Clustered Scheduling and Communication Pipelining for Efficient Resource Management of Wireless Federated Learning

Chihat Keçeci, Mohammad Shaqfeh, Member, IEEE, Fawaz Al-Qahtani, Member, IEEE, Muhammad Ismail, Senior Member, IEEE, and Erchin Serpedin, Fellow, IEEE

Abstract—This article proposes using communication pipelining to enhance the convergence speed of federated learning in mobile edge computing applications. Due to limited wireless sub-channels, a subset of the total clients is scheduled in each iteration of federated learning algorithms. On the other hand, the scheduled clients wait for the slowest client to finish its computation. We propose to first cluster the clients based on the time they need per iteration to compute the local gradients of the federated learning model. Then, we schedule a mixture of clients from all clusters to send their local updates in a pipelined manner. In this way, instead of just waiting for the slower clients to finish their computations, more clients can participate in each iteration. While the time duration of a single iteration does not change, the proposed method can significantly reduce the number of required iterations to achieve a target accuracy. We provide a generic formulation for optimal client clustering under different settings, and we analytically derive an efficient algorithm for obtaining the optimal solution. We also provide numerical results to demonstrate the gains of the proposed method for different data sets and deep learning architectures.

Index Terms—Clustered scheduling, communication pipelining, federated learning, mobile edge computing.

I. INTRODUCTION

FEDERATED learning is a promising distributed machine learning approach in which a central server (coordinator) and a number of data-holding clients cooperate in the training of a common machine learning model without exchanging data [1]. Only model parameters are shared and updated using an iterative gradient descent algorithm. Hence, learning from private clients’ data becomes possible while preserving the privacy of the clients’. Furthermore, federated learning enables achieving mutual benefits for all participating clients since training the algorithm using larger data sets by aggregating the private data of all participants leads to significant enhancements in the performance of data-hungry models. Federated learning can be applied for mobile edge computing in wireless networks [2]. The increase in the amount of client data generated by end devices, such as mobile phones and Internet of Things (IoT) sensors introduce new opportunities and challenges for wireless federated learning. In a wireless federated learning scenario, the systems have to handle the fast data processing for delay-sensitive systems under bandwidth and latency constraints [3]. Federated learning algorithms could be employed in mobile edge computing nodes for low latency and enhanced privacy for artificial intelligence (AI) applications for edge devices, such as drones and autonomous vehicles. Moreover, federated learning enables better utilization of distributed computational and storage resources.

Research in federated learning presents many aspects. One of them is exploring the variations among distributed gradient-descent training algorithms and analyzing the convergence performance of these algorithms with respect to the number of local and global iterations [4]. Another attractive research area is managing and optimizing the distributed computational and communication resources [5]. In particular, one of the main challenges in federated learning is the heterogeneity of the clients in terms of their computing and data resources. In this work, we focus on speeding up the convergence rate of a machine learning model and enhancing the prediction accuracy in a wireless federated learning scenario. Hence, achieving a target prediction accuracy becomes possible with fewer iterations.

We propose using communication pipelining to speed up the convergence of federated learning. The primary motivation behind the proposed concept is that the computation time varies a lot between devices due to several reasons, such as different processor speeds or different sizes of the local data set [6]. So, in an iterative gradient algorithm, the time of each iteration will be limited by the slowest client. To overcome this bottleneck, we propose that instead of just waiting for the slow clients to finish the computation of their local gradients, we can schedule an extra number of clients within the same iteration without requiring extra communication resources. A mixture of fast and slow clients should be scheduled in each iteration. The fast clients will use the wireless channels first, while the slow clients are still running their computation of

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
the local update. In this way, the number of clients who participate in each iteration can be increased while preserving the total duration of each iteration. So, the main idea is to cluster clients based on their computation time and then schedule a balanced mixture of clients from different clusters. The merit of this proposed concept is not necessarily in increasing the number of scheduled clients in each iteration but rather in allocating the wireless resources more efficiently. So, for example, fewer resources can be used while maintaining the number of clients per iteration. In principle, pipelining is an added degree of freedom in the resource allocation problem that can be adequately utilized to adjust the tradeoff between the allocated resources and the number of clients to be scheduled to achieve the best outcome of the federated learning process.

It is essential to highlight that the proposed concept of pipelined federated learning (PFL) is a general concept that can be applied to any variation of the gradient descent algorithm, whether batch or stochastic, and for both primal or dual optimization methods. In this article, we just show, as an example, the formulation for a primal method with client sampling in each iteration and one local iteration of full-gradient (batch) computation in each global iteration. We refer to this method as clustered-scheduling gradient-descent (CS-GD). The extension of the concept to other cases, such as having more local iterations per client, is straightforward as long as the needed computation time per client is quantified.

A. Related Work

The application of federated learning is discussed in several studies, such as [1], [7], [8], and [9]. A seminal work on federated learning introducing the federated averaging algorithm is proposed in [1]. The federated averaging algorithm aggregates the updated local weights from each client in each global iteration. A comprehensive survey on federated learning is provided in [7] and [8] by discussing the arising challenges, problems, and possible solutions regarding federated learning. An overview of the distributed learning algorithms proposed for the next-generation wireless communication networks is provided in [10]. A fog-assisted federated learning framework is proposed in [11] by introducing fog nodes responsible for local aggregation. The idea of federated learning is also applied to traffic prediction in [9].

Wadu et al. [12] investigated the problem of client scheduling and resource block (RB) allocation to enhance the performance of model training using FL. Tran et al. [13] adopted FL in wireless networks as a resource allocation optimization problem that captures the tradeoff between FL convergence wall clock time and energy consumption of UEs with heterogeneous computing and power resources. Yang et al. [14] proposed a novel over-the-air computation based-approach for fast global model aggregation by exploring the superposition property of a wireless multiple-access channel. This is achieved by joint device selection and beamforming design using the difference of convex function representation concept. In the same direction, Zhu et al. [15] proposed a novel digital version of broadband over-the-air aggregation, called one-bit broadband digital aggregation (OBDA), featuring one-bit gradient quantization followed by digital quadrature amplitude modulation (QAM) at edge devices and over-the-air majority-voting-based decoding at edge server.

Many works in the literature have considered the client scheduling problem, e.g., [16], [17], [18], [19], [20], and [21]. In [16], three practical scheduling policies: random scheduling, round Robin scheduling, and proportional fair scheduling were compared. In [17], a probabilistic scheduling setting that considers both channel quality and local update importance was proposed. In [18], a multiarmed bandit-based formulation was presented for online client scheduling. In [19], scheduling based on a metric called “age of information” was recommended, and in [20], a joint client scheduling and resource allocation scheme was advocated. The training and wireless transmission parameters were jointly optimized in [22]. The computational heterogeneity of the clients was considered in [21]. In this work, we develop the concept further by introducing client clustering and communication pipelining. We propose the novel concept of pipelined network access for sharing parameter updates in a federated learning setup. By taking into account the variations in the computational time among the clients, we schedule the clients efficiently to avoid wasting the extra time required by clients that need more time for computational tasks.

B. Contributions

This article brings three major contributions.

1) We propose the novel federated learning concept of PFL in which the communication of the model parameter updates is performed in a pipelined manner. The client updates in each round are pipelined by assigning the clients into clusters based on their computation time. The proposed pipelining scheme provides efficient utilization of the communication resources as well as balanced incorporation of all clients.

2) We investigate the problem of optimal client clustering in a generic setting that can be applied to many other scenarios. We show how to formulate the problem, and then we present rigorous analytical steps to derive an optimal and efficient algorithm to solve the problem. The solution assumes the change of variables step, Lagrangian dual-problem formulation, manipulations of the Karush–Kuhn–Tucker (KKT) necessary and sufficient conditions for optimality, hypothesis testing to find the active inequality constraints, and finally, an elegant geometric interpretation that helps in obtaining an efficient algorithm to solve the optimization problem.

3) We demonstrate the advantages of the proposed PFL via extensive numerical examples for different federated learning tasks. We assess the gain introduced by the proposed pipelining scheme relative to the conventional federated learning scheme using the MNIST [23], the federated EMNIST [24], and the federated Shakespeare [1], [25] data sets by employing a variety of neural network architectures, such as multilayer...
TABLE I
SUMMARY OF SYMBOLS AND NOTATIONS

| Notation | Description |
|----------|-------------|
| $M$      | Total number of clients |
| $n$      | Total number of data samples |
| $n_m$    | Number of data samples for client $m$ |
| $\tau_{\text{min}}$ | Min. data samples per client |
| $\tau_{\text{max}}$ | Max. data samples per client |
| $F(w)$  | Global loss function |
| $F_m(w)$ | Loss function for client $m$ |
| $l(\cdot)$ | Loss function for a data sample |
| $w$      | Model weights |
| $x^i_m$  | Input values for the model |
| $y^i_m$  | Output values for the model |
| $T$      | Number of global iterations |
| $S[t]$   | Active subset of clients at iteration $t$ |
| $\tilde{w}[t]$ | Model weights at iteration $t$ |
| $g_m[t]$ | Gradient for client $m$ |
| $\eta_t$ | Learning rate |
| $N$      | Number of sub-channels |
| $\tau_m$ | Computation time for client $m$ |
| $\tau_{\text{com}}$ | Communication time |
| $\tau_{\text{server}}$ | Server time |
| $\tau_{\text{global}}[t]$ | Duration of a global iteration |
| $\tau_{\text{total}}$ | Total duration of $T$ iterations |
| $\epsilon$ | Utilization efficiency |
| $K$      | Number of clusters |
| $C_k$    | $k$th cluster |
| $\theta_k$ | Computation time for cluster $k$ |
| $\sigma$ | Computation time per sample |
| $\Delta_{\text{global}}$ | Extra overall communication time |
| $S_m$    | Ordered index of client $m$ |
| $T[S_m]$ | Function that maps $S_m$ to $\tau_m$ |
| $\omega_k$ | Index of max. $\tau_m$ in cluster $k$ |
| $\tau_k$ | Threshold for cluster $k$ |
| $M_k$    | Number of clients in cluster $k$ |
| $\delta_k$ | Relaxed parameter $M_k$ |
| $|\cdot|$ | Cardinality of a set |
| $\lfloor \cdot \rceil$ | Floor function |
| $\lceil \cdot \rceil$ | Nearest integer |
| $\mathcal{L}$ | Lagrangian |
| $\lambda_{k, \nu}$ | Lagrangian dual variables |

perceptron (MLP), convolutional neural network (CNN), and recurrent neural network (RNN). The simulation results suggest that employing PFL decreases the number of communication rounds required to train a deep neural network.

The remainder of this article is organized as follows. The novel PFL scheme is introduced in Section II. A generic client clustering algorithm is proposed and detailed in Section III. Extensive simulations are performed to validate the proposed method in Section IV. Finally, the conclusions are drawn in Section V. A summary of symbols and notations is provided in Table I.

II. PIPELINED FEDERATED LEARNING

A typical federated learning problem is considered with a total of $M$ clients participating in the optimization algorithm without sharing their own data sets. Each client has $n_m$ data samples, where $m$ is the index of the client. The total number of data samples is $n = \sum_{m=1}^{M} n_m$. The goal is to minimize a loss function defined as follows:

$$\min_{w \in \mathbb{R}^d} F(w) \equiv \sum_{m=1}^{M} p_m F_m(w)$$

(1)

where $p_m = (n_m/n)$, and

$$F_m(w) = \frac{1}{n_m} \sum_{i=1}^{n_m} l(w, x^i_m, y^i_m)$$

(2)

where $l(w, x^i_m, y^i_m)$ is the loss function for the model and $w, x^i_m, y^i_m$ are the model weights, the input values, and the output values, respectively.

The model training mechanism is based on a number $T$ of iterations. In each iteration $t$, the server selects a subset of the total clients $S[t]$ and broadcasts an updated model $w[t]$ to the selected clients. The selected clients calculate the local gradient using their data samples and send the gradient to the server. A local gradient computation, at a given global round $t$, is based on averaging the gradient of all data points in the data set of the client $g_m[t] = \nabla F_m(w[t]) = \frac{1}{n_m} \sum_{i=1}^{n_m} \nabla l(w[t], x^i_m, y^i_m)$.

(3)

Then, the server averages the gradients

$$g[t] = \frac{\sum_{m \in S[t]} n_m g_m[t]}{\sum_{m \in S[t]} n_m}$$

(4)

and updates the model according to

$$w[t+1] = w[t] - \eta_t g[t]$$

(5)

where $\eta_t > 0$ is the learning rate in the $t$th iteration.

The uplink communication channel is considered to be an OFDM channel with a total of $N$ subchannels. So, at a given time, no more than $N$ users can communicate with the server. Each subchannel consists of multiple subcarriers, and the data packets may cover multiple OFDM frames in order to fit the exchanged gradient data between the clients and the server. The number of gradients may be in the order of a million since deep learning architectures usually have a large number of trainable variables. Furthermore, the total number of clients is assumed to be much larger than the number of subchannels $N \ll M$. Therefore, the server should randomly select a subset of the clients at each iteration based on the available communication resources (i.e., the number of subchannels).

The computation time to obtain the local gradients varies among the clients for several reasons, such as different processor speeds and different computation loads. Note that the computation load to calculate (3) depends on the total number of local data samples $n_m$. So, even if two clients have the same processor speed, a larger data set will need a larger computation time than a client with a smaller data set. The
variation in computation time presents practical relevance. On the other hand, since the packet size for communicating a local update to the server is the same across all clients (and it depends on the size of the gradient vector $g$), the communication time is considered to be the same for all clients. We know that wireless channels’ quality varies among different users due to small-scale and large-scale fading. Thus, some studies have taken channel quality into consideration in the selection of the clients per iteration [16], [17]. However, in this work, we do not want to compromise machine learning best practices for other advantages related to opportunistic communication in order to enhance the transmission data rate. Hence, we design a system that can give all clients an almost equal chance to be selected. In particular, we aim to use the largest possible amount of data in training by involving all clients and learning a model that is not biased toward the data of only a subset of the clients who get selected more frequently than others. In our communication scheme, we assume that the communication over the subchannels applies a fixed rate that is independent of the particular selected user. We assume that the fixed transmission rate can be reliably decoded by all users. This is a common scheme in practical communication systems. The wireless channel data rate efficiency could be enhanced by applying adaptive modulation and coding based on the channel conditions. However, this is out of the scope of this article. We denote the computation time for client $m$ as $\tau_m$, the communication time of every client as $\tau_{com}$, and the combined time for the server to update the global model and broadcast it to the clients alongside information about the scheduled clients for next round as $\tau_{server}$. In order for the server to update the global model and broadcast the new model $w[t]$ to the clients, it must wait for all local updates. So, the bottleneck will be to wait for the slowest client until it finishes its local computation. Therefore, the duration of one global iteration $t$ is expressed as follows:

$$\tau_{global}[t] = \tau_{server} + \max_{m \in S[t]} \tau_m + \tau_{com}$$  \hspace{1cm} (6)

and the total time to complete the whole training process over $T$ iterations is given by

$$\tau_{total} = \sum_{t=1}^{T} \tau_{global}[t] = T(\tau_{server} + \tau_{com}) + \sum_{t=1}^{T} \max_{m \in S[t]} \tau_m.$$  \hspace{1cm} (7)

One might think of excluding the clients with large values of $\tau_m$ to speed up the training process. However, this is not a good option since those clients are likely to be the clients with the largest data sets, and hence, their local updates have higher weights (i.e., they present higher importance). Furthermore, it is advantageous to involve more data samples in the training process in order to enhance overall learning accuracy. Thus, client exclusion is not recommended. Instead, client scheduling in each global iteration should be based on random sampling so that each client gets a chance to be involved in each global iteration. On the other hand, just waiting for the slowest client to finish its computation is a waste of valuable communication resources since the utilization efficiency $\epsilon$ of the $N$ uplink subchannels will be

$$\epsilon = \frac{T \tau_{com}}{\tau_{total}} \approx \frac{\tau_{com}}{\tau_{com} + \tau_{server} + \max_m \tau_m}.$$  \hspace{1cm} (8)

In this work, we propose to schedule a mixture of slow and fast clients in each iteration and make them use the available channels in a pipelined manner. So, the clients are grouped into a number of $K$ clusters based on their computation speed. More details about how to cluster the clients will be given in the next section. In each iteration of the federated learning algorithm, the server randomly samples $N$ clients from each cluster. The sampled clients from each cluster transmit their parameters in different communication intervals, i.e., a federated learning iteration spans multiple communication rounds. In this way, the total number of clients per federated learning iteration will be $KN$. The proposed PFL is illustrated in Fig. 1.

The utilization efficiency of the wireless subchannels based on our proposal becomes

$$\epsilon_{PFL} = \frac{T K \tau_{com}}{\tau_{total}} \approx \frac{K \tau_{com}}{\tau_{com} + \tau_{server} + \max_m \tau_m} \approx K \epsilon.$$  \hspace{1cm} (9)

Thus, PFL improves the performance of the learning algorithm since adding more clients in each iteration will make it converge faster to the globally optimal solution of (1). Although PFL does not decrease the time per iteration (6), it helps to achieve a targeted accuracy level for solving (1) with less number of iterations $T$. Thus, it does indeed speed up the convergence performance of the learning algorithm. This will be demonstrated in the numerical simulation section.

### III. Client Clustering

Clients are grouped into $K$ clusters, denoted as $C_1, C_2, \ldots, C_K$. Each client is placed in just one cluster. Thus, $\sum_{k=1}^{K} |C_k| = M$. The computation time for a given cluster $k$ is denoted as $\theta_k$. This represents the time duration given to the selected clients in the cluster to complete their gradient computations. Then, they synchronously get scheduled to access the communication channels at time $\theta_k$, for a communication duration $\tau_{com}$, as illustrated in Fig. 1.

Clustering the clients has some flexibility, and it can be done in several ways. However, there are two strict constraints to be taken into consideration and two favorable, but not strict, requirements. First, all clients in a given cluster must satisfy the constraint that

$$\tau_m \leq \theta_k \hspace{1cm} \forall m \in C_k, \hspace{0.5cm} k = 1, \ldots, K.$$  \hspace{1cm} (10)

Furthermore, we have a constraint for scheduling the clusters into consequent communication frames. Assuming that the clusters are ordered in an ascending way based on their $\theta_k$, the constraint is expressed as follows:

$$\theta_k - \theta_{k-1} \geq \tau_{com}, \hspace{0.5cm} k = 2, \ldots, K.$$  \hspace{1cm} (11)

A favorable, but not strict, requirement is to make the sizes of the clusters (i.e., number of clients) almost equal in order to give an equal chance for all clients to participate in the training process. Hence, we have

$$|C_k| \approx \frac{M}{K} \hspace{0.5cm} \forall k = 1, \ldots, K.$$  \hspace{1cm} (12)
Another favorable requirement is to satisfy that
\[ |C_k| \geq N, \quad \forall k = 1, \ldots, K \]  \hspace{1cm} (13)
in order to fully utilize the wireless resources to the maximum possible efficiency.

The CS-GD algorithm is summarized in Algorithm 1. This is a generic algorithm that can be applied for any arbitrary client clustering method given that (10) and (11) are satisfied. Next, we elaborate on the optimization of clustering based on (12) while maintaining the other two constraints in (10) and (11).

Algorithm 1 CS-GD

**Input:** The number of sub-channels \( N \); the clients’ clusters \( C_1, C_2, \ldots, C_K \); the initial model parameters \( \theta[0] \); the number of global iterations \( T \); the learning rate per iteration \( \eta_i \) for \( t = 1, 2, \ldots, T \); the number of data samples per client \( n_m, \forall m \).

**Output:** The final model \( \theta[T] \)

1: for global round \( t = 1 \) to \( T \) do
2: for cluster \( k = 1 \) to \( K \) do
3: Uniformly sample \( C_k \) to obtain a set of \( N \) clients, denoted by \( S_k[t] \)
4: \( S[t] = S[t] \cup S_k[t] \)
5: for each client \( m \) in \( S_k[t] \) do
6: Compute the gradient \( \theta_m[t] \) according to (3)
7: Send gradient \( \theta_m[t] \) to the server according to the pipeline timing \( \theta_k \) of its cluster
8: end for
9: end for
10: The server aggregates the gradients according to (4), and updates the model according to (5)
11: end for

A. Generic Optimization Problem Formulation

First, we sort the clients in ascending order of their computation time \( \tau_m \) and assign the integer index \( S_m \) for the order of client \( m \). Hence, if \( S_j > S_j \), then \( \tau_i \geq \tau_j \forall S_j, S_j \in \{1, 2, \ldots, M\} \). Define the discrete function \( T \) for the ordered computation times of the clients as follows:

\[ T(S_m) = \tau_m, \quad S_m = 1, 2, \ldots, M. \]  \hspace{1cm} (14)

By definition, \( T(S_m) \) is monotonically nondecreasing. Here, we would like to provide a generic optimization framework for client clustering that is applicable for any arbitrary monotonically nondecreasing \( T \). Extensions to special cases are straightforward. An example of special cases is the case when the computation time is uniformly distributed among the clients between \( \tau_{\min} \) and \( \tau_{\max} \), which results in

\[ T[i] = \tau_{\min} + \frac{\tau_{\max} - \tau_{\min}}{M - 1} (i - 1), \quad i \in \{1, 2, \ldots, M\}. \]  \hspace{1cm} (15)

Next, we use the notation \( \omega_k \) to denote the index \( S_m \) of the client that has the largest computation time among all clients in cluster \( C_k \). Hence, we have

\[ \omega_k = \max_{m \in C_k} S_m, \quad S_m \in \{1, 2, \ldots, M\}. \]  \hspace{1cm} (16)

Based on the introduced notations for sorting the clients by their computation time, clustering the clients will be based on the following rule:

\[ C_k = \{m : \omega_{k-1} < S_m \leq \omega_k \}, \quad \forall k \in \{1, 2, \ldots, K\} \]  \hspace{1cm} (17)

where \( \omega_M = M \) and \( \omega_0 = 0 \). Based on (17), it is straightforward to show that \( |C_k| = \omega_k - \omega_{k-1} \). Now, we can formulate the optimization problem to reduce the variance among the number of clients per cluster since it is favorable to have
clustering with almost equal sizes, according to (12)
\[
\begin{align*}
\min_{\omega_1, \ldots, \omega_{K-1}} & \quad \sum_{k=1}^{K} \left( \omega_k - \omega_{k-1} - \frac{M}{K} \right)^2 \\
\text{subject to } & \quad T[\omega_k] \leq \theta_k \quad \forall k \in \{1, 2, \ldots, K - 1\} \\
& \quad \omega_K = M 
\end{align*}
\] (18a)

where the constraint (18b) is based on (10). We do not need to include \(\omega_K\) as an optimization variable, but rather we added it as a constraint in (18c) to enforce the optimal solution to include all clients in the clustering method and not exclude any client.

Before we discuss the solution of (18), we will comment first on the selection of constrained thresholds \(\theta_k\) in (18b). First, we introduce an extra hyper-parameter in the problem formulation to allow for some extra overall time that can be added at each global iteration. The added extra time is denoted by \(\Delta_{\text{global}}\), where \(\Delta_{\text{global}} \geq 0\). The purpose of this added hyper-parameter is to enable more flexibility in adjusting the number of clients per cluster and consequently to enable reducing the variations among the clusters’ sizes. Furthermore, the extra time \(\Delta_{\text{global}}\) can also be useful in enabling a larger number of clusters \(K\) if needed. However, there is a tradeoff since adding more time per iteration will reduce the utilization efficiency to
\[
\epsilon = \frac{K \tau_{\text{com}}}{\tau_{\text{com}} + \tau_{\text{server}} + \max \tau_m + \Delta_{\text{global}}}. 
\] (19)

We consider \(\Delta_{\text{global}}\) and the total number of clusters \(K\) to be hyper-parameters that are set in advance before solving (18). The selection of \(\Delta_{\text{global}}\) will affect the values of the thresholds \(\theta_k\) in the optimization problem constraint (18b). We can start by examining \(\theta_k\) of the last cluster, i.e., the one with the largest computation time, which can be adjusted as \(\theta_K = \tau_{\text{max}} + \Delta_{\text{global}}\), and then we apply (11) with equality to adjust \(\theta_k\) for clusters \(K-1, K-2, \ldots, 1\) sequentially. As a result, we will have
\[
\theta_k = \tau_{\text{max}} + \Delta_{\text{global}} - (K-k) \tau_{\text{com}}, \quad k = K, \ldots, 1. 
\] (20)

It should be noted that using the procedure mentioned above to adjust the values of the thresholds \(\theta_k\) in (20) aims at setting these thresholds at the maximum possible values in order to enlarge the space of feasible solutions of the main optimization problem (18), and hence, to enable achieving a lower value of the objective function (18a) at the optimal solution of the problem.

The upper bound of the possible number of clusters \(K\) can be obtained by setting \(\theta_1 \geq \tau_{\text{min}}\), which can be simplified as \(\tau_{\text{max}} + \Delta_{\text{global}} - (K-1) \tau_{\text{com}} \geq \tau_{\text{min}}\). This inequality is equivalent to
\[
K \leq \frac{\tau_{\text{max}} - \tau_{\text{min}} + \Delta_{\text{global}}}{\tau_{\text{com}}} + 1. 
\] (21)

Furthermore, since the number of clusters is an integer, we can write
\[
K \leq \left\lfloor \frac{\tau_{\text{max}} - \tau_{\text{min}} + \Delta_{\text{global}}}{\tau_{\text{com}}} \right\rfloor 
\] (22)

where \(\lfloor x \rfloor\) is the floor (rounding to the highest lower integer) of \(x\).

The above upper bound of \(K\) is theoretically correct. However, it is not plausible in practice since it is based on assuming \(\theta_1 \geq \tau_{\text{min}}\) is satisfied at strict equality. This means that the first cluster will have just a single client that has the lowest computation time among all clients. This violates (12). Therefore, we alternatively prefer, from practical perspectives, to adjust \(K\) by reducing the upper bound by one, which yields

\[
K = \left\lfloor \frac{\tau_{\text{max}} - \tau_{\text{min}} + \Delta_{\text{global}}}{\tau_{\text{com}}} \right\rfloor - 1. 
\] (22)

B. Optimal Solution

It is not straightforward to solve the primal optimization problem in (18) analytically since the objective function will have nonlinear terms after expanding the square in (18a). Therefore, we propose a change of variable step to transform the problem formulation into a convex optimization problem. We consider the number of clients per cluster \(|C_k|\), denoted as \(M_k\), where \(k = 1, 2, \ldots, K\), as the optimization variables instead of \(\omega_k\). Thus, \(M_k = \omega_k - \omega_{k-1}\). Since the optimization variables \(M_k\) are integers, we relax the problem into the continuous domain by replacing \(M_k\) by \(\delta_k\), which is a real number. The resulting reformulated optimization problem is

\[
\begin{align*}
\min_{\delta_1, \ldots, \delta_K} & \quad \frac{1}{K} \sum_{k=1}^{K} \left( \delta_k - \frac{M}{K} \right)^2 \\
\text{subject to } & \quad \sum_{i=1}^{k} \delta_i \leq \pi_k \quad \forall k \in \{1, 2, \ldots, K - 1\} \\
& \quad \sum_{i=1}^{K} \delta_i = M 
\end{align*}
\] (23a)

where the thresholds \(\pi_k\) satisfy \(\pi_k = \{|m : \tau_m \leq \theta_k\}\}, which can be equivalently expressed as follows:
\[
\pi_k = \max Sm_{\delta_1} \text{ subject to } T[S_m] \leq \theta_k. 
\] (24)

By solving (23) for \(M_k\), \(k = 1, \ldots, K\), we can then easily obtain \(\omega_k = \sum_{i=1}^{k} M_i\), for \(k = 1, \ldots, K\). Then, we can apply clustering using (17). Problem (23) is a convex optimization problem, and it can be solved analytically. After obtaining \(\delta_k\), \(k = 1, \ldots, K\), we can obtain \(M_k\) for \(k = 1, \ldots, K\) by rounding to the nearest integer. We can also directly obtain \(\omega_k\) \((k = 1, \ldots, K - 1)\) of the primal optimization problem (18) by using
\[
\omega_k = \left\lfloor \sum_{i=1}^{k} \delta_i \right\rfloor, \quad k = 1, 2, \ldots, K - 1 
\]
that the optimal value in the discrete domain should be by rounding the relaxed number of clients to the nearest integer. Any other value will give a higher variance. Furthermore, rounding to the lower or upper integer will give an equal value. For example, if we have seven users and want to distribute 4 and the second group could have 3, or the first group could have 3 and the second group could have 4. Both solutions present the same variance. In principle, rounding the nearest integer will always be optimal, although it may not be the unique optimal solution. The Lagrangian dual problem is given by

$$\min \ L(\delta_1, \ldots, \delta_K, \lambda_1, \ldots, \lambda_{K-1}, \nu)$$
subject to
$$\lambda_i \geq 0 \ \forall k \in \{1, 2, \ldots, K-1\}$$

where the Lagrangian $L$ in (25a) is expressed as follows:

$$L = \sum_{k=1}^{K} \left( \delta_k \frac{M}{K} \right)^2 + \sum_{k=1}^{K-1} \lambda_k \left( \sum_{i=1}^{k} \delta_i - \pi_k \right) - \nu \left( \sum_{i=1}^{K} \delta_i - M \right)$$

(26)

where $\lambda_k$ $(k = 1, \ldots, K-1)$ are the dual variables associated with the inequality constraints of the primal problem (23b), and $\nu$ is the dual variable associated with the equality constraint of the primal problem (23c).

Based on the KKT necessary and sufficient conditions for optimality [26], the optimal solution of our problem must satisfy the following constraints.

1) The primal equality constraint (23c).
2) The primal inequality constraints in (23b), which are $K-1$ inequalities in total.
3) The dual constraints in (25b), which are also $K-1$ inequalities.
4) The complementary slackness, given by

$$\lambda_k \left( \sum_{i=1}^{k} \delta_i - \pi_k \right) = 0 \ \forall k \in \{1, 2, \ldots, K-1\}$$

(27)

which are $K-1$ equations related to the previous two conditions.

5) Vanishing of the gradient of the Lagrangian with respect to the primal optimization variables

$$\frac{\partial L}{\partial \delta_k} = 0 \ \forall k \in \{1, 2, \ldots, K\}$$

(28)

which yields $K$ equations in total.

Furthermore, since the problem is convex, the solution that satisfies all KKT conditions is unique. To solve this system of equations and inequalities, we will first perform some analytical manipulations based on these conditions. Then, we will develop a hypothesis testing approach to get to the optimal solution in an efficient and quick way. Based on the KKT conditions, we can characterize the optimal solution as follows.

**Lemma 1 (Primal Variables and Lagrangian Multipliers):**

The optimal solution must satisfy

$$\delta_k = \delta_{k-1} + \frac{1}{2} \lambda_{k-1} \ \forall k = 2, \ldots, K.$$  

(29)

**Proof:** From (28) we obtain

$$\frac{\partial L}{\partial \delta_k} = 2 \left( \delta_k - \frac{M}{K} \right) + \sum_{i=k}^{K-1} \lambda_i - \nu = 0$$

which can be rearranged to obtain

$$\delta_k = \frac{M}{K} - \frac{1}{2} \sum_{i=k}^{K-1} \lambda_i + \frac{1}{2} \nu, \ k = 1, \ldots, K.$$  

(30)

Based on (30), and by evaluating $\delta_k = \delta_{k-1}$, we obtain (29).

**Lemma 2 (Ordered Subsequent Clusters):**

The optimal solution must satisfy

$$\delta_k \geq \delta_{k-1} \ \forall k = 2, \ldots, K.$$  

(31)

**Proof:** This is a direct consequence of Lemma 1 since the Lagrangian multipliers $\lambda_k$ are nonnegative based on the dual constraint (25b).

**Lemma 3 (Two Possibilities Per Complementary Slackness Condition):**

The optimal solution must satisfy either

$$\delta_k = \delta_{k+1}, \quad \text{if} \quad \lambda_k = 0 \ \forall k \in \{1, 2, \ldots, K-1\}$$

(32a)

or

$$\delta_k = \pi_k - \sum_{i=1}^{k-1} \delta_i, \quad \text{if} \quad \lambda_k > 0 \ \forall k \in \{1, 2, \ldots, K-1\}.$$  

(32b)

**Proof:** From the complementary slackness conditions in (27), the optimal solution must satisfy either $\lambda_k = 0$, which results in (32a) by substituting for $\lambda_k = 0$ in (29), or $\sum_{i=1}^{k} \delta_i - \pi_k = 0$, which can be rearranged as in (32b). These two possibilities correspond to satisfying the primal inequality constraints in (23b) as strict inequality in the first case or strict equality in the second case. The primal constraint in the latter case can be described as “active.”

**C. Hypothesis Testing Approach**

The main bottleneck in solving (25) using the KKT conditions is that we cannot know in advance which inequality constraints will be active and which ones will be inactive at the optimal solution. All possibilities are likely in the generic problem formulation in which the thresholds $\pi_k$ $(k = 1, \ldots, K-1)$ can take any arbitrary ordered values. Thus, to overcome this bottleneck, we propose a hypothesis-testing approach to solve the KKT conditions. We can start by guessing which inequality constraints in (23b) will be active and which ones will be inactive. This will be a hypothesis that will be tested for validity. Based on the initial assumption of the hypothesis, we can then obtain all primal optimization variables $\delta_k$ $(k = 1, 2, \ldots, K)$ based on (32a) and (32b) of Lemma 3, in addition to (23c). We will have $K$ independent
equations and $K$ variables, so the solution of this system of equations will be unique. After we obtain the solution, we can check for the validity of the hypothesis by examining two things. First, we check that there is no contradiction in the hypothesis’s initial assumption by examining if all primal inequality constraints that were assumed to be inactive are indeed inactive after we substitute the obtained values of $\delta_k$ ($k = 1, 2, \ldots, K$). Second, we check if the obtained $\delta_k$ ($k = 1, 2, \ldots, K$) satisfy (31) of Lemma 2. This check is for the optimality of the solution since the result in Lemma 2 is based on manipulations of the last KKT condition about the vanishing gradient of the Lagrangian. So, in summary, we need to make sure that the hypothesis is both: 1) noncontradicting and 2) generating the unique optimal solution to the problem.

Since we have in total $K - 1$ inequality constraints in (23b), we can have a maximum of $2^{K-1}$ hypotheses to test. In a practical deployment scenario, the number of clusters $K$ is assumed to be limited to a single-digit number. So, checking all possible hypotheses would not be a prohibitively expensive step in such scenarios. Having said that, we also propose a search algorithm to find the correct hypothesis quickly. This can be particularly useful for scenarios of large $K$. However, before discussing the search algorithm, we will give numerical examples of hypothesis testing to clarify the concept. Then, we will formulate the hypothesis testing procedure in a systematic way that can be implemented in an algorithm.

**Example 1 (Three Scenarios of Hypothesis Testing):** Assume that $K = 4$, $M = 100$, $\pi_1 = 10$, $\pi_2 = 46$, and $\pi_3 = 80$. We can distinguish between three cases.

1) **Case A:** Assuming that only the second inequality constraint is active is a contradicting hypothesis.

2) **Case B:** Assuming that only the first and second inequality constraints are active is a noncontradicting hypothesis but not an optimal hypothesis.

3) **Case C:** Assuming that only the first inequality constraint is active is both a noncontradicting hypothesis and an optimal hypothesis.

In case A, since the second inequality constraint is assumed to be active, we should use (32b) for $k = 2$, and use (32a) for $k = 1$ and $k = 3$. The system of four equations that should be solved is expressed as follows:

\[
\begin{align*}
\delta_1 &= \delta_2 \quad \text{(33a)} \\
\delta_2 &= \pi_3 - \delta_1 \quad \text{(33b)} \\
\delta_3 &= \delta_4 \quad \text{(33c)} \\
\delta_4 &= M - (\delta_1 + \delta_2 + \delta_3). \quad \text{(33d)}
\end{align*}
\]

The numerical solution of these equations yields $\delta_1 = \delta_2 = 23$, and $\delta_3 = \delta_4 = 27$. By checking the first inequality constraint, we will find a contradiction since $\delta_1 = 23 > \pi_1 = 10$. So, this is a contradicting hypothesis.

In case B, we will have the same system of (33), but replacing (33a) by $\delta_1 = \pi_1$. The solution of the system of four equations, in this case, yields $\delta_1 = 10$, $\delta_2 = 36$, and $\delta_3 = \delta_4 = 27$. By checking for all inequality constraints, we do not find any contradiction. However, this solution is not optimal since $\delta_2 = 36 > \delta_3 = 27$, and this violates the optimality condition (31) that was proven in Lemma 2.

Finally, in case C, we will have (33a) replaced by $\delta_1 = \pi_1$, and (33b) replaced by $\delta_2 = \delta_3$. The other two equations will be the same. The solution in this case yields $\delta_1 = 10$ and $\delta_2 = \delta_3 = \delta_4 = 30$. This solution does not violate any inequality constraint and satisfies the optimality check in (31). So, it is the optimal solution to the problem.

To formulate the hypothesis testing approach in a systematic algorithm, we use the notation $h$ for a binary vector of $K$ entries that indicates which primal inequality constraints are assumed to be active by the hypothesis. The final entry in $h$ is for the equality constraint (23c), which is, by definition, always active. The entry of $h$ that corresponds to the index $k$ of an active constraint is given the value 1, and the entry that corresponds to the index of an inactive inequality is given the value 0. Thus, for case A in the previous example, we have $h = (0, 1, 0, 1)$. For case B, we have $h = (1, 1, 0, 1)$, and for case C, we have $h = (1, 0, 0, 1)$.

We reserve the notation $p(h)$ for a vector that is obtained from $h$ by indicating the indices of the entries that have value 1. Thus, in our previous examples, if $h = (0, 1, 0, 1)$, then $p = (2, 4)$, and if $h = (1, 1, 0, 1)$, then $p = (1, 2, 4)$, and if $h = (1, 0, 0, 1)$, then $p = (1, 4)$, and so on. It should be noted that while the number of entries of $h$ is fixed at $K$, the number of entries in $p(h)$ varies depending on the sum of all entries of the hypothesis $h$. Furthermore, we use the notation $p^{(i)}$ to indicate the $i$th entry of $p$. For example, if $p = (1, 2, 4)$, then $p^{(1)} = 1$, $p^{(2)} = 2$, and $p^{(3)} = 4$.

Based on the introduced notations for $h$ and $p$, we can obtain that for a given hypothesis $h$, the resulting system of equations to obtain $\delta_k$, $k = 1, \ldots, K$, will have the following generalized formula to obtain the solution for any $p(h)$

\[
\delta_i = \frac{\pi_i p^{(j)} - \pi_j p^{(i-1)}}{p^{(j)} - p^{(i-1)}} \quad \forall i \in \{p^{(j-1)} + 1, \ldots, p^{(j)}\} \quad (34)
\]

where $j = 1, 2, \ldots, |p|$, $p^{(0)} = 0$, $\pi_0 = 0$, and $\pi_K = M$. Implementing (34) in an algorithm is straightforward by running a for-loop for the values of $j$, and obtaining all $\delta_i$’s that are associated with $j$ in (34). After running through all $j$ values, all $\delta_i$, $i = 1, \ldots, K$, will be obtained.

**D. Quick Search Over Space of Possible Hypotheses**

To find the noncontradicting and optimal hypothesis in the minimum number of steps, we need to conduct a theoretical characterization of the space of possible hypotheses so that we can exclude many possibilities directly and find the optimal solution in a quick search.

**Theorem 1 (Inactive Primal Constraints):** If $\pi_k$ in the $k$th inequality constraint in (23b) satisfies

\[
\pi_k > \frac{k}{K}M, \quad k \in \{1, 2, \ldots, K - 1\} \quad (35)
\]

then this constraint will be inactive at the optimal solution of the problem.

**Proof:** Assume that (35) is true and the $k$th inequality constraint is active. The latter means

\[
\sum_{i=1}^{k} \delta_i = \pi_k. \quad (36)
\]
If the solution is optimal, then we must have (31) satisfied as proven in Lemma 2. Thus, we must have \( \delta_k \geq \delta_{k-1} \geq \cdots \geq \delta_1 \). Consequently, we must have that \( \delta_k \geq (\pi_k/k) \), which can be lower bounded based on (35) as follows:

\[
\delta_k \geq \frac{\pi_k}{k} > \frac{M}{K}. \tag{37}
\]

Furthermore, by simple manipulations of the equality constraint in (23c) with (36), we obtain

\[
\sum_{i=k+1}^{K} \delta_i = M - \pi_k. \tag{38}
\]

If the solution is optimal, then we must have (31) satisfied, which yields \( \delta_k \geq \delta_{k-1} \geq \cdots \geq \delta_{k+1} \). Consequently, we must have \( \delta_{k+1} \leq (\lfloor M - \pi_k \rfloor / k) \), which can be upper bounded based on (35) as follows:

\[
\delta_{k+1} \leq \frac{M - \pi_k}{K} \leq \frac{M}{K} - \frac{K}{K} \frac{M}{K} = \frac{M}{K}. \tag{39}
\]

From (37) and (39), we find that \( \delta_k > \frac{M}{K} \geq \delta_{k+1} \). However, based on the optimality check in (31), we must have \( \delta_{k+1} \geq \delta_k \). So, there is a contradiction. Thus, if (35) is true, then the associated constraint must be inactive at optimality.

The statement in Theorem 1 can help in excluding many hypotheses directly from the search algorithm. Furthermore, we can narrow the space of possible hypotheses even further based on the following theorem.

**Theorem 2 (Generalized Check for Inactive Primal Constraints):** Assume that we have any arbitrary two integers \( a \) and \( b \) such that \( 1 \leq a < b \leq K \). If \( \pi_k \) associated with the \( k \)-th inequality constraint in (23b) satisfies

\[
\frac{\pi_k - \pi_a}{b - a} > \frac{\pi_b - \pi_a}{b - a}, \quad k \in \{a + 1, a + 2, \ldots, b - 1\} \tag{40}
\]

then this constraint will be inactive at the optimal solution of the problem.

Before presenting the proof, it should be noted that the statement in Theorem 2 is a generalized version of the statement in Theorem 1. The latter is a special case in which we have \( a = 0 \) and \( b = K \), and similar to (34), we use the notation \( \pi_0 = 0 \) and \( \pi_K = M \).

**Proof:** Assume that both (40) is true and the \( k \)-th inequality constraint is active. The latter means that (36) and (38) are true as discussed in the proof of Theorem 1. Furthermore, we need to use the \( a \)-th and the \( b \)-th primal constraints in the construction of the proof. So, we have

\[
\sum_{i=1}^{a} \delta_i \leq \pi_a \quad \text{and} \quad \sum_{i=1}^{b} \delta_i \leq \pi_b. \tag{41a}
\]

By simple manipulations of (36) and (41a), we obtain that \( \sum_{i=a+1}^{b} \delta_i \geq \pi_k - \pi_a \). Furthermore, if the solution is optimal, then we must have (31) satisfied. Thus, we must have \( \delta_k \geq \delta_{k-1} \geq \cdots \geq \delta_{a+1} \). Consequently, we must have that \( \delta_k \geq (\lceil \pi_k - \pi_a \rceil / [b - a]) \), which can be lower bounded based on (40) as

\[
\delta_k \geq \frac{\pi_k - \pi_a}{b - a}, \quad k \in \{a + 1, a + 2, \ldots, b - 1\} \tag{42}
\]

Next, simple manipulations on the equality constraint in (23c) with (41b) lead to \( \sum_{i=a+1}^{b} \delta_i \geq M - \pi_b \), which can be combined with (38) to obtain

\[
\sum_{i=a+1}^{b} \delta_i \geq (b - k)\delta_{k+1}. \tag{43}
\]

Consequently, we must have

\[
\delta_{k+1} \leq \frac{\pi_b - \pi_k}{b - k}. \tag{44}
\]

Looking back at (40), we can rewrite the inequality as

\[
\pi_k - \pi_a = \{b - a\}(\pi_b - \pi_k), \quad \text{which can be used to bound (44) as follows:}
\]

\[
2\delta_{k+1} \leq \frac{\pi_b - \pi_k}{b - k} < \frac{\pi_b - \pi_a - \frac{k-a}{b-a}(\pi_b - \pi_a)}{b - k} = \frac{(\pi_b - \pi_a)(1 - \frac{k-a}{b-a})}{b - k} = \frac{\pi_b - \pi_a}{b - a}. \tag{45}
\]

From (42) and (45), we find that \( \delta_k > (\lceil \pi_k - \pi_a \rceil / [b - a]) > \delta_{k+1} \). However, based on the optimality check in (31), we must have \( \delta_{k+1} \geq \delta_k \). So, there is a contradiction. Thus, if (40) is true, then the associated constraint must be inactive at optimality.

A geometric interpretation of Theorem 2 is possible, as shown in Fig. 2, since the left term of the inequality in (40) is the slope of the line connecting the two points \((a, \pi_a)\) and \((k, \pi_k)\), while the right term is the slope of the line connecting the two points \((a, \pi_a)\) and \((b, \pi_b)\). Thus, we can equivalently...
constraint will be inactive at the optimal solution if the point $(k, \pi_k)$ is located above the line segment that connects the two points $(a, \pi_a)$ and $(b, \pi_b)$. This statement is true for any value of $a < k$ and any value of $b > k$. Hence, we can establish a nice geometric interpretation of our problem. Determining the active inequality constraints can be interpreted as finding the lower convex hull in a 2-D Cartesian coordinate system for the set of inequality thresholds, represented as

$$\mathcal{P} = \{(k, \pi_k) : k = 0, 1, \ldots, K\}.$$ (46)

Furthermore, the active constraints are only those that are located on the vertices of the piecewise linear boundary of the lower convex hull of $\mathcal{P}$, as illustrated in Fig. 3, which shows the numerical example: $\mathcal{P} = \{(0, 0), (1, 10), (2, 46), (3, 80), (4, 100)\}$. This particular $\mathcal{P}$ is related to the previously discussed Example 1. As shown in Fig. 3, the point $(1, 10)$ is a vertex in the boundary of the lower convex hull of $\mathcal{P}$, while the points $(2, 46)$ and $(3, 80)$ are not located on the boundary of the lower convex hull. Thus, the first constraint should be active at the optimal solution, while the second and third constraints should be inactive. This conclusion was already verified in Example 1. Thus, the geometric interpretation is consistent with the analytical algebraic derivations.

The significance of the geometric interpretation is that it enables us to define an efficient and quick procedure to find the optimal solution by searching for the boundary of the lower convex hull of $\mathcal{P}$ of our problem. This can be done sequentially segment-by-segment starting from the lower-left corner point $(0, 0)$ until we reach the upper right corner point $(K, M)$, as summarized in Algorithm 2.

### E. Complexity Analysis

In this section, we assess the time complexity of Algorithms 1 and 2. The time complexity for the gradient calculation for a client is denoted by $O(g(t))$. The computational complexity for Algorithm 1 is $O(TKNg(t))$, where $T$, $K$, and $N$ stand for the number of iterations, the number of clusters, and the number of subchannels, respectively. In Algorithm 2, we iterate for each cluster, and we have $K$ iterations. In the worst case in each iteration, we iterate through the rest of the clusters, which assumes $K-a$ iterations. Hence, for the worst case, the computational complexity of Algorithm 2 is $O(K^2)$, where $K$ denotes the number of clusters.

### IV. PERFORMANCE ANALYSIS AND COMPARISON

In this section, as a numerical example, we compare the performance of a conventional federated learning scheme (in which $N$ clients are randomly selected per iteration $t$ regardless of their number of data samples) with our suggested PFL scheme (which follows Algorithm 1 and the solution in Algorithm 2). We analyze the proposed PFL method using different data sets, such as the MNIST [23], the federated EMNIST [24], and the federated Shakespeare [1], [25] data sets and neural network architectures, such as MLP, CNN, and RNN to cover a wide range of federated learning applications from the literature. We used Python programming language and Tensorflow Federated framework [27] for the simulations.

Without loss of generality, we assume a scenario in which all clients have similar processors’ speeds, and the variation in computation time $\tau_m$ is solely dependent on the number of data samples per client, i.e., $\tau_m = \sigma n_m$, where $\sigma$ is the computation time per data sample.

#### A. MNIST Data Set

First, we have validated the proposed method using the MNIST data set [23], which consists of hand-written digits. In the simulation scenario, we assumed that there are $M = 1500$ clients. We assumed that the number of data samples (i.e., images of hand-written digits) per client $n_m$ is uniformly distributed between $n_{\min} = 10$ samples and $n_{\max} = 70$ samples. We compared the results of PFL to the conventional federated learning scheme where there is only one cluster. We considered four possible scenarios for the ratio between the variation in computation time to communication time in (22). We performed the simulations for 1, 2, 3, and 4 clusters, respectively. We also studied the effect of the number of subchannels on the performance of PFL. Hence, we varied the number of subchannels as 1, 2, 4, and 8, respectively.
TABLE II
NUMBER OF REQUIRED ROUNDS TO REACH 97% ACCURACY FOR THE CNN MODEL ON THE MNIST DATA SET. COLUMNS REPRESENT THE NUMBER OF CLUSTERS, AND ROWS REPRESENT THE NUMBER OF SUBCHANNELS

| K \ N | 1     | 2     | 4     | 8     |
|-------|-------|-------|-------|-------|
| 1     | 273 (0%) | 180 (0%) | 152 (0%) | 118 (0%) |
| 2     | 186 (32%) | 129 (28%) | 109 (28%) | 106 (10%) |
| 3     | 139 (49%) | 129 (28%) | 114 (25%) | 100 (15%) |
| 4     | 131 (52%) | 122 (32%) | 105 (30%) | 97 (18%) |

TABLE III
NUMBER OF REQUIRED ROUNDS TO REACH 96% ACCURACY FOR THE MLP MODEL ON THE MNIST DATA SET. COLUMNS REPRESENT THE NUMBER OF CLUSTERS, AND ROWS REPRESENT THE NUMBER OF SUBCHANNELS

| K \ N | 1     | 2     | 4     | 8     |
|-------|-------|-------|-------|-------|
| 1     | 981 (0%) | 767 (0%) | 528 (0%) | 445 (0%) |
| 2     | 754 (23%) | 514 (32%) | 480 (9%) | 436 (2%) |
| 3     | 613 (38%) | 451 (41%) | 479 (9%) | 432 (3%) |
| 4     | 506 (48%) | 472 (38%) | 443 (16%) | 424 (5%) |

can be used by a single client only at a given time instant. We used two different deep-learning models for the training of the MNIST data set. The first model is a two-layer CNN model with 32 and 64 $5 \times 5$ filters, respectively. The second model is an MLP model with two layers with 200 nodes in each layer. We run the training algorithm for up to 1000 rounds of global model (server) updates. We obtained the model prediction accuracy by testing the trained model on a separate test data set as a function of the number of communication rounds. The batch size is chosen as 16, and the number of local epochs is set to 1. The client learning rate is optimized using a multiplicative parameter grid, and the server learning rate (the scaling factor) is set to 1. The number of required rounds to reach 97% accuracy for the CNN model trained on the MNIST data set is shown in Table II. Similarly, the number of required rounds to reach 96% accuracy for the MLP model trained on the MNIST data set is shown in Table III. The evolution of the test accuracy with respect to the number of communication rounds is provided in Figs. 4 and 5 for training with the CNN and MLP models, respectively. The test accuracy values in both figures represent the maximum test accuracy obtained until that round.

As seen in Tables II and III, applying clustered scheduling enables achieving faster convergence (in terms of the number of iterations) in comparison with a baseline federated learning approach with no clustering. The conventional federated learning scheme with the CNN model achieves 97% accuracy in 273, 180, 152, and 118 rounds with 1, 2, 4, and 8 subchannels, respectively. The proposed PFL scheme provides 32% gain compared to the conventional federated learning scheme in terms of required rounds by employing only two clusters. The gain provided by clustering is improved for each additional cluster in the system. The marginal gain provided by the PFL scheme decreases with the increasing number of subchannels since the effective number of data samples per communication round is already enough for conventional federated learning.

The benefit of the proposed clustered pipelining scheme can be realized in a different way by decreasing the bandwidth requirements. Thus, we can decrease the number of subchannels needed to achieve the desired accuracy value. For instance, by using four clusters with one subchannel, we reach the same performance as the one achieved by the conventional federated learning with four subchannels.

The MLP model converges slower compared to the CNN model. The conventional federated learning scheme achieves 96% accuracy in 981, 767, 528, and 445 rounds with 1, 2, 4, and 8 subchannels, respectively. Similar to the results with the CNN model, introducing clustering into the federated learning setup decreases the number of required training rounds significantly. However, the PFL method provides little to no gain compared to the conventional federated learning scheme if the number of subchannels is high, e.g., if the number of subchannels is 8 in our simulation setup.

B. Federated EMNIST Data Set

In this section, we use the extended version of the MNIST data set, namely, EMNIST [24]. We use the federated EMNIST...
data set in which the data is already distributed to the clients by their writing styles. The EMNIST data set has letters additionally; however, we will use only the digits data set. The federated EMNIST digits data set has a total of 341,873 train and 40,832 test samples distributed among 3383 clients. There are ten classes in the data set, each corresponding to a handwritten digit. The data distribution among the clients is shown in Fig. 6. As seen from Fig. 6, the distribution of the number of data samples among the clients is unbalanced and not uniform. We formulate the client clustering problem as shown in (23) for clustering the clients for the federated EMNIST data set, and then we solve the problem efficiently using Algorithm 2. Accordingly, we first cluster the clients using the proposed optimization problem in the training step. Then, we train the model using the PFL scheme. We performed the simulations for 1, 2, 3, and 4 clusters and 1, 2, 4, and 8 subchannels, respectively. We used a two-layer CNN model with 32 and 64 $5 \times 5$ filters as the neural network model. The batch size is chosen as 16, and the learning rate is optimized using a multiplicative parameter grid. The number of local epochs is chosen as 1 in order to assess the convergence speed. The required number of rounds to achieve 97% accuracy for the EMNIST digits data set is given in Table IV.

The gain from the proposed PFL scheme decreases if the number of available subchannels increases. Therefore, we could conclude that the proposed scheme may not be feasible if the number of subchannels is sufficiently large, e.g., more than four subchannels in this case. The test accuracy versus the number of global communication rounds for the federated EMNIST data set is provided in Fig. 7. The test accuracy values in Fig. 7 represent the maximum test accuracy obtained until that round. Using the proposed pipelining scheme in the training of the federated EMNIST data set significantly improves the convergence speed of the model. As seen from Table IV, the model converges faster for a higher number of clusters in the system. However, the marginal gain obtained by each additional cluster decreases with the number of clusters already present in the system.

### C. Federated Shakespeare Data Set

For modeling time-series data, we used federated Shakespeare data set [1], [25] for training a stacked two-layer long short-term memory (LSTM) network. The federated Shakespeare data set is built by combining all works of Shakespeare. The data set consists of the speaking roles of 715 characters in each play, corresponding to the clients in our federated learning setup. The data set is unbalanced since most of the plays have a few lines, and a small number of clients have a high count of lines in their plays. Additionally, the number of data samples is not distributed uniformly to the clients. It is also clear that the data set is not independent identically distributed (i.i.d.) since the data set is distributed to the clients according to the played character. We use Algorithm 2 for determining the clusters for the clients. Then, we train the model using the proposed PFL scheme. The neural network model used for this task has an embedding layer at the input, which embeds the input characters into an 8-D space. After the embedding layer, it has two LSTM layers with 256 hidden nodes. Finally, it has an output layer with softmax activation with an output for each character. In training, we set the batch size to be
10 and optimized the learning rate through a multiplicative parameter grid. The simulation results for the application of PFL for the training of the two-layer LSTM network on the Shakespeare data set are given in Table V. The rows represent the number of clusters, and the columns represent the number of subchannels. The values are given in terms of the number of rounds required to reach 40% accuracy in the training. By examining the simulation results for both federated EMNIST and Shakespeare data sets, we conclude that the proposed pipelining scheme improves the results for nonuniformly distributed data more significantly.

V. CONCLUSION

This article proposed using communication pipelining to improve the convergence speed of federated learning in very general settings, including also mobile edge computing applications. In each iteration of a federated learning algorithm, a subset of the total clients is scheduled due to the limited bandwidth, and the scheduled clients wait for the slowest client to finish its computations. The proposed method aims to solve the problem by employing a communication pipelining mechanism to accommodate in a more efficient manner the slowest clients. The main idea is to cluster clients based on their computation time and then schedule a balanced mixture of clients from the different clusters. The faster clients access the communicate their parameters first, while the slower clients are still running their computations. By allowing more clients to be involved in each iteration, the time required to converge to a targeted prediction accuracy of the trained model is reduced significantly. The time per iteration does not change in comparison with conventional federated learning. Nevertheless, the required number of iterations and the total time to train the model are reduced. A generic formulation of the optimal client clustering problem under different settings and an efficient algorithm to obtain the optimal solution are presented. Extensive numerical simulations of the proposed scheme using representative data sets and deep learning architectures are conducted. It is shown that the required number of communication rounds also depends on the number of subchannels, and the gain offered by the proposed method is lower for a high number of subchannels.

ACKNOWLEDGMENT

The statements made herein are solely the responsibility of the authors.
Y. LeCun, C. Cortes, and C. Burges. “MNIST handwritten digit database.” ATT Labs. 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist

G. Cohen, S. Afshar, J. Tapson, and A. V. Schaik, “EMNIST: Extending MNIST to handwritten letters,” in Proc. IEEE Int. Joint Conf. Neural Netw., 2017, pp. 2921–2926.

W. Shakespeare, The Complete Works of William Shakespeare. Hertfordshire, U.K.: Wordsworth Editions, 2007.

S. Boyd and L. Vandenberghe, Convex Optimization. Cambridge, U.K.: Cambridge Univ. Press, 2004.

“Tensorflow federated.” 2021. Google. [Online]. Available: https://www.tensorflow.org/federated

Cihat Keçeci received the B.S. (Hons.) and M.S. degrees in electrical and electronics engineering from Bogazici University, Istanbul, Turkey, in 2017 and 2020, respectively. He is currently pursuing the Ph.D. degree in electrical and computer engineering with Texas A&M University, College Station, TX, USA.

He was with the Scientific and Technological Research Council of Turkey (TUBITAK), Informatics and Information Security Research Center (BILGEM), Izmit, Kocaeli, Turkey. His research interests include machine learning, smart grids, and wireless communications.

Muhammad Shaqfeh (Member, IEEE) received the B.Sc. degree in electrical engineering (communications stream) from United Arab Emirates University, Abu Dhabi, UAE, in 2003, the M.Sc. degree in communications technology from Ulm University, Ulm, Germany, in 2005, and the Ph.D. degree from The University of Edinburgh, Edinburgh, U.K., in 2009.

In 2009, he joined the Department of Electrical and Computer Engineering, Texas A&M University at Qatar, Doha, Qatar, where he is currently an Associate Research Scientist. His research interests include wireless communication networks, information theory, and smart transportation systems.

Fawaz Al-Qahtani (Member, IEEE) received the B.Sc. degree in electrical engineering from the King Fahad University of Petroleum and Minerals, Dhahran, Saudi Arabia, in 2000, the M.Sc. degree in digital communication systems from Monash University, Melbourne, VIC, Australia, in 2005, and the Ph.D. degree in electrical and computer engineering from RMIT University, Melbourne, in December 2009.

He is currently working as the IP Commercialization Manager of ICT Portfolio with Research, Development, and Innovation, Qatar Foundation, Doha, Qatar. From May 2010 to August 2017, he was a Research Scientist with Texas A&M University at Qatar, Doha, Qatar, and Education City, Doha. His research has been sponsored by the Qatar National Research Fund. He has authored or coauthored 100 technical articles published in scientific journals and presented at international conferences. His current research interests include wireless communication and artificial intelligence.

Dr. Al-Qahtani was awarded of the JSERP and NPRP projects.

Erchin Serpedin (Fellow, IEEE) is a Full Professor with the Electrical and Computer Engineering Department, Texas A&M University at College Station, College Station, TX, USA. His current research interests include signal processing, machine learning, artificial intelligence, cyber security, smart grids, and wireless communications.

Dr. Serpedin served as an Associate Editor for more than 12 journals, including journals, such as the IEEE TRANSACTIONS ON INFORMATION THEORY, IEEE TRANSACTIONS ON SIGNAL PROCESSING, IEEE TRANSACTIONS ON COMMUNICATIONS, IEEE SIGNAL PROCESSING LETTERS, IEEE COMMUNICATIONS LETTERS, IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE Signal Processing Magazine, Signal Processing (Elsevier), Physical Communication (Elsevier), and EURASIP Journal on Advances in Signal Processing.