FRACTAL: Data-aware Clustering and Communication Optimization for Decentralized Federated Learning

Qianpiao Ma, Jianchun Liu, Hongli Xu, *Member, IEEE*, Qingmin Jia, Renchao Xie, *Member, IEEE*

***Abstract*—Decentralized federated learning (DFL) is a promising technique to enable distributed machine learning over edge nodes without relying on a centralized parameter server. However, existing DFL network topologies, such as fully connected, partially connected, or lower-tier hierarchical topology often struggle to effectively address the unique challenges presented by edge networks, including edge heterogeneity, communication resource constraint, and data Non-IID. In order to tackle these challenges, we propose a data-aware clustering algorithm, called FRACTAL, to construct a multi-tier hierarchical topology in a bottom-up manner taking into consideration both data distribution and communication efficiency for DFL. We theoretically explore the quantitative relationship between the convergence bound of multi-tier FL and the data distribution among each-tier servers. To further improve communication efficiency and address edge heterogeneity, we deploy a time-sharing communication scheduling algorithm within each fractal unit (the basic structure in FRACTAL consisting of multiple nodes and an aggregator), called magic mirror method (MMM), to determine the optimal order of model distributing and uploading for nodes. We conduct extensive experiments on the classical models and datasets to evaluate the performance of FRACTAL, and the results show that FRACTAL can significantly accelerate the DFL model training by 48.6%-72.3% compared with the state-of-the-art solutions.**

***Index Terms*—Decentralized federated learning, Clustering, Non-IID, Time-sharing Scheduling, Heterogeneity.**

I. INTRODUCTION

INTERNET of Things (IoT) has become a prevalent technology that connects physical devices and generates massive amounts of data every day [1]–[3]. However, sending these data to the remote cloud for processing or training poses several challenges, such as privacy leakage, bandwidth consumption, and latency. Therefore, *edge computing* has emerged as a promising solution that enables local data processing at the network edge, leveraging the computation capabilities of edge nodes [4]. Moreover, edge computing facilitates the implementation of federated learning (FL), which is a distributed machine learning paradigm that allows edge nodes (also called workers) to collaboratively train a global model without sharing their local data [5]–[7].

FL can be categorized into two architectures: centralized and decentralized. In centralized federated learning (CFL) [8] [9], there is a centralized parameter server that coordinates the model training among workers. Each worker performs local updates on its own data and uploads its local model to the parameter server. The parameter server then aggregates the local models into a global model and distributes it back to the workers. However, the parameter server easy becomes system bottleneck in CFL since the it may suffer heavy communication overhead, leading to poor scalability [10]. To overcome the limitation in CFL, decentralized federated learning (DFL) has been proposed to eliminate the need for a parameter server and enable direct model synchronization between neighboring workers. DFL can achieve better scalability and robustness than CFL, , but it also faces more challenges on mobile edge networks:

- **Edge Heterogeneity:** In DFL, each worker need to communicate with each other directly, without the coordination of a central parameter server. This means that they have to cope with the heterogeneity of their neighbors, which may vary in terms of location, data size, computation power, and network quality [11]. This makes the model synchronization in DFL more complex and inefficient than in CFL with a centralized parameter server to coordinate model updating.
- **Communication Resource Constraint:** On one hand, since the absence of the parameter server, DFL typically require more training rounds to converge than CFL. On the other hand, each worker sends model to multiple workers in each round in DFL rather than sending them to a single server. Therefore, DFL usually consumes more communication resource than CFL [12].
- **Non-IID Data:** The data among workers is often non-independent-and-identically-distributed (Non-IID) [8] due to local data collection, which can significantly degrade FL performance [13]. This challenge is more pronounced in DFL, since workers may have less data diversity than the parameter server in CFL, which can obtain model trained from all workers' data.

Network topology describes the organization and communication of workers, affecting the model convergence, transmission, robustness of DFL significantly. Three types of network topologies are commonly used in DFL: fully connected, partially connected, and hierarchical clustering [14].

In fully connected topology [15] [16], each worker ex-

This work was supported in part by National Natural Science Foundation of China (No. 92267301), Jiangning Baijia Lake Plan Program (No.74072203-3).

Q. Ma and Q. Jia are with the Purple Mountain Laboratories, Nanjing, Jiangsu, China, 211111. E-mail: maqianpiao@pmlabs.com.cn, jiaqingmin@pmlabs.com.cn

J. Liu and H. Xu are with the School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui, China, 230027, and also with Suzhou Institute for Advanced Study, University of Science and Technology of China, Suzhou, Jiangsu, China, 215123. E-mail: jcliu17@ustc.edu.cn, xuhongli@ustc.edu.cn

R. Xie is with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China, 100083, and also with the Purple Mountain Laboratories, Nanjing, Jiangsu, China, 211111. E-mail: Renchao_xie@bupt.edu.cn

changes its model with all other workers. This topology possesses robustness against worker or link failures, maintaining network operability despite disruptions. However, the addition of a new worker mandates the creation of links with all other workers, potentially causing substantial communication overhead [16]. To mitigate communication overhead, some DFL solutions employ a partially connected topology for model transmission, *e.g.*, by constructing subgraphs with sparse communication links within the original topology [17] [18]. Nonetheless, sparse topologies may suffer from high variance and difficult convergence issues [19]. An alternative approach to curbing communication overhead is hierarchical clustering topology, wherein workers form clusters with a selected aggregator for each cluster responsible for local model aggregation and external network communication. Nevertheless, current hierarchical FL research encompasses at most two-tier servers. In expansive edge networks with numerous workers, two-tier servers may also not be able to withstand the heavy model transmission, thereby evolving into bottlenecks. Moreover, a less hierarchical topology might struggle to support more fine-grained clustering strategy to address edge heterogeneity.

In this paper, we propose a data-aware clustering mechanism, called FRACTAL[1], to construct a multi-tier hierarchical topology for DFL. As shown in Fig. 1, nodes are hierarchically clustered in a bottom-up manner and an H-tier topology is finally constructed as in Fig. 2. During this process, a node in each cluster is selected to assume the role of aggregator, thereby establishing the fundamental building block called a "fractal unit"[2] within the FRACTAL mechanism. In addition to the communication factor, we also consider the data distribution among the clusters while hierarchically clustering, such that we tend to cluster nodes to approximate an IID-like data distribution among clusters. In order to further improve communication efficiency and address edge heterogeneity, within each fractal unit, we propose a time-sharing communication scheduling algorithm, called magic mirror method (MMM), to determine the order of model distributing and uploading for nodes[3].

The main contributions of this paper are as follows:

- We theoretically explore the quantitative relationship between the convergence bound of multi-tier FL and the data distribution among each-tier servers.
- For the model transmission within one fractal unit, we propose a novel time-sharing communication scheduling algorithm MMM to determine the order of model distributing and uploading. We further prove that MMM can obtain a local optimal schedule.
- Based on both the convergence analysis for multi-tier FL and the time-sharing scheduling strategy within one fractal unit, we design the mechanism FRACTAL to construct a multi-tier hierarchical topology for DFL, by joint considering the data distribution and communication.
- Experimental results on the classical models and datasets show that, by deploying FRACTAL, the DFL model training can be greatly accelerated by 48.6%-72.3% compared with the state-of-the-art solutions.

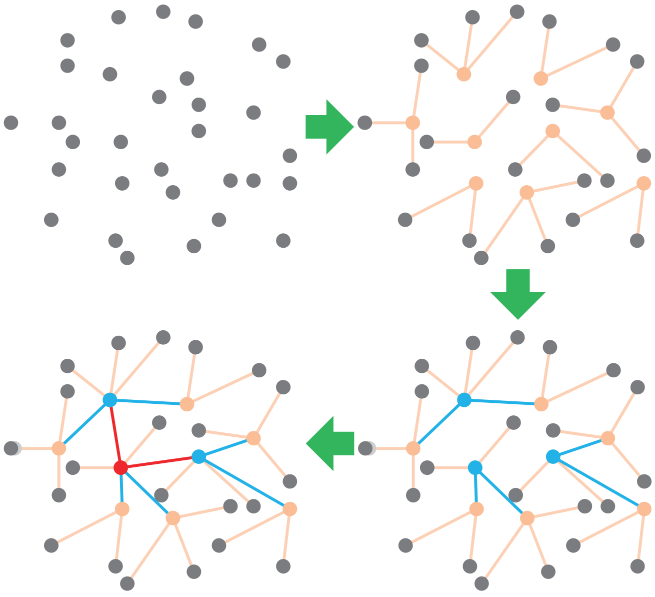


Fig. 1: Topology construction.

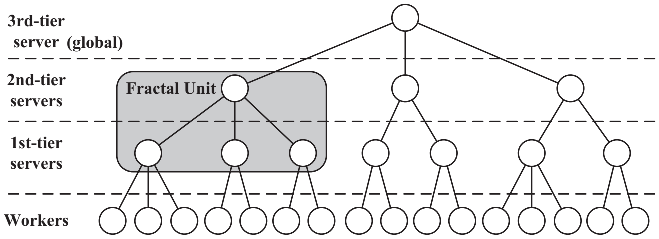


Fig. 2: Three-tier topology example with $N_0 = 16$, $N_1 = 7$, $N_2 = 3$, $N_3 = 1$.

The rest of this paper is organized as follows. Section II reviews the related works. Section III formalizes the multi-tier hierarchical topology in DFL, and Section IV gives the convergence analysis for it. We propose an time-sharing scheduling strategy within a fractal unit in Section V. The clustering algorithm FRACTAL is introduced in Section VI. The Experimental results are shown in Section VII. We conclude this paper in Section VIII.

II. Related Works

A. Network Topology for Decentralized Federated Learning

This section reviews the existing DFL architectures in terms of network topologies, which can be classified into three categories: fully connected, partially connected, and hierarchical clustering topologies [14].

Fully connected topologies [15] [16] allow workers to transmit their models to each other through peer-to-peer (P2P) communication. For instance, [15] proposes a fully connected topology for decentralized federated learning across multiple

[1] In mathematics, a fractal is a geometric shape containing detailed structure at arbitrarily small scales, called self-similarity property. We inspired by this concept and use it to name our multi-tier DFL structures with similar structures.

[2] A fractal unit is a term used in fractal geometry to describe a basic shape or pattern that is repeated at different scales to form a complex structure. We use fractal unit to represent a structure, that is, multiple workers/servers connected to an aggregator.

[3] We collectively refer to a worker or server as a node when the roles of them do not affect semantics.

medical centers without sharing data or relying on a central server. However, this topology requires each worker to share its model with all the others, which incurs significant communication overhead and may compromise the training performance.

Partially connected topologies reduce the communication overhead by enabling each worker to exchange its local model only with its neighboring workers. For example, GossipFL [12] proposes a framework where each client only needs to exchange its model with a single peer at each round to reduce the bandwidth costs. D2D-FL [20] leverages device-to-device communication for model training in DFL to save bandwidth and energy consumption. MATCHA [17] uses matching decomposition to split the original network topology into disjoint subgraphs and communicates over different subgraphs at different frequencies. L2PL [18] introduces a learning-driven method to dynamically construct an optimal partially connected topology at each training epoch. D-Cliques [21] proposes a novel topology that reduces gradient bias by grouping nodes in interconnected cliques such that the local joint distribution in a clique is representative of the global class distribution. However, high-variance problem among workers is particularly pronounced with sparse topologies, potentially hampering model convergence, especially in the presence of Non-IID data [19].

Hierarchical clustering topologies cluster workers into groups and organize them into a hierarchical structure, mainly explored via two aspects: clustering techniques and hierarchical structure. The existing clustering techniques are usually revolve around two considerations: communication and data distribution. For example, in [22]–[26], workers are clustered to the aggregator with the shortest communication time (or distance). On the other hand, [27] concentrate on data distribution, intending to align intra-cluster data distribution with the global while clustering. The organization of clusters consists of two fashions: centralized and decentralized. The centralized approach [22] [27] employs a 2nd-tier global server to aggregate the cluster models. However, this two-tier hierarchical structure still confronts high transmission burdens in large-scale edge networks. In the decentralized approach [24] [25], each cluster aggregator interacts with neighboring aggregators akin to partially connected topologies. However, it also confronts the convergence challenges similar to those in partially connected topologies.

In this paper, we propose a multi-tier hierarchical topology, aiming to ameliorate the limitations in less tier hierarchical topologies. The proposed approach to clustering nodes integrates considerations of data distribution and communication consumption among clusters.

B. Transmission Technology for Federated Learning in Mobile Edge Networks

Due to system bandwidth limitation and wireless interference in mobile edge networks, edge nodes typically communicate with their aggregator (*i.e.*, parameter server) using frequency-sharing (FS) or time-sharing (TS) methods for FL model aggregations. FS communication is employed in [28] [29], where frequency is allocated to multiple nodes for model uploading, effectively enhancing communication efficiency. However, due to edge heterogeneity, individual nodes may possess varying computation capabilities and communication quality with the aggregator. This leads to differences in model training and transmission times. Yet, the frequency assigned for nodes is static, potentially resulting in idle periods and wastage of bandwidth resources [30].

TS communication [30]–[32] allows nodes to take turns using the whole bandwidth resources, which can improve flexibility and adaptability for heterogeneous networks. Consequently, only one node is uploading its model to the aggregator in any time period. Therefore, TS communication requires careful scheduling of nodes to optimize the model updating time. However, most existing TS-based FL researches assume that the downlink transmission rate is much higher than the uplink transmission rate by default, and thus ignore the downlink time when the aggregator distributes the aggregated model. This assumption may not hold for decentralized FL on mobile edge networks, where edge nodes with limited transmission power act as aggregators. Therefore, this paper considers the disparities in model distribution, local training, and model uploading times among heterogeneous edge nodes, and proposes a TS scheduling strategy that minimizes the single round model updating time.

Another approach to improve wireless communication efficiency for FL model aggregation is over-the-air computation (AirComp) [33]–[35]. It is achieved by synchronizing nodes to transmit their local model vectors concurrently, leveraging the superposition property of wireless multiple access channels (MAC) to sum these vectors over-the-air [36]. However, AirComp-based aggregation is prone to noise and channel fading, resulting in inaccurate aggregated model. Moreover, AirComp requires strict synchronization among nodes, which is challenging to achieve on heterogeneous edge networks.

III. System Model and Problem Formulation

In this section, we first introduce the concept of federated learning (Section III-A). Then we describe the multi-tier hierarchical federated learning system (Section III-B). For ease of expression, some key notations are listed in Table I.

A. Federated Learning (FL)

We consider a K-class classification problem, where the label space is denoted as $\mathcal{C} = \{c_1, c_2, ..., c_K\}$. The classification task is performed through federated learning, which involves a set of N_0 workers denoted as $\mathcal{V} = \{v_1, v_2, ..., v_{N_0}\}$.

Each individual worker v_i is responsible for training a local model using a dataset of size d_i. The data belonging to class c_k available on worker v_i is denoted as d_i^k, and it holds that $\sum_{c_k \in \mathcal{C}} d_i^k = d_i$. The total data size across all workers is represented as $D = \sum_{v_i \in \mathcal{V}} d_i$. We define $\alpha_i = d_i/D$ and $\gamma_k = \sum_{v_i \in \mathcal{V}} d_i^k/D$ to represent the proportion of data size belonging to worker v_i and class c_k in the entire dataset, respectively. Considering the classification problem, we adopt

TABLE I: Key Notations.

Symbol	Semantics
$\mathcal{V}$	The set of workers $\{v_1, v_2, ..., v_{N_0}\}$
$\mathcal{C}$	The set of labels $\{c_1, c_2, ..., c_K\}$
j_h	The index of the hth-tier server, satisfying $\sum_{0\le h'\le h-1} N_{h'} + 1 \le j_h \le \sum_{0\le h'\le h} N_{h'}$
$\mathcal{J}_h$	The set of the indexes of the $(h-1)$th-tier nodes in cluster $\mathcal{V}_{j_h}$
v_{j_h}	The node at the hth-tier
$\mathcal{V}_{j_h}$	The set of nodes whose aggregator is v_{j_h}
d_i	The data size on worker v_i
d_i^k	The data size labeled as c_k on worker v_i
D_{j_h}	The total data size of workers in cluster $\mathcal{V}_{j_h}$
$D_{j_h}^k$	The data size labeled as c_k in cluster $\mathcal{V}_{j_h}$
D	The total data size on all workers
ϕ_{j_h}	The proportion of the data size of node v_{j_h} in its cluster
$f_i(\mathbf{w})$	The loss function of worker v_i
$F_j(\mathbf{w})$	The loss function of cluster $\mathcal{V}_j$
$F(\mathbf{w})$	The global loss function
$\mathbf{w}_t$	The global model at round t
$\mathbf{v}_t^i$	The local model of worker v_i at round t
$\tilde{\mathbf{v}}_t^i$	The trained local model of worker v_i at round t
$\mathbf{w}_t^{j_h}$	The model of hth-tier server v_{j_h} at round t
$\tilde{\mathbf{w}}_t^{j_h}$	The aggregated model of hth-tier server v_{j_h} at round t

the widely-used cross-entropy loss function [37] represented as follows:

$$F(\mathbf{w}) \triangleq \sum_{c_k\in\mathcal{C}} -\gamma_k \mathbb{E}_{\mathbf{x}|y=c_k}[\log p_k(\mathbf{x},\mathbf{w})], \tag{1}$$

where $p_k(\mathbf{x},\mathbf{w})$ denotes the predicted probability of input $\mathbf{x}$ belonging to class c_k under the model parameter $\mathbf{w}$. To account for the distributed nature of federated learning, we also define the local loss function for worker v_i as follows:

$$f_i(\mathbf{w}) \triangleq \sum_{c_k\in\mathcal{C}} -\alpha_i^k \mathbb{E}_{\mathbf{x}|y=c_k}[\log p_k(\mathbf{x},\mathbf{w})], \tag{2}$$

where $\alpha_i^k = d_i^k/d_i$ represents the proportion of the data size corresponding to class c_k on worker v_i. It is worth noting that the global loss function can be expressed as the sum of the local loss functions across all workers, weighted by their respective data proportions:

$$F(\mathbf{w}) = \sum_{v_i\in\mathcal{V}} \frac{d_i}{D} f_i(\mathbf{w}) = \sum_{v_i\in\mathcal{V}} \alpha_i f_i(\mathbf{w}). \tag{3}$$

Thus, the learning problem aims to find the optimal parameter vector $\mathbf{w}^*$ that minimizes the global loss function $F(\mathbf{w})$, formally expressed as $\mathbf{w}^* = \arg\min_{\mathbf{w}} F(\mathbf{w})$.

B. Multi-tier Hierarchical Federated Learning System

In this section, we formalize the multi-tier hierarchical federated learning system, and the workflow of the model updating is formally described in Alg. 1.

1) System Architecture: The system consists of N_0 decentralized workers $\mathcal{V} = \{v_1, v_2, ..., v_{N_0}\}$, which form a hierarchical structure with H tiers. These N_0 workers are organized into N_1 clusters $\mathcal{V}_{N_0+1}$, $\mathcal{V}_{N_0+2}$, ..., $\mathcal{V}_{N_0+N_1}$, which are associated with a 1st-tier server v_{N_0+1}, v_{N_0+2}, ..., $v_{N_0+N_1}$, respectively. Each tier server is acted as a worker within its cluster. Moving further, the N_1 1st-tier servers are grouped into N_2 clusters $\mathcal{V}_{N_0+N_1+1}$, $\mathcal{V}_{N_0+N_1+2}$, ..., $\mathcal{V}_{N_0+N_1+N_2}$, which are associated with a 2nd-tier server $v_{N_0+N_1+1}$, $v_{N_0+N_1+2}$, ..., $v_{N_0+N_1+N_2}$, respectively. This hierarchical arrangement continues up to the N_h hth-tier servers, which are organized into N_{h+1} clusters, with each cluster equipped with a $(h+1)$th-tier server. Ultimately, the $(H-1)$th-tier servers are interconnected with a global parameter server. Each server is responsible for receiving and aggregating models from nodes (workers or servers) at a lower tier within its cluster and communicating with its higher tier server.

For convenience, let j_h denote the index of the hth-tier server, satisfying $\sum_{0\le h'\le h-1} N_{h'} + 1 \le j_h \le \sum_{0\le h'\le h} N_{h'}$, and $\mathcal{J}_h$ denote the set of the indexes of $(h-1)$th-tier nodes in cluster $\mathcal{V}_{j_h}$. Specially, $j_0 = i$ denotes the index of a worker, satisfying $1 \le j_0 \le N_0$, and $j_H = \sum_{0\le h\le H} N_h = \sum_{0\le h\le H-1} N_h + 1$ is the index of the global server. The sum of the data size of workers in 1st-tier cluster $\mathcal{V}_{j_1}$ can be calculated as $D_{j_1} = \sum_{i\in\mathcal{J}_1} d_i$, $N_0 + 1 \le j_1 \le N_0 + N_1$. Sequentially, let D_{j_h} denote the total data size of the hth-tier cluster $\mathcal{V}_{j_h}$, satisfying

$$D_{j_h} = \begin{cases} d_i, & h = 0 \\ \sum_{j_{h-1}\in\mathcal{J}_h} D_{j_{h-1}}, & 1 \le h \le H, \end{cases} \tag{4}$$

2) Local Training: Let $\mathbf{v}_t$ denote the model of worker v_i at round t. At each round t, worker v_i performs local model updating based on the previous round's local model $\mathbf{v}_{t-1}$ as

$$\tilde{\mathbf{v}}_t = \mathbf{v}_{t-1} - \eta\nabla f_i(\mathbf{v}_{t-1}, \xi_{t-1}^i), \tag{5}$$

where η denotes the learning rate, ∇ denotes the gradient operator, and ξ_{t-1}^i is a sample uniformly chosen from the local data. Then v_i uploads its local model $\tilde{\mathbf{v}}_t$ to the corresponding 1-st tier server (Lines 3-6).

3) Cluster Aggregation at a Tier Server: Let $\mathbf{w}_t^{j_h}$ denote the model of server v_{j_h} at round t. Upon receiving the model from each node $v_{j_{h-1}}$ within cluster $\mathcal{V}_{j_h}$, the tier server v_{j_h} performs cluster aggregation as

$$\tilde{\mathbf{w}}_t^{j_h} = \sum_{j_{h-1}\in\mathcal{J}_h} \frac{D_{j_{h-1}}}{D_{j_h}} \tilde{\mathbf{w}}_t^{j_{h-1}} = \sum_{j_{h-1}\in\mathcal{J}_h} \phi_{j_{h-1}} \tilde{\mathbf{w}}_t^{j_{h-1}}. \tag{6}$$

where $\phi_{j_{h-1}}$ is the proportion of the data size of node $v_{j_{h-1}}$ in its cluster.

4) Global Aggregation: Upon receiving the cluster models from all $(H-1)$th-tier node, the global server aggregates them as

$$\mathbf{w}_t = \tilde{\mathbf{w}}_t^{j_H} = \sum_{j_{H-1}\in\mathcal{J}_H} \phi_{j_{H-1}} \tilde{\mathbf{w}}_t^{j_{H-1}}. \tag{7}$$

Then the recurrence relation for model updating can be expressed as

$$\tilde{\mathbf{w}}_t^{j_h} = \begin{cases} \mathbf{v}_t, & h = 0 \\ \sum_{j_{h-1}\in\mathcal{J}_h} \phi_{j_{h-1}} \tilde{\mathbf{w}}_t^{j_{h-1}}, & 1 \le h \le H, \end{cases} \tag{8}$$

5) Model Distributing: After global aggregation, the global server distributes the global model $\mathbf{w}_t$ to all nodes in the $(H-$

Algorithm 1 Hierarchical Aggregation Federated Learning

```
1: for $t = 1$ to $T$ do
2:     Processing at Each Worker $v_i$
3:     Train local model $\tilde{\mathbf{v}}_t^i$ by Eq. (5)
4:     Upload $\tilde{\mathbf{v}}_t^i$ to its upper server $v_{j_1}$
5:     Receive $\mathbf{w}_t^{j_1}$ from $v_{j_1}$
6:     $\tilde{\mathbf{v}}_t^i = \mathbf{w}_t^{j_1}$
7:
8:     Processing at Each $h$-th Tier Server $v_{j_h}$
9:     Receive $\tilde{\mathbf{w}}_t^{j_{h-1}}$ from each server $v_{j_{h-1}} \in \mathcal{V}_{j_h}$
10:    Obtain cluster model $\tilde{\mathbf{w}}_t^{j_h}$ by Eq. (6)
11:    Send $\tilde{\mathbf{w}}_t^{j_h}$ to its upper server $v_{j_{h+1}}$
12:    Receive $\mathbf{w}_t^{j_{h+1}}$ from its upper server
13:    $\mathbf{w}_t^{j_h} = \mathbf{w}_t^{j_{h+1}}$
14:    Distribute $\mathbf{w}_t^{j_h}$ to each server $v_{j_{h-1}} \in \mathcal{V}_{j_h}$
15:
16:    Processing at the Global Server
17:    Receive $\tilde{\mathbf{w}}_t^{h_{H-1}}$ from each server $v_{j_{H-1}} \in \mathcal{V}_{j_H}$
18:    Update global model $\mathbf{w}_t$ by Eq. (7)
19:    Distribute $\mathbf{w}_t$ to each server $v_{j_{H-1}} \in \mathcal{V}_H$
```

1)-th tier server (Line 19). Each tier server v_{j_h} updates its cluster model to match the global model, *i.e.*, $\mathbf{w}_t^{j_h} = \mathbf{w}_t$ (Line 13). At last, the global model is sent to all workers as their local models in the new round, *i.e.*, $\mathbf{v}_t^i = \mathbf{w}_t$ (Line 6).

IV. Convergence Analysis

In this section, we first give several general assumptions in federated learning (Section IV-A). Then we obtain the convergence bound through theoretical analysis (Section IV-B).

A. Assumptions

We make the following assumptions on the loss functions $f_i(\mathbf{w}), \forall v_i \in \mathcal{V}$ in Eq. (2) for convergence analysis, which are widely used in the existing literatures [38] [39].

Assumption 1 (Smoothness): *$f_i(\mathbf{w})$ is L-smooth with $L > 0$, i.e., $\forall \mathbf{w}_1, \mathbf{w}_2$, $f_i(\mathbf{w}_2) - f_i(\mathbf{w}_1) \leq \langle \nabla f_i(\mathbf{w}_1), \mathbf{w}_2 - \mathbf{w}_1 \rangle + \frac{L}{2}\|\mathbf{w}_2 - \mathbf{w}_1\|^2$.*

Assumption 2 (Strong convexity): *$f_i(\mathbf{w})$ is μ-strongly convex with $\mu \geq 0$, i.e., $\forall \mathbf{w}_1, \mathbf{w}_2$, $f_i(\mathbf{w}_2) - f_i(\mathbf{w}_1) \geq \langle \nabla f_i(\mathbf{w}_1), \mathbf{w}_2 - \mathbf{w}_1 \rangle + \frac{\mu}{2}\|\mathbf{w}_2 - \mathbf{w}_1\|^2$.*

Note that models with convex loss functions, such as linear regression and support vector machines, satisfy Assumption 2. The evaluation results in Section VII show that our mechanism can also work well for models (*e.g.*, AlexNet) with non-convex loss functions.

Assumption 3 (Variance Bounded): *The variance of stochastic gradients in each device is bounded, i.e., $\forall i, t$, $\mathbb{E}\|\nabla f_i(\mathbf{w}_i^t) - \nabla f_i(\mathbf{w}_i^t, \xi_i^t)\|^2 \leq \sigma_i^2$.*

Assumption 4 (Gradient Bounded): *The expected squared norm of stochastic gradients is uniformly bounded, i.e., $\forall k, \mathbf{w}$, $\mathbb{E}\|G_k(\mathbf{w})\|^2 \leq G^2$, where $G_k(\mathbf{w}) = \nabla \mathbb{E}_{\mathbf{x}|y=c_k}[\log p_k(\mathbf{x}, \mathbf{w})]$.*

Assumption 5 (Existence of global optimal): *Assume that the learning problem has at least one solution $\mathbf{w}^*$ minimizing the global loss function $F(\mathbf{w})$, i.e., $\nabla F(\mathbf{w}^*) = \mathbf{0}$.*

B. Analysis of Convergence Bounds

1) Convergence Analysis of Cluster Aggregation: For ease of expression, we introduce some notations. The earth mover distance (EMD) [40] is applied to represent the difference of data distribution between two datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ as

$$\text{EMD}(\mathcal{D}_1, \mathcal{D}_2) = \sum_{c_k \in \mathcal{C}} \|\frac{D_1^k}{D_1} - \frac{D_2^k}{D_2}\|. \tag{9}$$

We denote Γ_j as the EMD between cluster $\mathcal{V}_{j_h}$'s dataset $\mathcal{D}_j$ and the global dataset $\mathcal{D}$, *i.e.*,

$$\Gamma_j = \text{EMD}(\mathcal{D}, \mathcal{D}_j) = \sum_{c_k \in \mathcal{C}} \|\frac{D^k}{D} - \frac{D_{j_h}^k}{D_{j_h}}\|. \tag{10}$$

Theorem 1 : *$\mathbf{w}_0$ is the initial global model. If $\eta < \frac{1}{L}$, After the global aggregation Eq. (7) is performed T times, the trained global model $\mathbf{w}_T$ satisfies*

$$\begin{aligned}\mathbb{E}[F(\mathbf{w}_T)] - F(\mathbf{w}^*) &\leq (1 - \mu\eta)^T (F(\mathbf{w}_0) - F(\mathbf{w}^*)) \\ &+ \frac{1 - (1 - \mu\eta)^T}{2\mu} (\sum_{1 \leq h \leq H} 2^h \Phi(h) + 2^H \Delta).\end{aligned}$$

where $\Phi(h) = \sum_{j_{H-1} \in \mathcal{J}_H} \phi_{j_{H-1}} \sum_{j_{H-2} \in \mathcal{J}_{H-1}} \phi_{j_{H-2}} \cdots \sum_{j_{h-1} \in \mathcal{J}_h} \phi_{j_{h-1}} \Gamma_{j_{h-1}}$ and $\Delta = \sum_{j_{H-1} \in \mathcal{J}_H} \phi_{j_{H-1}} \sum_{j_{H-2} \in \mathcal{J}_{H-1}} \phi_{j_{H-2}} \cdots \sum_{i \in \mathcal{J}_1} \phi_i \sigma_i^2$.

Proof: According to Eqs. (5) and (6), it holds that

$$\begin{aligned}\mathbf{w}_t = &\sum_{j_{H-1} \in \mathcal{J}^H} \phi_{j_{H-1}} \tilde{\mathbf{w}}_t^{j_{H-1}} \\ = &\sum_{j_{H-1} \in \mathcal{J}_H} \phi_{j_{H-1}} \sum_{j_{H-2} \in \mathcal{J}_{H-1}} \phi_{j_{H-2}} \tilde{\mathbf{w}}_t^{j_{H-2}} \\ = &\sum_{j_{H-1} \in \mathcal{J}_H} \phi_{j_{H-1}} \sum_{j_{H-2} \in \mathcal{J}_{H-1}} \phi_{j_{H-2}} \cdots \sum_{j_1 \in \mathcal{J}_2} \phi_{j_1}^2 \sum_{j_0 \in \mathcal{J}_1} \phi_{j_0}^1 \tilde{\mathbf{w}}_t^{j_0} \\ = &\sum_{j_{H-1} \in \mathcal{J}_H} \phi_{j_{H-1}} \sum_{j_{H-2} \in \mathcal{J}_{H-1}} \phi_{j_{H-2}} \cdots \sum_{j_1 \in \mathcal{J}_2} \phi_{j_1}^2 \sum_{i \in \mathcal{J}_1} \phi_i^1 \tilde{\mathbf{v}}_t^i \\ = &\sum_{j_{H-1} \in \mathcal{J}_H} \phi_{j_{H-1}} \sum_{j_{H-2} \in \mathcal{J}_{H-1}} \phi_{j_{H-2}} \\ &\cdots \sum_{j_1 \in \mathcal{J}_2} \phi_{j_1}^2 \sum_{i \in \mathcal{J}_1} \phi_i^1 (\mathbf{w}_{t-1} - \eta \nabla f_i(\mathbf{w}_{t-1}, \xi_{t-1}^i)) \\ = &\mathbf{w}_{t-1} - \eta \sum_{j_{H-1} \in \mathcal{J}_H} \phi_{j_{H-1}}^H \\ &\cdots \sum_{j_1 \in \mathcal{J}_2} \phi_{j_1}^2 \sum_{i \in \mathcal{J}_1} \phi_i^1 \nabla f_i(\mathbf{w}_{t-1}, \xi_{t-1}^i) \\ = &\mathbf{w}_{t-1} - \eta \mathcal{F}_{j_H}(\mathbf{w}_{t-1})\end{aligned}$$

where

$$\mathcal{F}_{j_h}(\mathbf{w}_t) = \begin{cases} \nabla f_i(\mathbf{w}_t, \xi_t^i), & h = 0 \\ \sum_{j_{h-1} \in \mathcal{J}_h} \phi_{j_{h-1}} \mathcal{F}_{j_{h-1}}(\mathbf{w}_t), & 1 \leq h \leq H, \end{cases} \tag{11}$$

According to Assumption 1, it is obvious that F is L-smooth. If $\eta < \frac{1}{L}$, it follows

$$\begin{aligned}&F(\mathbf{w}_{t+1}) - F(\mathbf{w}^*) \\ \leq &F(\mathbf{w}_t) - F(\mathbf{w}^*) + \langle \nabla F(\mathbf{w}_t), \mathbf{w}_{t+1} - \mathbf{w}_t \rangle \\ &+ \frac{L}{2}\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2\end{aligned}$$

$$
\begin{aligned}
=&F(\mathbf{w}_t) - F(\mathbf{w}^*) - \eta\langle \nabla F(\mathbf{w}_t), \mathcal{F}(H)\rangle + \frac{L\eta^2}{2}\|\mathcal{F}_{j_H}(\mathbf{w}_t)\|^2 \\
\leq&F(\mathbf{w}_t) - F(\mathbf{w}^*) + \frac{\eta}{2}\|\nabla F(\mathbf{w}_t) - \mathcal{F}_{j_H}(\mathbf{w}_t)\|^2 \\
&- \frac{\eta}{2}\|\nabla F(\mathbf{w}_t)\|^2 \qquad (12)
\end{aligned}
$$

By using the AM-GM Inequality and the Jensen's Inequality, we deduce that

$$
\begin{aligned}
&\|\nabla F(\mathbf{w}_t) - \mathcal{F}_{j_H}(\mathbf{w}_t)\|^2 \\
=&\|\nabla F(\mathbf{w}_t) - \sum_{j_{H-1}\in\mathcal{J}_H} \phi^H_{j_{H-1}} \nabla F_{j_{H-1}}(\mathbf{w}_t) \\
&+ \sum_{j_{H-1}\in\mathcal{J}_H} \phi^H_{j_{H-1}} \nabla F_{j_{H-1}}(\mathbf{w}_t) - \mathcal{F}_{j_H}(\mathbf{w}_t)\|^2 \\
\leq&2\|\nabla F(\mathbf{w}_t) - \sum_{j_{H-1}\in\mathcal{J}_H} \phi_{j_{H-1}} \nabla F_{j_{H-1}}(\mathbf{w}_t)\|^2 \\
&+ 2\|\sum_{j_{H-1}\in\mathcal{J}_H} \phi_{j_{H-1}} \nabla F_{j_{H-1}}(\mathbf{w}_t) - \mathcal{F}_{j_H}(\mathbf{w}_t)\|^2 \\
\leq&2\sum_{j_{H-1}\in\mathcal{J}_H} \phi_{j_{H-1}}\|\nabla F(\mathbf{w}_t) - \nabla F_{j_{H-1}}(\mathbf{w}_t)\|^2 \\
&+ 2\sum_{j_{H-1}\in\mathcal{J}_H} \phi_{j_{H-1}}\|\nabla F_{j_{H-1}}(\mathbf{w}_t) - \mathcal{F}_{j_{H-1}}(\mathbf{w}_t)\|^2. \qquad (13)
\end{aligned}
$$

According to Assumption 4, we deduce that

$$
\begin{aligned}
&\mathbb{E}\|\nabla F(\mathbf{w}_t) - \nabla F_{j_{H-1}}(\mathbf{w}_t)\|^2 \\
=&\mathbb{E}\|\sum_{c_k\in\mathcal{C}}(\frac{D^k_{j_{H-1}}}{D_{j_{H-1}}} - \frac{D^k}{D})G_k(\mathbf{w}_t)\|^2 \\
\leq&\Gamma^2_{j_{H-1}}G^2, \qquad (14)
\end{aligned}
$$

Similar to Eqs. (13) and (14), when $h = H-1$, it holds

$$
\begin{aligned}
&\|\nabla F_{j_{H-1}}(\mathbf{w}_t) - \mathcal{F}_{j_{H-1}}(\mathbf{w}_t)\|^2 \\
\leq&2\sum_{j_{H-2}\in\mathcal{J}_{H-1}} \phi_{j_{H-2}}\|\nabla F_{j_{H-1}}(\mathbf{w}_t) - \nabla F_{j_{H-2}}(\mathbf{w}_t)\|^2 \\
&+ 2\sum_{j_{H-2}\in\mathcal{J}_{H-1}} \phi_{j_{H-2}}\|\nabla F_{j_{H-2}}(\mathbf{w}_t) - \mathcal{F}_{j_{H-2}}(\mathbf{w}_t)\|^2. \\
&\qquad (15)
\end{aligned}
$$

and

$$
\mathbb{E}\|\nabla F_{j_{H-1}}(\mathbf{w}_t) - \nabla F_{j_{H-2}}(\mathbf{w}_t)\|^2 \leq \Gamma^2_{j_{H-2}}G^2, \qquad (16)
$$

Iteratively, we deduce the bound of $\|\nabla F_{j_h}(\mathbf{w}_t) - \mathcal{F}_{j_h}(\mathbf{w}_t)\|^2$ for h ranges from H to 1. Specially, when $h = 0$, according to Assumption 3, it holds

$$
\begin{aligned}
\mathbb{E}\|\nabla F_{j_0}(\mathbf{w}_t) - \mathcal{F}_{j_0}(\mathbf{w}_t)\|^2 =&\mathbb{E}\|\nabla f_i(\mathbf{w}_t) - \nabla f_i(\mathbf{w}_t, \xi^i_t)\|^2 \\
\leq&\sigma^2_i. \qquad (17)
\end{aligned}
$$

By taking the bounds of $\|\nabla F_{j_h}(\mathbf{w}_t) - \mathcal{F}_{j_h}(\mathbf{w}_t)\|^2$, $\forall h \in [0, H-1]$ into Eq. (13), we deduce that

$$
\begin{aligned}
&\|\nabla F(\mathbf{w}_t) - \mathcal{F}_{j_H}(\mathbf{w}_t)\|^2 \\
\leq&2\Phi(H) + 4\Phi(H-1) + ... + 2^H\Phi(1) + 2^H\Delta \\
=&\sum_{1\leq h\leq H} 2^h\Phi(h) + 2^H\Delta, \qquad (18)
\end{aligned}
$$

where $\Phi(h) = \sum_{j_{H-1}\in\mathcal{J}_H}\phi_{j_{H-1}}\sum_{j_{H-2}\in\mathcal{J}_{H-1}}\phi_{j_{H-2}}\cdots$ $\sum_{j_{h-1}\in\mathcal{J}_h}\phi_{j_{h-1}}\Gamma^2_{j_{h-1}}$ and $\Delta = \sum_{j_{H-1}\in\mathcal{J}_H}\phi_{j_{H-1}}$ $\sum_{j_{H-2}\in\mathcal{J}_{H-1}}\phi_{j_{H-2}}\cdots\sum_{i\in\mathcal{J}_1}\phi_i\sigma^2_i$. According to Assumption 2, it is obvious that F is μ-strongly convex. Combining with Assumption 5, it follows

$$
\begin{aligned}
\|\nabla F(\mathbf{w}_t)\|^2 =&\|\nabla F(\mathbf{w}_t) - \nabla F(\mathbf{w}^*)\|^2 \\
\geq&2\mu(F(\mathbf{w}_t) - F(\mathbf{w}^*)). \qquad (19)
\end{aligned}
$$

By taking Eqs. (18) and (19) into Eq. (12), we deduce that

$$
\begin{aligned}
&F(\mathbf{w}_{t+1}) - F(\mathbf{w}^*) \\
\leq&(1-\mu\eta)(F(\mathbf{w}_t) - F(\mathbf{w}^*)) + \frac{\eta}{2}(\sum_{1\leq h\leq H} 2^h\Phi(h) + 2^H\Delta) \\
&\qquad (20)
\end{aligned}
$$

By using the recursive relation Eq. (20) from round T to round 0, we obtain that

$$
\begin{aligned}
&F(\mathbf{w}_T) - F(\mathbf{w}^*) \\
\leq&(1-\mu\eta)(F(\mathbf{w}_{T-1}) - F(\mathbf{w}^*)) + \frac{\eta}{2}(\sum_{1\leq h\leq H} 2^h\Phi(h) + 2^H\Delta) \\
\leq&(1-\mu\eta)^2(F(\mathbf{w}_{T-2}) - F(\mathbf{w}^*)) \\
&+ [1 + (1-\mu\eta)]\frac{\eta}{2}(\sum_{1\leq h\leq H} 2^h\Phi(h) + 2^H\Delta) \\
&... \\
\leq&(1-\mu\eta)^T(F(\mathbf{w}_0) - F(\mathbf{w}^*)) \\
&+ \frac{1-(1-\mu\eta)^T}{1-(1-\mu\eta)}\frac{\eta}{2}(\sum_{1\leq h\leq H} 2^h\Phi(h) + 2^H\Delta) \\
=&(1-\mu\eta)^T(F(\mathbf{w}_0) - F(\mathbf{w}^*)) \\
&+ \frac{1-(1-\mu\eta)^T}{2\mu}(\sum_{1\leq h\leq H} 2^h\Phi(h) + 2^H\Delta)
\end{aligned}
$$

Thus, we complete the proof. ■

V. Magic Mirror Method for Time-sharing Communication Scheduling

A. Time-sharing Communication

Recall that a fractal unit is a structure where multiple nodes (workers/servers) connected to an aggregator. We deploy time-sharing strategy [30]–[32] for each fractal unit in FRACTAL, *i.e.*, only one node is uploading or receiving models in any time period. Due to edge heterogeneity, nodes in a fractal may have diverse computation capability and communication quality with the aggregator, which results in different model training and transmission time. Let c_i denote the model training time on worker v_i. a_i and b_i denote the model distributing time and model uploading time between worker v_i and its aggregator, respectively. Since workers' model training can be performed in parallel, different scheduling strategies for model distributing and uploading will result in different completion time of aggregation.

Let $\mathcal{G} = (O^{d_1}_{k_1}, ...O^{d_n}_{k_n}, ..., O^{d_{2N}}_{k_{2N}})$ denote a schedule for model aggregation, where $O^{d_n}_{k_n}$ is the nth operation in schedule $\mathcal{G}$, and sequence k_1, k_2, ..., k_{2N} is a rearrangement of sequence 1, 1, 2, 2..., N, N. If the n^{th} operation $O^{d_n}_{k_n}$ is distributing model to worker v_{k_n}, $d_n = a$, and if it is uploading model from worker v_{k_n}, $d_n = b$.

Note that as long as a schedule is determined, the completion time of model aggregation can be calculated accordingly. Let L_n denote the completion time after executing the n^{th} operation $O^{c_n}_{k_n}$. Initially, we set $L_0 = 0$, and obtain the following recursive relation:

1) If $O^{d_n}_{k_n}$ is a distributing operation, *i.e.*, $d_n = a$, we have

$$
L_n = L_{n-1} + a_{k_n}. \qquad (21)
$$

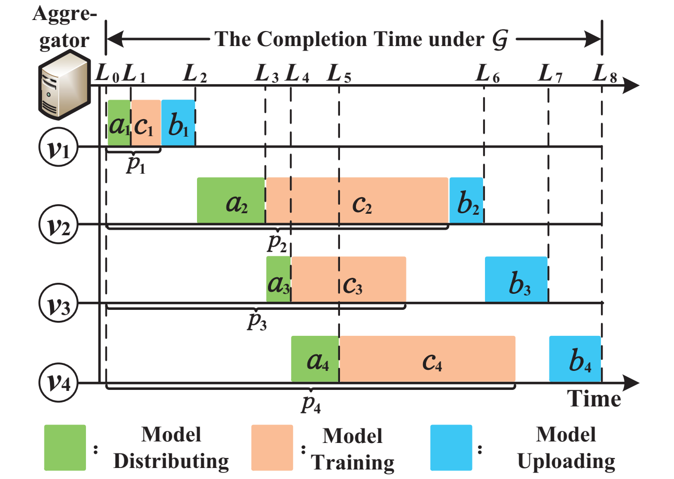


Fig. 3: One schedule $\mathcal{G} = (O_1^a, O_1^b, O_2^a, O_3^a, O_4^a, O_2^b, O_3^b, O_4^b)$.

Let p_i denote the completion time of local training on worker v_i. Therefore, the completion time of local training on worker v_{k_n} can be calculated as

$$p_{k_n} = L_n + c_{k_n} = L_{n-1} + a_{k_n} + c_{k_n}, \quad d_n = a. \tag{22}$$

2) If $O_{k_n}^{d_n}$ is a uploading operation, *i.e.*, $d_n = b$, its start time is equal to the larger of L_{n-1} and p_{k_n}, and its completion time is

$$L_n = \max\{L_{n-1}, p_{k_n}\} + b_{k_n}. \tag{23}$$

We derive the recursive formula as

$$L_n = \begin{cases} 0, & n = 0 \\ L_{n-1} + a_{k_n}, & 1 \le n \le 2N, d_n = a \\ \max\{L_{n-1}, p_{k_n}\} + b_{k_n}, & 1 \le n \le 2N, d_n = b. \end{cases} \tag{24}$$

Finally, the completion time of model aggregation under schedule $\mathcal{G}$ is L_{2N}, and our problem is to determine the schedule $\mathcal{G}$, so as to minimize L_{2N}.

To better illustrate the impact of a schedule on the TS-DU completion time, we give an example with four nodes v_1-v_4 and a aggregator in the system as shown in Fig. 3. The schedule is $\mathcal{G} = (O_1^a, O_1^b, O_2^a, O_3^a, O_4^a, O_2^b, O_3^b, O_4^b)$. For example, the 1^{st} operation O_1^a is distributing model to worker v_1, so its completion time is $L_1 = L_0 + a_1$, and completion time of local training on v_1 is $p_1 = L_0 + a_1 + c_1$. For another example, the 7^{th} operation O_3^b is uploading model from worker v_3. Since the completion time of the 6^{th} operation is larger than that of local training on v_3, *i.e.*, $L_6 > p_3$, the completion time of the 7^{th} operation is $L_7 = \max\{L_6, p_3\} + b_3 = L_6 + b_3$. L_8 is the completion time of model aggregation under schedule $\mathcal{G}$.

B. Magic Mirror Method

We propose a novel algorithm, called the magic mirror method (MMM), to obtain a time-sharing schedule for each fractal unit in our FRACTAL mechanism.

The main idea of the algorithm is to initially generate a distributing sequence and a uploading sequence (Section V-B1), then alternately fix the distributing/uploading sequence and reduce the completion time by reordering uploading/distributing sequence (Sections V-B2, V-B3 and V-B4). Finally, the algorithm converges to obtain a local optimal schedule, and a particle swarm optimization algorithm based on MMM is proposed to increase the probability of obtaining the global optimal (Section V-B5). The relationship between the workflow and the associated theorems is elaborated in Fig. 4.

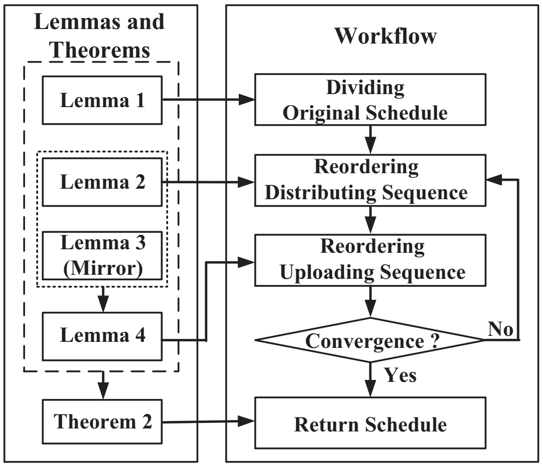


Fig. 4: The relationship between the workflow and the lemmas/theorems.

1) Dividing Original Schedule: In this section, we theoretically prove that there must be an optimal schedule where the first N operations are distributing operations and the last N are uploading operations.

Lemma 1 : *An schedule* $\mathcal{G} = (O_{k_1}^{c_1}, O_{k_2}^{c_2}, ..., O_{k_{2N}}^{c_{2N}})$ *can be transformed into a new schedule* $\hat{\mathcal{G}} = (O_{i_1}^a, O_{i_2}^a, ..., O_{i_N}^a, O_{j_1}^b, O_{j_2}^b, ..., O_{j_N}^b)$, *where the first N operations are distributing operations and the last N are uploading operations, such that the completion time under schedule $\hat{\mathcal{G}}$ is not greater than that under schedule $\mathcal{G}$.*

Proof. Suppose that the n^{th} operation in the schedule $\mathcal{G}$ is a uploading operation, while the $(n+1)^{th}$ operation is a distributing operation, *i.e.*, $O_{k_n}^b, O_{k_{n+1}}^a \in \mathcal{G}$. According to Eq. (24), the completion time of $O_{k_n}^b$ is

$$L_n = \max\{L_{n-1}, p_{k_n}\} + b_{k_n}, \tag{25}$$

then the completion time of $O_{k_{n+1}}^a$ is

$$\begin{aligned} L_{n+1} =& L_n + a_{k_{n+1}} \\ =& \max\{L_{n-1}, p_{k_n}\} + b_{k_n} + a_{k_{n+1}} \\ =& \max\{L_{n-1} + a_{k_{n+1}}, p_{k_n} + a_{k_{n+1}}\} + b_{k_n}. \end{aligned} \tag{26}$$

Now we swap the order of $O_{k_n}^b$ and $O_{k_{n+1}}^a$ in $\mathcal{G}$ to obtain $\hat{\mathcal{G}}$. Then the completion time of the n^{th} operation $O_{k_{n+1}}^a$ in $\hat{\mathcal{G}}$ becomes

$$\hat{L}_n = L_{n-1} + a_{k_{n+1}}, \tag{27}$$

and the completion time of the $(n+1)^{th}$ operation $O_{k_n}^b$ in $\hat{\mathcal{G}}$ becomes

$$\begin{aligned} \hat{L}_{n+1} &= \max\{\hat{L}_n, p_{k_n}\} + b_{k_n} \\ &= \max\{L_{n-1} + a_{k_{n+1}}, p_{k_n}\} + b_{k_n}. \end{aligned} \tag{28}$$

It is obvious that $L_{n+1} \ge \hat{L}_{n+1}$. Similarly, we can deduce that $L_{n+2} \ge \hat{L}_{n+2}$, $L_{n+3} \ge \hat{L}_{n+3}$, ..., $L_{2N} \ge \hat{L}_{2N}$. Thus, the TS-DU completion time does not increase after swapping the order of two adjacent operation, the first of which is a uploading operation and the second of which is a distributing operation.

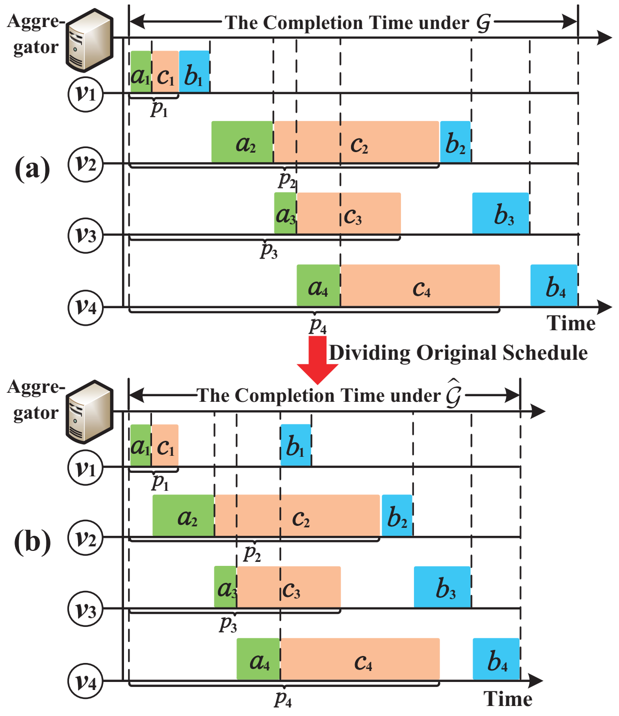


Fig. 5: Dividing original schedule to a distributing and a uploading sequence.

By swapping several times, all N distributing operations can be finally moved to the first N positions of the schedule without completion time increasing. □

We give an example to visualize Lemma 1 as shown in Fig. 5. For a schedule $\mathcal{G} = (O_1^a, O_1^b, O_2^a, O_3^a, O_4^a, O_2^b, O_3^b, O_4^b)$ in Fig. 5(a), the 2^{nd} operation is O_1^b, which is earlier than distributing operations O_2^a, O_3^a and O_4^a. By swapping in Lemma 1, the new strategy becomes $\hat{\mathcal{G}} = (O_1^a, O_2^a, O_3^a, O_4^a, O_1^b, O_2^b, O_3^b, O_4^b)$ as shown in Fig. 5(b). We can visually observe that the completion time under $\hat{\mathcal{G}}$ is smaller then that under $\mathcal{G}$.

By Lemma 1, we reconstruct a schedule where the first N operations form a distributing sequence $O_{i_1}^a, O_{i_2}^a, ...O_{i_N}^a$, and the last N operations form a uploading sequence $O_{j_1}^b, O_{j_2}^b, ...O_{j_N}^b$, where $i_1, i_2, ..., i_N$ and $j_1, j_2, ..., j_N$ are rearrangements of $1, 2, ..., N$. Based on this, the formulas in Eqs. (22) and (24) are re-derived. By setting n from 1 to N in turn, the completion time of local training on v_{i_n} can be calculated according to Eq. (22) as follows

$$p_{i_n} = \sum\nolimits_{m=1}^{n} a_{i_m} + c_{i_n}. \tag{29}$$

Let l_i denote the completion time of v_i's uploading operation O_i^b. From Eq. (24), the completion time of operation $O_{j_n}^b$ is calculated as the recursive formula

$$l_{j_n} = \begin{cases} \sum_{i=1}^{N} a_i, & n = 0 \\ \max\{l_{j_{n-1}}, p_{j_n}\} + b_{j_n}, & 1 \le n \le N. \end{cases} \tag{30}$$

Since $O_{j_N}^b$ is the last operation to be executed, the completion time of TS-DU is $L_{2N} = l_{j_N}$.

2) Reordering Uploading Sequence: To reorder the uploading sequence, we first give the following lemma.

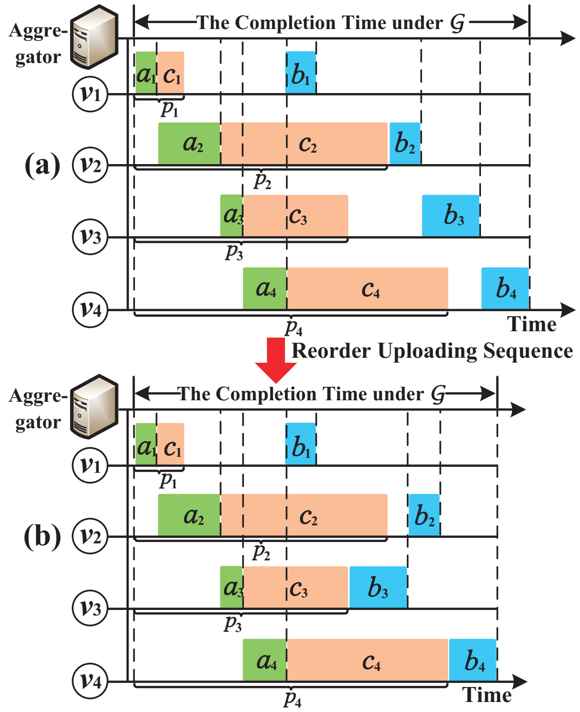


Fig. 6: Reorder Uploading Sequence.

Lemma 2 : *If the distributing sequence is fixed, the TS-DU completion time will be minimized by reordering the uploading sequence as $O_{j_1}^b, O_{j_2}^b, ...O_{j_N}^b$, such that $p_{j_1} \le p_{j_2} \le ... \le p_{j_N}$.*

Proof. Suppose that the completion time of local training on v_{j_n} and $v_{j_{n+1}}$ satisfies $p_{j_n} \ge p_{j_{n+1}}$. From Eq. (30), the completion time of $O_{j_n}^b$ is $l_{j_n} = \max\{l_{j_{n-1}}, p_{j_n}\} + b_{j_n}$. Then the completion time of $O_{j_{n+1}}^b$ is

$$\begin{aligned} l_{j_{n+1}} &= \max\{l_{j_n}, p_{j_{n+1}}\} + b_{j_{n+1}} \\ &= \max\{\max\{l_{j_{n-1}}, p_{j_n}\} + b_{j_n}, p_{j_{n+1}}\} + b_{j_{n+1}} \\ &= \max\{l_{j_{n-1}} + b_{j_n} + b_{j_{n+1}}, p_{j_n} + b_{j_n} + b_{j_{n+1}}, \\ &\qquad p_{j_{n+1}} + b_{j_{n+1}}\}. \end{aligned}$$

The completion time of $O_{j_n}^b$ in the reordered sequence $O_{j_1}^b$, ..., $O_{j_{n+1}}^b$, $O_{j_n}^b$, ..., $O_{j_N}^b$ is

$$\begin{aligned} \hat{l}_{j_n} &= \max\{\hat{l}_{j_{n+1}}, p_{j_n}\} + b_{j_n} \\ &= \max\{\max\{l_{j_{n-1}}, p_{j_{n+1}}\} + b_{j_{n+1}}, p_{j_n}\} + b_{j_n} \\ &= \max\{l_{j_{n-1}} + b_{j_{n+1}} + b_{j_n}, p_{j_{n+1}} + b_{j_{n+1}} + b_{j_n}, p_{j_n} + b_{j_n}\}. \end{aligned}$$

Since $p_{j_n} \ge p_{j_{n+1}}$, it holds $l_{j_{n+1}} \ge \hat{l}_{j_n}$. Then we can easily deduce $l_{j_{n+2}} \ge \hat{l}_{j_{n+2}}$,..., $l_{j_N} \ge \hat{l}_{j_N}$, *i.e.*, $L_{2N} \ge \hat{L}_{2N}$. Thus the TS-DU completion time will not increase after swapping the order of $O_{j_n}^b$ and $O_{j_{n+1}}^b$ in the uploading sequence $O_{j_1}^b, O_{j_2}^b, ..., O_{j_n}^b, O_{j_{n+1}}^b, ..., O_{j_N}^b$. By swapping several times, the uploading sequence can be transformed into $O_{j_1}^b, O_{j_2}^b, ...O_{j_N}^b$, satisfying $p_{j_1} \le p_{j_2} \le ... \le p_{j_N}$, without completion time increasing. □

By Lemma 2, we illustrate that if the distributing sequence $O_{i_1}^a, O_{i_2}^a, ..., O_{i_N}^a$ is fixed, the TS-DU completion time will be minimized by reordering the uploading sequence as $O_{j_1}^b$, $O_{j_2}^b$, ..., $O_{j_N}^b$, such that their prepare time satisfies $p_{j_1} \le p_{j_2} \le ... \le p_{j_N}$. Visually, as shown in Fig. 6(a), the prepare time of O_1^b, O_2^b, O_3^b and O_4^b is $p_1 = a_1 + c_1$, $p_2 = a_1 + a_2 + c_2$,

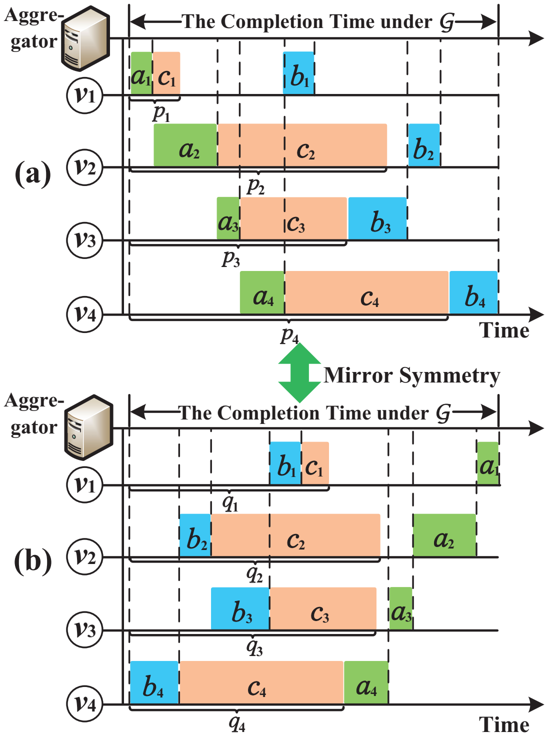


Fig. 7: Symmetry of the mirroring schedule.

$p_3 = a_1 + a_2 + a_3 + c_3$ and $p_4 = a_1 + a_2 + a_3 + a_4 + c_4$, respectively, satisfying $p_1 \le p_3 \le p_2 \le p_4$. Accordingly, the uploading sequence is reordered to $O_1^b, O_3^b, O_2^b, O_4^b$ as shown in Fig. 6(b).

3) The Symmetry of Mirror Schedule: From Lemma 2, it is easy to determine the order of the uploading sequence that minimizes the TS-DU completion time when the distributing sequence is fixed, whereas conversely, it is hard to determine the order of the distributing sequence when the uploading sequence is fixed. We introduce the concept of *mirror schedule* to assist in solving this problem.

Definition 1 : *The mirror schedule is the symmetry of a real schedule, adopting the reversed uploading sequence $O_{j_N}^b$, $O_{j_{N-1}}^b$, ..., $O_{j_1}^b$ as its distributing sequence, and adopting the reversed distributing sequence $O_{i_N}^a$, $O_{i_{N-1}}^a$, ..., $O_{i_1}^a$ as its uploading sequence.*

For example, a real schedule in Fig. 7(a) is with distributing sequence $O_1^a, O_2^a, O_3^a, O_4^a$, and uploading sequence $O_1^b, O_3^b, O_2^b, O_4^b$. Fig. 7(b) shows its corresponding mirror schedule, with distributing sequence $O_4^b, O_2^b, O_3^b, O_1^b$ and uploading sequence $O_4^a, O_3^a, O_2^a, O_1^a$.

It can be observed visually that the TS-DU completion time under a schedule is equal to that under its mirror schedule.

We theoretically prove the symmetry between an arbitrary schedule and its mirror schedule by Lemma 3. For theoretical proof, we first derive the completion time of each operation in the mirror schedule. Similar to the derivation of Eqs. (29) and (30), by setting n from N to 1 in turn, the prepare time of $O_{j_n}^a$ in the mirror schedule is calculated as

$$q_{j_n} = \sum\nolimits_{m=n}^{N} b_{j_m} + c_{j_n}. \tag{31}$$

and the completion time of $O_{i_n}^a$ in mirror schedule is calculated as

$$s_{i_n} = \begin{cases} \sum_{j=1}^{N} b_j, & n = N+1 \\ \max\{s_{i_{n+1}}, q_{i_n}\} + a_{i_n}, & 1 \le n \le N. \end{cases} \tag{32}$$

Since $O_{i_1}^a$ is the last operation in the mirror schedule, its completion time is equal to $\hat{L}_{2N} = s_{i_1}$.

Lemma 3 (Symmetry): *The TS-DU completion time under a schedule is equal to that under its mirror schedule.*

Proof. Recall that we consider a schedule with distributing sequence $O_{i_1}^a$, $O_{i_2}^a$, ..., $O_{i_N}^a$ and the uploading sequence $O_{j_1}^b$, $O_{j_2}^b$,..., $O_{j_N}^b$, where $i_1, i_2, ..., i_N$ and $j_1, j_2, ..., j_N$ are rearrangements of $1, 2, ..., N$. Let $\mathcal{R}(n)$ denote the position of $O_{j_n}^a$ in the distributing sequence, and $\mathcal{R}'(n)$ denote the position of $O_{i_n}^b$ in the uploading sequence. It is obvious that $\mathcal{R}(\mathcal{R}'(n)) = n$ and $\mathcal{R}'(\mathcal{R}(n)) = n$ for $1 \le n \le N$. By Eq. (30), the completion time under the schedule is

$$\begin{aligned} L_{2N} &= l_{j_N} \\ &= \max\{l_{j_{N-1}}, p_{j_N}\} + b_{j_N} \\ &= \max\{l_{j_{N-1}} + b_{j_N}, p_{j_N} + b_{j_N}\} \\ &= \max\{\max\{l_{j_{N-2}}, p_{j_{N-1}}\} + b_{j_{N-1}} + b_{j_N}, p_{j_N} + b_{j_N}\} \\ &= \max\{l_{j_{N-2}} + b_{j_{N-1}} + b_{j_N}, p_{j_{N-1}} + b_{j_{N-1}} + b_{j_N}, \\ &\qquad p_{j_N} + b_{j_N}\} \\ &= \ldots \\ &= \max\{l_{j_0} + \sum_{n=1}^{N} b_{j_n}, p_{j_1} + \sum_{n=1}^{N} b_{j_n}, p_{j_2} + \sum_{n=2}^{N} b_{j_n}, \\ &\quad ..., p_{j_{N-1}} + b_{j_{N-1}} + b_{j_N}, p_{j_N} + b_{j_N}\}. \end{aligned} \tag{33}$$

By Eq. (29), $p_{i_n} = \sum_{m=1}^{n} a_{i_m} + c_{i_n}$ for $n \in [1, N]$. Besides, $l_{j_0} = \sum_{i=1}^{N} a_i$ and $\sum_{n=1}^{N} b_{j_n} = \sum_{i=1}^{N} b_i$, thus we have

$$\begin{aligned} &L_{2N} \\ &= \max\{\sum_{i=1}^{N}(a_i + b_i), p_{j_1} + \sum_{n=1}^{N} b_{j_n}, p_{j_2} + \sum_{n=2}^{N} b_{j_n} \\ &\quad , ..., p_{j_{N-1}} + b_{j_{N-1}} + b_{j_N}, p_{j_N} + b_{j_N}\} \\ &= \max\{\sum_{i=1}^{N}(a_i + b_i), \sum_{m=1}^{\mathcal{R}(1)} a_{i_m} + c_{j_1} + \sum_{m=1}^{N} b_{j_m}, \\ &\quad \sum_{m=1}^{\mathcal{R}(2)} a_{i_m} + c_{j_2} + \sum_{m=2}^{N} b_{j_m}, ..., \sum_{m=1}^{\mathcal{R}(N)} a_{i_m} + c_{j_N} + b_{j_N}\} \\ &= \max\{\sum_{i=1}^{N}(a_i + b_i)\} \cup Q, \end{aligned} \tag{34}$$

where $Q = \{\sum_{m=1}^{\mathcal{R}(n)} a_{i_m} + c_{j_n} + \sum_{m=n}^{N} b_{j_n} | 1 \le n \le N\}$ and L_{2N} is the maximum among $\{\sum_{i=1}^{N}(a_i + b_i)\} \cup Q$. By Eq. (32), the completion time under the **mirror schedule** is

$$\begin{aligned} \hat{L}_{2N} =& s_{i_1} \\ &= \max\{s_{i_2}, q_{i_1}\} + a_{i_1} \\ &= \max\{s_{i_2} + a_{i_1}, q_{i_1} + a_{i_1}\} \\ &= \max\{\max\{s_{i_3}, q_{i_2}\} + a_{i_2} + a_{i_1}, q_{i_1} + a_{i_1}\} \\ &= \max\{s_{i_3} + a_{i_2} + a_{i_1}, q_{i_2} + a_{i_2} + a_{i_1}, q_{i_1} + a_{i_1}\} \\ &= \ldots \end{aligned}$$

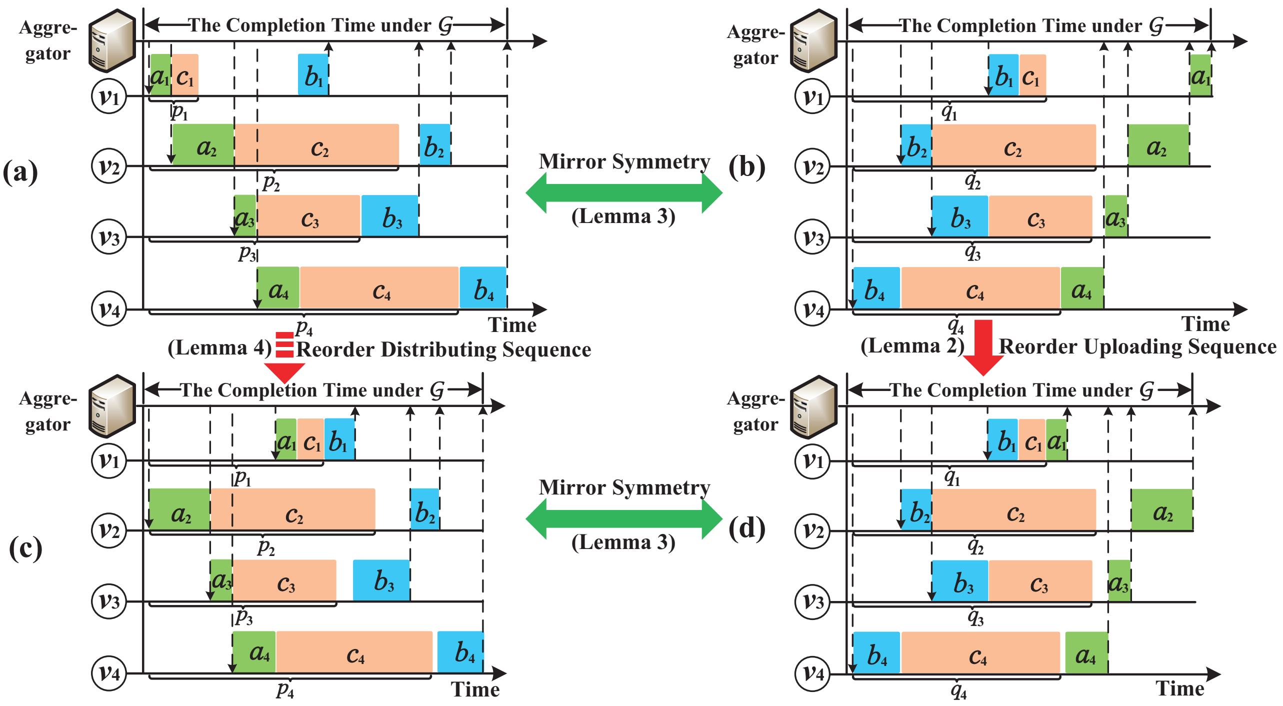


Fig. 8: Reorder Distributing Sequence.

$$
\begin{aligned}
&= \max\{s_{i_{N+1}} + \sum_{n=1}^{N} a_{i_n}, q_{i_N} + \sum_{n=1}^{N} a_{i_n},\\
&\qquad q_{i_{N-1}} + \sum_{n=1}^{N-1} a_{i_n}, ..., q_{i_1} + a_{i_1}\}.
\end{aligned}
$$

By Eq. (31), $q_{j_n} = \sum_{m=n}^{N} b_{j_m} + c_{j_n}$ for $1 \leq n \leq N$. Besides, $s_{i_{N+1}} = \sum_{i=1}^{N} b_i$ and $\sum_{n=1}^{N} a_{i_n} = \sum_{i=1}^{N} a_i$, thus we have

$$
\begin{aligned}
\hat{L}_{2N} &= \max\{\sum_{i=1}^{N}(b_i + a_i), q_{i_N} + \sum_{n=1}^{N} a_{i_n},\\
&\qquad q_{i_{N-1}} + \sum_{n=1}^{N-1} a_{i_n}, ..., q_{i_1} + a_{i_1}\}\\
&= \max\{\sum_{i=1}^{N}(b_i + a_i), \sum_{m=\mathcal{R}'(N)}^{N} b_{j_m} + c_{i_N} + \sum_{m=1}^{N} a_{i_m},\\
&\qquad \sum_{m=\mathcal{R}'(N-1)}^{N} b_{j_m} + c_{i_{N-1}} + \sum_{m=1}^{N-1} a_{i_m},\\
&\qquad ..., \sum_{m=\mathcal{R}'(1)}^{N} b_{j_m} + c_{i_1} + a_{i_1}\}\\
&= \max\{\sum_{i=1}^{N}(b_i + a_i)\} \cup \hat{Q}, \qquad (35)
\end{aligned}
$$

where $\hat{Q} = \{\sum_{m=\mathcal{R}'(n)}^{N} b_{j_m} + c_{i_n} + \sum_{m=1}^{n} a_{i_m} | 1 \leq n \leq N\}$ and $\hat{L}_{2N}$ is the maximum among set $\{\sum_{i=1}^{N}(b_i + a_i)\} \cup \hat{Q}$. We consider an arbitrary element $\hat{Q}_n$ in $\hat{Q}$, it holds

$$
\begin{aligned}
\hat{Q}_n &= \sum_{m=\mathcal{R}'(n)}^{N} b_{j_m} + c_{i_n} + \sum_{m=1}^{n} a_{i_m}\\
&= \sum_{m=1}^{n} a_{i_m} + c_{i_n} + \sum_{m=\mathcal{R}'(n)}^{N} b_{j_m}\\
&= \sum_{m=1}^{\mathcal{R}(\mathcal{R}'(n))} a_{i_m} + c_{i_n} + \sum_{m=\mathcal{R}'(n)}^{N} b_{j_m}. \qquad (36)
\end{aligned}
$$

Since $\mathcal{R}'(n) \in [1, N]$, $\hat{Q}_n$ is also an element in Q. Similarly, we can prove that any element in Q is an element in $\hat{Q}$. Therefore, it holds $Q = \hat{Q}$ and $L_{2N} = \hat{L}_{2N}$, *i.e.*, the TS-DU completion time under a schedule is equal to that under its mirror schedule. □

4) Reordering Distributing Sequence: To reorder distributing sequence, we first give the following lemma.

Lemma 4 : *If the uploading sequence is fixed, the TS-DU completion time will be minimized by reordering the distributing sequence as $O^a_{i_1}, O^a_{i_2}, ...O^a_{i_N}$, such that $q_{i_1} \geq q_{i_2} \geq ... \geq q_{i_N}$.*

Proof. By Lemma 2, if the mirror schedule's distributing sequence $O^b_{j_N}, O^b_{j_{N-1}}, ...O^b_{j_1}$ is fixed, the completion time under the mirror schedule will be minimized by reordering the mirror schedule's uploading sequence as $O^a_{i_N}, O^a_{i_{N-1}}, ...O^a_{i_1}$, such that $q_{i_N} \leq q_{i_{N-1}} \leq ... \leq q_{i_1}$. By Lemma 3, the TS-DU completion time under a schedule is equal to that under its mirror schedule, so the reordering of the mirror schedule's uploading sequence is equal to minimize the TS-DU completion time under the real schedule, *i.e.*, reordering the real schedule's distributing sequence as $O^a_{i_1}, O^a_{i_2}, ...O^a_{i_N}$, such that $q_{i_1} \geq q_{i_2} \geq ... \geq q_{i_N}$. □

Intuitively, as shown in Fig. 8, plot (b) shows the mirror schedule of the real schedule in plot (a). The prepare time of O^a_4, O^a_2, O^a_3 and O^a_1 in the mirror schedule is calculated as $q_4 = b_4 + c_4$, $q_2 = b_4 + b_2 + c_2$, $q_3 = b_4 + b_2 + b_3 + c_3$, and $q_1 = b_4 + b_2 + b_3 + b_1 + c_1$, respectively, satisfying $q_1 \leq q_4 \leq q_3 \leq q_2$. According to Lemma 2, the uploading sequence in the mirror schedule is reordered to $O^a_1, O^a_4, O^a_3, O^a_2$ to minimize the completion time as shown in Fig. 8(d). Meanwhile, this

Algorithm 2 Magic Mirror Method Based Scheduling Strategy (MMM)

```
Input: distributing time $a_i$, training time $c_i$, uploading time $b_i$, $\forall i \in [1, N]$
Output: A schedule $\mathcal{G}$ and its completion time $L^*$
1:  Randomly generate a distributing sequence $O^a_{i_1}$, $O^a_{i_2}$, ..., $O^a_{i_N}$ and a uploading sequence $O^b_{j_1}$, $O^b_{j_2}$,..., $O^b_{j_N}$
2:  $L^* = +\infty$
3:  while true do
4:      Step 1: Reorder the uploading sequence.
5:      for $n \in \{1, 2, ..., N\}$ do
6:          $p_{i_n} = \sum_{m=1}^{n} a_{i_m} + c_{i_n}$
7:      Reorder uploading sequence as $O^b_{j_1}, O^b_{j_2}, ..., O^b_{j_N}$, such that $p_{j_1} \leq p_{j_2} \leq ... \leq p_{j_N}$
8:      Step 2: Reorder the distributing sequence.
9:      for $n \in \{N, N-1, ..., 1\}$ do
10:         $q_{j_n} = \sum_{m=n}^{N} b_{j_m} + c_{j_n}$
11:     Reorder distributing sequence as $O^a_{i_1}, O^a_{i_2}, ..., O^a_{i_N}$, such that $q_{i_1} \geq q_{i_2} \geq ... \geq q_{i_N}$
12:     Step 3: Calculate the TS-DU completion time under the current strategy.
13:     $l_{j_0} = \sum_{i=1}^{N} a_i$
14:     for $n \in \{1, 2, ..., N\}$ do
15:         $l_{j_n} = \max\{l_{j_{n-1}}, p_{j_n}\} + b_{j_n}$
16:     if $l_{j_N} < L^*$ then
17:         $L^* = l_{j_N}$
18:     else
19:         Return schedule $\mathcal{G} = (O^a_{i_1}, O^a_{i_2}, ..., O^a_{i_N}, O^b_{j_1}, O^b_{j_2}, ..., O^b_{j_N})$ and completion time $L^*$
```

is equivalent to minimizing the completion time of TS-DU by reordering the distributing sequence in the real schedule to $O^a_2, O^a_3, O^a_4, O^a_1$ as shown in Fig. 8(c).

Remark 1 : *Note that the mirror schedule is not the real scheduling procedure, but only used for virtual numerical calculations. The key point of MMM is to minimize the TS-DU completion time under the mirror schedule by reordering the its uploading sequence (it is equivalent to reordering the distributing sequence in the real schedule), then the TS-DU completion time under the real schedule will also be magically minimized.*

5) The overall of MMM: We summarize our magic mirror method (MMM) as shown in Alg. 2. We first randomly generate a distributing sequence $O^a_{i_1}$, $O^a_{i_2}$, ..., $O^a_{i_N}$ and a uploading sequence $O^b_{j_1}$, $O^b_{j_2}$,..., $O^b_{j_N}$. Next, we obtain the scheduling by an iterative procedure, where each iteration consists of three steps. In the first step (Lines 4-7), we reorder the uploading sequence as described in Section V-B2. In the second step (Lines 8-11), we reorder the distributing sequence as described in Section V-B4. In the third step (Lines 12-19), we calculate the completion time under the current schedule $l^r_{j_N}$ and judge whether the algorithm has converged. If $l^r_{j_N}$ is equal to the completion time L^* at the last iteration, it means that the algorithm will terminate and the final schedule is returned. Otherwise, L^* is set as l_{j_N} (Lines 16-19).

Theorem 2 : *The proposed MMM algorithm in Alg. 2 can obtain a local optimal schedule, i.e., the TS-DU completion time cannot be reduced by unilaterally reordering the distributing or gathering sequence.*

Proof. Consider an arbitrary schedule with $2N$ coupled operations, we can transform it into a sequence where the first N are distributing operations and the last N are uploading operations. In the iterative procedure, we alternately fix the distributing/uploading sequence, and minimize the completion time by reordering uploading/distributing sequence. In all the above processes, the TS-DU completion time does not increase according to Lemmas 1, 2 and 4, thus we can obtain a local optimal schedule by performing Alg. 2 for finite iterations. □

VI. Data-Aware Clustering and Communication Optimization for Decentralized Federated Learning

Based on both the convergence analysis for multi-tier federated learning in Section IV and the proposed time-sharing communication scheduling strategy in Section V, we define the data-aware clustering and communication optimization for decentralized federated learning.

A. Determine the Number of Tiers

The initial step involves ascertaining the number of tiers H, within the devised topology and the quantification of servers in each respective tier. Specifically, we define the number of servers in the hth-tier to conform to the relation:

$$N_h = \lfloor \sqrt{N_{h-1}} \rfloor, \tag{37}$$

where $h \in [1, H]$. As an illustrative example, when constructing topology for 100 workers, the allocation of servers across the 1st, 2nd, 3rd tiers would amount to 10, 3, and 1 servers, respectively. The square root function helps prevent overloading in the lower tiers while guaranteeing resources required in a given tier does not increase linearly but follows a logarithmic pattern.

Subsequently, we proceed to cluster the nodes (workers or servers) within the range of h from 1 to H. The clustering process for each tier encompasses two steps. The initial step entails partitioning N_{h-1} nodes into N_h clusters, followed by the subsequent selection of a server from each cluster to serve as the aggregator for that particular cluster. However, the effectiveness of the clustering strategy is intertwined with the determination of the aggregator within each cluster. We therefore commence by delineating the approach for aggregator determination within a cluster.

B. Determine the Aggregator for a Fractal Unit by MMM

Consider a cluster $\mathcal{V} = \{v_1, v_2, ...v_N\}$ with N nodes. For the construction of a fractal unit, the assignment of an aggregator becomes imperative. The algorithmic details are presented formally in Alg. 3. Let a^j_i denote the model distributing time from v_j to v_i, and b^j_i represent the model uploading time from v_i to v_j. Additionally, c_i is the model training time on v_i.

Algorithm 3 Determine the Aggregator for Cluster (DA)

```
Input: $\mathcal{V} = \{v_1, v_2, ...v_N\}$, $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$
Output: $j^*, L^*$
1: $L^* = +\infty$
2: for $v_j \in \mathcal{V}$ do
3:     for each $v_i \in \mathcal{V}$ do
4:         $a_i = a_i^j$,$b_i = b_i^j$,$c_i = c_i$
5:     $L = \text{MMM}(a_i, b_i, c_i, \forall i \in [1, N])$(Alg. 2)
6:     if $L < L^*$ then
7:         $j^* = j$
8:         $L^* = L$
9: return the Aggregator $v_{j^*}$ and completion time $L^*$
```

Subsequently, the model distributing/uploading/training time within cluster $\mathcal{V}$ can be succinctly expressed as

$$\mathbf{A} = \begin{pmatrix} a_1^1 & \cdots & a_1^N \\ \vdots & \ddots & \vdots \\ a_N^1 & \cdots & a_N^N \end{pmatrix},\mathbf{B} = \begin{pmatrix} b_1^1 & \cdots & b_1^N \\ \vdots & \ddots & \vdots \\ b_N^1 & \cdots & b_N^N \end{pmatrix},\mathbf{C} = \begin{pmatrix} c_1 \\ \vdots \\ c_N \end{pmatrix},$$

respectively.

The iterative process entails evaluating each $v_j \in \mathcal{V}$ as a potential aggregator for cluster $\mathcal{V}$ (Line 2). Subsequently, the MMM is employed to determine the scheduling strategy and calculate the completion time L via Alg. 2 (Line 5). Following this iterative exploration across all $v_j \in \mathcal{V}$, the aggregator v_{j^*} is ultimately identified as the choice that minimizes the completion time within cluster $\mathcal{V}$.

C. Data-Aware Clustering for Decentralized FL

Let x_{j_{h-1},j_h} denote the indicator denoting whether server $v_{j_{h-1}}$ belongs to cluster $\mathcal{V}_{j_h}$ or not. If v_{j_h} serves as the aggregator of server v_{h_j}, $x_{j_h,j_{h+1}} = 1$; otherwise, $x_{j_j,j_{h+1}} = 0$. Our objective is to determine a clustering strategy denoted by $\boldsymbol{x} = \{x_{h_{j-1},h_j}\}_{h\in[1,H]}$, geared towards enhancing decentralized federated learning efficiency.

Recall Theorem 1, the convergence bound is give by

$$\begin{aligned}\mathbf{F}(\mathbf{w}_0, T) =&(1 - \mu\eta)^T(F(\mathbf{w}_0) - F(\mathbf{w}^*)) \\ &+ \frac{1 - (1 - \mu\eta)^T}{2\mu}(\sum_{1\leq h\leq H} 2^h\Phi(h) + 2^H\Delta),\end{aligned} \tag{38}$$

where Δ is a constant related to the gradients within workers. $\Phi(h) = \sum_{j_{H-1}\in\mathcal{J}_H} \phi_{j_{H-1}} \sum_{j_{H-2}\in\mathcal{J}_{H-1}} \phi_{j_{H-2}} \cdots \sum_{j_{h-1}\in\mathcal{J}_h} \phi_{j_{h-1}}\Gamma_{j_{h-1}}$ relates to the EMD $\Gamma_{j_{h-1}}$, signifying the data distributing of the $(h-1)$th-tier nodes. The subsequent corollary can be intuitively deduced:

Corollary 1 : *The degree of data Non-IID among clusters positively impacts $\Gamma_{j_{h-1}}$ for cluster $\mathcal{V}_{j_h}$, as well as $\Phi(h)$ and the convergence bound $\mathbf{F}(\mathbf{w}_0, T)$. For clusters within the hth-tier adhering to IID data, $\Gamma_{j_{h-1}} = 0$ and $\Phi(h) = 0$, leading to a reduction in the convergence bound $\mathbf{F}(\mathbf{w}_0, T)$.*

Corollary 1 shows the potential to cluster nodes to approximate an IID-like data distribution among clusters, thereby curbing the convergence bound $\mathbf{F}(\mathbf{w}_0, T)$ and enhancing training efficacy. Motivated by this insight, we propose a data-aware clustering algorithm that applies this principle at each tier of clustering. The details are formally described in Alg. 4.

Algorithm 4 Muti-tier Data-Aware Clustering Algorithm for Decentralized Federated Learning (FRACTAL)

```
Input: Data size $D_i$, $D_i^k$, $\forall v_i \in \mathcal{V}$, $\forall c_k \in \mathcal{C}$; completion time
    threshold $L_j^{max}$, $\forall s_j \in \mathcal{S}$
Output: Final clustering strategy $\boldsymbol{x}$
1: for $h \in [1, h-1]$ do
2:     $\Phi_{temp} = +\infty$
3:     for $j_{h-1} \in [\sum_{0\leq h'\leq h-2} N_{h'} + 1, \sum_{0\leq h'\leq h-1} N_{h'}]$ do
4:         for $j_h \in [\sum_{0\leq h'\leq h-1} N_{h'} + 1, \sum_{0\leq h'\leq h} N_{h'}]$ do
5:             $x_{j_{h-1},j_h} = 0$
6:     for $j_{h-1} \in [\sum_{0\leq h'\leq h-2} N_{h'} + 1, \sum_{0\leq h'\leq h-1} N_{h'}]$ do
7:         for $j_h \in [\sum_{0\leq h'\leq h-1} N_{h'} + 1, \sum_{0\leq h'\leq h} N_{h'}]$ do
8:             $x_{j_{h-1},j_h} = 1$
9:             if $\text{DA}(\mathcal{V}_{j_h}, \mathbf{A}_{j_h}, \mathbf{B}_{j_h}, \mathbf{C}_{j_h}) > L_{j_h}^{max}$ then
10:                //Calculate the completion time by Alg. 3
11:                $x_{j_{h-1},j_h} = 0$
12:                continue
13:            if $\Phi(h) < \Phi_{temp}$ then
14:                $\Phi_{temp} = \Phi(h)$
15:                $j_h^* = j_h$
16:            $x_{j_{h-1},j_h} = 0$
17:        $x_{j_{h-1},j_h^*} = 1$
18:    for $j_h \in [\sum_{0\leq h'\leq h-1} N_{h'} + 1, \sum_{0\leq h'\leq h} N_{h'}]$ do
19:        $j^*, L^* = \text{DA}(\mathcal{V}_{j_h}, \mathbf{A}_{j_h}, \mathbf{B}_{j_h}, \mathbf{C}_{j_h})$
20:        //Select $v_{j^*}$ as the aggregator of $\mathcal{V}_{j_h}$ by Alg. 3
21:        $v_{j_h} = v_{j^*}$
22:        //Update $\mathcal{V}_{j_h}$'s completion time by Alg. 3
23:        $c_{j_h} = L^*$
24: return final clustering strategy $\boldsymbol{x}$
```

For every tier $h \in [1, H-1]$, we greedily determine the allocation of each node to clusters one by one, so as to minimize the current $\Phi(h)$. The process is initiated by configuring the states of the $(h-1)$th-tier nodes, with $x_{j_{h-1},j_h} = 0$ (Lines 3-5). Subsequently, an attempt is made to assign $v_{j_{h-1}}$ to each hth-tier cluster, *i.e.*, setting $x_{j_{h-1},j_h} = 1$, and subsequently calculating the current $\Phi(h)$ value. In detail, let $\mathbf{A}_{j_h}$, $\mathbf{B}_{j_h}$ and $\mathbf{C}_{j_h}$ denote the model distributing/uploading/training time within cluster $\mathcal{V}_{j_h}$, respectively. On one hand, if the completion time within cluster $\mathcal{V}_{j_h}$, *i.e.*, $\text{DA}(\mathcal{V}_{j_h}, \mathbf{A}_{j_h}, \mathbf{B}_{j_h}, \mathbf{C}_{j_h})$, surpasses the threshold $L_{j_h}^{max}$, then node $v_{j_{h-1}}$ remains unallocated to cluster $\mathcal{V}_{j_h}$, *i.e.*, resetting $x_{i,j} = 0$ (Lines 9-12). On the other hand, after traversing all clusters within the hth-tier, node $v_{j_{h-1}}$ is assigned to the cluster $\mathcal{V}_{j_h^*}$ that minimizes the $\Phi(h)$ (Line 13-17). Once all $(h-1)$th-tier nodes are clustered, the aggregator of each cluster is accordingly determined, and the completion time on v_{j_h} is set as the completion time of cluster $\mathcal{V}_{j_h}$ by Alg. 3,*i.e.*, $c_{j_h} = L^*$ (Lines 18-23). Subsequently, the node clustering process for the hth-tier commences. The algorithm terminates upon the H-tier topology is constructed.

VII. Performance Evaluation

A. System Setup

We implement a large-scale decentralized federated learning system using PySyft (version 0.2.9) [41], a Python library for privacy-preserving deep learning under the PyTorch framework. PySyft allows the virtual worker creation for FL training, where each worker simulates an individual machine and trains a local model on its own dataset. We simulate a typical edge computing system with 100 workers randomly deployed in a 50m×50m region. Our experimental evaluations are conducted on an AMAX deep learning workstation with an 8-core Intel Xeon CPU (E5-2620v4) and 4 NVIDIA GeForce RTX 2080Ti GPUs with 11GB GDDR6. The system environment is Ubuntu 18.04, CUDA v10.0, and cuDNN v7.5.0.

1) Simulation of Edge Heterogeneity: We introduce a scaling factor κ_i for each worker v_i, which is a random float number drawn uniformly from [1,10], to simulate the edge heterogeneity. $\hat{c}_i$ denote the actual local training time on worker v_i. Since we deploy the virtual workers v_1-v_{100} on the same servers for experimentation, their local training times are roughly equal, *i.e.*, $\hat{c}_1 \approx \hat{c}_2 \approx ... \approx \hat{c}_{100}$. Then, we set the local training time on worker v_i as $c_i = \kappa_i \hat{c}_i$, which means that after completing local training, v_i idles for a period between 0 and 9 times of $\hat{c}_i$.

For the model delivering among workers, the transmission rate from v_i to v_j can be given by the Shannon capacity [42], $r_{i,j} = W_{i,j} \log_2(1+\frac{p_i^u h_{i,j}}{\sigma^2})$, where p_i denotes the transmission power of worker v_i, set as 100mWatts [43], and σ^2 denotes the noise power, set as -100dBm [43]. $W_{i,j}$ represents the bandwidth assign for model delivering from v_i to v_j. $W_i = \sum_{v_j \in \mathcal{V}}(W_{i,j} + W_{j,i})$ is the total available bandwidth at each worker v_i, set as 10 MHz. $h_{i,j}$ denotes the channel gain from worker v_i to v_j, modeled by the pass-loss $h_{i,j} = h_0 d_{i,j}^{-4}$ model [44], where $h_0 = -40$dB [42] is the path-loss constant and $d_{i,j}$ is the distance between worker v_i and aggregator s_j. Let ξ denote the model size. Consequently, the model transmission time from worker v_i to aggregator s_j can be calculated by $L_{i,j} = \frac{\xi}{r_{i,j}}$.

2) Models and Datasets: We train two classical models (*i.e.*, LR [45], AlexNet [46]) and on datasets MNIST [47] and CIFAR-10 [48]. MNIST consists of 60,000 handwritten digits for training and 10,000 for testing, and owns ten types of labels from "1" to "9", while CIFAR-10 includes 50,000 images for training and 10,000 for testing, and has ten different types of objects. We adopt the same mini-batch size (*i.e.*, 64) during FL training.

To implement the Non-IID data among workers, we adopt the label skewed method to partition dataset [49] [50]. Specifically, the data in MNIST labeled as '0' are distributed to workers v_1-v_{10}, the data labeled as '1' are distributed to workers v_{11}-v_{20},..., and the data labeled as '9' are distributed to workers v_{91}-v_{100}.

3) Benchmarks and Performance Metrics: We compare our **Fractal** algorithm against three other typical FL solutions.

- **FedAvg [8]:** The classic CFL algorithm that all workers participating in each round of global aggregation. We assume that the most centered worker acts as the parameter server in our simulation environment.

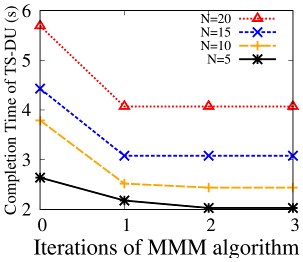


Fig. 9: Completion Time of a single Round vs. Iterations.

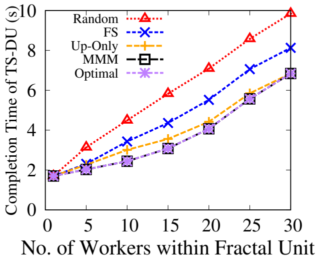


Fig. 10: Comparison with other communication solutions.

- **MATCHA [17]:** A DFL solution with partially connected topology, where the original network topology is split into disjoint subgraphs and communicates on different subgraphs in different training rounds.
- **TiFL [26]:** A DFL solution with two-tier hierarchical topology, where each worker is clustered to the aggregator with the shortest communication time.

For one round model updating within a fractal unit, we first compare our **MMM** algorithm with other communication solutions:

- **Frequency-sharing (FS)** [28] [29]: Each worker is allocated with static bandwidth and the bandwidth allocation remains unchanged during the model updating process.
- **Random**: A time-sharing scheduling algorithm where the models are distributed/uploaded in random order.
- **Up-Only** [30]: A time-sharing scheduling algorithm, where the distributing sequence is in random order, and the uploading sequence is sorted as Step 1 in Alg. 2.
- **Optimal**: The TS-DU process is formalized as a nonlinear integer programming problem, and can be solving to obtain a optimal schedule by CVXPY package [51].

To evaluate the training performance, we adopt three performance metrics. 1) *Loss Function* reflects the training process of the model and whether convergence has been achieved. 2) *Accuracy* is the most common performance metric in classification problems, which is defined as the proportion of right data classified by the model to all test data. 3) *Training Time* is adopted to measure the training speed.

B. Evaluation Results

1) Comparison of Communication Solutions: Fig. 9 shows the change of the completion time of one TS-DU process in the iterative procedure of our MMM algorithm. As is shown, MMM can converge in two iterations for different number N (from 5 to 20) of nodes within a fractal unit.

Fig. 10 shows the completion time of one TS-DU process of different solutions FS, Random, Up-Only, MMM and Optimal when the number of nodes within a fractal unit increases from 1 to 30. We test 1000 times and average to smooth the curve. As shown, our MMM algorithm can always achieve almost the same completion time as Optimal. When the number of nodes is between 5 and 15, MMM can greatly reduce the completion time compared with Random, FS and Up-Only.

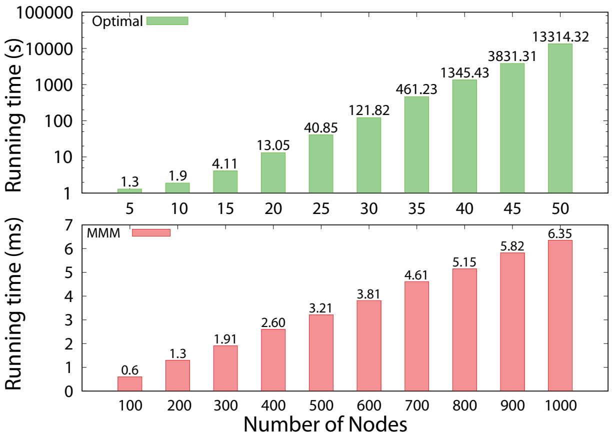


Fig. 11: Symmetry of the mirroring schedule.

For example, when N=10, the completion time of Random, FS, Up-Only, MMM and Optimal is 4.49s, 3.43s, 3.01s, 2.44s and 2.44s, respectively. MMM can reduce the completion time by about 45.7%, 28.9% and 18.9% compared with Random, FS and Up-Only. Whereas, when the number of node exceeds 20, the performance of Up-Only is close to that of MMM in practice. This is because the more nodes in the cluster, the better the parallelism of model training of MMM and Up-Only, and the completion time of both algorithms will tend to the sum of all nodes' model distributing and uploading time. For example, when X=30, the intra-cluster aggregation time of Random, FS, Up-Only, MMM and Optimal is 9.85s, 8.13s, 6.84s, 6.84s and 6.84s, respectively. MMM and Up-Only can reduce the completion time by about 16.1% compared with FS. The average gap between the obtained local optimal strategy and the global optimal is less than 0.1%. Furthermore, we observed that among the results of 1000 tests, 912 propose strategies by MMM are also global optimal.

Fig. 11 shows the running time of our MMM algorithm when the number of nodes in a fractal unit ranges from 100 to 1000. We observe that even in a fractal unit with amounts of nodes, MMM can still obtain scheduling strategy in milliseconds, and the algorithm running time increases evenly with the increase of the number of nodes in the fractal unit. For example, when there are 1000 nodes in the system, the scheduling strategy can be obtained in 6.35ms by MMM. As a comparison, we evaluate the running time of Optimal when the number of nodes ranges from 5 to 50. We observe that the algorithm running time of Optimal increases dramatically with the increase of the number of nodes. For example, even only 50 nodes in the system, the running time of Optimal has reached an unacceptable 13314.32s.

The experimental results in Fig. 9-11 show that our MMM algorithm can achieve near-optimal performance in a short and linear time and is practical for the topology construction in FRACTAL.

2) The Chosen of Learning Rate: In this section, we train AlexNet on MNIST use our FRACTAL to obtain the suitable learning rate η. Fig. 12 shows the effect of different learning rates on the training performance. A low learning rate of 0.001 leads to slow convergence, reaching only 72.1% accuracy after 5000s of training. A moderate learning rate of 0.01 results in faster and more stable convergence, achieving 91.3% accuracy after the same training time. However, a high learning rate of 0.1 causes the loss and accuracy curves to fluctuate wildly and prevents the model from converging. The accuracy curve oscillates between 78.5% and 92.1% after 5000s of training. Therefore, we choose $\eta = 0.01$ as the learning rate for the subsequent experiment.

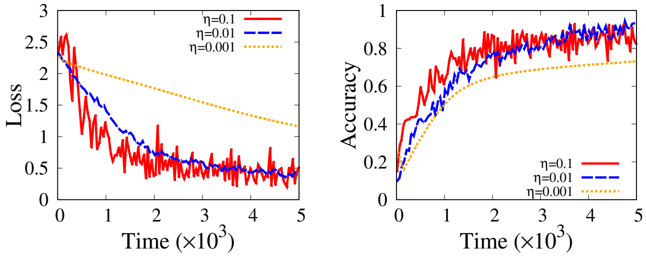


Fig. 12: The Comparison of Different Learning Rates (AlexNet on MNIST). *Left*: Loss; *Right*: Accuracy.

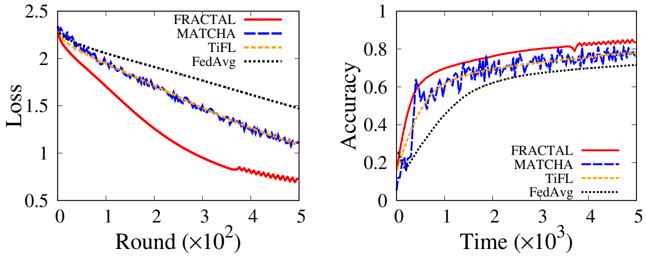


Fig. 13: Loss/Accuracy vs. Time (LR on MNIST). *Left*: Loss; *Right*: Accuracy.

3) Comparison of Federated Learning Solutions: In this section, we compare our FRACTAL with other typical FL solutions: FedAvg, MATCHA and TiFL. For model updates within the fractal units, we employ the MMM algorithm in our FRACTAL, while deploying the most commonly used FS communication for other benchmarks.

TABLE II: The impact of the clustering methods on EMD

Clustering methods	Original	TiFL	FRACTAL	
	$\overline{\Gamma}_{(0)}$	$\overline{\Gamma}_{(1)}$	$\overline{\Gamma}_{(1)}$	$\overline{\Gamma}_{(2)}$
EMD	1.8	0.69	0.19	0.102

Table II shows the average EMD of nodes on different tiers after applying TiFL and FRACTAL clustering methods. We use $\overline{\Gamma}_{(j)}$ to denote the average EMD of nodes on the j^{th} tier. As shown in Table II, the EMD of the 0^{th} tier is $\overline{\Gamma}_{(0)} = |\frac{1}{10} - 1| + |\frac{1}{10} - 0| \times 9 = 1.8$, since each worker has data with the same label. After the clustering the TiFL, the EMD of the 1^{th} tier is reduced to 0.69. However, by applying our proposed FRACTAL, the EMD of the 1^{th} tier is further reduced to 0.19, and that of the 2^{th} tier is only 0.102. This demonstrates that FRACTAL can achieve a more balanced data distribution among clusters on each tier, which is closer to IID.

Figs. 13-15 show that FRACTAL can greatly accelerate FL at decentralized edge network compare with other solutions. For example, in Fig. 13, after 5000s of training, FRACTAL attains a stable training accuracy of 83.5%, outperforming MATCHA, TiFL, and FedAvg, which reach 73.3%, 78.1%, and 71.6%, respectively. The training time to achieve 70%

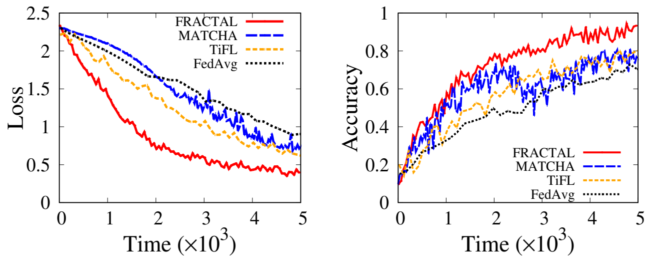


Fig. 14: Loss/Accuracy vs. Time (AlexNet on MNIST). *Left*: Loss; *Right*: Accuracy.

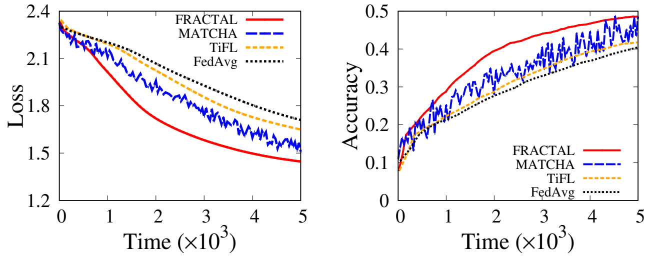


Fig. 15: Loss/Accuracy vs. Time (AlexNet on CIFAR-10). *Left*: Loss; *Right*: Accuracy.

accuracy is 1140s, 2733s, 2220s and 4112s for FRACTAL, MATCHA, TiFL, and FedAvg, respectively. FRACTAL can reduce the training time by about 58.3%, 48.6% and 72.3% compared with MATCHA, TiFL and FedAvg, respectively. The superiority of FRACTAL over TiFL and FedAvg arises from its utilization of a multi-tier topology, distributing communication load across more nodes and consequently accelerating the training process. Additionally, note that during the training process, the loss and accuracy curves of MATCHA jitter more violently compared with that of other solutions. This volatility is attributed to the high-variance problem among workers, particularly prominent in sparse topologies for MATCHA, which is aggravated by non-IID data distribution. In contrast, FRACTAL mitigates this problem by constructing hierarchical topology while promoting data distribution among clusters close to IID, resulting in improved stability during training.

VIII. Conclusion

In this paper, we have proposed a novel data-aware clustering algorithm, called FRACTAL, to construct a multi-tier hierarchical topology for decentralized federated learning. We have theoretically explored the quantitative relationship between the convergence bound of multi-tier FL and the data distribution among each-tier servers. To address edge heterogeneity and communication resource constraint, within each fractal unit, we have proposed a time-sharing communication scheduling algorithm, called magic mirror method (MMM), to determine the order for model distributing and uploading, and proved its convergence. The extensive experimental results show that FRACTAL can significantly accelerate the DFL model training compared with the state-of-the-art solutions.

References

[1] A. Pantelopoulos and N. G. Bourbakis, "A survey on wearable sensor-based systems for health monitoring and prognosis," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 1, pp. 1–12, 2009.

[2] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones." in *Esann*, 2013.

[3] G. Zhu, D. Liu, Y. Du, C. You, J. Zhang, and K. Huang, "Towards an intelligent edge: Wireless communication meets machine learning," *arXiv preprint arXiv:1809.00343*, 2018.

[4] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[5] H. B. McMahan and D. Ramage., "http://www.googblogs.com/federated-learning-collaborative-machine-learning-without-centralized-training-data/," Google, 2017.

[6] X. Wei, Q. Li, Y. Liu, H. Yu, T. Chen, and Q. Yang, "Multi-agent visualization for explaining federated learning," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 2019, pp. 6572–6574.

[7] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–7.

[8] B. McMahan, E. Moore, D. Ramage, S. Hampson, and A. y. B. Arcas, "Communication-efficient learning of deep networks from decentralized data," *AISTATS*, pp. 1273–1282, 2017.

[9] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*, 2014, pp. 583–598.

[10] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[11] Y. Liao, Y. Xu, H. Xu, L. Wang, and C. Qian, "Adaptive configuration for heterogeneous participants in decentralized federated learning," in *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 2023, pp. 1–10.

[12] Z. Tang, S. Shi, B. Li, and X. Chu, "Gossipfl: A decentralized federated learning framework with sparsified and adaptive communication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 3, pp. 909–922, 2022.

[13] M. Chen, Y. Xu, H. Xu, and L. Huang, "Enhancing decentralized federated learning for non-iid data on heterogeneous devices," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2023, pp. 2289–2302.

[14] E. T. M. Beltrán, M. Q. Pérez, P. M. S. Sánchez, S. L. Bernal, G. Bovet, M. G. Pérez, G. M. Pérez, and A. H. Celdrán, "Decentralized federated learning: Fundamentals, state-of-the-art, frameworks, trends, and challenges," *arXiv preprint arXiv:2211.08413*, 2022.

[15] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger, "Braintorrent: A peer-to-peer environment for decentralized federated learning," *arXiv preprint arXiv:1905.06731*, 2019.

[16] J. Wang and G. Joshi, "Adaptive communication strategies to achieve the best error-runtime trade-off in local-update sgd," *arXiv: Learning*, 2018.

[17] J. Wang, A. K. Sahu, Z. Yang, G. Joshi, and S. Kar, "MATCHA: Speeding Up Decentralized SGD via Matching Decomposition Sampling," in *2019 Sixth Indian Control Conference (ICC)*, 2019, pp. 299–300.

[18] H. Xu, M. Chen, Z. Meng, Y. Xu, L. Wang, and C. Qiao, "Decentralized machine learning through experience-driven method in edge networks," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 2, pp. 515–531, 2022.

[19] J.-w. Lee, J. Oh, S. Lim, S.-Y. Yun, and J.-G. Lee, "Tornadoaggregate: Accurate and scalable federated learning via the ring-based architecture," *arXiv preprint arXiv:2012.03214*, 2020.

[20] Y. Li, W. Liang, J. Li, X. Cheng, D. Yu, A. Y. Zomaya, and S. Guo, "Energy-aware, device-to-device assisted federated learning in edge computing," *IEEE Transactions on Parallel and Distributed Systems*, 2023.

[21] A. Bellet, A.-M. Kermarrec, and E. Lavoie, "D-cliques: Compensating for data heterogeneity with topology in decentralized federated learning," in *2022 41st International Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2022, pp. 1–11.

[22] M. S. H. Abad, E. Ozfatura, D. Gunduz, and O. Ercetin, "Hierarchical federated learning across heterogeneous cellular networks," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 8866–8870.

[23] Z. Wang, H. Xu, J. Liu, H. Huang, C. Qiao, and Y. Zhao, "Resource-efficient federated learning with hierarchical aggregation in edge computing," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[24] Z. Zhong, Y. Zhou, D. Wu, X. Chen, M. Chen, C. Li, and Q. Z. Sheng, "P-FedAvg: Parallelizing Federated Learning with Theoretical Guarantees," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[25] Y. Sun, J. Shao, Y. Mao, J. H. Wang, and J. Zhang, "Semi-decentralized federated edge learning for fast convergence on non-iid data," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2022, pp. 1898–1903.

[26] Z. Chai, A. Ali, S. Zawad, S. Truex, A. Anwar, N. Baracaldo, Y. Zhou, H. Ludwig, F. Yan, and Y. Cheng, "TiFL: A Tier-based Federated Learning System," in *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing*, 2020, pp. 125–136.

[27] N. Mhaisen, A. Awad, A. Mohamed, A. Erbad, and M. Guizani, "Optimal user-edge assignment in hierarchical federated learning based on statistical properties and network topology constraints," *IEEE Transactions on Network Science and Engineering*, pp. 1–1, 2021.

[28] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Convergence time optimization for federated learning over wireless networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 4, pp. 2457–2471, 2021.

[29] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 3, pp. 1935–1949, 2021.

[30] B. Luo, X. Li, S. Wang, J. Huang, and L. Tassiulas, "Cost-effective federated learning in mobile edge networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3606–3621, 2021.

[31] N. H. Tran, W. Bao, A. Zomaya, M. N. H. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 1387–1395.

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

[33] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE transactions on wireless communications*, vol. 19, no. 3, pp. 2022–2035, 2020.

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

[35] M. M. Amiri and D. Gündüz, "Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air," *IEEE Transactions on Signal Processing*, vol. 68, pp. 2155–2169, 2020.

[36] X. Cao, G. Zhu, J. Xu, Z. Wang, and S. Cui, "Optimized power control design for over-the-air federated edge learning," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 1, pp. 342–358, 2021.

[37] R. A. Dunne and N. A. Campbell, "On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function," in *Proc. 8th Aust. Conf. on the Neural Networks, Melbourne*, vol. 181. Citeseer, 1997, p. 185.

[38] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, 2019.

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

[40] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[41] A. Ziller, A. Trask, A. Lopardo, B. Szymkow, B. Wagner, E. Bluemke, J.-M. Nounahon, J. Passerat-Palmbach, K. Prakash, N. Rose *et al.*, "Pysyft: A library for easy federated learning," in *Federated Learning Systems*. Springer, 2021, pp. 111–139.

[42] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[43] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, 2014.

[44] T. S. Rappaport *et al.*, *Wireless communications: principles and practice*. prentice hall PTR New Jersey, 1996, vol. 2.

[45] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*. John Wiley & Sons, 2013, vol. 398.

[46] A. Krizhevsky, I. Sutskever, and E. G. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, pp. 84–90, 2017.

[47] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[48] A. Krizhevsky, G. Hinton *et al.*, *Learning multiple layers of features from tiny images*. Citeseer, 2009.

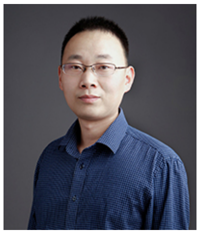
[49] Q. Ma, Y. Xu, H. Xu, Z. Jiang, L. Huang, and H. Huang, "FedSA: A Semi-Asynchronous Federated Learning Mechanism in Heterogeneous Edge Computing," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3654–3672, 2021.

[50] K. Hsieh, A. Phanishayee, O. Mutlu, and P. Gibbons, "The non-iid data quagmire of decentralized machine learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 4387–4398.

[51] S. Diamond and S. Boyd, "CVXPY: A python-embedded modeling language for convex optimization," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2909–2913, 2016.

Qianpiao Ma received his Ph.D. degree in computer software and theory from the University of Science and Technology of China in 2022. Prior to that, he received his B.S. degree in computer science from the University of Science and Technology of China in 2014. He is currently a post-doctoral researcher at Purple Mountain Laboratories. His primary research interests include federated learning, mobile edge computing and distributed machine learning.

jianchun Liu received his Ph.D. degree in computer software and theory from the University of Science and Technology of China in 2022. He is currently an associate researcher in the School of Computer Science and Technology, University of Science and Technology of China. His primary research interests include federated learning, mobile edge computing and distributed machine learning.



Hongli Xu (Member, IEEE) received his Ph.D. degree in computer software and theory from the University of Science and Technology of China, China, in 2007. He is a professor at the School of Computer Science and Technology, University of Science and Technology of China. He has won the best paper awards in several famous conferences. He has published more than 100 papers in famous journals and conferences. His primary research interest is software defined networks, edge computing and Internet of Thing.

Qingmin Jia is currently a researcher at Purple Mountain Laboratories, Nanjing, China. He received the B.S. degree in communication engineering from Qingdao University of Technology, Qingdao, China, in 2014, and received the Ph.D. degree in information and communication engineering from Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2019. His current research interests include Edge Intelligence and Industrial Internet of Things.

Renchao Xie (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from the Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2012. He is currently a Professor with the school of information and communication engineering, BUPT. His research interests include edge computing, information centric networking, and Industrial Internet of Things.