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

Afsaneh Mahmoudi, Graduate Student Member, IEEE, Hossein S. Ghadikolaei, Member, IEEE, and Carlo Fischione, Senior Member, IEEE

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 iterative-stop-termination method that trades off 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 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 in 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 learning 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 to the learning process without sharing their own data with other workers and the master node. Specifically, FL methods refer to a class of privacy-preserving distributed learning algorithms in which individual workers \([M]\) execute some local iterations 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., \(\{f_j : \mathbb{R}^d \rightarrow \mathbb{R}, j \in [M]\}\) with corresponding local parameters \(\{w^j \in \mathbb{R}^d\}_{j \in [M]}\). The common solution to such a problem involves an iterative procedure wherein at each global communication iteration \(k\), workers have to find the local parameter \(\{w^j_k\}_{j \in [M]}\) and upload them to a central controller. Then, the master node updates the model parameters as \(w_{k+1}\) and broadcasts it to all the nodes to start the next iteration [5].

The FL algorithm alleviates computation and privacy by parallel computations at workers using their local private data [5]. However, such an algorithm introduces a communication cost: parameter vectors, such as weight and bias, must be communicated between the master and the workers to run a new iteration. The weights can be vectors 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, e.g., computation,

© 2024 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
latency, communication resource utilization, and 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 training purposes.

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-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 test 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], [10], [11].

A. Literature Survey

Cost-efficient distributed training is addressed in the literature through communication-efficiency [4], [12], [13], [14], [15], [16], [17], [18], [19] or tradeoff between computation and communication primarily by resource allocation [11], [20]. 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], [13]. 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 [14]. 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], [15].

The second class of approaches focuses on the reduction of the communication iterations by eliminating the communication between some of the workers and the master node in some iterations [16]. The work [16] 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 [17] by sending quantized versions of the gradient vectors. In [19], local SGD techniques reduce the number of communication rounds needed to solve an optimization problem. In a generic FL setting, adding more local iterations 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 [18] 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. Furthermore, [18] has shown that adding a memory unit at the master node and using ideas from SAGA [21] reduce the upload frequency of each worker, thus improving the communication efficiency.

The two classes mentioned above present opportunities for reducing the cost of running distributed training algorithms and adapting them to wireless communication protocols. However, these classes focus primarily on the complexity of the iterative algorithm in terms of bits per communication round or the number of communication rounds [9]. Moreover, they neglect other crucial costs associated with solving federated learning (FL) problems, such as latency [3] and energy consumption [20]. These costs can render distributed algorithms ineffective in bandwidth or battery-limited wireless networks, where latency and energy consumption are critical factors.

Recent works have explored the co-design of optimization problems and communication networks, particularly in the context of computational offloading [10], [11], [22]. These works have addressed task offloading, resource allocation optimization, and joint learning of wireless resource allocation and user selection. In contrast to existing literature, our approach differs by proactively designing stopping criteria to optimize tradeoffs rather than treating them as hyper-parameters set through cross-validations. This distinction makes our approach original and distinct from current state-of-the-art algorithms.

In our preliminary works, we have characterized the overall communication-computation of solving a distributed gradient descent problem where the workers had background traffic and followed a channel from medium access control (MAC) protocols using random access, such as slotted-ALOHA [23] or CSMA/CA [24] 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 achievable accuracy before the algorithm’s design phase. This paper constitutes a major step in addressing this important research gap. Previously in [23] and [24], we proposed a cost-efficient framework considering the cost of each iteration 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 the 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], [20]. 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 [25]. 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 in 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, for the multi-objective optimization above, one with a focus on original FL and the others with a 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 MNIST and CIFAR-10 datasets, 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 [26], or fixed assignment access protocol like OFDMA [27];
- We apply our proposed FedCau on top of top-\(q\) sparsification and lazily aggregated quantized gradient (LAQ) methods showing vast applications of the FedCau [7], [8];
- The experimental results highlight the ability of our proposed FedCau to achieve efficient and accurate training. We conclude that a co-design of distributed optimization algorithms and communication protocols is 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 \(w\), bold-font capital letter \(W\), and calligraphic font \(W\) denote scalar, vector, matrix, and set, respectively. We define the index set \([N] = \{1, 2, \ldots, N\}\) for any integer \(N\). We denote by \(||\cdot||\) the \(l_2\)-norm, by \(|A|\) the cardinality of set \(A\), by \([w]_i\), the entry \(i\) of vector \(w\), by \(w^\top\) the transpose of \(w\), and \(1_x\) is an indicator function taking 1 if and only if \(x\) is true, and takes 0 otherwise.

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 afterward, we propose the main approach of this paper.

A. Federated Learning

Consider a star network of \(M\) workers that cooperatively solve a distributed training problem involving a loss function \(f(w)\). Consider \(D\) as the whole dataset distributed among each worker \(j \in [M]\) with \(D_j\) data samples. Let tuple \((x_{ij}, y_{ij})\) denote data sample \(i\) of \(D_j\) samples of worker \(j\) and \(w \in \mathbb{R}^d\) 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

\[
    w^* \in \arg \min_{w \in \mathbb{R}^d} f(w) = \sum_{j=1}^M \rho_j f_j(w), \tag{1}
\]
where \( f_j(w) := \sum_{i=1}^{|D_j|} f(w; x_{ij}, y_{ij})/|D_j| \). Optimization problem (1) applies to a large group of functions as convex and non-convex (such as deep neural networks).

The standard iterative procedure to solve problem (1) with the initial vector \( w_0 \) is

\[
w_k = \sum_{j=1}^{M} \rho_j w_{k,j}, \quad k = 1, \ldots, K.
\]

(2)

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

Initializing the training process with \( w_0, \) Federated Averaging (FedAvg) is a distributed learning algorithm in which the master node sends \( w_{k-1} \) 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 \( f \), and computes its local parameter \( w_{i,k}^{j} \), considering the initial point of \( w_{0,k}^{j} = w_{k-1} \), [28], for any \( k = 1, \ldots, K \),

\[
w_{i,k}^{j} = w_{i-1,k}^{j} - \frac{\alpha_k}{\xi_k} \sum_{n=1}^{\xi_k} \nabla w f(w_{i-1,k}; x_{nj}, y_{nj}),
\]

(3)

where \( w_{k}^{j} = w_{i,k}^{j} \). Then each worker transmits \( w_{k}^{j} \) to the master node for updating \( w_{k} \) according to (2). Note that in FedAvg, when \( E = 1 \) and we use the exact gradient vector in 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 \( f(w) \), we use the following Remark throughout the paper.

Remark 1: (Theorem 11.7 of [29]) Consider Any Differentiable Loss Function \( f(w) : \mathbb{R}^d \mapsto \mathbb{R} \) with Lipschitz continuous gradient \( \nabla_w f(w) \), i.e., \( \|\nabla_w f(w_1) - \nabla_w f(w_2)\| \leq L\|w_1 - w_2\| \), for some constant \( 0 < L < \infty \), and let \( w_1, \ldots, w_K \) be the sequence obtained from the FL algorithm updates in Eq. (2). Then, by \( \alpha_k \geq \gamma \|\nabla_w f(w_k)\|^2 \) and for an appropriate constant \( \gamma > 0 \), the following inequality holds:

\[
f(w_1) \geq \ldots \geq f(w_K).
\]

The workers use the FedAvg algorithm (3) to compute their local parameters \( w_{i,k}^{j} \), while the master node performs the iterations of (2) until a convergence criteria for \( \|f(w_k) - f(w^*)\| < \epsilon \) is met [30]. We denote by \( K \) the first iteration at which the stopping criteria of the FedAvg algorithm is met, namely

\[
K := \text{the first value of } k \mid \|f(w_k) - f(w^*)\| < \epsilon,
\]

(4)

where \( \epsilon > 0 \) is the decision threshold for terminating the algorithm at iteration \( K \) and \( f(w^*) \) is the optimum of the loss function at the optimal parameter \( w^* \). The state-of-the-art literature defines the threshold \( \epsilon \) independently before training. However, an optimal threshold must be designed to optimize communication-computation resources in solving (1). The termination iteration \( K \) is crucial to prevent potentially adverse effects on communication-computation resource utilization in FedAvg (3) over wireless networks.

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 \( k \);
- **Energy consumption**: \( c_k \) is the energy needed for performing a global iteration to receive \( w_k \) at a worker and send \( \{w_k^{j}\}_{j \in [M]} \) 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 latency components:

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

See Section IV for more detailed modeling of the components of \( c_k \) for 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) **Communication-computation cost**: It lacks the incorporation of computation and communication costs related to local parameters and model updates. These costs depend on factors such as computation power, communication protocols, energy consumption, and overall communication resources of the local device;
2) **Number of iterations**: The termination iteration \( K \) in (4) significantly impacts the communication-computation cost of the algorithm (3). A lower \( K \) would consume fewer resources while leading to a negligible degradation in training performance, compared to a higher \( K \) that can result in substantial communication-computation costs without significant improvements in training optimality.

The termination iteration \( K \) in (4) strongly impacts the overall training costs for solving the optimization problem (1). Thus, selecting an appropriate value for \( K \) is crucial to prevent potentially adverse effects on communication-computation resource utilization in FedAvg (3) over wireless networks.

We propose an original optimization of the termination iteration \( K \) in the FedAvg algorithm (3) to tackle the mentioned challenges. By explicitly considering the cost of training iterations, we aim at obtaining an optimal stopping iteration that solves the following optimization problem:

\[
\text{minimize } K \left[ f(w_K), \sum_{k=1}^{K} c_k \right]
\]

(5a)
subject to \( w_k = \sum_{j=1}^{M} \rho_j w^j_k \), \( k = 1, \ldots, K \) \hspace{1cm} (5b)
\[ w^j_{0,k} = w_{k-1}, \quad k = 1, \ldots, K \]
\[ w^j_{i,k} = w^j_{i-1,k} \]
\[ \frac{\alpha_k}{\xi_k} \sum_{n=1}^{K} \nabla w f(w^j_{i-1,k}; x_{nj}, y_{nj}), \quad k \leq K, \] \hspace{1cm} (5c)
where \( \sum_{k=1}^{K} c_k \) quantifies the overall iteration-cost expenditure for the training of loss function \( f(w) \) 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 \( f(w) \), 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 \( w_k \), but neither \( c_k \) nor \( w_k \) 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 \( f(w) \). Thus, by solving optimization problem (5a), we can obtain the optimal number of iterations for FedAvg algorithm (3), which minimizes 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 [31], 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 \( k^* \).

Solving (5) presents several challenges: it is multi-objective, involves integer variables, and contains non-analytical objective and constraint functions with non-explicit dependencies on \( K \). Additionally, the problem is non-causal, making it difficult to determine the optimal \( K \) without knowing \( w_k \)’s in advance. Thus, addressing such non-explicit and non-causal optimization problems can be highly challenging [25]. In the next section, we propose a practical solution to problem (5).

### III. SOLUTION ALGORITHMS

In this section, we present preliminary technical results, propose an iterative solution to (5), and demonstrate that the proposed methods achieve optimal or sub-optimal solutions while converging in a finite number of iterations.

#### A. Preliminary Solution Steps

In this subsection, we develop some preliminary results to arrive at a solution to the optimization problem (5). We start by transforming (5) according to the scalarization procedure of multi-objective optimization [25]. 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 \( f(w_K) \).

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

\[ k^* \in \arg \min_K G(K) \] \hspace{1cm} (6a)
subject to \( w_k = \sum_{j=1}^{M} \rho_j w^j_k \), \( k \leq K \) \hspace{1cm} (6b)
\[ w^j_{0,k} = w_{k-1}, \quad k \leq K \] \hspace{1cm} (6c)
\[ w^j_{i,k} = w^j_{i-1,k} \]
\[ \frac{\alpha_k}{\xi_k} \sum_{n=1}^{K} \nabla w f(w^j_{i-1,k}; x_{nj}, y_{nj}), \quad k \leq K, \] \hspace{1cm} (6d)
where \( G(K) \) and \( C(K) \) are defined as

\[ G(K) := (\beta C(K) + (1 - \beta)f(w_K)) \] \hspace{1cm} (7)
\[ C(K) := \sum_{k=1}^{K} c_k \] \hspace{1cm} (8)

\( C(K) \) is the iteration-cost function representing 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 [25].

The following lemma states that if \( G(K) \) is monotonically decreasing, we can find \( k^* \) where \( G(K) \) 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 [32] of \( G(K) \) changes for the first time, i.e.

\[ k^* \in \min \{ K | G(K + 1) - G(K) > 0 \} \] \hspace{1cm} (9)

**Proof:** See Appendix A-A in [33].

In the following section, we present three algorithms to solve optimization problem (6). First, we discuss the non-causal setting for characterizing the minimizer, then, introduce a causal setting to design algorithms that achieve practical minimizers for convex and non-convex loss functions. Finally, we establish the optimality and convergence of the algorithms.

#### B. Non-Causal Setting

An ideal approach to solve problem (6) is an exhaustive search over the discrete set of \( K \in [0, +\infty) \). However, this approach requires knowing in advance the sequences \( (f(w_k))_k \) and \( (c_k)_k \) for all \( k \in [0, +\infty) \), which is not practical as the sequence of parameters \( (w_k)_k \), and consequently \( (f(w_k))_k \), are not available in advance. For analytical purposes, our non-causal setting assumes that all these values are available at \( k = 0 \), enabling us to find the ultimate
Algorithm 1 Cost-Efficient Batch FedCau

1: Inputs: \( w_0, (x_{ij}, y_{ij})_{i,j}, \alpha_k, M, \{D_j\}_{j \in [M]} \), \( \rho_j \).
2: Initialize: \( k_c = +\infty \), \( G(0) = +\infty \).
3: Master node broadcasts \( w_0 \) to all nodes \( \triangleright \) Global iterations
4: for \( k \leq k_c \) do
5: for \( j \in [M] \) do
6: \( f_j^k := \sum_{i=1}^{D_j} f(w_k; x_{ij}, y_{ij})/|D_j| \)
7: Set \( w_{k+1}^j := w_k \)
8: for \( h \in [E] \) do
9: Compute \( w_{h,k+1}^j = w_{h-1,k+1} - \alpha_k \nabla f_j^k \)
10: end for
11: Set \( w_{k+1}^j = w_{k+1}^j \)
12: Send \( w_{k+1}^j \) and \( f_j^k \) to the master node \( \triangleright \) Local iterations
13: end for
14: Wait until master node collects all \( \{w_{k+1}^j\} \) and set \( w_{k+1} = \sum_{j=1}^{M} \rho_j w_{k+1}^j \).
15: Calculate \( f(w_k) := \sum_{j=1}^{M} \rho_j f_j^k \)
16: Calculate \( c_k \) and \( G(k) \)
17: if \( G(k) < G(k-1) \) then \( \triangleright \) Evaluating (9)
18: Master node broadcasts \( w_{k+1} \) to the workers
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 \( w_{k_c}, k_c, G(k_c) \).

minimizer \( k^* \). While this approach is not feasible in practice, we investigate it to establish a benchmark for the performance evaluation of subsequent causal solution algorithms (see Section V).

C. FedCau for Convex Loss Functions

Here, we propose an approximation of the optimal stopping iteration \( k^* \), referred to as \( k_c \). Our analysis demonstrates that \( k_c \) can be practically computed using a causal setting scenario. Under certain conditions, we establish that \( k_c \) corresponds to \( k^* \) or \( k^* + 1 \). Specifically, when \( k^* = K^{\text{max}} \), with \( K^{\text{max}} \) denoting the maximum allowable number of iterations, we have \( k_c = k^* \), otherwise, \( k_c = k^* + 1 \) (see Section III-E). Thus, we develop three implementation variations of FedAvg algorithm (3), Algorithms 1-3, with our causal termination approach, FedCau, for solving (6). Algorithms 1 and 2 are batch and mini-batch implementations using convex loss functions, while Algorithm 3 considers non-convex loss functions.

In the batch update of Algorithm 1, workers compute \( \{w_{k}^j, f_k^j\} \in [M] \) 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 \( w_{k}^j \) and the local loss function \( f_k^j \).

The master node updates \( w_k \) and \( f(w_k) \) upon receiving all local parameters \( \{w_{k}^j, f_k^j\}_{j \in [M]} \) from workers at each iteration \( k \) (see lines 14-15). Then, the iteration cost \( c_k \), representing the iteration cost, is calculated. To prevent termination in the first iteration, we initialize \( G(0) = +\infty \), and subsequently, the multi-objective cost function \( G(k) \) is updated (line 16). A comparison between \( G(k) \) and its previous value \( G(k-1) \) is made (see line 17) to determine the termination of iterations (see lines 19-24).

\footnote{We assumed that \( f_k^j \in \mathbb{R} \) and \( w_{k}^j \in \mathbb{R}^d \), then the communication overhead, in term of the number of bits, for transmission of \( f_k^j \) is negligible compared to the local FL model \( w_{k}^j \). Thus, we consider the local parameters to consist of both the local FL model and the local loss function value.}

In FedAvg, there are many scenarios where specific workers can upload their local parameters to the master node, resulting in implicit sub-sampling and approximations of \( f(w_k) \) denoted as \( f(w_k) \). This sub-sampling results in approximating the joint communication-computation and FL cost function, \( G(K) \). Algorithm 2 employs mini-batch updates to avoid excessive resource consumption for marginal test accuracy improvements. It leverages the descent property of FedAvg algorithm (3) for a monotonic decreasing loss function \( f(w) \), as described in Remark 1. Algorithm 2 aims at achieving non-increasing sequences of \( f(w_k) \) and \( G(k) \).

Algorithm 2 introduces partial worker participation and fairness in training FedCau. \( M_k^2 \) represents the node selection subset at each communication iteration \( k \), and Fair-count\([j]\) denotes the counter for the number of successfully-sent local parameters by worker \( j \in M_k^2 \). We introduce a “Fairness-Factor” \( F_f \leq K^{\text{max}} \) that restricts workers from transmitting more than \( F_f \) local parameters until all workers satisfy \( F_f \) local parameter transmission. At the first communication iteration \( k = 1 \), once a worker \( j \in M_k^2 \) successfully transmits its local parameter \( w_{k}^j \), it is removed from the selected node subset \( M_k^2 \) (see lines 7-21). Thus, worker \( j \) will not transmit any packets until all workers send their local parameters. The master node computes the resource used to perform the first communication iteration as \( T_n \). It considers \( T_n \) as a benchmark to determine \( T \leq T_n \) as the maximum allowable time slots for future iterations \( k = 2, \ldots, K \) (see line 25). Note that in \( k = 1 \), low-power workers have a higher probability of transmitting their local parameter, and the latency \( T \) is smaller compared to full worker participation. After completing communication iteration \( k = 1 \), partial worker participation begins at \( k \geq 2 \) when the master node updates \( M_k^2 \) (see line 26).

For \( k \geq 2 \), the selected workers \( j \in M_k^2 \) have a time budget \( T \) to compute and transmit their local parameters. This constraint creates competition among the selected workers to communicate with the master node. However, some workers may fail to send their local parameters. To address this challenge, we introduce the set \( M_k^2 \) which contains the indexes of the successful workers \( j_s \in M_k^2 \) that managed to transmit during iteration \( k \) (see line 31). Additionally, the fairness counter of each successful worker, Fair-count\([j_s]\), is increased (see line 32) to influence future selections for communication iterations. Afterward, the master node updates the global parameter by the local parameters it has received, \( w_{k}^j, j \in M_k^2 \), and then replaces the missing local parameters by the values of the previous iteration, for the local parameters \( w_{k}^j = w_{k-1}^j, j \notin M_k^2 \) and local functions \( f_k^j = f_{k-1}^j, j \notin M_k^2 \) (see lines 30-31). Algorithm 2 utilizes this replacement strategy to ensure the non-increasing behavior of \( G(k) \), \( k = 1, \ldots, k^* \), and maintain a descent sequence of \( f(w_k), k = 1, \ldots, k^* \). Since Algorithm 2 considers convex loss functions, the replacement of missing parameters guarantees the descent behavior of the sequence \( f(w_k), k = 1, \ldots, k^* \) (Lemma 2). Additionally, the master node updates the selected workers
Algorithm 2 Stochastic Cost-Efficient Mini-Batch FedCau.

1: Inputs: $w_0$, $(x_{ij}, y_{ij})_{i,j} \in [\mathcal{D}_j]$, $f_j$, $\alpha_k$, $\beta$, $\mathcal{M}$.
2: Initialize: $k_0 = +\infty$, $T_0 = +\infty$, $j_0 = 0$, $G(0) = +\infty$, $\mathcal{M}_0^k = \{\mathcal{M}\}$, and $\text{Fair-Count} = 1_{\mathcal{M} \ni k}$.
3: Master node broadcast $w_0$ to all nodes.
4: for $k \leq k_0$ do \hspace{1cm} \triangleright Global iterations
5:     $\mathcal{M}_k^k = \{\}$
6:     for $j \in [\mathcal{M}]$ do
7:         Calculate $f_j^k := \sum_{i=1}^{\mathcal{D}_j} f(x_{ij}, y_{ij}) / |\mathcal{D}_j|$.
8:         Set $w_j^k := w_k$.
9:     end for
10:     for $h \in [E]$ do \hspace{0.5cm} \triangleright local iterations
11:         Compute $w_{h,k}^{k+1} := w_{h-1,k}^{k+1} - \alpha_k \nabla f_h^k$.
12:     end for
13:     Set $w_{k+1}^k = w_{E,k+1}$.
14:     if $j \in \mathcal{M}_k^k$ then \hspace{0.5cm} \triangleright Partial participation
15:         Send $w_{k+1}^k$ to the master node.
16:     end if
17: end for
18: for $t_k \leq T_k$ do \hspace{0.5cm} \triangleright Assigning time budget $T$
19:     Every node $j \in \mathcal{M}_k^k$ sends $w_{k+1}^k$.
20:     if Successful node $j_j \in \mathcal{M}_k^k$ then
21:         $\mathcal{M}_k^k \leftarrow \mathcal{M}_k^k \cup \{j\}$
22:     end if
23: end for
24: Set $T_k = t_k$.
25: Compute $f_k := \sum_{j \in \mathcal{M}_k^k} f_j w_j$.
26: Set a time budget $T$.
27: if $t_k \geq 2$ then \hspace{0.5cm} \triangleright if $k \geq 2$
28:     $\text{Fair-Count}[j] = \text{Fair-Count}[j] + 1$
29:     $\text{Global Update} = \{\mathcal{M}\}$.
30: else
31:     $\text{Global Update} = \{\mathcal{M}\}$.
32: end if
33: for $t_k \leq T_k$ do \hspace{0.5cm} \triangleright Updating global parameters
34:     $\text{Master node set}$ \hspace{0.5cm} \triangleright Global update with replacements
35:     $w_{k+1}^k := \sum_{j \in \mathcal{M}_k^k} \rho_j w_j^k + \sum_{j \in \mathcal{M}_k^k} \rho_j^2 w_j^k$.
36: Master node calculate $F_k := \sum_{j \in \mathcal{M}_k^k} \beta_k^2 f_j w_j$.
37: Master node update $\mathcal{M}_k^k = \{j_{\text{Failure-Count}}[j] < F_k\}$ \hspace{0.5cm} \triangleright Fairness evaluation.
38: if $\mathcal{M}_k^k = \{\}$ then \hspace{0.5cm} \triangleright Update partial participation
39:     $\text{Fair-Count} = 0_{\mathcal{M} \ni k}$.
40: else
41:     $\text{Master node broadcast } w_{k+1}^k \text{ to the workers}$.
42:     $\text{Set } k \leftarrow k+1, \text{ and } w_{k+1}^k \leftarrow w_{k+1}^k$.
43:     Break and go to line 51 \hspace{0.5cm} \triangleright Terminating the training.
44: end if
45: end for
46: Return $w_{k+1}^k, k_c, G(k_c)$.

Based on the fairness factor, ensuring fair worker participation for the upcoming communication iterations (see lines 37-40). This process requires the master node to retain a memory of all previous local parameters. The remaining part of Algorithm 2 (lines 43-51) handles parameter updates and checks for the potential stopping iteration $k_c$, similar to lines 12-20 in Algorithm 1.

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

**Proof:** See Appendix A-B in [33].

As explained above, Algorithm 2 allows for both full and partial participation, offering fairness in worker participation based on the parameter $T$. The distinction between full and partial participation lies in the fact that in partial participation, the update of the global parameter $w_k$ depends on the new local parameters from the subset $\mathcal{M}_k^k$. However, it remains uncertain which workers within the subset successfully transmit their local parameters and which ones fail, particularly when workers possess non-aid training data. To address this challenge, we introduce the fairness-factor $F_f$ to mitigate the impact on the global update. The value of $F_f$ can be tailored to the specific training application, enabling customization of the partial participation scheme.

Another challenge in the partial participation of Algorithm 2 is determining the appropriate time budget $T$ for each iteration. Algorithm 2 suggests selecting a value for $T < T_k$ by causally computing $T_k$ in the first iteration, considering full worker participation and excluding background traffic. However, the choice of $T$ depends on the specific learning application, such as healthcare, autonomous driving, or video surveillance. One should consider a suitable time budget of $T$ based on the requirements of the learning application. For latency-sensitive scenarios like autonomous driving, where quick decisions are crucial to prevent accidents, a smaller value of $T$ is preferred.

### D. FedCau for Non-Convex Loss Functions

Here, we extend Algorithms 1 and 2 to include non-convex loss functions. We consider FedAvg [5], where each worker $j$ performs $E \geq 1$ local iterations over its local data subset, $\mathcal{X}_k \subset [\mathcal{D}_j]$. The master node updates the global parameter $w_{k+1}$ by averaging and broadcasting the local parameters to the workers. Additionally, the master node calculates $\bar{F}(w_k)$ by averaging the local loss functions of the workers [5].

We design a cost-efficient algorithm which optimizes $G(K)$, an estimate of the multi-objective cost function $G(K)$ defined as $G(K) := \beta C(K) + (1 - \beta) \bar{F}(w_k)$, where recall that $C(K)$ is the iteration-cost function at $K$. We design such an estimate since the stochastic nature of the sequences $(\bar{F}(w_k))_k$, arises from the local updates by $\xi_k \leq |\mathcal{D}_j|$ using mini-batches, results in a stochastic sequence of $(G(K))_k$. This sequence $(\bar{G}(k))_k$ 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 to establish non-increasing upper and lower bounds, $G_u(K)$ and $G_l(K)$, for the stochastic sequence $(\bar{G}(k))_k$. As this sequence is not necessarily non-increasing and may have multiple local optimum points, we aim at obtaining an interval $k_u \leq k_c \leq k_l$, where $k_u$ and $k_l$ represent the stopping 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

$$G_u(K) := \beta C(K) + (1 - \beta) F_u(w_K), \quad (10a)$$
$$G_l(K) := \beta C(K) + (1 - \beta) F_l(w_K), \quad (10b)$$
Algorithm 3 Stochastic Non-Convex Cost-Efficient Mini-Batch FedCau
1: Inputs: $w_0$, $(x_{ij}, y_{ij})_{i,j}$, $\alpha_k$, $M$, $\{|D_j|\}_{j \in [M]}$, $\rho_j$, $E$, $\xi_k$
2: Initialize: $k_0 = 0$, $k_{\text{max}} = 2$
3: Master node broadcasts $w_0$ to all nodes
4: for $k \geq 1$ do
5:    $M_a = \emptyset$
6:    Each node $j$ calculates:
7:    for $j \in [M]$ do
8:        Randomly select a subset of data with size $\xi_k$
9:        $F_j^k := \sum_{i=1}^{\xi_k} F(u_j; x_{ij}, y_{ij}) / \xi_k$
10:    end if
11:    Set $w_{j,k+1} = u_j$, $F_j^{k+1} = F_j^k$
12:    for $i \in [E]$ do
13:        Randomly select a subset of data with size $\xi_k$
14:        $w_{i,k+1} = u_i - \alpha_k \nabla w F_{i,-1,k+1}^{j,k}$
15:    end for
16:    Set $w_{j,k+1} = u_j$, and $F_{j,k+1} = F_j^{k+1}$
17:    Send $w_{j,k+1}$ and $F_{j,k+1}$ to the master node
18: end for
19: Master node calculates:
20:    $M_a \leftarrow M_a \cup \{\text{Successful workers}\}$
21:    $w_{k+1} \leftarrow \sum_{j \in [M], \rho_j} w_{j,k+1} / \sum_{j \in [M], \rho_j} \rho_j$
22:    $F_{w,k} := \sum_{i \in [M], \rho_i} F_{i,k} + \sum_{j \in [M], \rho_j} F_j^{k+1}$
23:    Update $C(k)$
24:    if $k < 2$ then
25:        Set $F_{w,k} = F_{1}(w_k) = \hat{F}(w_k)$
26:    else if $\hat{F}(w_k) \geq \hat{F}(w_{k-1})$ then
27:        Set $F_{w,k} = \hat{F}(w_k)$
28:    else if $\hat{F}(w_k) \geq F_{w,k-1}$ then
29:        $k_{\text{max}} = \max \{k_i | F_a(w_{k_i}) > \hat{F}(w_k)\}$
30:        Update $C(k)$
31:    end if
32: else
33:    Calculate $\delta_k^1 = F_{w,k} - F_{w,k-1}$
34:    Set $F_{w,k} = F_{w,k-1} + \delta_k^1$
35:    Update $(G_u(K))_{K=k_{\max}, \ldots, k}$ and $G_1(k)$
36: else
37:    if $\hat{F}(w_{k}) < \hat{F}(w_{k-1})$ then
38:        Set $F_{w,k} = \hat{F}(w_k)$
39:        Update $F_1(w_1) = \hat{F}(w_k)$, as (11)
40:    else if $\hat{F}(w_{k}) < F_{w,k-1}$ then
41:        Set $F_{w,k} = F_{w,k-1}$
42:    else if $G_1(K) = G_u(K)$ then
43:        Set $k_{\text{max}} = k$
44:    else
45:        Set $F_{w,k} = \hat{F}(w_k)$
46:        Update $\delta_k^1 = F_{w,k} - F_{w,k-1}$
47:        Set $F_{w,k} = F_{w,k-1} + \delta_k^1$
48:        Update $G_u(k)$ and $G_1(k)$
49:    end if
50: else
51:    Evaluate (9) for $G_1$ and $G_u$
52:    $k_i = \min \{K | G_1(K) > G_1(K-1)\}$
53:    $k_i = \min \{K | G_u(K) > G_u(K-1)\}$
54:    if $k_i \neq 0$, $k_i \neq 0$ then
55:        Break and go to line 60
56: else
57:    Master node broadcast $w_{k+1}$ to the workers
58:    Set $k = k + 1$
59: end if
60: end for
61: Return $k_{\text{stop}}$, $(w_{k})_{k=k_{\text{stop}}}$

where $F_u(w_{k})$ and $F_1(w_K)$ represent the estimation of the loss function at upper and lower bounds. To obtain the sequences of $(G_u(k))_k$ and $(G_1(k))_k$, the master node computes the upper and lower bounds for $\hat{F}(w_k)$ while ensuring the monotonic decreasing behavior of $(F_u(w_{k}))_k$ and $(F_1(w_k))_k$ to satisfy Remark 1. In the following, we now concentrate on the process of obtaining the bounds for $\hat{F}(w_k)$.

Algorithm 3 shows the steps required for the cost-efficient FedAvg with causal setting and non-convex loss function $\hat{F}(w)$. Lines 3-18 summarize the local and global iterations of FedAvg. Here, we introduce $M_a$ as the set of workers which successfully transmit their local parameters to the master node (see line 20). We initialize $F_u(w_{k}) = F_1(w_k) = \hat{F}(w_k)$, $k \leq 2$ for the first two iterations (see line 25). For iterations $k \geq 3$, if the new value of loss function fulfills $\hat{F}(w_k) \geq \hat{F}(w_{k-1})$, the algorithm updates $F_u(w_{k}) = \hat{F}(w_k)$ (see line 28). Then, the algorithm checks if $\hat{F}(w_k)$, which is now equal to $F_u(w_k)$, is greater than the previous value of $F_u(w_{k-1})$ (see line 29). This checking is important because we must develop a monotonic decreasing sequence of $F_u(w_j)_{j=1:k-2}$. When $\hat{F}(w_k) \geq \hat{F}(w_{k-1})$, the master node returns to the history of $F_u(w_j)_{j=1:k-1}$ and checks for $i < k$, when the condition $F_u(w_i) > F_u(w_k)$ is satisfied. Since at each communication iteration $k$ we carefully check the monotonic behavior of $F_u(w_k)$, we are sure that if we find the proper maximum communication iteration $i$ that fulfills $i < k$, for which $F_u(w_{i+1}) < F_u(w_k) < F_u(w_i)$, we have the result of $F_u(w_j) < F_u(w_j), j < i$. Let us define this communication iteration $i$ as $k_{\text{stop}}$ (see line 30). Thus, it is enough to find such $i$ to update the sequence of $F_u(w_j)_{j=1:k-2}$.

Now, we need to update the sequences of $F_u(w_j)_{j=1:k}$, $i_2 = k_{\text{stop}}$, $k$ to obtain the monotonic decreasing upper bound. We choose the monotonic linear function because it satisfies the sufficient decrease condition (see [29], Section 11.5). Therefore, we satisfy the decreasing behavior for $F_u(w_k)$ and the upper bound behavior, which means that the maximum values of $\hat{F}(w_k)$ are always lower than $F_u(w_k)$. Thus, we update the sequences of $F_u(w_i), i = k_{\text{max}}, \ldots, k$ according to (11), with $k_1 = k_{\text{max}}, k_{2} = k$, and $F_1(k_i) = F_{\text{linear}}(k_i), k_i \in [k_1, k_2]$. We define $F_{\text{linear}}(k)$ as the linear approximation of $\hat{F}(w_k)$ in an interval $k \in [k_1, k_2]$

$$F_{\text{linear}}(k_i) = ak_i + b, k_1 \leq k_i \leq k_2,$$

(11)

where

$$a = \frac{\hat{F}(w_{k_2}) - \hat{F}(w_{k_1})}{k_2 - k_1},$$

(12a)

$$b = \hat{F}(w_{k_1}) - ak_1.$$  

(12b)

Next, we need to update $F_1(w_k)$. Here, let us define the difference between two consecutive values of $F_u(w_k)$ and $F_u(w_{k-1})$ as $\delta_k^u$, and the difference between $\hat{F}(w_k)$ and $F_1(w_{k-1})$ as $\delta_k^1$. Then, we update the corresponding values for $(G_u(K))_{K=k_{\text{max}}, \ldots, k}$ and $G_1(k)$, respectively (see lines 34-35). Afterward, we need to check the condition at which $\hat{F}(w_k) < F_1(w_{k_{\text{max}}})$, where $k_{\text{max}}$ represents the last communication iteration at which the value of $\hat{F}(w_{k_{\text{max}}})$ has been considered as $F_1(w_{k_{\text{max}}})$. If $\hat{F}(w_k) < \hat{F}(w_{k_{\text{max}}})$, we need to update the lower bound sequences (see
lines 37-38) to avoid over-decreasing the lower bound function $F_l(w_k)$ by the approximation of line 34. Subsequently, we need to calculate $\delta^k_l = F_l(w_k) - F_l(w_{k-1})$, then update $F_u(w_k) = F_u(w_{k-1}) + \delta^k_l$ and the value of $k_{\text{max}}^l = k$ (see lines 41-43).

The last condition to check is when $F_l(w_{k_{\text{max}}}) < \tilde{F}(w_k) < \tilde{F}(w_{k-1})$. In this condition, the monotonic decreasing behavior of $\tilde{F}(w_k)$ is satisfied, whereas the decreasing behavior is not satisfied for the lower bound $F_l(w_k)$. Thus we set $F_u(w_k) = \tilde{F}(w_k)$, and $\delta^k_l = F_u(w_k) - F_u(w_{k-1})$, $F_l(w_k) = F_l(w_{k-1}) + \delta^k_l$, and update $G_u(k)$ and $G_l(k)$ (see lines 45-48).

Finally, lines 52-61 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 monotonic behavior of $G(K)$, $K \leq k^*$. In practice, we have this desired monotonically decreasing behavior, as we show in the following proposition:

**Proposition 1:** Consider $G(K)$ defined in Eq. (7). Define $\Delta_k := l_k - f_k$, $\Delta_0 = f_0$, by choosing $\beta$ as

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

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

**Proof:** See Appendix A-C in [33].

**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 $\beta$ 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 Appendix A-D in [33].

Proposition 2 implies that when $G(K)$ is monotonically decreasing with $K$, $k^*$ is equal to $K^{\text{max}}$. According to the training setup, the maximum number of iterations is set as $K^{\text{max}}$ at the beginning of the training. Thus, monotonically decreasing $G(K)$ results in $k^* = K^{\text{max}}$, which means that the value of the FL loss function is dominant in $G(K)$, and the FedCau procedure is similar to the FedAvg method.

The following theorem clarifies an important relation between the non-causal and causal solutions of Algorithms 1 and 2.

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

\[ k_c \in \{k^*, k^* + 1\}, \quad f(w_{k_c}) \leq f(w_{k^*}), \quad \text{and} \]

\[ G(k_c) \geq G(k^*). \]

**Proof:** See Appendix A-E in [33].

**Remark 4:** Note that $k^*$ and $k_c$ are fundamentally different. $k_c$ is obtained from Algorithms 1 or 2, while $k^*$ is the optimal stopping 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 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 causal setting solve problem (6) by taking at most one extra 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 $F_u(w_k)$ and $F_l(w_k)$ as the upper and lower bound functions for $\tilde{F}(w_k)$, respectively, such that for every $k \geq 1$, inequalities $F_l(w_k) \leq F_l(w_k) \leq F_u(w_k)$ hold. The following proposition highlights the important monotonic behavior of $F_u(w_k)$ and $F_l(w_k)$.

**Remark 5:** The proposed functions $F_u(w_k)$ and $F_l(w_k)$ are monotonically decreasing w.r.t. $k$, i.e., $F_u(w_k) < F_u(w_{k-1})$, and $F_l(w_k) < F_l(w_{k-1})$ for all $k \geq 1$. These results hold because we consider a linear function, which is monotonically decreasing, w.r.t. $k$, for updating each value of $F_l(w_k)$ and $F_u(w_k)$ for $k \geq 1$. Since the monotonically decreasing linear function fulfills the sufficient decreasing condition (see [29], Section 11.5), we claim that $F_u(w_k)$ and $F_l(w_k)$ are monotonically 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 $k^u_c$ and $k^l_c$, 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 $k^u_c$ and $k^l_c$, 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 $G(K)$ with $k^u_c$ and $k^l_c$.

**Proposition 3:** Let $k_c$, $k^u_c$, and $k^l_c$ be the causal stopping iterations for minimizing $G(K)$, $G_u(K)$, and $G_l(K)$, respectively. Then, the inequalities $k^u_c \leq k_c \leq k^l_c$ hold.

**Proof:** See Appendix A-F in [33].

Proposition 3 characterizes an interval in which $k_c$ can take values to stop Algorithm 3. As $k^u_c \leq k_c \leq k^l_c$, it is enough that we find $k^u_c$ and terminate the algorithm. However, the maximum allowable number of iterations is $k^l_c$, 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 $F_u(w_k)$ and $F_l(w_k)$ be respectively the upper bound and the lower bound of $\tilde{F}(w_k)$ obtained from the stochastic non-convex cost-efficient mini-batch FedCau Algorithm 3. Let us define $\tilde{F}_{\text{max}} := \max_{k \in [2, K]} \tilde{F}(w_k)$ and $\tilde{F}_{\text{min}} := \min_{k \in [2, K]} \tilde{F}(w_k)$. Assuming that $|\tilde{F}(w_k)| < \infty$, $k = 1, \ldots, K$, then $|F_u(w_k) - F_l(w_k)| \leq \tilde{F}_{\text{max}} - \tilde{F}_{\text{min}}$ is the tightness between the upper bound $F_u(w_k)$ and the lower bound $F_l(w_k)$.

**Proof:** See Appendix A-A.
Lemma 3 specifies that the maximum distance between the upper and lower bound functions $F_u(w_k)$ and $F_l(w_k)$, $k = 1, \ldots, K$, is determined by the variations of non-convex sequence $\tilde{F}(w_k), k = 1, \ldots, K$. In the following Proposition, we investigate the tightness of the interval $(k^u_c, k^l_c)$.

**Proposition 4:** Let $K^{\max}$, $k^u_c$ and $k^l_c$ be the maximum number of iterations, the causal stopping iteration for minimizing $G_u(K)$, and the causal stopping iteration for minimizing $G_l(K)$, respectively. Recall the definition of $k^u_{\max}$ in line 29 of Algorithm 3. Then,

$$
k^l_c = 1 + \max \left\{ k^u_c, k^u_{\max} + \frac{1 - \beta}{\beta c_{k_{d}}} \left\{ F_u(w_{k_{max}}^u) - \tilde{F}(w_{k_{d}}) \right\} \right\},$$

where, for $k \in [k^u_c + 1, K^{\max}]$,

$$k_{d} := \text{the first value of } k \mid \tilde{F}(w_{k}) > F_u(w_{k-1}).$$

Proof: See Appendix A-B. □

Proposition 4 denotes that the tightness of the interval $(k^u_c, k^l_c)$ is mainly determined by $c_k$ and the variations of the non-convex sequence $\tilde{F}(w_k), k = k^u_c + 1, \ldots, K^{\max}$.

To summarize, FedCau is applicable for both full and partial worker participation, as well as when $f(w_k)$ is monotonically 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 $f(w_k)$ is not monotonically decreasing.

**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 assuming the training is done considering a neural network with $N_l$ number of layers, $d_{N_l}$ is the maximum number of neurons, the backpropagation of local gradients in each worker $j$ after $E$ local iterations, results in a complexity of $O(E N_l d_{N_l}^2)$. Thus, Algorithm 1 has the complexity of $O(k_c E(|D|d + N_l d_{N_l}^2))$, which is less than or equal to the complexity of FedAvg as $O(K^{\max} E(|D|d + N_l d_{N_l}^2))$. Similarly, the complexity of Algorithm 2 is obtained as $O(|D|Ed + (k_c - 1)Ed^{-1}|D|Ed + Ek_c N_l d_{N_l}^2)$. Finally, the complexity of Algorithm 3, by considering the complexity from the neural network setting we mentioned before, is obtained as $O(|D|Ed(k_c^3 + Ed_{N_l}^2))$.

**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 [34] and CSMA/CA [26] with a binary exponential backoff transmission policy [35], and OFDMA [27] 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 the line of each node’s queue and ready to be transmitted. Thus, upon receiving $w_k$, each worker $j \in [M]$ computes its local parameter $w_j^k$ and puts it in the head of the 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 a 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). The critical point to consider is that we should choose a stable network in which packet saturation will not happen. We only consider the latency of transmitting local parameters, positioned at the head of line queues, which is influenced by the number of workers $M$, transmission probability $p_s$, and packet arrival probability $p_r$ at each time slot. Local parameters at each iteration $k$ are distinct from background traffic packets influenced by the 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 $b/R$. Also, it is natural to assume that $\ell_{4,k}$ is a given constant for updating parameters at the master node [36]. The computation latency $\ell_{2,k}$ in each iteration $k$ at each worker $j \in [M]$ is calculated as $\ell_{2,k}^j = a_k^j |D_j^k|/\nu k_j^2$, where $a_k^j$ is the number of processing cycles to execute one sample of data (cycles/sample), $|D_j^k| \leq |D_j|$ is a subset of local dataset each worker chooses to update its local parameter $w_j^k$, and $\nu k_j^2$ is the central processing unit (cycles/s) [37]. Without any loss of generality, we consider that $|D_j^k| = |D_j|, k = 1, \ldots, K$. We assume that all the worker nodes start transmitting their local parameters simultaneously. Thus, the network must wait for the slowest worker to complete its transmission. Therefore, $\ell_{2,k} = \max_{j \in [M]} \ell_{2,k}^j$. The third term, $\ell_{3,k}$, is determined by the channel capacity, resource allocation policy, and network traffic. We characterize this term for two batch and mini-batch update cases with a defined time budget. Further, every specific broadcast channel imposes a particular $R$ and $b$, which do not change during the 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 the network follow MAC protocols, such as slotted-ALOHA and CSAMA/CA, to transmit their local parameters of FedAvg algorithm (3) to the master node [35], [38]. 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 puts its local parameter at the head of the line in its queue and makes it 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 $w_k$. Accordingly, we 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 latency $\ell_{3,k}$.

**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

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

where

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

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

and

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

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

**Proof:** See Appendix A-G in [33].

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

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

which helps us to design the communication-computation parameters for FedCau. Note that we consider a setup where the transmission starts simultaneously 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 (18). The assumption that all workers transmit 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 $f(w)$ 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 $s = 1, \ldots, S_c$ orthogonal subchannels [39]. Let $h_s^l$, $p_s^l$ 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 $\text{SNR}(p_s^l, h_s^l) := p_s^l h_s^l / \sigma_s^l$. The corresponding data rate (bps/Hz) is as $R_s = \sum_{s=1}^{S_c} \log_2 (1 + \text{SNR}(p_s^l, h_s^l))$. The master node randomly decides at each iteration $k$ which worker should use which subchannel link, and the remaining workers will not participate in the parameter uploading.

## V. Numerical Results

In this section, we illustrate our results from the previous sections. We numerically show the extensive impact of the iteration costs when running the FedAvg algorithm (3) training problem over a wireless network. We use a network with $M$ workers 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 the well-known MNIST dataset with non-iid distribution among workers and the CIFAR-10 dataset with both iid and non-iid cases. For the non-iid implementation, we first sort the dataset w.r.t. the label numbers of $y_i$, where $i \in \{0, 1, \ldots, 9\}$, where $i$ is the index of each data sample with size $|D_i|$. Moreover, in the MNIST dataset, the labels are the same as the digits, while in CIFAR-10, the labels demonstrate airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Afterward, we assign an equal portion of data to each worker $j \in \{1, \ldots, M\}$, starting from the beginning of the sorted dataset. According to the size of each dataset, CIFAR-10 with 50000 and MNIST with 60000 data samples, the data portion of every class in the datasets assigned to each worker is different. Finally, 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 $M$ workers, each having $\{(x_{ij}, y_{ij})\}$, where $x_{ij} \in \mathbb{R}^{784}$ is a data sample $i$, and
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 $f(w_k)$ b) Iteration-cost function $C(K)$ c) Multi-objective cost function $G(K)$, and d) Test accuracy.

Fig. 2. Illustration of the effect of $\beta$ 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 $C(k)$ decreases while $\beta$ increases. b) The causal stopping iteration $k_c$ is smaller for larger $\beta$. c) Test accuracy also decreases when $\beta$ increases.

a vectorized image at node $j \in [M]$ with corresponding digit label $y_{ij} \in \{-1,+1\}$. We use the loss function [40]

$$f(w) = \frac{1}{M} \sum_{j=1}^{M} \frac{|D_j|}{|D|} \log \left(1 + e^{-w^T x_{ij} y_{ij}} \right),$$

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

Second, we consider a non-convex image classification problem with the workers using convolutional neural networks (CNNs) with a cross-entropy loss function. The architecture of the CNN consists of a convolutional layer, Conv2D$(32, 3, 3)$, a MaxPooling2D layer with a pool size of $(2, 2)$, a Flatten layer, two Dense (fully connected) layers with size 64 and 10, and a final layer that produces probability distributions over 10 classes of the CIFAR-10 dataset. Overall, the CNN has 462410 parameters.

We implement the network with $M$ workers performing local updates of $w_k^j, \forall j \in [M]$ and imposing computation latency of $\ell_{2,k}$ to the system. We assume a synchronous network in which all workers start the local iteration of $w_k^j$ simultaneously right after receiving $w_{k-1}$. Note that the latency counting of $c_k$ at each iteration $k$ starts from the beginning of the local iterations until the uplink process is complete. Regarding the computation latency, we consider $\nu_k \in [10^6, 3 \times 10^9]$ cycles/s, and $\omega_k = [10, 30]$ 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 $\mu$s, 50 $\mu$s, and 10 $\mu$s respectively [41] and run the network for 1000 times. In the OFDMA setup, 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 $h^c_k = \zeta/r^3$, following the Rayleigh fading, where $\zeta$ 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 $f(w_k)$ 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 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 $f(w_k)$ and the latency over the communication protocols.
Fig. 2 characterizes the effect of $\beta$ on the performance of batch FedCau update of Algorithm 1 with $M = 50, 100$ and CSMA/CA protocol for parameter upload. Fig. 2(a) shows that $C(k_c)$ decreases while $\beta$ takes the values between $(0, 1)$. This decreasing behavior is a valid result since the higher values of $\beta$ 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 $C(K)$ result in stopping at the smaller causal iterations, called $k_c$. Finally, Fig. 2(c) demonstrates the test accuracy we achieve while changing $\beta$. Since $k_c$ decreases as $\beta$ increases, the corresponding test accuracy decreases. Therefore, choosing $\beta \in [0.2, 0.5]$ gives us a lower causal iteration cost and sub-optimal test accuracy.

Fig. 3 represents the mini-batch FedCau update of Algorithm 2 and the FedAvg baseline 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, $C(k_c)$ decreases, while Fig. 3(b) shows the similarity in the test accuracy. Fig. 3(c)-(3(e) compare the test accuracy of FedCau in Algorithm 2 with the FedAvg by assigning the time budget $T = 0.5, 0.9, 1.1$s respectively. For the time budget $T = 0.5, 0.9, 1.1$s, the test accuracy of FedAvg is lower than the results of mini-batch FedCau update of Algorithm 2 with a similar time budget $T$. These results highlight the role of $F_f$ combined with $T$, where $F_f$ ensures participation fairness, especially for the smaller $T$, such as $T = 0.5$. Therefore, with the equal $T$, the FedCau in Algorithm 2 outperforms FedAvg in test accuracy and fairness in worker participation. Figs. 3(f)-(3(g) reveal the behavior of the mini-batch FedCau causal latency and test accuracy for $M = 50$, and $T = 1.1$s w.r.t. $F_f$. Fig. 3(a) demonstrates that the causal latency increases for small and large fairness factors $F_f$. Meanwhile, Fig. 3(g) shows that the test accuracy decreases while $F_f$ increases due to the lack of participation fairness. For smaller $F_f$, the participation fairness results in better test accuracy, while a higher causal latency arises from more frequent transmission of low-power workers.

C. Impact of Communication Parameters on FedCau Performance

Fig. 4 characterizes the iteration-cost function $C(K)$ for the same setup as in Fig. 1. The iteration-cost function for slotted-ALOHA is larger than CSMA/CA, as we see in Figs. 4(a) and 4(c). On the other hand, the iteration-cost function for CSMA/CA increases exponentially when the probability $p_r$ increases, as shown in Fig. 4(b). This result also holds for the bounds of the iteration cost in Eq. (17), as Fig. 4 shows. Furthermore, the results from Fig. 4(c) show that $C(k_c)$ increases on a slower rate than $M$ increases, such that

$$\frac{C(k_{c_2}) - C(k_{c_1})}{M_2 - M_1} < 1, \quad M_2 > M_1,$$  

(20)
where $M_2$ and $M_1$ are number of workers, and $C(k_{c_2})$ and $C(k_{c_1})$ are the total communication-computation with the stopping causal iterations $k_{c_2}$ and $k_{c_1}$, respectively. Thus, considering full worker participation as the worst case when investigating the scalability, we conclude that the total communication-computation cost of FedCau is scalable in $M$.

**D. Performance of Non-Convex FedCau From Algorithm 3**

The experimental results presented in Fig. 5 investigate the impact of the number of local iterations ($E$) on the performance of mini-batch FedCau updates of Algorithm 3. The study focuses on the CIFAR-10 iid dataset and CNN architecture, employing CSMA/CA with different values of $M = 50$ and $M = 100$, along with $p_x = 0.8$ and $p_r = 0.01$. Fig. 5(a) reveals distinct behaviors in the causal test accuracy concerning $E$ for $M = 50$ and $M = 100$. While the changes in test accuracy are less pronounced for $M = 50$, the corresponding values are lower than $M = 100$. Fig. 5(b) shows the causal stopping iterations ($k_{c_2}$ and $k_{c_1}$), which decreases as $E$ increases. Additionally, the tightness of the interval $(k_{c_1}, k_{c_2})$ established in Proposition 4 is validated, according to the variations in the non-convex sequences of $\tilde{F}(w_k)$. Moreover, Fig. 5(c) shows the causal iteration-cost $C(K)$ as a function of $E$, which increases as $E$ increases. This observation highlights the significant impact of computation latency on the performance of the FedCau. Based on the findings in Fig. 5, selecting $E = 10$ as the optimal number of local iterations is recommended, which provides the best accuracy with a lower causal iteration cost compared to $E > 10$. These results offer valuable insights into selecting $E$ and understanding the trade-off between $E$, test accuracy, iteration cost, and causal stopping iterations.

Fig. 6 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. 6(a) compares the test accuracy of training the mini-batch FedCau update of Algorithm 3 by iid and non-iid data obtained by $k_{c_1}^l$, the highest test accuracy achieved by Algorithm 3 for any non-convex loss function. We observe that for the iid case, with $k_{c_1}^l = 160$, the test accuracy is higher than the case for non-iid with $k_{c_1}^l = 93$. Figs. 6(b) and 6(c) show the loss functions $\tilde{F}(w_k)$ and the corresponding upper and lower bounds $F_u(w_k)$ and $F_l(w_k)$ of iid and non-iid data, and Fig. 6(d) compares the test accuracy of FedCau and FedAvg with stopping iterations $k_{c_1}^l$ and $K_{\text{max}} = 200$ after 100 realizations to have smoother curves. Notably, FedAvg with $K_{\text{max}} = 200$ increases the total iteration cost by 55% (non-iid) and 20% (iid), but the test accuracy improvement over FedCau is only 2.2% (non-iid) and 0.65% (iid). We observe that non-iid FedCau terminates at iteration $k_{c_1}^l = 93$ while the FedAvg test accuracy curve becomes flat at iteration $k = 101$. Moreover, the test accuracy of non-iid FedCau with $k_{c_1}^l = 93$ is 1.2% higher than non-iid FedAvg at iteration $k = 101$. The communication costs of the local parameters for every extra iteration are high; thus, stopping the training at a proper iteration saves a huge amount of communication resources (14.7 Mbits per iteration per worker). As a result, FedCau, with the knowledge of when to terminate the training, i.e., $k_{c_1}^l = 93$, is significantly superior to FedAvg in terms of saving communication-computation resources and achieving higher test accuracy.

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

We choose LAQ because it achieves the same linear convergence as the gradient descent while effecting major savings in the communication resources [8]. Among all the compression
the FedCau stopping iteration of contains 32. We change the percentage of the dimension of each local LAQ and Top-
in practice. When terminating FedAvg at iteration 55, these stopping iterations for FedAvg cannot be set beforehand considering FedAvg at stopping iteration 60, FedAvg achieves a test accuracy of 94.46% with an iteration cost of 4.8 Mbits. Compared to FedCau at $k_0 = 57$, FedAvg incurs a 5.27% increase in iteration cost while gaining only a marginal 0.26% improvement in test accuracy. This comparison highlights that beyond $k_0$, the increase in iteration cost becomes significantly higher compared to the increase in test accuracy.

We conclude that FedCau obtains the optimal stopping iteration regarding the iteration cost and the achievable test accuracy, even when applying it on top of existing communication-efficient methods, such as LAQ and top-

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 algorithms 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. We optimized the iteration-termination criteria to minimize the trade-off between FL’s achievable objective value and the overall training cost. To this end, we proposed a causal setting, FedCau, utilized in two convex scenarios for batch and mini-batch updates, and 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 the 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. In the end, we applied the FedCau method on top of the existing methods like top-

methods, we choose top-

| Method          | Stop iteration | Total cost (MBits) | Test accuracy (%) |
|-----------------|----------------|--------------------|-------------------|
| FedCau LAQ, $b = 2$ | 57             | 4.56               | 91.2              |
| FedCau LAQ, $b = 10$ | 43             | 16.92              | 87.8              |
| FedCau Top-

APPENDIX A

A. Proof of Lemma 3

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

### B. 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 \( F_I(w_k) \) according to the behavior of \( \tilde{F}(w_k) \) for \( k \geq k_c^u + 1 \). There are three different scenarios, which are explained in the following:

- \( \tilde{F}(w_k) < \tilde{F}(w_{k-1}) \) and \( \tilde{F}(w_k) > F_I(w_{k-1}) \): According to Algorithm 3 (see lines 44-47), for \( k = k_c^u + 1 \), we have \( F_u(w_k) = \tilde{F}(w_k) \) and \( F_I(w_k) = F_I(w_{k-1}) + \delta_k^u \) where \( \delta_k^u = F_u(w_{k-1}) - F_u(w_{k-1}) = \tilde{F}(w_k) - F_u(w_{k-1}) \). Moreover, since \( k \geq k_c^u + 1 \), the inequality of \( G_u(k) > G_u(k) \) gives us

\[
G_u(k) - G_u(k^u) = (\beta C(k) + (1 - \beta) F_u(w_k)) - (\beta C(k - 1) + (1 - \beta) F_u(w_{k-1})) = \beta c_k + (1 - \beta) F_u (w_{k-1}) - F_u (w_k) > 0. \tag{21}
\]

Then, we compute \( G_I(k) \) as in Eq (10b):

\[
G_I(k) = \beta C(k) + (1 - \beta) F_I(w_k) = \beta C(k - 1) + \beta c_k + (1 - \beta) (F_I(w_{k-1}) + \delta_k^u) = \beta c_k + \beta C(k - 1) + (1 - \beta) (F_I(w_{k-1}) - F_u(w_{k-1}) + \tilde{F}(w_k)) = \beta c_k + G_I(k - 1) - (1 - \beta) (F_u(w_{k-1}) - \tilde{F}(w_k)) \geq G_I(k - 1). \tag{22}
\]

Therefore, \( k_c^l = k_c^u + 1 \).

- \( \tilde{F}(w_k) < \tilde{F}(w_{k-1}) \) and \( \tilde{F}(w_k) < F_I(w_{k-1}) \): According to Algorithm 3 (see lines 36-44), for \( k = k_c^u + 1 \), we have \( F_I(w_k) = \tilde{F}(w_k) \) and \( F_u(w_k) = F_u(w_{k-1}) + \delta_k^l \), where \( \delta_k^l = F_I(w_k) - F_I(w_{k-1}) = \tilde{F}(w_k) - F_I(w_{k-1}) \). Thus, we calculate \( G_I(k) \) as

\[
G_I(k) = \beta C(k) + (1 - \beta) F_I(w_k) = \beta C(k - 1) + \beta c_k + (1 - \beta) \tilde{F}(w_k) = \beta C(k - 1) + \beta c_k + (1 - \beta) \left( F_u(w_k) + F_I(w_{k-1}) - F_u(w_{k-1}) \right) = (1 - \beta) \left( F_u(w_{k-1}) + F_u(w_k) \right) + G_I(k - 1) + \beta c_k \geq G_I(k - 1), \tag{23}
\]

which results in \( k_c^l = k_c^u + 1 \).

- \( \tilde{F}(w_k) > F_u(w_{k-1}) \) (see lines 28-34 in Algorithm 3): In this case, the update of \( F_u(w_k) \) and \( F_I(w_k) \) are according to the linear update we proposed in Section III-D in the revised manuscript. Thus, the update of \( \delta_k^u \) is as

\[
\delta_k^u = \frac{\tilde{F}(w_k) - F_u(w_{1max})}{k - k_{1max}}, \quad k \geq k_c^u + 1, \tag{24}
\]

and \( F_I(w_k) = F_I(w_{k-1}) + \delta_k^u \), and \( F_u(w_k) = \tilde{F}(w_k) \). Next, we calculate \( G_I(k) \) as

\[
G_I(k) = \beta C(k) + (1 - \beta) F_I(w_k) = \beta C(k - 1) + \beta c_k + (1 - \beta) (F_I(w_{k-1}) + \delta_k^u) = G_I(k - 1) + \beta c_k + (1 - \beta) \delta_k^u, \tag{25}
\]

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

\[
-\delta_k^u = \frac{F_u(w_{k_{1max}}) - \tilde{F}(w_k)}{k - k_{1max}} < \frac{\beta}{1 - \beta} c_k, \quad k \geq k_c^u + 1, \tag{26}
\]

where \( k_c^l \) is

\[
k_c^l = k_c^u + \left\{ \frac{(1 - \beta) (F_u(w_{k_{1max}}) - \tilde{F}(w_k))}{\beta c_k} \right\}, \tag{27}
\]

where \( k_c^l := \text{the first value of } k \mid \tilde{F}(w_k) > F_u(w_{k-1}) \). \tag{29}

## References

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

[2] O. Simeone, “A very brief introduction to machine learning with applications to communication systems,” IEEE Trans. Cogn. Comput. Netw., vol. 4, no. 4, pp. 648–664, Dec. 2018.

[3] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, “Wireless network intelligence at the edge,” Proc. IEE, vol. 107, no. 11, pp. 2204–2239, Nov. 2019.

[4] P. Kairouz et al., “Advances and open problems in federated learning,” Found. Trends Mach. Learn., vol. 14, nos. 1–2, pp. 1–210, Oct. 2021.

[5] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, “Federated learning: Strategies for improving communication efficiency,” 2016, arXiv:1610.05492.

[6] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, “Deep learning’s diminishing returns: The cost of improvement is becoming unsustainable,” IEEE Spectr., vol. 58, no. 10, pp. 50–55, Oct. 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 Trans. Neural Netw. Learn. Syst., vol. 31, no. 9, pp. 3400–3413, Sep. 2020.

[8] J. Sun, T. Chen, G. B. Giannakis, Q. Yang, and Z. Yang, “Lazily aggregated quantized gradient innovation for communication-efficient federated learning,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 4, pp. 2031–2044, Apr. 2022.
[9] S. Caldas, J. Konečný, H. B. McMahan, and A. Talwalkar, “Expanding the reach of federated learning by reducing client resource requirements,” 2018, arXiv:1812.07210.

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

[11] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, “Energy efficient federated learning over wireless communication networks,” IEEE Trans. Wireless Commun., vol. 20, no. 3, pp. 1935–1949, Mar. 2021.

[12] S. U Stich, J.-B. Cordonnier, and M. Jaggi, “Sparsified SGD with memory,” in Proc. Adv. Neural Inf. Process. Syst., 2018, pp. 4447–4458.

[13] S. Di, D. Tao, X. Liang, and F. Cappello, “Efficient lossy compression for scientific data based on pointwise relative error bound,” IEEE Trans. Parallel Distrib. Syst., vol. 30, no. 2, pp. 331–345, Feb. 2019.

[14] K. Yuan, Q. Ling, and Z. Tian, “Communication-efficient decentralized event monitoring in wireless sensor networks,” IEEE Trans. Parallel Distrib. Syst., vol. 26, no. 8, pp. 2198–2207, Aug. 2015.

[15] J. Wangni, J. Wang, J. Liu, and T. Zhang, “Gradient sparsification for communication-efficient distributed optimization,” in Proc. Int. Conf. Adv. Neural Inf. Process. Syst., 2018, pp. 1299–1309.

[16] T. Chen, G. Giannakis, T. Sun, and W. Yin, “LAG: Lazily aggregated gradient for communication-efficient distributed learning,” in Proc. Adv. Neural Inf. Process. Syst., 2018, pp. 5050–5060.

[17] J. Sun et al., “Communication-efficient distributed learning via lazily aggregated quantized gradients,” in Adv. Neural Inf. Process. Syst., 2019, pp. 3370–3380.

[18] K. Hsieh et al., “GAIA: Geo-distributed machine learning approaching Convex Optimization,” Cambridge, U.K.: Cambridge Univ. Press, 2004.

[19] D. Bertsekas et al., Convex Optimization, 2nd ed. Philadelphia, PA, USA: SIAM, 2009.

[20] T. D. Hoang and L. Bao Le, “Joint prioritized scheduling and resource allocation for OFDMA-based wireless networks,” IEEE Trans. Wireless Commun., vol. 17, no. 1, pp. 310–323, Jan. 2018.

[21] V.-D. Nguyen, S. K. Sharma, T. X. Xu, S. Chatzinotas, and B. Ottersten, “Efficient federated learning algorithm for resource allocation in wireless IoT networks,” IEEE Internet Things J., vol. 8, no. 5, pp. 3394–3409, May 2021.

[22] F. S. Samani, H. Zhang, and R. Stadler, “Efficient learning on high-dimensional operational data,” in Proc. 15th Int. Conf. Netw. Service Manage. (CNSM), Oct. 2019, pp. 1–9.

[23] E. Ziouva and T. Antonakopoulos, “CSMA/CA performance under high contention diversity slotted ALOHA (CRDSA): An enhanced random access scheme for satellite access packet networks,” IEEE Trans. Wireless Commun., vol. 6, no. 4, pp. 1408–1419, Apr. 2007.

[24] K. Koh et al., “An interior-point method for large-scale f1-regularized logistic regression,” J. Mach. Learn. Res., vol. 8, no. Jul, pp. 1519–1555, 2007.

[25] IEEE Standard for Information Technology—Telecommunications—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, Standard IEEE Std 802.11-2016, 2016, pp. 1–3534.

Afsaneh Mahmoudi (Graduate Student Member, IEEE) received the bachelor’s degree from the Sharif University of Technology, Tehran, Iran, in 2015, and the master’s degree in electrical engineering from the University of Tehran, Tehran, in 2018. She is currently pursuing the Ph.D. degree with the Department of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden. Her research area focuses on federated learning, cell-free massive MIMO, machine learning, and optimization.

Hossein S. Ghadikolaei received the B.Sc. degree in electrical engineering from the Iran University of Science and Technology in 2009, the M.Sc. degree in electrical engineering from the Sharif University of Technology, Tehran, Iran, in 2011, and the Ph.D. degree in electrical engineering and computer science from the KTH Royal Institute of Technology, Stockholm, Sweden, in 2018. He is currently with Ericsson Research, Kista, Sweden. His research interests include distributed optimization and machine learning, with applications in communication networks. He was a recipient of the IEEE Communications Society Stephen O. Rice Prize, in 2018, the Premium Award for the Best Paper in IET Communications, in 2014, and the Program of Excellence Award from KTH, in 2013, among others. He was selected as an Exemplary Reviewer for the IEEE TRANSACTIONS ON COMMUNICATIONS, in 2017 and 2018, and served as an Associate Editor of IEEE COMMUNICATIONS LETTERS.
José Mairton Barros Da Silva Jr. (Member, IEEE) received the B.Sc. (Hons.) and M.Sc. degrees in telecommunications engineering from the Federal University of Ceará, Brazil, in 2012 and 2014, respectively, and the Ph.D. degree from the KTH Royal Institute of Technology, Stockholm, Sweden, in 2019. He is currently an Assistant Professor with the Division of Computer Systems, Uppsala University, Sweden. He was a Postdoctoral Researcher at the KTH Royal Institute of Technology, between 2019-2021. He was a Marie Skłodowska-Curie Postdoctoral Fellow with Princeton University, USA, and KTH Royal Institute of Technology, between 2022-2023. He is currently a Workshop, Tutorials, and Symposia Officer of the IEEE Communications Society Emerging Technology Initiative on Machine Learning for Communication. He has been involved in the organization of many conferences and workshops, including co-chairing IEEE ICMLCN 2024, IEEE SECON 2022-2023. He gave several tutorials at many IEEE flagship conferences, including ICASSP, PIMRC, ICC, and GLOBECOM. His research interests include distributed machine learning and optimization over wireless communications.

Carlo Fischione (Senior Member, IEEE) received the Laurea degree (summa cum laude) in electronic engineering and the Ph.D. degree in electrical and information engineering from the University of L’Aquila, Italy, in May 2001 and 2005, respectively. He has held various research positions, such as a Visiting Professor with the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2015; an Associate Professor with Harvard University, Cambridge, in 2015; and a Visiting Scholar with the University of California at Berkeley, CA, USA, from 2004 to 2005, where he was a Research Associate from 2007 to 2008. He is currently an Honorary Professor with the Department of Mathematics, Information Engineering, and Computer Science, University of L’Aquila. He is also a Full Professor with the Department of Electrical Engineering and Computer Science, Division of Network and Systems Engineering, KTH Royal Institute of Technology, Stockholm, Sweden. He is also the Director of the KTH-Ericsson Data Science Micro Degree Program directed to Ericsson globally. His current research interests include applied optimization, wireless, sensor networks, the Internet of Things, and machine learning. He is an ordinary member of the Italian Academy of History Deputazione Abruzzese di Storia Patria (DASP). He received several awards, such as the IEEE Communication Society S. O. Rice Award for the 2018 Best IEEE TRANSACTIONS ON COMMUNICATIONS Paper and the Best Paper Award of IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS in 2007. He is also a Distinguished Lecturer of the IEEE Communication Society, the Funding Chair of the IEEE International Conference on Machine Learning for Communication and Networking (IEEE ICMLCN 2024), and the Chair of the IEEE Machine Learning for Communications Emerging Technologies Initiative. He is an Editor of IEEE TRANSACTIONS ON COMMUNICATIONS (Machine Learning for Communications Area) and IEEE TRANSACTIONS ON MACHINE LEARNING FOR COMMUNICATION AND NETWORKING and served as an Associated Editor for Automatica (IFAC) from 2014 to 2019.