Information-Exchangeable Hierarchical Clustering for Federated Learning With Non-IID Data

Chen-Han Shih[§], Jian-Jhih Kuo[†], and Jang-Ping Sheu[§]
[§]Dept. of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
[†]Dept. of Computer Science & Information Engineering, National Chung Cheng University, Chiayi, Taiwan
E-mail: s108062515@m108.nthu.edu.tw, lajacky@cs.ccu.edu.tw, sheujp@cs.nthu.edu.tw

***Abstract*—Federated Learning (FL) allows Internet-of-Things (IoT) devices to train a global model collaboratively and keep their data locally to address privacy concerns. However, the current FL framework has three main drawbacks, high communication cost, single point of failure, and low accuracy on non-independent and identically distributed (non-IID) data. To this end, we propose a novel FL framework, IHC-FL, to 1) group devices into clusters based on communication cost and model distance, 2) distribute model aggregation over cluster heads, and 3) construct a topology to guide cluster heads to exchange model updates. To the best of our knowledge, this paper makes the first attempt to jointly optimize grouping user devices into clusters and exchanging model updates among cluster heads to enhance model performance. The numeric results show that IHC-FL can reduce 38%∼89% of total communication cost over time than other heuristics with non-IID data on FMNIST and CIFAR-10 to achieve the target accuracy.**

***Index Terms*—Federated learning, clustering, communication topology, machine-learned advice**

I. Introduction

Widespread Internet-of-Things (IoT) devices generate massive amounts of data (e.g., sound, video, text) in our daily lives. Thus, Machine Learning (ML) has been widely used to analyze user data for many applications [1]. In traditional, application servers collect user data and train a global model, but such a centralized approach may violate data privacy. To address this issue, Federated Learning (FL) has emerged to collaboratively train the global model by distributed devices with on-device training data and become a promising paradigm [2]. However, FL has the following drawbacks. 1) Communication cost is considerable due to many model updates exchanged between the central server and a massive number of user devices. 2) FL suffers from the single point of failure. The entire training process will stop if the server fails since the model aggregation only occurs on the server. 3) Devices collect and generate non-independent and identically distributed (non-IID) data across the network. The non-IID data distributions may severely degrade model accuracy and postpone training convergence [1].

To alleviate communication cost of FL, [3] proposed structured updates (i.e., learning an update from a restricted space that can be parameterized using a smaller number of variables) and sketch updates (i.e., compressing the model before sending it to the server) to reduce up-link communication cost. Then, [4] proposed a distillation mechanism that exchanges model outputs instead of model updates to reduce communication cost with a cost of performance degradation. Moreover, Decentralized Learning (DL) further makes user nodes exchange model updates with their neighbors directly [5]. DL does not need a central parameter server, and thus it avoids communication cost from the central server and overcomes the single point of failure. Nevertheless, the test accuracy of the above methods may drop severely with typical non-IID data of user devices.

To solve the problem caused by non-IID data in FL, many researchers proposed hierarchical and clustering mechanisms that encourage users with similar data distributions or model parameters to exchange model updates with each other [6]–[9]. [6] set up K global models in advance and made each user select the best global model among them (i.e., the global model that generates the lowest loss on its local data). [7] proposed Clustered Federated Learning (CFL), which iteratively divides user nodes into two clusters based on their cosine similarities between model updates. Then, the user nodes of each cluster train a model for their local data collaboratively to mitigate the effect of non-IID data. Unlike CFL, Hierarchical Federated Learning (HFL) is a two-level FL, having only one clustering step to group users based on model distance to reduce clustering cost [8]. The nodes in each cluster train a cluster model via FL, and then all nodes' models are aggregated by a central server periodically. [9] took a regrouping mechanism in FL to iteratively update the members of clusters based on model distance and loss. However, the above methods require long-term additional computation on the permanent parameter server and deteriorate the effect of the single point of failure.

Inspired by DL and CFL, we propose a novel framework of FL with Information-exchangeable Hierarchical Clustering (IHC-FL) to solve the mentioned drawbacks of conventional FL. To the best of our knowledge, IHC-FL is the first one to leverage both DL and CFL to avoid the hot spot of the communication and computation and mitigate the effect of non-IID data. To this end, IHC-FL groups user nodes into clusters (i.e., training groups), and each cluster has a cluster head, similar to CFL. Meanwhile, it exploits the concept of DL to establish a communication topology connecting cluster heads. Thus, for each epoch, each cluster head coordinates its members to collaboratively train a model by FL and exchanges model updates with its neighboring cluster heads based on the constructed communication topology. The proposed promising framework combines the advantages of DL and CFL but also raises three research challenges as follows. 1) *Effect of the number of clusters on communication cost*. Intuitively,

This work was supported by the National Science and Technology Council, Taiwan, under Grants NSTC 110-2221-E-007-037-MY2 and 111-2628-E-194-001-MY3 and Qualcomm Technologies, Inc. under grant SOW NAT-487844.

grouping closer user nodes into a cluster can lead to more clusters and a higher data transfer rate within a cluster, reducing intra-cluster communication cost. However, inter-cluster communication cost (i.e., between cluster heads) may increase significantly as the number of clusters inclines. Thus, to minimize total communication cost (including intra- and inter-cluster communication costs), it has to choose the number of clusters carefully. 2) *Effect of the number of clusters on model performance.* Grouping user nodes with similar model updates together can lead to a high performance of each cluster's model for their local data [7]. Besides, the more clusters, the smaller the cluster size, the faster the convergence speed. However, it may make the training data insufficient in a cluster and generate a less robust model. Thus, to ease the effect of non-IID data, it has to fine-tune the number of clusters. 3) *Communication topology construction for cluster heads.* Selecting the cluster heads with similar models for each cluster head as neighbors to exchange model updates can further mitigate the effect of non-IID data and improve model performance for local data of their members. However, excessive distant neighbors selected for each cluster head will cause significant inter-cluster communication cost. Thus, to alleviate the effect of non-IID data and minimize total communication cost, it has to balance model distance and communication cost.

To further solve the above challenges in IHC-FL, we present a new optimization problem. More specifically, given 1) a set of user nodes and 2) the network information, the optimization problem aims to group user nodes into clusters, select cluster heads, and construct a communication topology for the cluster heads to minimize total communication cost while optimizing model performance. However, simultaneously making the above decisions to minimize total communication cost is complicated and intractable. To overcome the difficulty, we rewrite the original optimization problem to a novel optimization problem termed Cluster Formation (CF) that adopts an *oracle* trained to predict communication cost for exchanging the models among any given set of selected cluster heads, inspired by the idea of value function in reinforcement learning. To further train the oracle, we introduce another optimization problem, Topology Construction (TC), to construct a connected communication topology with the minimum communication cost for any given set of selected cluster heads while bounding model distance between any adjacent nodes in the communication topology. The solutions generated by TC will be used as samples to train the oracle later. Last, we design an efficient framework to alternately solve CF and TC and train the *oracle* until the solution converges. Afterward, IHC-FL will follow the solution to train user models, aggregate local updates on cluster heads, and exchange models among cluster heads.

With the framework's assistance, IHC-FL takes the advantages of DL and CFL to address the three challenges above. Overall, IHC-FL can 1) alleviate communication cost by selecting a suitable number of clusters, 2) distribute model aggregation over cluster heads to mitigate the effect of the single point of failure, and 3) balance model distances between user nodes and their cluster heads and between neighboring clusters on the communication topology to avoid performance degradation with non-IID data distribution. Finally, extensive simulation results manifest that IHC-FL can outperform the state-of-the-art frameworks by up to 63.221%.

II. Optimization Problem for IHC-FL

This section first provides an overview of IHC-FL. Subsequently, we observe the effect of the number of cluster heads on total communication cost (including intra- and inter-cluster communication costs) in IHC-FL. Then, we conduct experiments to show the effect of the number of cluster heads on the model accuracy in IHC-FL, where user devices have non-IID data. After that, we formulate the optimization problem to solve the three challenges mentioned in the introduction section according to the observed effects.

A. The Overview of IHC-FL

IHC-FL is run over an IoT system that consists of a platform and user devices. IHC-FL has two main stages. In the first stage, the platform collects the effect of user data on model training. To this end, it sends the initial model to all participant nodes, and each node train the model with its local data for a few epochs (not yet converged) and uploads the model updates back to the platform. Afterward, the platform calculates model distance[1] and communication cost (e.g., transmission time) between any two nodes to group the nodes into several clusters. It determines the communication topology based on model distance and communication cost to minimize total communication cost (including intra- and inter-cluster communication costs) while accelerating later training. Note that the platform no longer needs to monitor the training after the first stage.

In the second stage, the training process starts. Each cluster head sends its model to its members. Each cluster member receives the model, performs training using its local data, and uploads the model updates to its cluster head. Each cluster head then aggregates the received models and sends the aggregated model to its neighboring cluster heads in the communication topology. Afterward, each cluster head aggregates the received models as its new model. All cluster heads will repeat the above operations in the second stage until convergence.

To optimize the training process in the second stage, we will observe the effects of the number of clusters and formulate a novel optimization problem for IHC-FL in the following.

B. Information-exchangeable Hierarchical Clustering

In FL, the model parameter server performs aggregation and sends back and forth the model parameters. Communication cost will be severe for the central parameter server. Thus, splitting the parameter server into multiple cluster heads is intuitive to remedy the single point of failure and distribute communication cost over cluster heads. Then, we cannot help wondering whether communication cost can also be reduced significantly by doing so. To this end, we observe the effects of splitting the parameter center into one, two, four, and eight cluster heads for 12 participant nodes, as shown in Fig. 1. The solid and dashed edges denote the intra- and inter-cluster communications, respectively. Assume that the communication

[1]In this paper, we use cosine distance [10] to represent model distance.

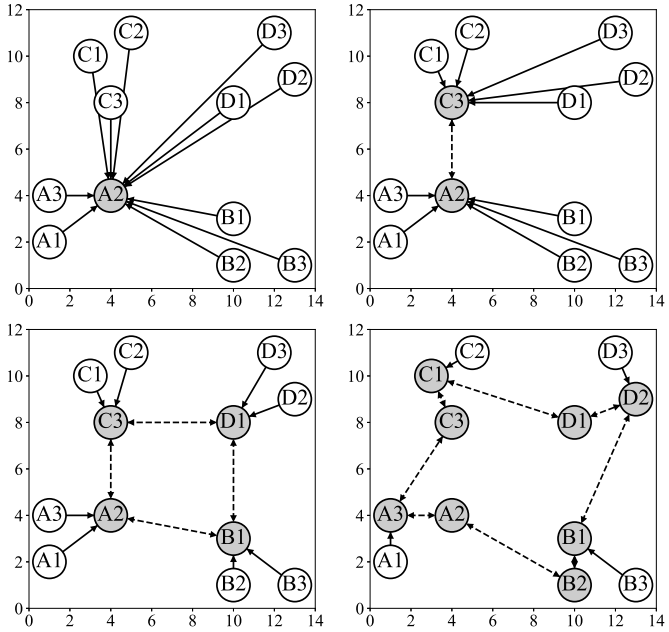

Fig. 1. Effect of the number of clusters on total communication cost.

cost between every two nodes is the Euclidean distance, and the communication graph between cluster heads adopts a ring, which is common for DL [11]. The total communication cost is $6.08+7.07+4.0+3.0+3.61+7.21+10.63+10.3+6.08+6.71+9.49 = 74.18$ for the one-cluster scenario, $3.0+3.61+6.08+6.71+9.49+2.24+3.16+8.54+9.06+6.0+4.0 = 61.89$ for the two-cluster scenario, $2.24+3.16+4.0+3.0+3.61+6.0+3.61+3.16+5.0+3.61+2.0+6.08 = 45.47$ for the four-cluster scenario, and $2.24+2.24+7.28+3.16+2.24+6.71+3.61+2.0+6.71+3.0+2.0+5.0 = 46.19$ for the eight-cluster scenario. This shows that choosing the number of cluster heads can also reduce communication cost significantly (i.e., from 74.18 to 45.47). However, excessive cluster heads may increase communication cost conversely.

C. Effect of Number of Clusters on Model Performance

To observe the effect of the number of clusters on model performance, we split the data into non-IID user data based on symmetric Dirichlet distribution with a parameter α according to [12]. To be general, we set α to different levels, ranging from 0.3 to 1.0, and conducted ten trials for each level to get the average result. Then, the K-means algorithm [13] was adopted here to obtain K cluster heads and the total number of nodes in each cluster based on model distance, where K ranged from 1 to 16. Afterward, the K cluster heads formed a ring communication topology with minimum total model distance. For each training epoch, every cluster head aggregates the models from its cluster members and then exchanges the aggregated model with its neighboring cluster heads.

Fig. 2 shows the best accuracy occurs when the cluster head number is around four. The more cluster heads, the smaller the maximum model distance between neighboring cluster heads (denoted by γ). An increase in cluster heads can benefit model performance until reaching four. After that, the cluster members in each cluster become insufficient, causing a decrease in the accuracy of each cluster.

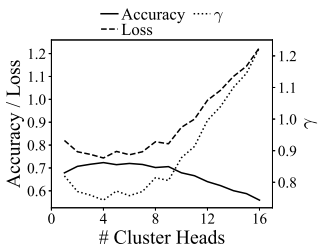

Fig. 2. Effect of number of clusters on accuracy, loss, and γ using the FMNIST dataset.

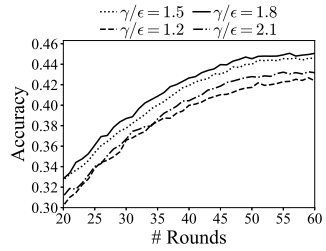

Fig. 3. Effect of ratio γ/ϵ on accuracy using non-IID data distributions of CIFAR-10 dataset.

D. System Model and Problem Formulation

Following Section II-A, the platform has to jointly consider model distance and communication cost to group the nodes into several clusters to minimize total communication cost (including intra- and inter-cluster communication costs) while speeding up convergence. To this end, the optimization problem is formulated as Integer Linear Programming (ILP) as follows.

$$\text{min.} \quad \sum_{n\in V}\sum_{a\in V} z_{n,a}\cdot p_{n,a} + \sum_{i\in V}\sum_{j\in V} x_{i,j}\cdot p_{i,j} \tag{1a}$$

$$\text{s. t.} \quad z_{n,a} \le y_a, \qquad \forall n,a \in V \tag{1b}$$

$$\sum_{a\in v} z_{n,a} = 1, \qquad \forall n \in V \tag{1c}$$

$$\delta_{n,a}\cdot z_{n,a} \le \epsilon, \qquad \forall n,a \in V \tag{1d}$$

$$\Delta_{i,j}\cdot x_{i,j} \le \gamma, \qquad \forall i,j \in V \tag{1e}$$

$$x_{i,j} = x_{j,i}, \qquad \forall i,j \in V \tag{1f}$$

$$x_{i,j} \le y_i, \qquad \forall i,j \in V \tag{1g}$$

$$\sum_{j\in V} x_{i,j} = \mathcal{D}\cdot y_i, \qquad \forall i \in V \tag{1h}$$

$$x_{i,j} \ge f^d_{i,j}, \qquad \forall i,j,d \in V\setminus\{r\} \tag{1i}$$

$$x_{r,i} + z_{r,i} \ge f^d_{r,i}, \qquad \forall i,d \in V\setminus\{r\} \tag{1j}$$

$$\sum_{i\in V} f^d_{j,i} - \sum_{i\in V} f^d_{i,j} = 0,\ \forall j,d \in V\setminus\{r\}, j\ne d \tag{1k}$$

$$\sum_{i\in V} f^d_{d,i} - \sum_{i\in V} f^d_{i,d} = -y_d, \qquad \forall d \in V\setminus\{r\} \tag{1l}$$

$$\sum_{i\in V} f^d_{r,i} - \sum_{i\in V} f^d_{i,r} = y_d, \qquad \forall d \in V\setminus\{r\} \tag{1m}$$

In the above ILP, binary decision variable $y_a \in \{0,1\}$ represents whether node a is selected as a cluster head. That is, a is a cluster head if and only if $y_a = 1$. In addition, we make binary decision variable $z_{n,a} \in \{0,1\}$ indicate whether node n is connected to cluster head node a. In other words, n is connected to a that we select as a cluster head if and only if $z_{n,a} = 1$. Moreover, binary decision variable $x_{i,j} \in \{0,1\}$ denotes whether node i and node j are both cluster heads and they exchange model parameters with each other.

The objective (1a) is to minimize total communication cost, including 1) communication cost between each cluster head and its members (i.e., intra-cluster communication cost)

and 2) communication cost between cluster heads (i.e., inter-cluster communication cost), where parameter $p_{i,j}$ denotes the communication cost from node i to node j.

Constraint (1b) states that node n selects node a as its cluster head only if a is a cluster head, and constraint (1c) indicates that each node must join one cluster. Let $\delta_{i,j}$ be the model distance between node i and node j before model aggregation, while $\Delta_{i,j}$ represents the model distance between cluster heads i and j after model aggregation. Note that the platform can calculate model distance between two participant nodes based on their models trained for a few epochs. Constraint (1d) prevents node n from joining the cluster of head a if their models are dissimilar (i.e., their model difference is greater than a given threshold ϵ) before model aggregation. In contrast, constraint (1e) bounds the model difference between cluster heads i and j within a given threshold γ if they determine to exchange their cluster model parameters on the communication graph. Subsequently, constraint (1f) ensures that if cluster head i sends its cluster model to cluster head j, then j will send j's cluster model to i to enable DL among the cluster heads.

Constraints (1f) and (1g) jointly guarantee that head i sends the cluster model parameters to head j only if node i and node j are both selected as cluster heads. Therefore, the selected cluster heads will form a communication graph, where each link on such a graph is between two cluster heads. Constraint (1h) make every selected cluster head has the same degree in the communication graph to speed the DL convergence among cluster heads according to [11]. If $y_i = 1$, then the degree of node i will be limited to $\mathcal{D}$ by constraint (1h), where $\mathcal{D} \in \mathbb{Z}^+$ is user-defined parameter (no less than two) to make the communication graph regular. $\mathcal{D}$ is often set to 2 or 4, leading to a ring or a 4-regular graph, respectively [11]. Otherwise, the degree will be zero due to constraint (1g). Constraint (1h) make each cluster head exchange model update with exactly $\mathcal{D}$ cluster heads to ensure the connectivity among headers.

Afterward, constraints (1i), (1j), (1k), (1l), and (1m) guarantee that the communication graph is connected by finding a *virtual flow* from an arbitrarily selected source r to each cluster head. Note that the virtual flow is not an actual flow in the network but is only used to ensure any two cluster heads in the formed communication graph are connected. That is, the two cluster heads will share their model parameters with each other later. Specifically, r is a node randomly pre-selected from V, and it does not have to be a cluster head. After that, let binary decision variable $f_{i,j}^d$ denote whether the flow from r to a cluster head d visits the link between nodes i and j. By doing so, constraints (1i) and (1j) can ensure that the virtual flows will pass only cluster heads in the communication graph. Finally, flow-like constraints (1k), (1l), and (1m) construct a virtual flow from r to each selected cluster head d through only the cluster heads in the induced communication graph.

The problem is NP-hard since the well-known NP-Complete problem, the Hamiltonian cycle problem (HCP) [14], can be reduced to our problem. Also, it does not has any polynomial-time approximation algorithm with any constant approximation ratio [15]. Otherwise, such an algorithm can know whether there is a Hamiltonian cycle for any instance of the HCP in polynomial time. The proof is detailed in Appendix A of [16].

To solve the problem efficiently, in the following section, we introduce how to leverage machine-learned advice to find the appropriate set of cluster heads to overcome the three IHC-FL's challenges mentioned in the introduction section.

III. Reformulation and Clustering for IHC-FL

In this section, we first reformulate the problem to select the cluster heads with machine-learned advice and formulate a sub-problem to construct a communication graph for any given selected cluster heads. After that, we develop a framework to solve the above two problems alternately until convergence. Finally, we discuss how to select ϵ and γ with experiments.

A. Problem Reformulation with Machine-Learned Advice

Intuitively, solving problem (1) is time-consuming since it has numerous variables and constraints. Instead, we reformulate the problem as a novel Mixed-Integer Linear Programming (MILP) as shown in problem (3) termed CF using a Neural Network (NN) parameterized by Θ (i.e., weights $\boldsymbol{w}$ and biases $\boldsymbol{b}$) to estimate the inter-cluster communication cost (i.e., machine-learned advice). In this way, problem (1) can be narrowed down to identify a suitable set of cluster heads without explicitly finding an exact communication graph for the selected cluster heads. To this end, the sub-problem, minimizing inter-cluster communication cost for any given cluster heads, is separated from problem (1), termed TC, and shown in problem (4), while the NN is trained to act as an oracle, predicting the inter-cluster communication cost. With the reformulation, the running time of solving problem (1) can be reduced by 99.99%. Due to the page limit, it is shown in Table III of Appendix B in [16].

Specifically, the NN consists of an input layer, a fully-connected hidden layer, and a scalar output. The input layer includes the values of all decision variables y_a (i.e., whether node a is selected as a cluster head), while the hidden layer has 16 neurons. To simulate the behavior of a hidden layer in a neuron network with Rectified Linear Unit (ReLU) activation functions in a MILP formulation, we impose the constraints in the following definition from [17].

Definition 1. *Let I and H denote the input of hidden layer and the set of neurons in the hidden layer, respectively. The MILP formulation for simulating a hidden layer in a neural network with ReLU activation functions is defined as follows:*

$$y^{(h)} \geq \boldsymbol{w^{(h)}} \cdot \boldsymbol{\chi} + b^{(h)}, \qquad \forall h \in H, \quad \text{(2a)}$$

$$y^{(h)} \leq \boldsymbol{w^{(h)}} \cdot \boldsymbol{\chi} + b^{(h)} + M \cdot (1 - z^{(h)}), \qquad \forall h \in H, \quad \text{(2b)}$$

$$y^{(h)} \leq M \cdot z^{(h)}, \qquad \forall h \in H, \quad \text{(2c)}$$

$$(\boldsymbol{\chi}, y^{(h)}, z^{(h)}) \in \mathbb{R}_{\geq 0}^{|I|} \times \mathbb{R}_{\geq 0} \times \{0, 1\}, \qquad \forall h \in H, \quad \text{(2d)}$$

where 1) $\boldsymbol{\chi}$ is the input vector of hidden layer, 2) $y^{(h)}$ is the ReLU's output of neuron $h \in H$, 3) $\boldsymbol{w^{(h)}}$ is the weight vector of neuron $h \in H$, 4) $b^{(h)}$ is the bias of the neuron $h \in H$, 5) $z^{(h)}$ indicates whether the ReLU's output of neuron $h \in H$ is positive, and 6) M is a sufficiently large constant. Therefore, if $\boldsymbol{w^{(h)}} \cdot \boldsymbol{\chi} + b^{(h)} > 0$, then $z^{(h)}$ will be one, and $y^{(h)}$ will be squeezed to $\boldsymbol{w^{(h)}} \cdot \boldsymbol{\chi} + b^{(h)}$ by constraints (2a) and (2b). Otherwise, constraints (2c) and (2d) will make $y^{(h)} = 0$.

Thus, we derive the reformulated problem (CF) by replacing constraints (1e)−(1m) with constraints (3e)−(3g), where the latter constraints are used to simulate an NN to predict inter-cluster communication cost, as follows:

$$\text{min.} \quad \sum_{n\in V}\sum_{a\in V} z_{n,a}\cdot p_{n,a} + \mathcal{C} \tag{3a}$$

$$\text{s. t.} \quad z_{n,a} \le y_a, \qquad \forall n,a \in V \tag{3b}$$

$$\sum_{a\in v} z_{n,a} = 1, \qquad \forall n \in V \tag{3c}$$

$$\delta_{n,a}\cdot z_{n,a} \le \epsilon, \qquad \forall n,a \in V \tag{3d}$$

$$\mathcal{C} = \sum_{h\in H} w_{(h)}^{out}\cdot y^{(h)} + b^{out} \tag{3e}$$

$$y^{(h)} \ge \sum_{a\in V} w_a^{(h)}\cdot y_a + b^{(h)}, \qquad \forall h \in H \tag{3f}$$

$$y^{(h)} \le \sum_{a\in V} w_a^{(h)}\cdot y_a + b^{(h)} + M\cdot(1-z^{(h)}), \quad \forall h \in H \tag{3g}$$

$$y^{(h)} \le M\cdot z^{(h)}, \qquad \forall h \in H \tag{3h}$$

The objective (3a) aims to minimize total communication cost, where decision variable $\mathcal{C}$ is the estimated inter-cluster communication cost predicted by the NN. The NN with weights $\boldsymbol{w}$ and biases $\boldsymbol{b}$ is trained by the sub-problem TC in (4), which will be illustrated in detail later in Section III-B. Following Definition 1, constraints (3e)−(3h) simulate the NN's structure with ReLU activation functions. Constraint (3e) calculates the NN's output (i.e., inter-cluster communication cost), while constraints (3f)−(3h) represent the hidden layers.

Subsequently, for any given set of cluster heads denoted by $\mathcal{V}$ (i.e., the set of nodes with $y_a = 1$ in problem (3)), the sub-problem TC separated from problem (1) is defined as follows:

$$\text{min.} \quad \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}} x_{i,j}\cdot p_{i,j} \tag{4a}$$

$$\text{s. t.} \quad \Delta_{i,j}\cdot x_{i,j} \le \gamma, \qquad \forall i,j \in \mathcal{V} \tag{4b}$$

$$x_{i,j} = x_{j,i}, \qquad \forall i,j \in \mathcal{V} \tag{4c}$$

$$\sum_{j\in\mathcal{V}} x_{i,j} = \mathcal{D}, \qquad \forall i \in \mathcal{V} \tag{4d}$$

$$x_{i,j} \ge f_{i,j}^d, \qquad \forall i,j,d \in \mathcal{V}\setminus\{r\}, \tag{4e}$$

$$\sum_{i\in\mathcal{V}} f_{j,i}^d - \sum_{i\in\mathcal{V}} f_{i,j}^d = 0, \ \forall j,d \in \mathcal{V}\setminus\{r\}, j \ne d, \tag{4f}$$

$$\sum_{i\in\mathcal{V}} f_{d,i}^d - \sum_{i\in\mathcal{V}} f_{i,d}^d = -1, \qquad \forall d \in \mathcal{V}\setminus\{r\}, \tag{4g}$$

$$\sum_{i\in\mathcal{V}} f_{r,i}^d - \sum_{i\in\mathcal{V}} f_{i,r}^d = 1, \qquad \forall d \in \mathcal{V}\setminus\{r\}, \tag{4h}$$

It is not difficult to observe that most constraints in problem (4) are identical to those in problem (1). The underlying reason is that problem (4) is extracted from problem (1). Besides, it is worth noting that problem (4) does not consider the nodes that are not selected as cluster heads, and constraint (4d) is derived directly from (1h). Moreover, variable y_d is not needed for constraints (4g) and (4h).

B. Clustering Framework for IHC-FL

With (3) and (4), we develop a clustering framework to select the suitable set of clusters and build the communication graph. It includes three phases: 1) Oracle Initialization Phase (OIP), 2) Cluster Formation Phase (CFP), and 3) Topology Construction Phase (TCP). OIP initializes an NN as an oracle to learn and predict the result of problem (4) later. CFP uses an MILP solver to solve problem (3) with the simulated NN inside to obtain suitable cluster heads and their members. Given the set of cluster heads selected by CFP, TCP uses an MILP solver to solve problem (4) to derive inter-cluster communication cost. Then, it randomly draws samples from the previous solutions and trains the NN to predict inter-cluster communication cost of any set of selected cluster heads. Overall, the clustering framework first executes OIP. After that, CFP and TCP are called alternately to repeatedly select a set of cluster heads, calculate the communication graph and its cost, and train the NN until the set of cluster heads converges.

Recall that in Section II-A, IHC-FL will start the second stage after the set of cluster heads is determined. In the following, we observe the trade-off between intra- and inter-cluster model distance and their effects on model accuracy. Then, we discuss how to tune the model distance thresholds ϵ and γ based on the observation to achieve high performance.

C. Trade-off between intra- and inter-cluster model distances

To avoid insufficient or excessive clusters that will degrade model performance significantly, we set ϵ with the formula $\epsilon = \max_{i\in V}\left(\min_{j\in V\setminus\{i\}}\delta_{i,j}\right)$. Then, to get the best setting for ϵ and γ, we distributed the CIFAR-10 dataset to 32 nodes using symmetric Dirichlet distribution (i.e., non-IID) and conducted experiments to observe the effect of the ratio $\frac{\gamma}{\epsilon}$ on model accuracy. Fig. 3 shows the best setting of $\frac{\gamma}{\epsilon}$ for model accuracy is 1.8, also works effectively on the FMNIST dataset.

IV. EVALUATION

A. Simulation Settings

1) Training Settings: We perform experiments using two well-known datasets: FMNIST (70,000 samples) and CIFAR-10 (60,000 samples). For both datasets, we adopt a convolutional neural network (CNN) with two 5×5 layers. The 1^{st} and 2^{st} layers have six and sixteen output channels, respectively. The two layers are both followed by a 2×2 max pooling. Following [12], we use the symmetric Dirichlet distribution with a parameter α to randomly distribute samples to nodes. The smaller the parameter α, the more non-IID the nodes' data distributions. Each node employs 70% of its assigned data for training and 30% for testing and aims to acquire a model optimized for its local test data distribution. For the simulations, we set $\mathcal{D} = 2$ for IHC-FL according to [11].

2) Communication Cost: Nodes with higher spatial similarities usually have more similar data distributions and smaller communication cost [18]. Thus, we randomly generate communication cost from 1 to 1.5 seconds for simulating the time required by exchanging model updates between nodes with similar data, and 1 to 2 seconds between dissimilar others.

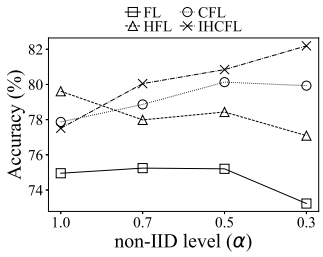

Fig. 4. Effect of different levels of non-IID data on the 32-device accuracy of FMNIST dataset.

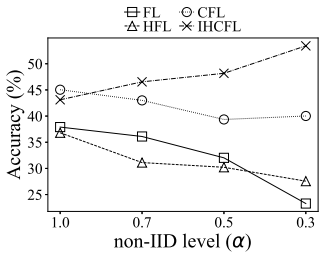

Fig. 5. Effect of different levels of non-IID data on the 32-device accuracy for CIFAR-10 dataset.

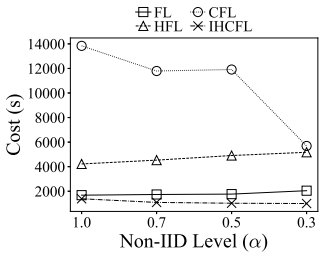

Fig. 6. Effect of different levels of non-IID data on 32-device total cost over time for the FMNIST dataset.

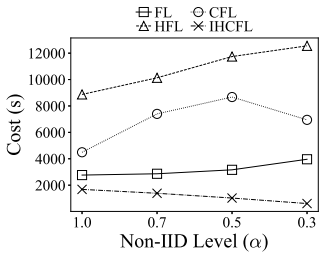

Fig. 7. Effect of different levels of non-IID data on 32-device total cost over time for the CIFAR-10 dataset.

TABLE I
IMPROVEMENT IN ACCURACY FOR 32 DEVICES

Method	non-IID	FMNIST (25 rounds)	CIFAR-10 (80 rounds)
FedAvg	$\alpha = 0.3$	74.798% (1.0x)	27.283% (1.0x)
HFL		76.738% (1.026x)	26.117% (0.957x)
CFL		82.925% (1.109x)	42.915% (1.573x)
IHC-FL		**85.416% (1.142x)**	**53.714% (1.969x)**

3) Performance metrics: Two metrics are adopted to evaluate performance: 1) test accuracy and 2) communication cost. Note that each result is averaged over ten trials. The target accuracy ζ is set to 80% in FMNIST and 37% in CIFAR-10.

B. Effects of Non-IID Level and Number of Nodes

We compare IHC-FL with Federated Averaging (FedAvg) [2], CFL [7], and HFL [8] with different data distribution and the number of devices. Figs. 4 and 5 show that IHC-FL handles non-IID data distributions better than other methods and outperforms other methods in most cases and increases the accuracy by 0.803% to 63.221% in average. Table I further demonstrates IHC-FL improves accuracy significantly in the network with 32 nodes. More experiment results are provided in Figs. 8 and 9 and Table IV in Appendix B in [16].

C. Communication Cost and Number of Training Rounds

We compare IHC-FL with FedAvg, CFL, HFL on total communication cost and the number of training rounds required to achieve the target accuracy with different datasets. Table II show total communication cost, the number of training rounds to achieve the target accuracy, and total communication cost over time for each method with $\alpha = 0.5$ and 32 nodes and the CIFAR-10 dataset. Then, Figs. 6 and 7 further show the effect of the non-IID level and number of nodes on total communication cost over time for different methods. Overall, IHC-FL can reduce the number of rounds by 3.9%–72.0% and total communication cost over time by 38.4%–89.3%, while achieving the target accuracy. More training results are given in Figs. 10 and 11 and Table V in Appendix B in [16].

TABLE II
ROUNDS TO ACHIEVE THE TARGET ACCURACY AND TOTAL COMM. COST OVER TIME WHEN 32 DEVICES AND $\alpha = 0.5$ ON CIFAR-10 DATASET.

Method	Total comm. cost (s)	# rounds to achieve the target acc.	Total comm. cost over time (s)
FedAvg	42 (1.0x)	65 (1.0x)	2730 (1.0x)
HFL	107 (2.55x)	78 (1.2x)	8346 (3.06x)
CFL	71 (1.69x)	49 (0.75x)	3479 (1.27x)
IHC-FL	**43 (1.02x)**	**20 (0.31x)**	**860 (0.32x)**

V. CONCLUSION

In this paper, we proposed a new clustering and hierarchical FL framework termed IHC-FL to tackle three drawbacks of the FL framework. First, IHC-FL significantly decreases communication cost by selecting a suitable number of clusters. Second, it performs model aggregation on cluster heads to remedy the single point of failure. Third, it mitigates the effect of the non-IID data on convergence. The paper indicates that a carefully-selected set of clusters for clients with a subtly-built topology for cluster heads can help speed up model training substantially in non-IID environment. Finally, IHC-FL can reduce 3.9%–72.0% training rounds and 38.4%–89.3% communication cost compared with the other methods.

REFERENCES

[1] T. Li *et al.*, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, 2020.

[2] B. McMahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *Proc. AISTATS*, 2017.

[3] J. Konečný *et al.*, "Federated learning: Strategies for improving communication efficiency," in *Proc. NeurIPS Workshop on PMPML*, 2016.

[4] H. Seo *et al.*, "Federated knowledge distillation," *Machine Learning and Wireless Communications*, pp. 457–485, 2022.

[5] I. Hegedűs, G. Danner, and M. Jelasity, "Gossip learning as a decentralized alternative to federated learning," in *Proc. IFIP DAIS*, 2019.

[6] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," in *Proc. NeurIPS*, 2020.

[7] F. Sattler *et al.*, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, pp. 3710–3722, 2020.

[8] C. Briggs *et al.*, "Federated learning with hierarchical clustering of local updates to improve training on non-iid data," in *Proc. IJCNN*, 2020.

[9] G. Long *et al.*, "Multi-center federated learning: clients clustering for better personalization," *World Wide Web*, vol. 26, pp. 481–500, 2023.

[10] P.-N. Tan *et al.*, *Introduction to data mining*. Pearson Edu. India, 2016.

[11] Y.-C. Lin, J.-J. Kuo, W.-T. Chen, and J.-P. Sheu, "Reinforcement based communication topology construction for decentralized learning with Non-IID data," in *Proc. IEEE GLOBECOM*, 2021.

[12] S. P. Sturluson *et al.*, "Fedrad: Federated robust adaptive distillation," *arXiv preprint arXiv:2112.01405*, 2021.

[13] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *J. R. Stat. Soc. Ser. C Appl. Stat.*, vol. 28, pp. 100–108, 1979.

[14] R. J. Gould, "Advances on the hamiltonian problem–a survey," *Graphs and Combinatorics*, vol. 19, no. 1, pp. 7–52, 2003.

[15] D. P. Williamson and D. B. Shmoys, *The design of approximation algorithms*. Cambridge university press, 2011.

[16] C.-H. Shih, J.-J. Kuo, and J.-P. Sheu, "Information-exchangeable hierarchical clustering for federated learning with non-iid data (technical report)," May 2023. [Online]. Available: https://github.com/hankle10/IHC-FL/blob/main/TR.pdf

[17] R. Anderson *et al.*, "Strong mixed-integer programming formulations for trained neural networks," *Math. Program.*, vol. 183, pp. 3–39, 2020.

[18] Y. Shi *et al.*, "Beyond IID: learning to combine non-IID metrics for vision tasks," in *Proc. AAAI*, 2017.

APPENDIX A
NP-HARDNESS

To prove problem 1 is NP-Hard, we reduce the well-known NP-Complete problem, the HCP [14], to our problem. Recall that the HCP asks whether there is a cycle that visits every node once and then goes back to the start node for a given graph. Subsequently, for any given instance of the HCP, i.e., a graph $G = \{V, E\}$, we show how to construct a corresponding instance of our problem, i.e., a graph $G' = \{V', E'\}$, as follows.

First, for each node in V, we create a corresponding node and add it to V'. Then, for any two nodes in V', we create a link between them and add it to E'. Moreover, the weight of each link $e \in E'$ is set to a positive value according to the following formula:

$$p_{i,j} = \begin{cases} 1, & \text{if } (i,j) \in E \\ 2, & \text{otherwise.} \end{cases} \tag{5a}$$

After that, we set ϵ in the instance of our problem to zero such that each node in V' must become a cluster head and there is no intra-cluster communication cost. Then, we set $\mathcal{D}$ to two, and thus constraint (1h) will limit each cluster head's degree to two. That is, the solution should be a ring. Finally, we set γ to sufficiently large, and thus $\Delta_{i,j}$ for any nodes $i, j \in V$ would always be less than γ. Clearly, the above construction can be done in polynomial time.

In this way, graph G contains a Hamiltonian cycle if and only if the optimal solution of G' has the total communication cost (including intra- and inter-cluster communication costs) of $|V|$. By contrast, there is no Hamiltonian cycle in G if and only if the optimal solution must select a link with cost 2 such that the total communication cost greater than $|V|$. Therefore, if our problem (i.e., problem 1) can be solved in polynomial time, such an algorithm can be used to answer the HCP. That is, our problem's NP-hardness is proven.

It is worth noting that Theorem 2.9 in [15] shows that there is no α-approximation algorithm for Traveling Salesman problem (TSP) unless $P = NP$, where $\alpha > 1$ is a constant. This implies that our problem does not admit any approximation algorithm with any constant approximation ratio since TSP is a special case of our problem. Specifically, any TSP instance can be mapped to an instance of ours as follows. 1) Set ϵ and γ to 0 and a sufficiently large constant, respectively. 2) Create a corresponding node for each city of the TSP instance and add it to our problem's instance. 3) Create a link between the two corresponding nodes for any two nodes of the TSP instance, set the weight of such a link to the corresponding weight of the link in the TSP instance, and add such a link to our problem's instance. Clearly, the reduction can also be done in polynomial time. Therefore, problem 1 does not admit any α-approximation algorithm, where $\alpha > 1$ is a constant.

APPENDIX B
MORE EXPERIMENT RESULTS

TABLE III
RUNNING TIME OF MILP SOLVER WITH INTEL I5-7500 CPU (SEC)

Solver	Devices	FMNIST	CIFAR-10
Prob.1	16	30.205 (1.0x)	14.421(1.0x)
CFP		0.202 (0.007x)	0.195 (0.014x)
TCP		0.043 (0.001x)	0.049 (0.003x)
IHC-FL		**0.245 (0.008x)**	**0.244 (0.017x)**
Prob.1	24	7200+ (1.0x)	3888.64 (1.0x)
CFP		0.34 (0.00005x)	0.326 (0.00008x)
TCP		0.091 (0.00001x)	0.118 (0.00003x)
IHC-FL		**0.431 (0.00006x)**	**0.444 (0.00011x)**
Prob.1	32	7200+ (1.0x)	7200+ (1.0x)
CFP		0.49 (0.00007x)	0.482 (0.00007x)
TCP		0.226 (0.00003x)	0.243 (0.00003x)
IHC-FL		**0.716 (0.0001x)**	**0.725 (0.0001x)**

TABLE IV
IMPROVEMENT IN ACCURACY FOR 32 DEVICES

Method	non-IID	FMNIST (25 rounds)	CIFAR-10 (80 rounds)
FedAvg	$\alpha = 0.5$	74.36% (1.0x)	37.708%(1.0x)
HFL		77.456% (1.042x)	31.708% (0.841x)
CFL		80.104% (1.077x)	43.928% (1.165x)
IHC-FL		**80.507% (1.083x)**	**49.545% (1.314x)**
FedAvg	$\alpha = 0.7$	75.403% (1.0x)	35.864% (1.0x)
HFL		75.729% (1.004x)	33.77% (0.942x)
CFL		78.906% (1.046x)	45.929% (1.281x)
IHC-FL		**79.364% (1.053x)**	**47.183% (1.316x)**

TABLE V
ROUNDS TO ACHIEVE THE TARGET ACCURACY AND TOTAL COMM. COST OVER TIME WHEN 32 DEVICES AND α = 0.5 ON FMNIST DATASET.

Method	Total comm. cost (s)	# rounds to achieve the target acc.	Total comm. cost over time (s)
FedAvg	43 (1.0x)	46 (1.0x)	1978 (1.0x)
HFL	132 (3.07x)	35 (0.76x)	4620 (2.34x)
CFL	155 (3.6x)	25 (0.54x)	3875 (1.96x)
IHC-FL	**43 (1.0x)**	**25 (0.54x)**	**1075 (0.54x)**

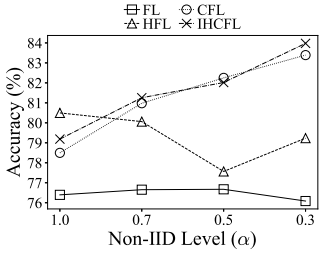

Fig. 8. Effect of different levels of non-IID data on the 24-device accuracy of FMNIST dataset.

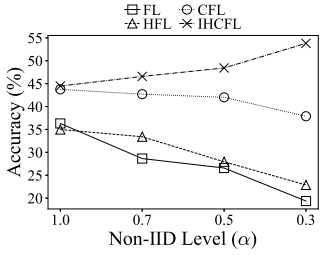

Fig. 9. Effect of different levels of non-IID data on the 24-device accuracy of CIFAR-10 dataset.

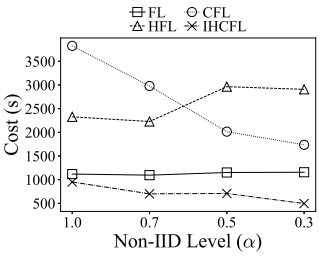

Fig. 10. Effect of different levels of non-IID data on 24-device total cost over time for the FMNIST dataset.

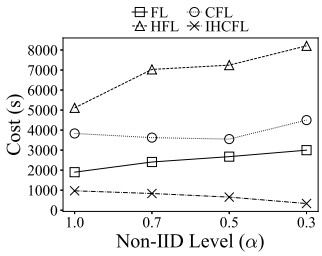

Fig. 11. Effect of different levels of non-IID data on 24-device total cost over time for the CIFAR-10 dataset.