cost-efficient mini-batch FedCau.

```
1: Inputs: $\boldsymbol{w}_0$, $(\boldsymbol{x}_{ij}, y_{ij})_{i,j}$, $\alpha_k$, $M$, $\{|D_j|\}_{j\in[M]}$, $\rho_j$, $E$, $\xi_k^j$
2: Initialize: $k_c^u = k_c^l = 0$, $k_{\max}^l = 2$
3: Master node broadcasts $\boldsymbol{w}_0$ to all nodes
4: for $k \geq 1$ do
5:     $\mathcal{M}_a = \{\}$
   Each node $j$ calculates:
6:     for $j \in [M]$ do
7:         if $k = 0$ then
8:             Randomly select a subset of data with size $\xi_k^j$
9:             $F_0^j := \sum_{i=1}^{\xi_k^j} F(\boldsymbol{w}_0; \boldsymbol{x}_{ij}, y_{ij})/\xi_k^j$
10:        end if
11:        Set $\boldsymbol{w}_{0,k+1}^j = \boldsymbol{w}_k$, $F_{0,k+1}^j = F_k^j$
12:        for $i \in [E]$ do
13:            Randomly select a subset of data with size $\xi_k^j$
14:            $\boldsymbol{w}_{i,k+1}^j \leftarrow \boldsymbol{w}_{i_{E-1},k+1} - \alpha_k \nabla_w F_{i_{E-1},k+1}^j$
15:            $F_{i,k+1}^j = \sum_{i=1}^{\xi_k^j} F(\boldsymbol{w}_{i,k+1}^j; \boldsymbol{x}_{ij}, y_{ij})/\xi_k^j$
16:        end for
17:        Set $\boldsymbol{w}_{E,k+1}^j = \boldsymbol{w}_{k+1}^j$, and $F_{k+1}^j = F_{E,k+1}^j$
18:        Send $\boldsymbol{w}_{k+1}^j$ and $F_{k+1}^j$ to the master node
19:    end for
   Master node calculates:
20:    $\mathcal{M}_a \leftarrow \mathcal{M}_a \cup \{\text{Successful nodes}\}$
21:    $\tilde{F}(\boldsymbol{w}_k) := \sum_{j\in\mathcal{M}_a} \rho_j F_k^j + \sum_{j'\notin\mathcal{M}_a} \rho_{j'} F_{k-1}^{j'}$
22:    Update $C(k)$
23:    if $k \leq 2$ then
24:        Set $F_u(\boldsymbol{w}_k) = F_l(\boldsymbol{w}_k) = \tilde{F}(\boldsymbol{w}_k)$
25:    else
26:        if $\tilde{F}(\boldsymbol{w}_k) \geq \tilde{F}(\boldsymbol{w}_{k-1})$ then
27:            Set $F_u(\boldsymbol{w}_k) = \tilde{F}(\boldsymbol{w}_k)$
28:            if $\tilde{F}(\boldsymbol{w}_k) \geq F_u(\boldsymbol{w}_{k-1})$ then
29:                $k_{\max}^u = \max_{k_u<k}\{k_u | F_u(\boldsymbol{w}_{k_u}) > \tilde{F}(\boldsymbol{w}_k)\}$
30:                Update $(F_u(\boldsymbol{w}_i))_{i=k_{\max}^u,\dots,k}$ as (11)
31:            end if
32:            Calculate $\delta_k^u = F_u(\boldsymbol{w}_k) - F_u(\boldsymbol{w}_{k-1})$
33:            Set $F_l(\boldsymbol{w}_k) = F_l(\boldsymbol{w}_{k-1}) + \delta_k^u$
34:            Update $(G_u(K))_{K=k_{\max}^u,\dots,k}$ and $G_l(k)$
35:        else
36:            if $\tilde{F}(\boldsymbol{w}_k) < F_l(\boldsymbol{w}_{k_{\max}^l})$ then
37:                Set $F_l(\boldsymbol{w}_k) = \tilde{F}(\boldsymbol{w}_k)$
38:                Update $(F_l(\boldsymbol{w}_i))_{i=k_{\max}^l,\dots,k}$ as (11)
39:                Calculate $\delta_k^l = F_l(\boldsymbol{w}_k) - F_l(\boldsymbol{w}_{k-1})$
40:                Set $F_u(\boldsymbol{w}_k) = F_u(\boldsymbol{w}_{k-1}) + \delta_k^l$
41:                $(G_l(K))_{K=k_{\max}^l,\dots,k}$ and $G_u(k)$
42:                $k_{\max}^l = k$
43:            else
44:                Set $F_u(\boldsymbol{w}_k) = \tilde{F}(\boldsymbol{w}_k)$
45:                Calculate $\delta_k^u = F_u(\boldsymbol{w}_k) - F_u(\boldsymbol{w}_{k-1})$
46:                Set $F_l(\boldsymbol{w}_k) = F_l(\boldsymbol{w}_{k-1}) + \delta_k^u$
47:                Update $G_u(k)$ and $G_l(k)$
48:            end if
49:        end if
50:    end if
51:    $k_c^l = \min\{K | G_l(K) > G_l(K-1)\}$
52:    $k_c^u = \min\{K | G_u(K) > G_u(K-1)\}$
53:    if $k_c^u \neq 0, k_c^l \neq 0$ then
54:        Break and go to line 60
55:    else
56:        Master node broadcast $\boldsymbol{w}_{k+1}$ to the worker nodes
57:        Set $k \leftarrow k + 1$
58:    end if
59: end for
60: Return $k_c^l$, $k_c^u$, $(\boldsymbol{w}_k)_{k=k_c^l,\dots,k_c^u}$
```

several local optimum points Thus, we aim to develop the non-increasing upper and lower bounds for $(\tilde{G}(k))_k$ to obtain an interval of k_c as $k_c^u \leq k_c \leq k_c^l$, in which k_c^u and k_c^l represent the stopping communication iteration for $G_u(K)$ and $G_l(K)$ functions, respectively. According to the definition of $G(K)$ function, we define $G_u(K)$, and $G_l(K)$ functions as

$$G_u(K) := \beta C(K) + (1-\beta)F_u(\boldsymbol{w}_K), \tag{10a}$$

$$G_l(K) := \beta C(K) + (1-\beta)F_l(\boldsymbol{w}_K)\,, \tag{10b}$$

where $F_u(\boldsymbol{w}_K)$ and $F_l(\boldsymbol{w}_K)$ represent the estimation of the loss function at upper and lower bounds. Thus, in order to obtain the sequences of $(G_u(k))_k$ and $(G_l(k))_k$, the master node first needs to compute the upper and lower bounds for $\tilde{F}(\boldsymbol{w}_k)$. To do so, the master node must guarantee the monotonic decreasing behavior of $(F_u(\boldsymbol{w}_k))_k$ and $(F_l(\boldsymbol{w}_k))_k$ to satisfy Remark 1. In the following, we focus on how the master node obtains the bounds for $\tilde{F}(\boldsymbol{w}_k)$.

Algorithm 3 shows the steps required for the cost-efficient FedAvg with causal setting and non-convex loss function $F(\boldsymbol{w})$. Lines 3-18 summarize the local and global iterations of FedAvg. Here, we introduce $\mathcal{M}_a$ as the set of workers which successfully transmit their local parameters to the master node (see line 20). We initialize $F_u(\boldsymbol{w}_k) = F_l(\boldsymbol{w}_k) = \tilde{F}(\boldsymbol{w}_k), k \leq 2$ for the first two iterations (see line 24). For iterations $k \geq 3$, if the new value of loss function fulfils $\tilde{F}(\boldsymbol{w}_k) \geq \tilde{F}_(\boldsymbol{w}_{k-1})$, the algorithm updates $F_u(\boldsymbol{w}_k) = \tilde{F}(\boldsymbol{w}_k)$ (see line 27). Then, the algorithm checks if $\tilde{F}(\boldsymbol{w}_k)$, which is now equal to $F_u(\boldsymbol{w}_k)$, is greater than the previous value of $F_u(\boldsymbol{w}_{k-1})$ (see line 28). This checking is important because we must develop a monotonic decreasing sequence of $F_u(\boldsymbol{w}_i)_{i=1:k_c^u}$. When $\tilde{F}(\boldsymbol{w}_k) \geq \tilde{F}_u(\boldsymbol{w}_{k-1})$, the master node returns to the history of $F_u(\boldsymbol{w}_i)_{i=1:k-1}$ and checks for $i < k$, when the condition $F_u(\boldsymbol{w}_i) > F_u(\boldsymbol{w}_k)$ is satisfied. Since at each communication iteration k we carefully check the monotonic behavior of $F_u(\boldsymbol{w}_k)$, we are sure that if we find the proper maximum communication iteration i that fulfils $i < k$, for which $F_u(\boldsymbol{w}_{i+1}) < F_u(\boldsymbol{w}_k) < F_u(\boldsymbol{w}_i)$, we have the result of $F_u(\boldsymbol{w}_j) < F_u(\boldsymbol{w}_k), j < i$. Let us define this communication iteration i as k^u_{max} (see line 29). Thus, it is enough to find such i to update the sequence of $F_u(\boldsymbol{w}_{i_1}), i_1 = i, \ldots, k$.

Now, we need to update the sequences of $F_u(\boldsymbol{w}_{i_2}), i_2 = k^u_{\text{max}}, \ldots, k$ to obtain the monotonic decreasing upper bound. We choose the monotonic linear function because it satisfies the sufficient decrease condition (see [31], Section 11.5). Therefore, we satisfy the decreasing behavior for $F_u(\boldsymbol{w}_k)$ and the upper bound behavior, which means that the maximum values of $\tilde{F}(\boldsymbol{w}_k)$ are always lower than $F_u(\boldsymbol{w}_k)$. Thus, we update the sequences of $F_u(\boldsymbol{w}_i), i = k^u_{\text{max}}, \ldots, k$ according to (11), with $k_1 = k^u_{\text{max}}$, $k_2 = k$, and $F_u(k_i) = F_{\text{linear}}(k_i), k_i \in [k_1, k_2]$. We define $F_{\text{Apxt}}(k)$ as the linear approximation of $F(\boldsymbol{w}_k)$ in an interval $k \in [k_1, k_2]$

$$F_{\text{Apxt}}(k_i) = ak_i + b, \;\; k_1 \leq k_i \leq k_2, \text{where} \tag{11}$$

$$a = \frac{\tilde{F}(\boldsymbol{w}_{k_2}) - \tilde{F}(\boldsymbol{w}_{k_1})}{k_2 - k_1}, \tag{12a}$$

$$b = F(\boldsymbol{w}_{k_2}) - ak_2. \tag{12b}$$

Next, we need to update $F_l(\boldsymbol{w}_k)$. Here, let us define the difference between two consecutive values of $F_u(\boldsymbol{w}_k)$ and $F_u(\boldsymbol{w}_{k-1})$ as δ^u_k, and the difference between $F_l(\boldsymbol{w}_k)$ and $F_l(\boldsymbol{w}_{k-1})$ as

δ_k^l. Then, we update the corresponding values for $(G_u(K))_{K=k_{\max}^u,\ldots,k}$ and $G_l(k)$, respectively (see lines 33-34). Afterward, we need to check the condition at which $\tilde{F}(\boldsymbol{w}_k) < F_l(\boldsymbol{w}_{k_{\max}^l})$, where $k_{\max}^l$ represents the last communication iteration at which the value of $\tilde{F}(\boldsymbol{w}_{k_{\max}^l})$ has been considered as $F_l(\boldsymbol{w}_{k_{\max}^l})$. If $\tilde{F}(\boldsymbol{w}_k) < \tilde{F}(\boldsymbol{w}_{k_{\max}^l})$, we need to update the lower bound sequences (see lines 36-37) to avoid over-decreasing the lower bound function $F_l(\boldsymbol{w}_k)$ by the approximation of line 33. Subsequently, we need to calculate $\delta_k^l = F_l(\boldsymbol{w}_k) - F_l(\boldsymbol{w}_{k-1})$, then update $F_u(\boldsymbol{w}_k) = F_u(\boldsymbol{w}_{k-1}) + \delta_k^l$ and the value of $k_{\max}^l = k$ (see lines 40-42).

The last condition to check is when $F_l(\boldsymbol{w}_{k_{\max}^l}) < \tilde{F}(\boldsymbol{w}_k) < \tilde{F}(\boldsymbol{w}_{k-1})$. In this condition, the monotonic decreasing behavior of $\tilde{F}(\boldsymbol{w}_k)$ is satisfied, whereas the decreasing behavior is not satisfied for the lower bound $F_l(\boldsymbol{w}_k)$. Thus we set $F_u(\boldsymbol{w}_k) = \tilde{F}(\boldsymbol{w}_k)$, and $\delta_k^u = F_u(\boldsymbol{w}_k) - F_u(\boldsymbol{w}_{k-1})$, $F_l(\boldsymbol{w}_k) = F_l(\boldsymbol{w}_{k-1}) + \delta_k^u$, and update $G_u(k)$ and $G_l(k)$ (see lines 44-47). Finally, lines 51-60 show when to stop the algorithm.

E. Optimality and Convergence Analysis

In this subsection, we investigate the existence and optimality of the solution to problem (6) and the convergence of the algorithms that return the optimal solutions. We are ready to give the following proposition, which provides us with the required analysis of Algorithms 1, and 2.

First, we start with the monotonically behavior of $G(K), K \leq k^*$. In practice, we have this desired monotonically decreasing behaviour, as we show in the following proposition:

Proposition 1. *Consider $G(K)$ defined in Eq. (7). Define $\Delta_k := f_{k-1} - f_k$, $\Delta_0 = f_0$, by choosing β as*

$$0 < \frac{1}{1 + \frac{\max_k c_k}{\Delta_0}} \leq \beta \leq \frac{1}{1 + \frac{\min_k c_k}{\Delta_0}} < 1, \;\; k \leq k^*, \tag{13}$$

the function $G(K)$, $K \leq k^*$, is non-increasing at K.

Proof: See Appendix A-C. □

Remark 3. *The previous proposition implies that, without loss of generality, we can assume that max$_k$ c_k is high enough and min$_k$ c_k is close to zero (setting the initial cost to zero for example). Thus β can in practice vary between* 0 *and* 1, *without restricting the applicability range of the multi-objective optimization.*

Proposition 2. *Optimization problem* (6) *has a finite optimal solution k^*.*

Proof: See AppendixA-D. □

The following Theorem clarifies an important relation between the non-causal and causal solutions of Algorithm 1:

Theorem 1. *Let k^* be the solution to optimization problem* (6)*, and let and k_c denote the approximate solution obtained in the non-causal and causal settings of Algorithm 1, and 2, respectively. Then, the following statements hold*

$$k_c \in \{k^*, k^*+1\}\,, \tag{14a}$$
$$f(\boldsymbol{w}_{k_c}) \leq f(\boldsymbol{w}_{k^*})\,,\ \textit{and} \tag{14b}$$
$$G(k_c) \geq G(k^*)\,. \tag{14c}$$

Proof: See AppendixA-E. □

Remark 4. *Observe that k^* and k_c are fundamentally different. k_c is obtained from Algorithms 1 or 2, while k^* is the optimal stopping communication iteration that we would compute if we knew beforehand the evolution of the iterations of FedAvg algorithm* (3)*, thus non-causal. Nevertheless, we show that the computation of the stopping communication iteration k_c that we propose in the causal setting of Algorithms 1 and 2 is almost identical to k^*.*

Theorem 1 is a central result in our paper, showing that we can develop a simple yet close-to-optimal algorithm for optimization problem (6). In other words, Algorithms 1 and 2 in the causal setting solve problem (6) by taking at most one extra communication iteration compared to the non-causal to compute the optimal termination communication iteration number.

Next, we focus on the convergence analysis of Algorithm 3. From Section III-D, we define $F_u(\boldsymbol{w}_k)$ and $F_l(\boldsymbol{w}_k)$ as the upper and lower bound functions for $\tilde{F}(\boldsymbol{w}_k)$, respectively, such that for every $k \geq 1$, inequalities $F_l(\boldsymbol{w}_k) \leq \tilde{F}(\boldsymbol{w}_k) \leq F_u(\boldsymbol{w}_k)$ hold. The following remark highlights the important monotonic behavior of $F_u(\boldsymbol{w}_k)$ and $F_l(\boldsymbol{w}_k)$.

Remark 5. *The proposed functions $F_u(\boldsymbol{w}_k)$ and $F_l(\boldsymbol{w}_k)$ are monotonic decreasing w.r.t. k, i.e., $F_u(\boldsymbol{w}_k) < F_u(\boldsymbol{w}_{k-1})$, and $F_l(\boldsymbol{w}_k) < F_l(\boldsymbol{w}_{k-1})$ for $\forall k \geq 1$. These results hold because we consider a linear function, which is monotonically decreasing, w.r.t. k, for updating each value of $F_l(\boldsymbol{w}_k)$ and $F_u(\boldsymbol{w}_k)$ for $k \geq 1$. Since the monotonically decreasing linear function fulfils the sufficient decreasing condition (see [31], Section 11.5), we claim that $F_u(\boldsymbol{w}_k)$ and $F_l(\boldsymbol{w}_k)$ are monotonic decreasing w.r.t. k.*

Remark 5 indicates that we can apply the batch FedCau update of Algorithm 1 to obtain the causal stopping point for $G_u(K)$ and $G_l(K)$ denoted as k_c^u and k_c^l, respectively. Therefore, according to Proposition 2, there are finite optimal stopping iterations for minimizing $G_u(K)$ and $G_l(K)$. Thus, Theorem 1 is valid for k_c^u and k_c^l, and we guarantee the convergence of $G_u(K)$ and $G_l(K)$. The following Proposition characterizes the relation of causal stopping iteration k_c of $\tilde{G}(K)$ with k_c^u and k_c^l.

Proposition 3. *Let k_c, k_c^u, and k_c^l be the causal stopping iterations for minimizing $\tilde{G}(K)$, $G_u(K)$, and $G_l(K)$, respectively. Then, the inequalities $k_c^u \leq k_c \leq k_c^l$ hold.*

Proof: For the sake of saving space, we move the proof to extended version of FedCau in [36], Appendix A-F. □

Proposition 3 characterizes an interval in which k_c can take values to stop Algorithm 3.

As $k_c^u \leq k_c \leq k_c^l$, it is enough that we find k_c^u and terminate the algorithm. However, the maximum allowable number of iterations is k_c^l, which can be achieved if the resource budget allows us. Using Proposition 3, we can obtain a sub-optimal k_c by applying the FedCau update Algorithm 3 to non-convex loss functions.

Lemma 3. *Let $F_u(\boldsymbol{w}_k)$ and $F_l(\boldsymbol{w}_k)$ be respectively the upper bound and the lower bound of $\tilde{F}(\boldsymbol{w}_k)$ obtained from the stochastic non-convex cost-efficient mini-batch FedCau Algorithm 3. Let us define $\tilde{F}_{\max} := \max_{k\in[2,K]} \tilde{F}(\boldsymbol{w}_k)$ and $\tilde{F}_{\min} := \min_{k\in[2,K]} \tilde{F}(\boldsymbol{w}_k)$. Assuming that $|\tilde{F}(\boldsymbol{w}_k)| < \infty, k = 1, \dots, K$, then $|F_u(\boldsymbol{w}_k) - F_l(\boldsymbol{w}_k)| \leq \tilde{F}_{\max} - \tilde{F}_{\min}$ is the tightness between the upper bound $F_u(\boldsymbol{w}_k)$ and the lower bound $F_l(\boldsymbol{w}_k)$.*

Proof: See Appendix A-F. □

Lemma 3 specifies that the maximum distance between the upper and lower bound functions $F_u(\boldsymbol{w}_k)$ and $F_l(\boldsymbol{w}_k)$, for $k = 1, \dots, K$, is determined by the variations of non-convex sequence $\tilde{F}(\boldsymbol{w}_k), k = 1, \dots, K$. However, our proposed stochastic non-convex cost-efficient mini-batch FedCau Algorithm 3, provides a sub-optimal k_c and a close-to-optimal test accuracy (see numerical results in Section V) while we have non-convex sequence $\tilde{F}(\boldsymbol{w}_k), k = 1, \dots, K$.

To summarize, FedCau is applicable for both full and partial worker participation, as well as when $f(\boldsymbol{w}_k)$ is monotonic decreasing and not monotonic decreasing. Specifically, we have used the FedCau theory to propose Algorithm 3 that obtains a suboptimal solution for k_c when $f(\boldsymbol{w}_k)$ is not monotonically decreasing.

Proposition 4. *Let k_c^u and k_c^l be the causal stopping iterations for minimizing $G_u(K)$, and $G_l(K)$, respectively. Then,*

$$k_c^l = 1 + \max\left\{k_c^u, k_{\max}^u + \left\lceil (1-\beta)\frac{\tilde{F}(\boldsymbol{w}_k) - F_u(\boldsymbol{w}_{k_{\max}^u})}{\beta c_k}\right\rceil\right\}. \quad (15)$$

Proof: See Appendix A-G. □

F. Complexity Analysis of Algorithms 1-3

In this part, we analyze the computation complexities of Algorithms 1-3 and compare them with the computation complexity of FedAvg. Recall that in FedCau of Algorithms 1-3, the stopping iteration $k_c \leq K^{\max}$, $K^{\max}$ is the number of FedAvg global iterations. By ssuming the training is done considering a neural network with N_l number of layers, d_N as the maximum number of neurons, the backpropagation of local gradients in each worker j after E local iterations, results in a complexity of $\mathcal{O}(EN_l d_N^3)$. Thus, Algorithm 1 has the complexity of $\mathcal{O}(k_c E(|D|d + N_l d_N^3))$, which is less than or equal to the complexity of FedAvg as $\mathcal{O}(K^{\max} E(|D|d + N_l d_N^3))$. Similarly, the complexity of Algorithm 2 is obtained as $\mathcal{O}(|D|Ed + (k_c - 1)F_f^{-1}|D|Ed + Ek_c N_l d_N^3)$. Finally, the complexity of Algorithm 3, by considering the complexity from neural network setting we mentioned before, is obtained as $\mathcal{O}(|D|Ed(k_c^l)^3 + Ek_c^l N_l d_N^3)$.

IV. Application to communication Protocols

We consider wireless communication scenarios with a broadcast channel in the downlink from the master node to the workers. In the uplink, we consider three communication protocols, slotted-ALOHA [37] and CSMA/CA [28] with a binary exponential backoff retransmission policy [38], and OFDMA [29] by which the workers transmit their local parameters to the master node. We assume that in each communication iteration k, local parameters are set at the head of line of each node's queue and ready to be transmitted. Thus, upon receiving $\boldsymbol{w}_k$, each worker node $j \in [M]$ computes its local parameter $\boldsymbol{w}_k^j$ and puts it in the head of line of its transmission queue. In a parallel process, each worker may generate some background traffic and put them on the same queue, and send them by the first-in-first-out queuing policy. We obtain the average end-to-end communication-computation latency at each iteration k, denoted by c_k, for slotted-ALOHA and CSMA/CA protocols: by taking an average over the randomness of the protocols. Hence, at the end of each communication iteration K, the network has faced the latency equal to $\sum_{k=1}^{K} c_k$. It means that we consider each time slot (in ms) and sum up the spent computation delay and time slots in each communication iteration k to achieve c_k, thus following the Algorithms 1, 2, and 3 to solve optimization problem (6).

Here, the critical point to consider is that we should choose a stable network in which the packet saturation will not happen. Since we only consider the latency of transmission of local parameters, which are in the head of line of queues, and we assume a stable network the queue size does not play a role in the delay of the training procedure. However, the other parameters, such as number of worker nodes M, transmission probability p_x, and packet arrival probability p_r at each time slot, impact on the training latency. Note that the local parameters are the head of line packets at each communication iteration k and are not considered as the background traffic packets arrived by probability of p_r.

Recall the definition of the communication-computation cost components $\ell_{1,k}, \ell_{2,k}, \ell_{3,k}$ and $\ell_{4,k}$ in Section II-A. For $\ell_{1,k}$, we consider a broadcast channel with data rate R bits/s and parameter size of b bits (which includes the payload and headers), leading to a constant latency of $\ell_{1,k} = b/R$ s. Also, it is natural to assume that $\ell_{4,k}$ is a given constant for updating parameters at the master node [39]. The computation latency $\ell_{2,k}^j$ in each iteration k at each worker node $j \in [M]$ is calculated as $\ell_{2,k}^j = a_k^j |D_k^j| / \nu_k^j$, where a_k^j is the number of processing cycles to execute one sample of data (cycles/sample), $|D_k^j| \leq |D_j|$ is subset of local dataset each worker chooses to update its local parameter $\boldsymbol{w}_k^j$, and ν_k^j is the central processing unit (cycles/s) [40]. Without any loss of generality we consider that $|D_k^j| = |D_j|, k = 1. \dots, K$.

We assume that all the worker nodes start transmitting their local parameters simultaneously, thus the network has to wait for the slowest worker to finish its computation. Therefore, $\ell_{2,k} = \max_{j\in[M]} \ell^{j}_{2,k}$. The third term, $\ell_{3,k}$, is determined by the channel capacity, resource allocation policy, and the network traffic. We characterize this term for two cases of batch and mini-batch updates with a defined time budget. Further, every specific broadcast channel imposes a particular R and b, which are not changing during the whole optimizing process. Therefore, to compute the iteration-cost function $\sum_k c_k$, we take into account the $\ell_{3,k}$ and $\ell_{2,k}$ terms and ignore the latency terms of $\ell_{1,k}$, and $\ell_{4,k}$ because they do not play a role in the optimization problem (6) in the presence of shared wireless channel for the uplink. Note that in this paper, without loss of generality, $c_k := \ell_{2,k} + \ell_{3,k}$, in which $\ell_{2,k}$ is independent of the communication channels/protocols. We wish to obtain the upper bound for communication delay when the users in network follow MAC protocols, such as slotted-ALOHA and CSAMA/CA, to transmit their local parameters of FedAvg algorithm (3) to the master node [41]–[43]. There are many papers in the literature computing the average transmission delay for MAC protocols. However, we have a specific assumption that at each communication iteration k, each worker node puts its local parameter at the head of line in its queue and makes them ready for transmission. Note that in FedAvg algorithm (3) the master node needs to receive all the local parameters to update the new global parameter $\boldsymbol{w}_k$. Accordingly, we need to calculate the average latency of the system while all workers must successfully transmit at least one packet to the master node. The following Proposition establishes bounds of the average transmission delay $\ell_{3,k}$ which we use it in the numerical results.

Proposition 5. *Consider random access MAC protocols in which the local parameters of FedAvg algorithm* (3) *are head-of-line packets at each iteration k. Let M, p_x and p_r be the number of nodes, the transmission probability at each time slot, and the background packet arrival probability at each time slot. Consider each time slot to have a duration of t_s seconds. Then, the average transmission delay, $\mathbb{E}\{\ell_{3,k}\}$ is bounded by*

$$\sum_{i=0}^{M-1} t_s p_{i,i+1} \leq \mathbb{E}\left\{\ell_{3,k}\right\} \leq \sum_{i=0}^{M-1} t_s \left\{ p_{i,i+1} + \frac{p_{i,i}}{(1-p_{i,i})^2} \right\}, \tag{16}$$

where

$$p_{i,i} = \hat{p} + (1-p_x)^{M-i}, i = 0, 1,$$

$$p_{i,i} = i p_r p_x \sum_{j=1}^{i-1} \frac{(i-1)!}{j!(i-1-j)!} \left\{ p_r^j (1-p_x)^j (1-p_r)^{i-1-j} \right\} + \hat{p} + (1-p_x)^M, i \geq 2,$$

$$p_{i,i+1} = (M-i)\, p_x (1-p_x)^{M-i-1} \left\{ \sum_{j=1}^{i} p_r^j (1-p_x)^j (1-p_r)^{i-j} \right\},$$

where $\hat{p}$ is the probability of an idle time slot.

Proof: See Appendix A-H. □

Proposition 5 introduces the bounds for transmission delay, thus for c_k, while considering slotted-ALOHA and CSMA/CA communication protocols. Recall that we defined $c_k = \ell_{2,k} + \ell_{3,k}$, then by considering the slowest worker in local computation, the iteration cost c_k is bounded by

$$\sum_{i=0}^{M-1} t_s p_{i,i+1} + \min_{j\in[M]} \left\{ \frac{|D_j| a_k^j}{\nu_k^j} \right\} \leq c_k \leq |D| \max_{j\in[M]} \left\{ \frac{a_k^j}{\nu_k^j} \right\} + t_s \sum_{i=0}^{M-1} \left\{ p_{i,i+1} + \frac{p_{i,i}}{(1-p_{i,i})^2} \right\}, \quad (17)$$

which helps us to design the communication-computation parameters for FedCau. Note that we consider a set up in which the transmission starts at the same time for all the workers. This is an important setup by which we have developed Algorithms 1-3, and the bounds on the iteration-cost c_k in Proposition 5 and inequalities (17). The assumption that all of the workers are transmitting at each iteration k is only for Algorithm 1. However, in the updated Algorithm 2, we can consider either partial or full worker participation, which allows us to skip the slowest worker and not wait for it at *each* iteration k. Finally, in Algorithm 3, we have developed a general approach by which FedCau can be applied to any scenario, e.g., full or partial worker participation, non-convex loss functions $f(\boldsymbol{w})$ or any $G(K)$ with various local optimum points. Thus, the assumption that workers start transmissions to the master node simultaneously does not contradict the cost-efficiency of FedCau, because we have considered various scenarios, like full or partial worker participation, in Algorithms 1-3.

Finally, in OFDMA, we consider uplink transmissions in a single-cell wireless system with $s = 1, 2, \dots, S_c$ orthogonal subchannels [44]. Let h_l^s, p_l^s be the channel gain and the transmit power of link l on subchannel s by which worker j sends its local parameters to the master node. Therefore, the signal to noise ratio (SNR) for the uplink is defined by SNR(p_l^s, h_l^s) as

$$\text{SNR}(p_l^s, h_l^s) = \frac{p_l^s h_l^s}{\sigma_l^s}, \quad (18)$$

The corresponding data rates (bps/Hz) is as $R_p(\text{SNR}) = \sum_{s=1}^{S_c} \log_2(1 + \text{SNR}(p_l^s, h_l^s))$. The master node randomly decides at each communication iteration k which worker should use which subchannel link, and the remaining workers will not participate in the parameter uploading at that iteration.

V. Numerical Results

In this section, we illustrate our results from the previous sections and we numerically show the extensive impact of the iteration costs when running FedAvg algorithm (3) training problem over a wireless network. We use a network with M worker nodes and simulation to implement slotted-ALOHA, CSMA/CA (both with binary exponential backoff), and the OFDMA. In each of these networks, we apply our proposed Algorithms 1, 2, and 3. We train the FedCau by well-known MNIST datset with non-iid distribution among workers, and CIFAR-10 dataset with both iid and non-iid cases. Moreover, we apply our proposed FedCau on top of existing methods from the literature, such as top-q and LAQ.

A. Simulation Settings

First, we consider solving a convex regression problem over a wireless network using a real-world dataset. To this end, we extract a binary dataset from MNIST (hand-written digits) by keeping samples of digits 0 and 1 and then setting their labels to -1 and +1, respectively. We then randomly split the resulting dataset of 12600 samples among M worker nodes, each having $\{(\boldsymbol{x}_{ij}, y_{ij})\}$, where $\boldsymbol{x}_{ij} \in \mathbb{R}^{784}$ is a data sample i, and a vectorized image at node $j \in [M]$ with corresponding digit label $y_{ij} \in \{-1, +1\}$. We use the loss function [45]

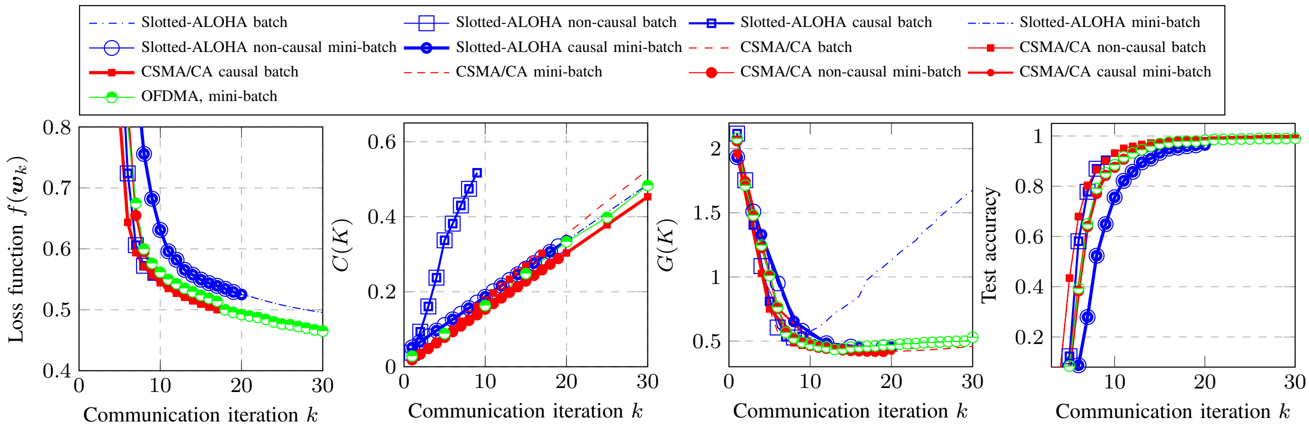
$$f(\boldsymbol{w}) = \sum_{j=1}^{M} \rho_j \sum_{i=1}^{|D_j|} \frac{1}{|D_j|} \log\left(1 + e^{-\boldsymbol{w}^T \boldsymbol{x}_{ij} y_{ij}}\right), \tag{19}$$

where we consider that each worker node $j \in [M]$ has $|D_j| = |D_i| = |D|/M, \forall i, j \in [M]$.

We implement the network with M worker nodes performing local updates of $\boldsymbol{w}_k^j, \forall j \in [M]$ and imposing computation latency of $\ell_{2,k}$ to the system. We assume a synchronous network in which all worker nodes start the local computation of $\boldsymbol{w}_k^j$ simultaneously right after receiving $\boldsymbol{w}_{k-1}$. Note that the latency counting of c_k at each iteration k starts from the beginning of the local computations until the uplink process is complete. Regarding the computation latency, we consider $\nu_k \in [10^6, 3 \times 10^9]$ cycles/s, and $a_k = [10, 30]$ cycle/sample for $k = 1, \ldots, M$. In slotted-ALOHA, we consider a capacity of one packet per slot and a slot duration of 1 ms. In CSMA/CA, we consider the packet length of 10 kb with a packet rate of 1 k packets per second, leading to a total rate of 1 Mbps. We set the duration of SIFS, DIFS, and each time slot to be 10 µs, 50 µs, and 10 µs respectively [46] and run the network for 1000 times. In OFDMA set up, we consider the uplink in a single cell system with the coverage radius of $r_c = 1$ Km. There are L_p cellular links on S_c subchannels. We model the subchannel power gain $h_l^s = \zeta/r^3$, following the Rayleigh fading, where ζ has an exponential distribution with unitary mean. We consider the noise power in each subchannel as -170 dBm/Hz and the maximum transmit power of each link as 23 dBm. We assume that $S_c = 64$ subchannels, the total bandwidth of 10 MHz, and the subchannel bandwidth of 150 KHz. We define c_k as the latency caused by the slowest worker to send the local parameters to the master node.

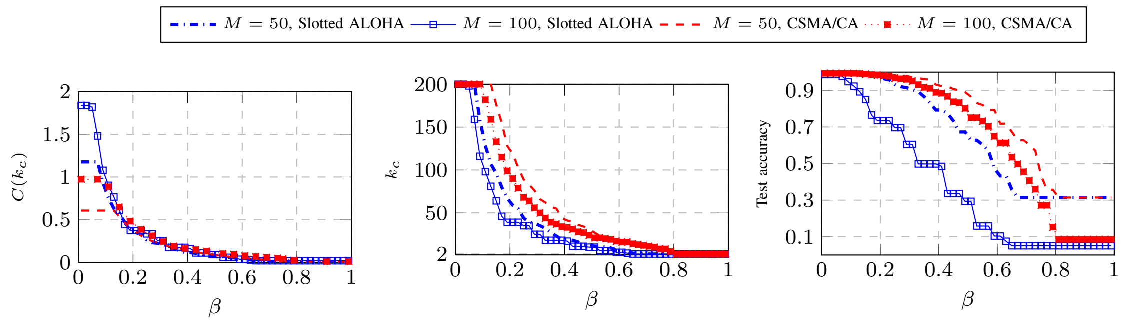
B. Performance of FedCau Update from Algorithms 1, 2 and Non-causal Approach

Fig. 1 characterizes the non-causal and causal behaviors along with the performance of FedCau update of Algorithms 1 and 2 for slotted-ALOHA and CSMA/CA protocols. The general network setup has $M = 100$, $p_x = 1$, $p_r = 0.2$, and the mini-batch time budget of $T = 0.3$s. We observe that while the behavior of $f(\boldsymbol{w}_k)$ is similar across the protocols in Fig. 1(a), the iteration-cost function $C(K)$ of the batch update for slotted-ALOHA is much larger among all the setups in Fig. 1(b). This behavior affects the multi-objective function $G(K)$ in Fig. 1(c) and causes an earlier stop. However, the test accuracy is not sacrificed as



(a) $f(\boldsymbol{w}_k)$.
(b) $C(K)$.
(c) $G(K)$.
(d) Test accuracy.

Fig. 1: Illustration of non-causal and FedCau batch update of Algorithm 1 and FedCau mini-batch update of Algorithm 2 with $T = 0.3$s the presence of slotted ALOHA and CSMA/CA, and OFDMA for $M = 100$, $p_x = 1$, and $p_r = 0.2$. a) Loss function $f(\boldsymbol{w}_k)$ b) Iteration-cost function $C(K)$ c) Multi-objective cost function $G(K)$, and d) Test accuracy.

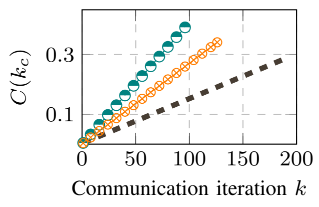


(a) $C(k_c)$ vs β.
(b) k_c vs β.
(c) Test accuracy vs β.

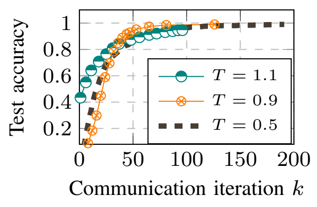
Fig. 2: Illustration of the effect of β on the performance of batch FedCau update of Algorithm 1, $M = 50, 100$ and CSMA/CA with $p_x = 1$, $p_r = 0.01$. a) The causal iteration-cost $C(k_c)$ decreases while β increases. b) The causal stopping iteration k_c is smaller for larger β. c) Test accuracy also decreases when β increases.

shown in Fig. 1(d). From Fig. 1, we conclude that the batch update of Algorithm 1 satisfies the causal setting, and preserves the test accuracy while optimizing both the loss function $f(\boldsymbol{w}_k)$ and the latency over the communication protocols.

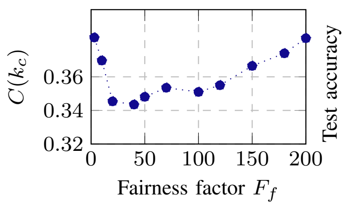
Fig. 2 characterizes the effect of β on the performance of batch FedCau update of Algorithm 1 assuming network sizes $M = 50, 100$ and CSMA/CA protocol for parameter upload. Fig. 2(a) shows that the causal iteration-cost $C(k_c)$ decreases while β takes the values between $(0, 1)$. This decreasing behavior is a valid result, since the higher values of β increase the effect of the term $C(K)$ in scalarized version (6). Since, $C(K)$ is an increasing function of K, the higher values of $C(K)$ results in stopping at the smaller causal iterations, called k_c. Thus, Fig. 2(b), shows the decreasing behavior of the causal stopping iteration k_c vs β. Finally, Fig. 2(c) demonstrates the test accuracy that we achieve while changing β. Since k_c decreases as β increases, the corresponding test accuracy decreases. Therefore, choosing $\beta \in [0.2, 0.5]$ provides us with the lower causal iteration-cost and sub-optimal test accuracy.



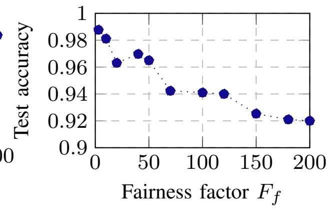
(a) Causal iteration-cost $M = 50$, $F_f = 10$.



(b) Mini-batch test accuracy $M = 50$, $F_f = 10$.



(c) Causal iteration-cost.



(d) Test accuracy.

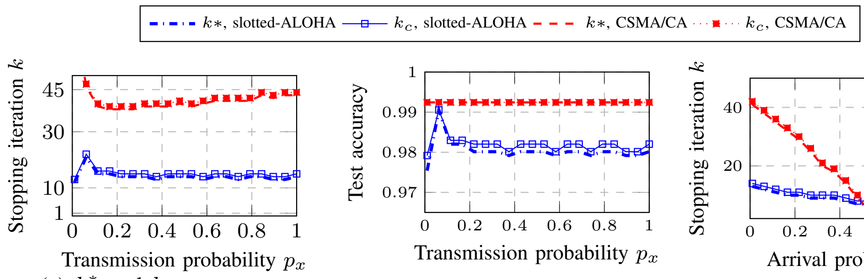
Fig. 3: Illustration of the mini-batch FedCau update of Algorithm 2 for CSMA/CA, $p_x = 1$, $p_r = 0.01$, and $M = 50$. a) $C(k_c)$ for $F_f = 10$, $T = 0.5, 0.9, 1.1$s, b) Test accuracy for $F_f = 10$, $T = 0.5, 0.9, 1.1$s, c) Iteration-cost function $C(k_c)$ for $T = 1.1$s vs different fairness factor F_f, and d) Test accuracy for $T = 1.1$s vs different fairness factor F_f.

Fig. 3 represents the mini-batch FedCau update of Algorithm 2 for CSMA/CA with $p_x = 1$, $p_r = 0.01$ and $M = 50$. Figs. 3(a)-3(b) show the results for $M = 50$, with $T = 0.5, 0.9, 1.1$s. Fig. 3(a) highlights that with a smaller time budget, the causal latency decreases, while Fig. 3(b) shows the similarity in the test accuracy. Figs. 3(c)-3(d) reveal the behavior of causal latency and test accuracy for $M = 50$, and $T = 1.1$s w.r.t. the fairness factor F_f. Fig. 3(a) demonstrates that the causal latency increases for small and large fairness factors F_f. However, 3(d) shows that the test accuracy decreases while F_f becomes large due to the lack of node fairness for higher F_f. We observe that for smaller F_f, the node fairness results in better test accuracy while a higher causal latency. This high causal latency results from the fairness condition that enforces a more frequent packet transmission of low-power nodes.

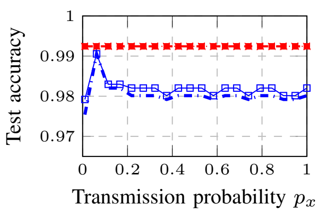
C. *Impact of Communication Parameters on The Performance of FedCau*

Fig. 4 shows the impact of communication parameters, the transmission p_x, and arrival probabilities p_r, and the network size M on the non-causal stopping iteration k^* and the causal stopping iteration k_c obtained from batch FedCau update Algorithm 1 along with the corresponding test accuracy. Figs. 4(a)-4(b) respectively characterize the stopping iterations and test accuracy of the FL algorithm (3) (with $E = 1$, and considering the whole local data size) considering $M = 50$, and $p_r = 0.01$. We note that the transmission probability p_x has more negative impacts when we regulate the channel by slotted-ALOHA. Figs. 4(c)-4(d) represent the same results, but by swiping the background packet arrival probability p_r, when $M = 50$ and $p_x = 0.8$. The numerical results shows that when facing a higher arrival traffic, slotted-ALOHA performs better than CSMA/CA. Finally, we characterize the same results by varying the network size M, when $p_x = 0.8$ and $p_r = 0.01$. Fig. 4(e) represents that slotted-ALOHA and CSMA/CA have similar decreasing behavior with increasing M. However, as seen in Fig. 4(f), slotted-ALOHA reduces the test accuracy in larger network size.

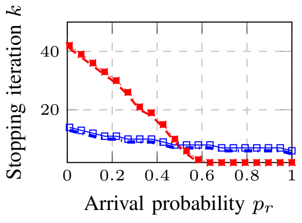
Fig. 5 characterizes the iteration-cost function $C(K)$ for the same setup we mentioned in Fig. 1. We observe that the iteration-cost function for slotted-ALOHA is larger than CSMA/CA, as we see in Figs. 5(a) and 5(c). On the other hand, the iteration-cost function for CSMA/CA



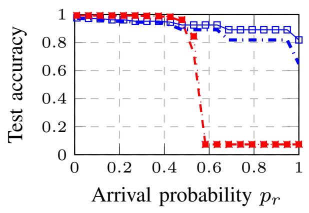
(a) k^* and k_c vs p_x.



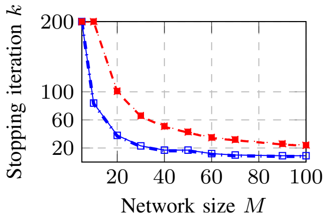
(b) Test accuracy vs p_x.



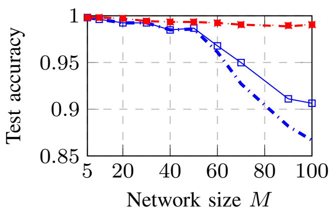
(c) k^* and k_c vs p_r.



(d) Test accuracy vs p_r.

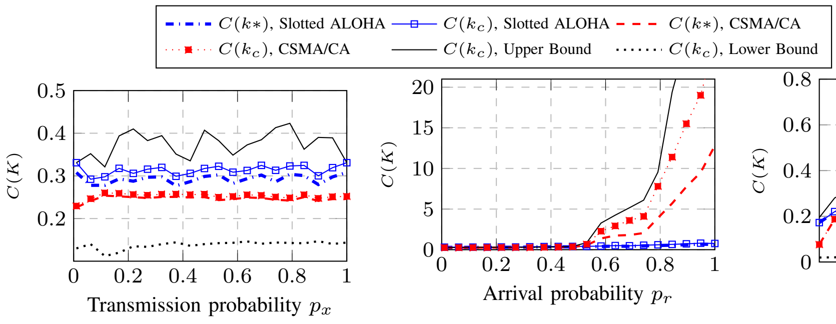


(e) k^* and k_c vs M.

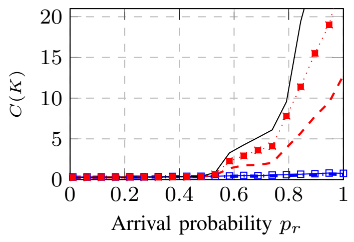


(f) Test accuracy vs M.

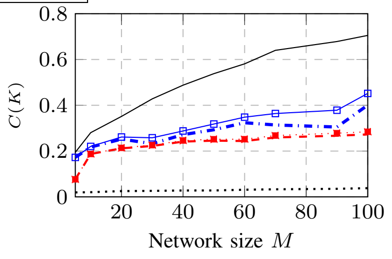
Fig. 4: Illustration of the batch FedCau update Algorithm 1: optimal stopping communication iteration k^* and causal stopping communication iteration k_c vs transmission probability p_x, arrival probability p_r, and network size M.



(a) Iteration-cost $C(k^*)$ and $C(k_c)$ vs p_x.



(b) Iteration-cost $C(k^*)$ and $C(k_c)$ vs p_r.



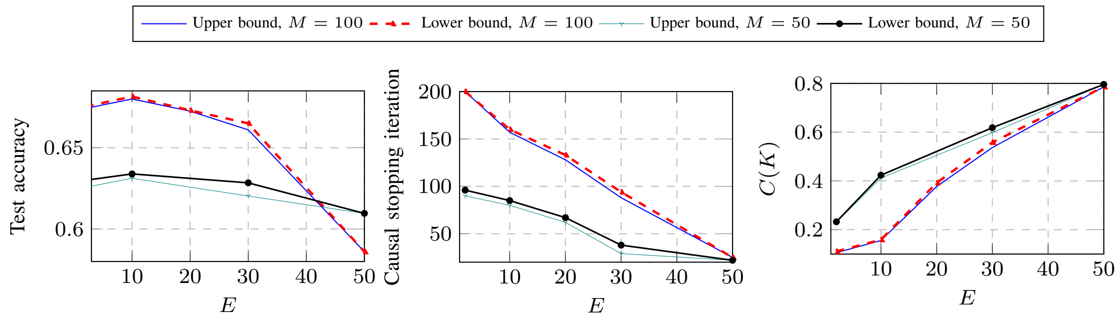
(c) Iteration-cost $C(k^*)$ and $C(k_c)$ vs M.

Fig. 5: Illustration of the batch FedCau update Algorithm 1: iteration-cost $C(k^*)$ and $C(k_c)$ and the bounds of Eq. (16) vs transmission probability p_x, arrival probability p_r, and network size M.

increases exponentially when the probability p_r increases, as shown in Fig. 5(b). This result also holds for the bounds of the iteration cost in Eq. (16), as Fig. 5 shows.

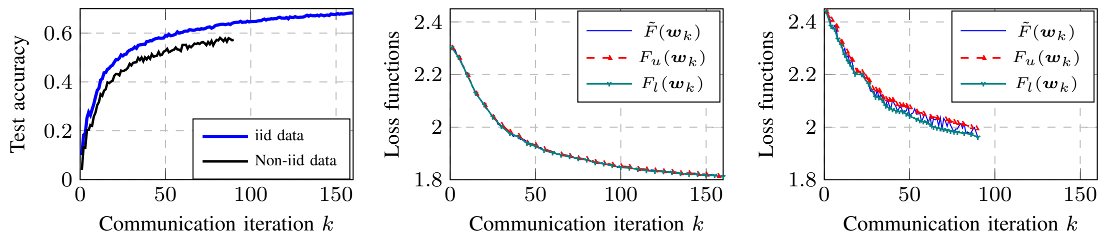
D. Performance of Non-convex FedCau from Algorithm 3

Fig. 6 characterizes the effect of the number of local iterations E on the performance of the bounds obtained from mini-batch FedCau update of Algorithm 3 with non-convex loss functions, CIFAR-10 iid dataset and CSMA/CA for $M = 50, 100$, $p_x = 0.8$, and $p_r = 0.01$. Fig. 6(a) shows that number of local iterations E has different effect on the causal test accuracy for $M = 50$ and $M = 100$. For $M = 50$, the changes in the test accuracy vs E is less than the case for $M = 100$. However, the value of test accuracy in the case with $M = 50$ is lower than the case with $M = 100$. Fig. 6(b), shows the causal stopping iterations of k_c^u and k_c^l for $M = 50$ and $M = 100$. We observe that in Fig. 6(b), the causal stopping iterations decrease while E increases. Finally, Fig. 6(c), demonstrates the causal iteration-cost $C(K)$ vs E, which



(a) Accuracy vs E. (b) Causal stopping iteration vs E. (c) Iteration-cost vs E.

Fig. 6: Illustration of the effect of number of local iterations E on the performance of mini-batch FedCau update of Algorithm 3 for non-convex loss functions with CIFAR-10 iid dataset and CSMA/CA, $M = 50, 100$, $p_x = 0.8$, and $p_r = 0.01$.



(a) Test accuracy by k_c^l. (b) Loss function, iid data. (c) Loss function, non-iid data

Fig. 7: Performance comparison of CIFAR-10 iid and non-iid data in mini-batch FedCau update of Algorithm 3 for non-convex loss functions with CSMA/CA, $M = 100$, $E = 10$, $p_x = 0.8$, and $p_r = 0.01$. a) Test accuracy of CIFAR-10 iid and non-iid dataset obtained by the lower bound causal stopping iteration k_c^l. b) Loss function $\tilde{F}(\boldsymbol{w}_k)$ with its upper bound $F_u(\boldsymbol{w}_k)$ and lower bound $F_l(\boldsymbol{w}_k)$ for iid data, and c) non-iid data.

increases while E increases. This behavior shows a strong effect of the computation latency on the FedCau performance. The interesting results in Fig. 6 bring us with a new insight of the choosing E and the trade-off between E and test accuracy, iteration-cost and the causal stopping iterations. These results also highlight the significant role of the computation latency when it comes to the large dataset. According to Fig. 6, the best choice for the number of local iterations E is $E = 10$, which provides us the best possible accuracy with the lower causal iteration-cost compared to $E > 10$.

Fig. 7 compares the performance of mini-batch FedCau update of Algorithm 3 in iid and non-iid data distribution of CIFAR-10, for non-convex loss functions with CSMA/CA, $M = 100$, $E = 10$, $p_x = 0.8$, and $p_r = 0.01$. Fig. 7(a) compares the test accuracy of training the mini-batch FedCau update of Algorithm 3 by iid and non-iid data obtained by k_c^l, the highest test accuracy achieved by Algorithm 3 for any non-convex loss function. We observe that for iid case, with $k_c^l = 160$ the test accuracy is higher than the case for non-iid with $k_c^l = 90$. Figs. 7(b) and 7(c) show the loss functions $\tilde{F}(\boldsymbol{w}_k)$ and the corresponding upper and lower bounds $F_u(\boldsymbol{w}_k)$ and $F_l(\boldsymbol{w}_k)$. Considering $K^{\max} = 200$, and comparing with FedAvg, FedCau reduces in 20% the overall iteration-cost while losing only 2% of the test accuracy in iid case. In the non-iid case, FedCau reduces the overall iteration cost by 55%, while losing only 4% of the test accuracy. Thus, FedCau outperforms FedAvg in saving iteration costs while sacrificing a tiny percentage of test accuracy in the iid case. The comparison between Fig. 7(b) and Fig. 7(c) reveals that the iid case results in a lower value of loss function and a higher test accuracy as shown in Fig. 7(a). Besides, the difference between the upper bound and the lower bound functions $F_u(\boldsymbol{w}_k)$ and $F_l(\boldsymbol{w}_k)$ are small in Figs. 7(b) and 7(c), which shows the high tightness of the bounds.

E. Performance of FedCau Update from Algorithm 1 on Top of LAQ and Top-q

Here, we choose LAQ because it achieves the same linear convergence as the gradient descent in the strongly convex case, while effecting major savings in the communication in terms of transmitted bits and communication iterations [8]. Among all the compression methods, we choose top-q sparsification because it suffers least from non-i.i.d. data, and the training still converges. Moreover, applying top-q for the logistic regression classifier trained on MNIST, the convergence does not slow down [7].

Despite the previous numerical results, which characterize the overall latency as the iteration-cost $c_k, k \geq 1$, here we consider the number of bits per each communication-iteration as $c_k, k \geq 1$ for these methods. In LAQ, the number of bits b shows the maximum amount of bits, element-wise, for the local parameters. Besides, in the top-q method, we change the percentage of the dimension of each local parameter, as $0 < q \leq 1$, but considering that each element contains 32 bits. TABLE I shows the comparison between FedAvg and FedCau with and without considering the communication-efficient methods LAQ and Top-q. Among the methods in TABLE I, FedCau LAQ with $b = 2$ achieves the test accuracy of 94.2% and spends the least number of bits (total cost of $4.56M$bits). After that, FedCau Top-q, $q = 0.1$, with a test accuracy of 92.4% and a total cost of $6.19M$bits, is the second best method.

The other baseline we consider is FedAvg with the number of iterations $K^{\max} = 200$, as shown in the last row of Table I in Section V-E. The numerical results show that FedAvg requires $250.88M$bits to perform $K^{\max}$ iterations to achieve a test accuracy of 99%, while FedCau LAQ with $b = 2$ achieves the test accuracy of 94.2% with a total cost of $4.56M$bits. This result shows us that FedCau LAQ with $b = 2$ reduces approximately 90% of the total communication cost while sacrificing less than 5% of the test accuracy. Moreover, the same analysis is valid for FedCau Top-q, $q = 0.1$, which achieves a test accuracy of 92.4% and a total cost of $6.19M$bits. Compared to FedAvg, FedCau Top-q, $q = 0.1$, reduces more than 90% the communication cost while resulting in less than 8% reduction of the test accuracy.

VI. Conclusion

In this paper, we proposed a framework to design cost-aware FL over networks. We characterized the communication-computation cost of running iterations of generic FL algo-

TABLE I: Comparison between FedCau and FedAvg with and without LAQ and Top-q, $M = 50$, and $K^{\max} = 200$.

Method	**Stop iteration**	**Total cost (Mbits)**	**Test accuracy (%)**
FedCau LAQ, $b = 2$	57	4.56	94.2
FedCau LAQ, $b = 10$	43	16.92	87.8
FedCau Top-q, $q = 0.1$	49	6.19	92.4
FedCau Top-q, $q = 0.6$	43	32.4	80.9
FedCau	56	70.24	96.4
FedAvg, LAQ, $b = 2$	200	16	98
FedAvg LAQ, $b = 10$	200	78.7	98.7
FedAvg, Top-q, $q = 0.1$	200	25	98.9
FedAvg Top-q, $q = 0.6$	200	150.72	97.5
FedAvg	200	250.88	99.02

rithms over a shared wireless channel, regulated by slotted-ALOHA, CSMA/CA, and OFDMA protocols. We posed the communication-computation latency as the iteration-cost function of FL, and showed that increasing the network size and background traffic may prohibit the application of FL over wireless networks. We optimized the iteration-termination criteria to minimize the trade-off between FL achievable objective value and the overall training cost. To this end, we proposed a causal setting, named FedCau, utilized in two convex scenarios for batch and mini-batch updates, and proposed a solution for non-convex scenarios as well.

The numerical results showed that in the same background traffic, time budget, and network situation, CSMA/CA has less communication-computation cost than slotted-ALOHA. We also showed that the mini-batch FedCau update could perform more cost-efficiently than the batch update by choosing proper time budgets. Moreover, the numerical results of the non-convex scenario provided a sub-optimal interval of the causal optimal solution close to the optimal interval, which provides many opportunities for non-convex FL problems. At the end, we applied FedCau method on top of the existing methods like top-q sparsification and LAQ with characterizing the iteration-cost as number of communication bits. We concluded that FedCau provides us a more efficient training even when we apply it on the top of efficient methods.

Our future work will extend the FedCau update of non-convex scenario and design communication protocols for cost efficient FL. We will also extend this work to consider optimal power allocation jointly with causal stopping points, which will investigate the problem of using FL low-power in devices such that the training stops at a causal point to avoid further unnecessary iterations.

Appendix A

A. Proof of Lemma 1

The proof is obtained from the definition of discrete derivative of a discrete function $G(K)$ [34]. Consider K_1 such that $G(K_1 - 1) \geq G(K_1)$, and $G(K_1 + 1) \geq G(K_1)$. Therefore, $G(K_1)$ is the minimum value of $G(K), K = 1, \dots, K_1 + 1$, and $k^* = K_1$.

B. Proof of Lemma 2

As $f_j(\boldsymbol{w}), j \in [M]$ is convex, the sequence of $f_k^j, j \in [M], k \geq 1$, (see Remark 1) converges to a global minima for each $f_j(\boldsymbol{w}), j \in [M]$. Consider the procedure of updating each $\hat{f}(\boldsymbol{w}_k) =$

$\sum_{j\in\mathcal{M}_a}\rho_j f_k^j+\sum_{j'\notin\mathcal{M}_a}\rho_{j'} f_{k-1}^{j'}$ in Algorithm 2 which replaces the missing parameter $f_k^{j'}, j'\notin\mathcal{M}_a$ with $f_{k-1}^{j'}$ of the previous iteration. Note that in the first iteration, all the workers were successful in sending their local parameters to the master node, which results in existing at least one local parameter for each worker in the master node's memory. According to Remark 1, $f_k^j\le f_{k-1}^j$, thus, if all the local parameters of worker $j\in[M]$ are available in the master node, $\sum_{j=1}^M\rho_j f_k^j\le\sum_{j=1}^M\rho_j f_{k-1}^j$. This inequality provides us with

$$\sum_{j=1}^{M}\rho_j f_k^j\le\sum_{j\in\mathcal{M}_a}\rho_j f_k^j+\sum_{j'\notin\mathcal{M}_a}\rho_{j'} f_{k-1}^{j'}\le\sum_{j=1}^{M}\rho_j f_{k-1}^j, \tag{20}$$

where $\sum_{j\in\mathcal{M}_a}\rho_j+\sum_{j'\notin\mathcal{M}_a}\rho_{j'}=1$. Inequalities (20) show that $f(\boldsymbol{w}_k)\le\hat{f}(\boldsymbol{w}_k)\le f(\boldsymbol{w}_{k-1})$.

C. Proof of Proposition 1

Define $\Delta_k:=f_{k-1}-f_k$, $\Delta_0:=f_0$, and $\eta_k:=c_k/\Delta_k$. According to the diminishing return rule, we obtain $\Delta_k\le\Delta_{k-1}$. For the sake of mathematical modelling, we express the amount of the cost of each communication iteration k, c_k, as $\mathcal{O}(c_k)\le\mathcal{O}(\Delta_k/z)\le\mathcal{O}(\Delta_0/z)$, where $z\ge1$ is the intended scale factor of cost.

The mapping should be fixed before starting the computation of $G(K)$. As we do not know the future information, we have to design the mapping with the very first information received from workers. We assume that the master node designs such a mapping at the beginning of communication iteration $k=1$ and does not change it during the training. Sine at the communication iteration $k=1$, the information Δ_0 and c_1 are available at the master node, the mapping has to be determined by these information to lead to a causal design. Thus, defining Ex_k as the total expenses (in seconds or bits), the master node defines the mapping function as $M_{\text{map}}:(\text{Ex}_k,\Delta_0,z)\to c_k$ and finds c_k named iteration-cost. Next, by the definition of $G(K)$ in (7), the scalarization parameter β which satisfies Lemma 1, takes the value between

$$0<\frac{1}{1+\eta_{k^*}}\le\beta\le\frac{1}{1+\eta_k},\ \ k\le k^*. \tag{21}$$

Inequalities in (21) are non-causal and need the information of Δ_k and c_k from the future communication iterations. Thus, we must demonstrate (21) by some causal information and bounds on c_k and Δ_k for $k=1,\dots,k^*$. Considering the overall latency when applying MAC protocols for communication iterations, we obtained the bounds on c_k in (17). Thus, the interval

$$0<\frac{1}{1+\frac{\max_k\,c_k}{\Delta_0}}\le\beta\le\frac{1}{1+\frac{\min_k\,c_k}{\Delta_0}}<1,\ \ k\le k^*, \tag{22}$$

in which $\Delta_0\ge\Delta_k, k\ge1$ is the only causal information about Δ_k that is available at $k=1$.

D. Proof of Proposition 2

The proof is ad-absurdum. Let us assume that $k^*\to\infty$, which means there is no optimal stopping communication iteration to solve (6). Using the descent property of FedAvg algorithm (3), which imposes $f(\boldsymbol{w}_1) \leq f(\boldsymbol{w}_2) \leq \ldots \leq f(\boldsymbol{w}_{k^*})$, together with $k^* \to \infty$, we have that $f(\boldsymbol{w}_{k^*})$ converges to f^*, $f(\boldsymbol{w}_{k^*}) \to f^* < \{f(\boldsymbol{w}_k)\}_k$. Besides, the inequality $G(K+1) < G(K)$ is still held because the algorithm has not found the optimal communication iteration in which the $G(K)$ is minimized. As a result, when $k^* \to \infty$, we also have $G(k^*) < G(k^* - 1)$, which results in $\sum_{k=1}^{k^*} c_k < \sum_{k=1}^{k^*-1} c_k$, thus $c_{k^*} < 0$. This is in contradiction with our assumption that $c_{k=1,2,\ldots}$ are positive. We conclude that there is a finite k^*. Therefore,there is a finite k_c for the optimization problem (6).

E. *Proof of Theorem 1*

In the beginning of the iterations, the master node sets $\epsilon = +\infty$ to ensure a proper running of the iterations. Inequality $G(K + 1) < G(K)$ implies that the loss function is still in the decreasing region and we have not found the minimum yet. As soon as the sign changes, the algorithm would terminate by setting the value of $\epsilon = 0$, the current k as k_c. Then, the algorithm reports the corresponding values for $G(k_c)$ and $f(\boldsymbol{w}_{k_c})$. Clearly, the penalty is up to one additional communication iteration when we observe an increase in $G(K)$ and the master node terminates the process, leading to $k_c = k^* + 1, \forall k^* < K^{\max}$, and by setting $k^* = K^{\max}$, then $k_c = k^*$ as well. Given (14a), the “descent property” of FedAvg algorithm (3) yields (14b). Then, inequality (14c) is a direct consequence of the definition of k^* in (6).

F. *Proof of Lemma 3*

The proof is directly obtained from the definitions of $F_u(\boldsymbol{w}_k)$, $F_l(\boldsymbol{w}_k)$, δ_k^u and δ_k^l in Algorithm 3. Recall that $F_u(\boldsymbol{w}_k) = \tilde{F}(\boldsymbol{w}_k)$ (see line 24, 27, 30, 44) or $F_u(\boldsymbol{w}_k) = F_u(\boldsymbol{w}_{k-1}) + \delta_k^l$ (see line 40) and the same arguments considering are valid for $F_l(\boldsymbol{w}_k) = \tilde{F}(\boldsymbol{w}_k)$ or $F_l(\boldsymbol{w}_k) = F_l(\boldsymbol{w}_{k-1}) + \delta_k^u$ (see lines 24, 33, 37, 46). Thus, by assuming a finite sequence of $|\tilde{F}(\boldsymbol{w}_k)|, k = 1, \ldots, K$, the inequality $|F_u(\boldsymbol{w}_k) - F_l(\boldsymbol{w}_k)| \leq \tilde{F}_{\max} - \tilde{F}_{\min}$ is the tightness between the upper bound $F_u(\boldsymbol{w}_k)$ and the lower bound $F_l(\boldsymbol{w}_k)$.

G. *Proof of Proposition 4*

The stopping iteration k_c given by Algorithm 3 is in the form of an interval of $k_c \in [k_c^u, k_c^l]$. This interval's tightness depends on different scenarios, as we explain in the following. Assuming that Algorithm 3 has obtained k_c^u, after which we face several situations for updating $F_l(\boldsymbol{w}_k)$ according to the behavior of $\tilde{F}(\boldsymbol{w}_k)$ for $k \geq k_c^u + 1$. The different scenarios are

- If $\tilde{F}(\boldsymbol{w}_k) < \tilde{F}(\boldsymbol{w}_{k-1})$ and $\tilde{F}(\boldsymbol{w}_k) > F_l(\boldsymbol{w}_{k-1})$, according to Algorithm 3 (see lines 44-47), for $k = k_c^u + 1$, we have $F_u(\boldsymbol{w}_k) = \tilde{F}(\boldsymbol{w}_k)$ and $F_l(\boldsymbol{w}_k) = F_l(\boldsymbol{w}_{k-1}) - \delta_k^u$ where

$\delta_k^u = F_u(\boldsymbol{w}_{k-1}) - F_u(\boldsymbol{w}_k) = F_u(\boldsymbol{w}_{k-1}) - \tilde{F}(\boldsymbol{w}_k)$. Moreover, since $k \geq k_c^u + 1$, the inequality of $G_u(k) > G_u(k)$ gives us

$$\begin{aligned} G_u(k) - G_u(k_c^u) =& \beta C(k) + (1-\beta)F_u(\boldsymbol{w}_k) - \beta C(k-1) - (1-\beta)F_u(\boldsymbol{w}_{k-1}) \\ =& \beta c_k - (1-\beta)(F_u(\boldsymbol{w}_{k-1}) - F_u(\boldsymbol{w}_k)) > 0, \end{aligned} \tag{23}$$

then we compute $G_l(k)$

$$\begin{aligned} G_l(k) =& \beta C(k) + (1-\beta)F_l(\boldsymbol{w}_k) = \beta C(k-1) + \beta c_k + (1-\beta)(F_l(\boldsymbol{w}_{k-1}) - \delta_k^u) \\ =& \beta C(k-1) + (1-\beta)(F_l(\boldsymbol{w}_{k-1}) - F_u(\boldsymbol{w}_{k-1}) + \tilde{F}(\boldsymbol{w}_k)) + \beta c_k \\ =& G_l(k-1) + \beta c_k - (1-\beta)(F_u(\boldsymbol{w}_{k-1}) - \tilde{F}(\boldsymbol{w}_k)) \overset{(23)}{>} G_l(k-1). \end{aligned} \tag{24}$$

Therefore, $k_c^l = k_c^u + 1$.

- If $\tilde{F}(\boldsymbol{w}_k) < \tilde{F}(\boldsymbol{w}_{k-1})$ and $\tilde{F}(\boldsymbol{w}_k) < F_l(\boldsymbol{w}_{k_{\max}^l})$, according to Algorithm 3 (see lines 36-44), for $k = k_c^u + 1$, we have $F_l(\boldsymbol{w}_k) = \tilde{F}(\boldsymbol{w}_k)$, and $F_u(\boldsymbol{w}_k) = F_u(\boldsymbol{w}_{k-1}) - \delta_k^l$, where $\delta_k^l = F_l(\boldsymbol{w}_{k-1}) - F_l(\boldsymbol{w}_k) = F_l(\boldsymbol{w}_{k-1}) - \tilde{F}(\boldsymbol{w}_k)$. Thus, we calculate $G_l(k)$ as

$$\begin{aligned} G_l(k) =& \beta C(k) + (1-\beta)F_l(\boldsymbol{w}_k) = \beta C(k-1) + \beta c_k + (1-\beta)\tilde{F}(\boldsymbol{w}_k) \\ =& \beta C(k-1) + (1-\beta)(F_u(\boldsymbol{w}_k) + F_l(\boldsymbol{w}_{k-1}) - F_u(\boldsymbol{w}_{k-1})) + \beta c_k \\ =& G_l(k-1) + \beta c_k - (1-\beta)(F_u(\boldsymbol{w}_{k-1}) - F_u(\boldsymbol{w}_k)) \overset{(23)}{>} G_l(k-1), \end{aligned} \tag{25}$$

which results in $k_c^l = k_c^u + 1$.

- The last scenario is when $\tilde{F}(\boldsymbol{w}_k) > F_u(\boldsymbol{w}_{k-1})$ (see lines 28-34 in Algorithm 3). In this case, the update of $F_u(\boldsymbol{w}_k)$ and $F_l(\boldsymbol{w}_k)$ are according to the linear update we proposed in Section III-D in the revised manuscript. Thus, the update of δ_k^u is as

$$\delta_k^u = \frac{\tilde{F}(\boldsymbol{w}_k) - F_u(\boldsymbol{w}_{k_{\max}^u})}{k - k_{\max}^u}, \quad k \geq k_c^u + 1, \tag{26}$$

and $F_l(\boldsymbol{w}_k) = F_l(\boldsymbol{w}_{k-1}) - \delta_k^u$, and $F_u(\boldsymbol{w}_k) = \tilde{F}(\boldsymbol{w}_k)$. Next, we calculate $G_l(k)$ as

$$\begin{aligned} G_l(k) =& \beta C(k) + (1-\beta)F_l(\boldsymbol{w}_k) = \beta C(k-1) + \beta c_k + (1-\beta)(F_l(\boldsymbol{w}_{k-1}) - \delta_k^u) \\ =& G_l(k-1) + \beta c_k - (1-\beta)\delta_k^u, \end{aligned} \tag{27}$$

where $G_l(k) - G_l(k-1) = \beta c_k - (1-\beta)\delta_k^u$. Thus, k_c^l is obtained when $\beta c_k > (1-\beta)\delta_k^u$, namely

$$\delta_k^u = \frac{\tilde{F}(\boldsymbol{w}_k) - F_u(\boldsymbol{w}_{k_{\max}^u})}{k - k_{\max}^u} < \frac{\beta}{1-\beta} c_k, \quad k \geq k_c^u + 1, \tag{28}$$

where k_c^l is

$$k_c^l = k_{\max}^u + \left\lceil (1-\beta) \frac{\tilde{F}(\boldsymbol{w}_k) - F_u(\boldsymbol{w}_{k_{\max}^u})}{\beta c_k} \right\rceil + 1 \tag{29}$$

Therefore, according to the three scenarios we mentioned, we obtain

$$k_c^l = 1 + \max\left\{ k_c^u, k_{\max}^u + \left\lceil (1-\beta) \frac{\tilde{F}(\boldsymbol{w}_k) - F_u(\boldsymbol{w}_{k_{\max}^u})}{\beta c_k} \right\rceil \right\}.$$

H. Proof of Proposition 5

Consider a set of sates defined as $\mathcal{S} := \{0, 1, \ldots, M\}$ where each state $i \in \mathcal{S}$ represents the nodes which have successfully transmitted their local parameters to the master node. The critical point to consider is that once a node successfully sends its local parameter (or its head-of-line packet) to the master node, the state i jumps to the sate $i+1$. While if the successfully transmitted packet is one of the background packets, the system still stays in the state i. Due to this reason, we consider the current state as the number of worker nodes which have been successful in transmitting at least one packet. As a result, the possible states in our proposed approach is from 0 (none of the nodes have transmitted packets) to M (all of the nodes have transmitted packets). The first step to calculate the average communication delay $\mathbb{E}\left\{\ell_{3,k}\right\}$, is to obtain the transition probabilities between the states, namely $p_{i,i}$ and $p_{i,i+1}$. We consider each probability to show the corresponding action at each time slot, and denote them as:

- $p_{i,i}$: the probability that no new node transmits. Thus, we have one of the following events: 1) No successful transmission in the system, 2) Idle time slot, or Just one node $j \in \{1, 2, \ldots, i\}$ transmits a background packet successfully.
- $p_{i,i+1}$: the probability of a new node transmits successfully;

where these probabilities are mathematically defined in Proposition 5.

Next, we consider that each of the probabilities of $p_{i,i}$ and $p_{i,i+1}$ lasts for $t_{i,i}, t_{i,i+1} \in \mathbb{Z}^+$ time slots, respectively. We know that $t_{i,i+1} = 1$ as each successful transmission spends one time slot. Then, consider t_s s to be the time slot duration, in which we calculate an upper bound for the average communication latency $\mathbb{E}\left\{\ell_{3,k}\right\}$ when $t_{i,i} \to +\infty$ as:

$$\mathbb{E}\left\{\ell_{3,k}\right\} \leq \sum_{i=0}^{M-1} t_s \left\{ p_{i,i+1} + \lim_{t_{i,i}\to+\infty} \sum_{u=0}^{t_{i,i}} u p_{i,i}^{u} \right\}, \tag{30}$$

where $\lim_{t_{i,i}\to+\infty} \sum_{u=0}^{t_{i,i}} u p_{i,i}^{u} = p_{i,i}/(1-p_{i,i})^2$. Then,

$$\mathbb{E}\left\{\ell_{3,k}\right\} \leq \sum_{i=0}^{M-1} t_s \left\{ p_{i,i+1} + \frac{p_{i,i}}{(1-p_{i,i})^2} \right\}. \tag{31}$$

Next, the lower bound is obtained easily by considering $t_{ii} = 0$, thus

$$\sum_{i=0}^{M-1} t_s p_{i,i+1} \leq \mathbb{E}\left\{\ell_{3,k}\right\}. \tag{32}$$

REFERENCES

[1] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[2] O. Simeone, "A very brief introduction to machine learning with applications to communication systems," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 4, pp. 648–664, 2018.

[3] J. Park, S. Samarakoon *et al.*, "Wireless network intelligence at the edge," *Proc. IEEE*, vol. 107, no. 11, pp. 2204–2239, Nov. 2019.

[4] P. Kairouz *et al.*, "Advances and open problems in federated learning," 2019.

[5] J. Konečnỳ, H. B. McMahan *et al.*, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[6] N. C. Thompson, K. Greenewald *et al.*, "Deep learning's diminishing returns: The cost of improvement is becoming unsustainable," *IEEE Spectrum*, vol. 58, no. 10, pp. 50–55, 2021.

[7] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient Federated Learning from non-i.i.d. data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3400–3413, 2020.

[8] J. Sun, T. Chen *et al.*, “Lazily Aggregated Quantized Gradient (LAQ) innovation for communication-efficient federated learning,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 4, pp. 2031–2044, 2022.

[9] S. Caldas, J. Konečny, H. B. McMahan, and A. Talwalkar, “Expanding the reach of federated learning by reducing client resource requirements,” *arXiv preprint arXiv:1812.07210*, 2018.

[10] Q. Fan and N. Ansari, “Application aware workload allocation for edge computing-based IoT,” *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2146–2153, 2018.

[11] Z. Yang, M. Chen *et al.*, “Energy efficient federated learning over wireless communication networks,” *IEEE Transactions on Wireless Communications*, pp. 1–1, 2020.

[12] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, “Sparsified SGD with memory,” in *Advances in Neural Information Processing Systems*, 2018, pp. 4447–4458.

[13] S. Magnússon, C. Enyioha, N. Li *et al.*, “Convergence of limited communications gradient methods,” *IEEE Transactions on Automatic Control*, vol. 63, no. 5, pp. 1351–1371, May 2017.

[14] S. Di, D. Tao *et al.*, “Efficient lossy compression for scientific data based on pointwise relative error bound,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 2, pp. 331–345, 2018.

[15] K. Yuan, Q. Ling, and Z. Tian, “Communication-efficient decentralized event monitoring in wireless sensor networks,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 8, pp. 2198–2207, 2014.

[16] J. Wangni, J. Wang *et al.*, “Gradient sparsification for communication-efficient distributed optimization,” in *Advances in Neural Information Processing Systems*, 2018, pp. 1299–1309.

[17] T. Chen, G. Giannakis *et al.*, “LAG: Lazily aggregated gradient for communication-efficient distributed learning,” in *Advances in Neural Information Processing Systems*, 2018, pp. 5050–5060.

[18] J. Sun, T. Chen *et al.*, “Communication-efficient distributed learning via lazily aggregated quantized gradients,” in *Advances in Neural Information Processing Systems*, 2019, pp. 3370–3380.

[19] K. Hsieh, A. Harlap *et al.*, “Gaia: Geo-distributed machine learning approaching {LAN} speeds,” in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, 2017, pp. 629–647.

[20] H. Yu, S. Yang, and S. Zhu, “Parallel restarted SGD for non-convex optimization with faster convergence and less communication,” *arXiv preprint arXiv:1807.06629*, 2018.

[21] S. Luo, X. Chen *et al.*, “HFEL: Joint edge association and resource allocation for cost-efficient Hierarchical Federated Edge Learning,” *IEEE Transactions on Wireless Communications*, vol. 19, no. 10, pp. 6535–6548, 2020.

[22] H. S. Ghadikolaei, S. Stich, and M. Jaggi, “LENA: Communication-efficient distributed learning with self-triggered gradient uploads,” in *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 130. PMLR, 13–15 Apr 2021, pp. 3943–3951.

[23] A. Defazio, F. Bach, and S. Lacoste-Julien, “SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives,” in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, Eds., vol. 27. Curran Associates, Inc., 2014.

[24] M. Chen, Z. Yang *et al.*, “A joint learning and communications framework for Federated Learning over wireless networks,” *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 269–283, 2021.

[25] A. Mahmoudi, H. S. Ghadikolaei, and C. Fischione, “Cost-efficient distributed optimization in machine learning over wireless networks,” in *IEEE International Conference on Communications (ICC)*, 2020.

[26] ——, “Machine learning over networks: Co-design of distributed optimization and communications,” in *IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2020.

[27] S. Boyd and L. Vandenberghe, *Convex Optimization*. USA: Cambridge University Press, 2004.

[28] E. Ziouva and T. Antonakopoulos, “CSMA/CA performance under high traffic conditions: Throughput and delay analysis,” *Computer communications*, vol. 25, no. 3, pp. 313–321, 2002.

[29] D. Bankov, A. Didenko *et al.*, “OFDMA uplink scheduling in IEEE 802.11ax networks,” in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6.

[30] X. Li, K. Huang *et al.*, “On the convergence of FedAvg on non-iid data,” *arXiv preprint arXiv:1907.02189*, 2019.

[31] I. Griva, S. G. Nash, and A. Sofer, *Linear and Nonlinear Optimization (2. ed.)*. SIAM, 2008.

[32] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Prentice hall Englewood Cliffs, NJ, 1989, vol. 23.

[33] A. Balatsoukas-Stimming and C. Studer, "Deep unfolding for communications systems: A survey and some new directions," in *2019 IEEE International Workshop on Signal Processing Systems (SiPS)*. IEEE, 2019, pp. 266–271.

[34] H.-G. Weigand, "A discrete approach to the concept of derivative," *ZDM – Mathematics Education*, vol. 46, no. 4, pp. 603–619, 2014.

[35] L. Bottou, F. Curtis *et al.*, "Optimization methods for large-scale machine learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018.

[36] A. Mahmoudi *et al.*, "FedCau: A proactive stop policy for communication and computation efficient Federated Learning," *arXiv preprint arXiv:2204.07773*, 2022.

[37] D. P. Bertsekas, R. G. Gallager, and P. Humblet, *Data networks, second edition*. Prentice-Hall International New Jersey, 2004, vol. 2.

[38] Y. Yang and T. Yum, "Delay distributions of slotted ALOHA and CSMA," *IEEE Transactions on Communications*, vol. 51, no. 11, pp. 1846–1857, Nov. 2003.

[39] F. S. Samani, H. Zhang, and R. Stadler, "Efficient learning on high- dimensional operational data," in *15th IEEE International Conference on Network and Service Management (CNSM)*, 2019.

[40] V. D. Nguyen, S. K. Sharma, T. X. Vu, S. Chatzinotas, and B. Ottersten, "Efficient federated learning algorithm for resource allocation in wireless IoT networks," *IEEE Internet of Things Journal*, 2020.

[41] F. Malandra and B. Sansò, "A markov-modulated end-to-end delay analysis of large-scale RF mesh networks with time-slotted ALOHA and FHSS for smart grid applications," *IEEE Transactions on Wireless Communications*, vol. 17, no. 11, pp. 7116–7127, 2018.

[42] Y. Yang and T.-S. Yum, "Delay distributions of slotted ALOHA and CSMA," *IEEE Transactions on Communications*, vol. 51, no. 11, pp. 1846–1857, 2003.

[43] E. Casini, R. De Gaudenzi, and O. Del Rio Herrero, "Contention resolution diversity slotted ALOHA (CRDSA): An enhanced random access schemefor satellite access packet networks," *IEEE Transactions on Wireless Communications*, vol. 6, no. 4, pp. 1408–1419, 2007.

[44] T. D. Hoang and L. Bao Le, "Joint prioritized scheduling and resource allocation for OFDMA-based wireless networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 1, pp. 310–323, 2018.

[45] K. Koh, S.-J. Kim, and S. Boyd, "An interior-point method for large-scale ℓ_1-regularized logistic regression," *Journal of Machine Learning Research*, vol. 8, no. Jul, pp. 1519–1555, 2007.

[46] "IEEE Standard for Information technology—Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pp. 1–3534, Dec 2016.