Reducing Training Time in Cross-Silo Federated Learning using Multigraph Topology

Tuong Do[1†], Binh X. Nguyen[1†], Vuong Pham[1], Toan Tran[2],
Erman Tjiputra[1], Quang D.Tran[1], Anh Nguyen[3]
[1]AIOZ, Singapore
[2]VinAI Research, Vietnam
[3]University of Liverpool, UK
{tuong.khanh-long.do, binh.xuan.nguyen}@aioz.io

Abstract

Federated learning is an active research topic since it enables several participants to jointly train a model without sharing local data. Currently, cross-silo federated learning is a popular training setting that utilizes a few hundred reliable data silos with high-speed access links to training a model. While this approach has been widely applied in real-world scenarios, designing a robust topology to reduce the training time remains an open problem. In this paper, we present a new multigraph topology for cross-silo federated learning. We first construct the multigraph using the overlay graph. We then parse this multigraph into different simple graphs with isolated nodes. The existence of isolated nodes allows us to perform model aggregation without waiting for other nodes, hence effectively reducing the training time. Intensive experiments on three public datasets show that our proposed method significantly reduces the training time compared with recent state-of-the-art topologies while maintaining the accuracy of the learned model. Our code can be found at:https://github.com/aioz-ai/MultigraphFL

1. Introduction

Federated learning entails training models via remote devices or siloed data centers while keeping data locally to respect the user's privacy policy [43]. According to [29], there are two popular training scenarios: the *cross-device* scenario, which encompasses a variety (millions or even billions) of unreliable edge devices with limited computational capacity and slow connection speeds; and the *cross-silo* scenario, which involves only a few hundred reliable data silos with powerful computing resources and high-speed access links. Recently, the cross-silo scenario becomes popular in different federated learning applications such as healthcare [88, 28, 21], robotics [68, 94, 83], medical imaging [8, 51], and finance [76, 50].

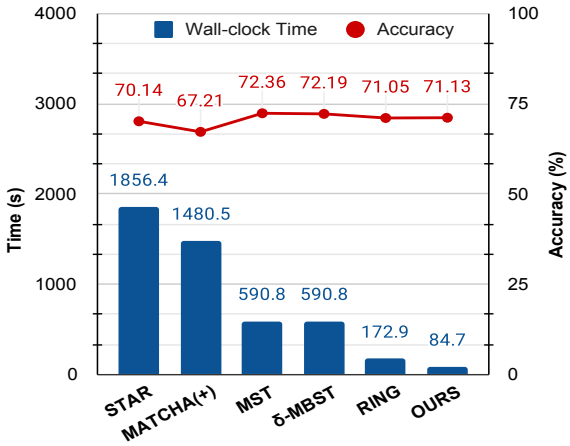


Figure 1. Comparison between different topologies on FEMNIST dataset and Exodus network [63]. The accuracy and total wall-clock training time (or overhead time) are reported after $6,400$ communication rounds. Our method significantly reduces the training time while maintaining the model accuracy.

In practice, federated learning is a promising research direction where we can utilize the effectiveness of machine learning methods while respecting the user's privacy. Critical challenges in federated learning include model convergence, communication congestion, and imbalance of data distributions in different silos [29]. A popular federated training method is to set a central node that orchestrates the training process and aggregates contributions of all clients [60]. The main limitation of this client-server approach is that the server node potentially represents a communication congestion point in the system, especially when the number of clients is large. To overcome this limitation, recent research has investigated the decentralized (or peer-

† indicates equal contribution.

to-peer) federated learning approach [48, 16, 58, 40, 26], in which the communication is performed via a peer-to-peer topology without the need for a central node. However, the main challenge of decentralized federated learning is to achieve fast training time while assuring model convergence and maintaining model accuracy.

In federated learning, the communication topology plays an important role. In particular, an efficient topology leads to faster convergence and reduces the training time and energy usage, quantifying by the worst-case convergence bounds in the topology design [27, 66, 84]. Furthermore, topology design is directly related to other problems during the training process such as network congestion, the overall accuracy of the trained model, or energy consumption [89, 68, 30]. Designing a robust topology that can reduce the training time while maintaining the model accuracy is still an open problem in federated learning [29]. Our paper aims to design a new topology for cross-silo federated learning, the most common training scenarios in practice.

Recently, different topologies have been proposed for cross-silo federated learning. In [3], the STAR topology is designed where the orchestrator averages all models throughout each communication round. The authors in [85] propose MATCHA to decompose the set of possible communications into pairs of clients. At each communication round, they randomly select some pairs and allow them to transmit models. The authors in [58] introduce RING topology using max-plus linear systems. While some progress has been made in the field, there are challenging problems that need to be addressed such as congestion at access links [85, 89], straggler effect [67, 71], or identical topology in all communication rounds [27, 58].

In this paper, we propose a new multigraph topology based on the recent RING topology [58] to reduce the training time for cross-silo federated learning. Our method first constructs the multigraph based on the overlay of the RING topology. We then parse this multigraph into simple graphs containing only one edge between two nodes. We refer to each simple graph as a *state* of the multigraph. Each state of the multigraph may have isolated nodes, and these nodes can do model aggregation without waiting for other nodes. This strategy significantly reduces the cycle time in each communication round. Intensive experiments show that our proposed topology outperforms other state-of-the-art methods in terms of training time for cross-silo federated learning by a large margin (See Figure 1).

2. Literature Review

Federated Learning. Federated learning has been regarded as a system capable of safeguarding data privacy [39, 14, 92, 41, 10]. Contemporary federated learning is based on a centralized network design in which a central node receives gradients from the client nodes to update a global model. Early findings of federated learning research include the work of [37], as well as a widely circulated article from [61]. Then [90, 74, 55, 24], and [77] extend the concept of federated learning and its related distributed optimization algorithms. Federated Averaging (FedAvg) was proposed by [60], its variations such as FedSage [91] and DGA [97], or other recent state-of-the-art model aggregation methods [20, 56, 93, 49, 9, 25, 6] are introduced to address the convergence and non-IID (non-identically and independently distributed) problem. Despite its simplicity, the client-server approach suffers from the communication and computational bottlenecks in the central node, especially when the number of clients is large [16, 73].

Decentralized Federated Learning. Decentralized (or peer-to-peer) federated learning allows each silo data to interact with its neighbors directly without a central node [16]. Due to its nature, decentralized federated learning does not have communication congestion at the central node, however, optimizing a fully peer-to-peer network is a challenging task [65, 47, 18, 48, 85, 58, 57, 40]. Noticeably, the decentralized periodic averaging stochastic gradient descent [84] is proved to converge at a comparable rate to the centralized algorithm while allowing large-scale model training [87, 75, 69]. Besides, systematic analysis of the decentralized federated learning has been explored by [46, 11, 36]. Recently, Jeong *et al*. [26] leverage knowledge distillation mechanism to ensure the collaboration between silos in the decentralized federated scenario while preserving the privacy between neighbor nodes.

Communication Topology. The topology has a direct impact on the complexity and convergence of federated learning [7]. Many works have been introduced to improve the effectiveness of topology, including star-shaped topology [3, 39, 62, 60, 29] and optimized-shaped topology [67, 85, 58, 1, 81, 23]. Particularly, a spanning tree topology based on [72] algorithm was introduced by [58] to reduce the training time. As mentioned by [3], STAR topology is designed where an orchestrator averages model updates in each communication round. The authors in [85] introduce MATCHA to speed up the training process through decomposition sampling. Since the duration of a communication round is dictated by stragglers effect [32, 42], [67] explore how to choose the degree of a regular topology. RING topology is proposed in [58] for cross-silo federated learning using the theory of max-plus linear systems. Huang *et al*. [23] introduce sample-induced topology which is able to recover the effectiveness of existing SGD-based algorithms along with their corresponding rates. Recently, Wu *et al*. [86] conducts a comprehensive survey of models, frameworks, and algorithms on network topologies.

Multigraph. The definition of multigraph has been introduced as a traditional paradigm in math [12, 82]. A typical "graph" usually refers to a simple graph with no loops

or multiple edges between two nodes. Different from a simple graph, a multigraph allows multiple edges between two nodes. In deep learning, multigraph has been applied in different domains, including clustering [59, 53, 31], medical image processing [52, 95, 2], traffic flow prediction [54, 96], activity recognition [78], recommendation system [79], and cross-domain adaptation [70]. In this paper, we construct the multigraph to enable isolated nodes and reduce the training time in cross-silo federated learning.

3. Preliminaries

3.1. Federated Learning

In federated learning, silos do not share their local data, but still periodically transmit model updates between them. Given N siloed data centers, the objective function for federated learning is:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^{N} p_i \mathbb{E}_{\xi_i}\left[L_i\left(\mathbf{w}, \xi_i\right)\right], \tag{1}$$

where $L_i(\mathbf{w}, \xi_i)$ is the loss of model parameterized by the weight $\mathbf{w} \in \mathbb{R}^d$, ξ_i is an input sample drawn from data at silo i, and the coefficient $p_i > 0$ specifies the relative importance of each silo. Recently, different distributed algorithms have been proposed to optimize Eq. 1 [38, 60, 44, 85, 45, 84, 33]. In this work, DPASGD [84] is used to update the weight of silo i in each training round as follows:

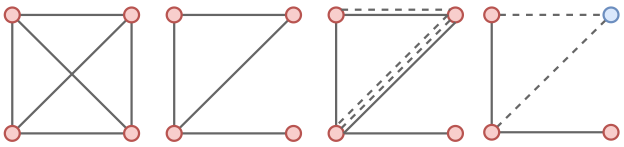
$$\mathbf{w}_i\left(k+1\right) = \begin{cases} \sum_{j\in\mathcal{N}_i^+\cup\{i\}} \mathbf{A}_{i,j}\mathbf{w}_j\left(k\right), \\ \qquad\qquad \text{if k} \equiv 0 \left(\text{mod } u+1\right), \\ \mathbf{w}_i\left(k\right) - \alpha_k \frac{1}{b}\sum_{h=1}^{b} \nabla L_i\left(\mathbf{w}_i\left(k\right), \xi_i^{(h)}\left(k\right)\right), \\ \qquad\qquad \text{otherwise.} \end{cases} \tag{2}$$

where b is the batch size, i, j denote the silo, u is the number of local updates, $\alpha_k > 0$ is a potentially varying learning rate at k-th round, $\mathbf{A} \in \mathbb{R}^{N\times N}$ is a consensus matrix with non-negative weights, and $\mathcal{N}_i^+$ is the in-neighbors set that silo i has the connection to.

3.2. Multigraph for Federated Learning

Connectivity and Overlay. Following [58], we consider the *connectivity* $\mathcal{G}_c = (\mathcal{V}, \mathcal{E}_c)$ as a graph that captures possible direct communications among silos. Based on its definition, the connectivity is often a fully connected graph and is also a directed graph. The *overlay* $\mathcal{G}_o$ is a connected subgraph of the connectivity graph, i.e., $\mathcal{G}_o = (\mathcal{V}, \mathcal{E}_o)$, where $\mathcal{E}_o \subset \mathcal{E}_c$. Only nodes directly connected in the overlay graph $\mathcal{G}_o$ will exchange the messages during training. We refer the readers to [58] for more in-depth discussions.

Multigraph. While the connectivity and overlay graph can represent different topologies for federated learning, one of their drawbacks is that there is only one connection between two nodes. In our work, we construct a *multigraph* $\mathcal{G}_m = (\mathcal{V}, \mathcal{E}_m)$ from the overlay $\mathcal{G}_o$. The multigraph can contain multiple edges between two nodes [5]. In practice, we parse this multigraph to different graph states, each state is a simple graph with only one edge between two nodes.



(a) Connectivity (b) Overlay (c) Multigraph (d) State of Multigraph

Figure 2. Example of connectivity, overlay, multigraph, and a state of our multigraph. Blue node is an isolated node. Dotted line denotes a weakly-connected edge.

In the multigraph $\mathcal{G}_m$, the connection edge between two nodes has two types: *strongly-connected* edge and *weakly-connected* edge [34]. Under both strong and weak connections, the participated nodes can transmit their trained models to their out-neighbours $\mathcal{N}_i^-$ or download models from their in-neighbours $\mathcal{N}_i^+$. However, in a strongly-connected edge, two nodes in the graph must wait until all upload and download processes between them are finished to do model aggregation. On the other hand, in a weakly-connected edge, the model aggregation process in each node can be established whenever the previous training process is finished by leveraging up-to-date models which have not been used before from the in-neighbours of that node.

State of Multigraph. Given a multigraph $\mathcal{G}_m$, we can parse this multigraph into different simple graphs with only one connection between two nodes (either strongly-connected or weakly-connected). We denote each simple graph as a state $\mathcal{G}_m^s$ of the multigraph.

Isolated Node. A node is called isolated when all of its connections to other nodes are weakly-connected edges. Figure 2 shows the graph concepts and isolated nodes.

3.3. Delay and Cycle Time in Multigraph

Delay. Following [58], a delay to an edge $e(i, j)$ is the time interval when node j receives the weight sending by node i, which can be defined by:

$$d(i,j) = u \times T_c(i) + l(i,j) + \frac{M}{O(i,j)}, \tag{3}$$

where $T_c(i)$ denotes the time to compute one local update of the model; u is the number of local updates; $l(i, j)$ is the link latency; M is the model size; $O(i, j)$ is the total network traffic capacity. However, unlike other communication infrastructures, the multigraph only contains connections between silos without other nodes such as routers or amplifiers. Thus, the total network traffic capacity $O(i,j) = \min\left(\frac{C_{\text{UP}}(i)}{|\mathcal{N}_i^-|}, \frac{C_{\text{DN}}(j)}{|\mathcal{N}_i^+|}\right)$ where C_{UP} and C_{DN} denote the upload and download link capacity. Note that the

upload and download processes can happen in parallel.

Since multigraph can contain multiple edges between two nodes, we extend the definition of the delay in Eq. 3 to $d_k(i,j)$, with k is the k-th communication round during the training process, as:

$$d_{k+1}(i,j) = \begin{cases} d_k(i,j), & \text{if } e_{k+1}(i,j) = \mathbb{1} \text{ and } e_k(i,j) = \mathbb{1} \\ \max(u \times T_c(j), d_k(i,j) - d_{k-1}(i,j)), & \text{if } e_{k+1}(i,j) = \mathbb{1} \text{ and } e_k(i,j) = \mathbb{0} \\ \tau_k(\mathcal{G}_m) + d_{k-1}(i,j)), & \text{if } e_{k+1}(i,j) = \mathbb{0} \text{ and } e_k(i,j) = \mathbb{0} \\ \tau_k(\mathcal{G}_m), & \text{otherwise} \end{cases} \tag{4}$$

where $e(i,j)=\mathbb{0}$ is weakly-connected edge, $e(i,j)=\mathbb{1}$ is strongly-connected edge; $\tau_k(\mathcal{G}_m)$ is the cycle time at the k-th computation round during the training process. In general, using Eq. 4, *the delay of the next communication round d_{k+1} is updated based on the delay of the previous rounds* and other factors, depending on the edge type connection.

Cycle Time. The cycle time per round is the time required to complete a communication round [58]. In this work, we define the cycle time per round is the maximum delay between all silo pairs with strongly-connected edges. Therefore, the average cycle time of the entire training is:

$$\tau(\mathcal{G}_m) = \frac{1}{k} \sum_{k=0}^{k-1} \left(\max_{j \in \mathcal{N}_i^{++} \cup \{i\}, \forall i \in \mathcal{V}} \left(d_k\left(j,i\right) \right) \right), \tag{5}$$

where $\mathcal{N}_i^{++}$ is an in-neighbors silo set of i whose edges are strongly-connected.

4. Multigraph Topology

Our method first constructs the multigraph based on an overlay. Then we parse this multigraph into multiple states that may have isolated nodes. Note that, we do not choose isolated nodes randomly, but rely on the delay time. In practice, we observe that *node with long delay time is the main reason for increasing the training time* since other nodes have to wait for it. Therefore, we want long delay nodes to become isolated nodes, and skip the model aggregation step on them. In the next communication round, the delay time of the isolated node will be updated using Eq. 4, therefore it can become a normal node. This strategy allows us to reduce the waiting time in isolated nodes while ensuring that isolated nodes can become normal nodes and contribute to the training after a number of communication rounds.

4.1. Multigraph Construction

Algorithm 1 describes our methods to generate the multigraph $\mathcal{G}_m$ with multiple edges between silos. The algorithm

Algorithm 1: Multigraph Construction.

Input: Overlay $\mathcal{G}_o = (\mathcal{V}, \mathcal{E}_o)$;
Maximum edge between two nodes t.
Output: Multigraph $\mathcal{G}_m = (\mathcal{V}, \mathcal{E}_m)$;
List number of edges between silo pairs $\mathcal{L}$.

```
// Delay computation for overlay
```
$D_o \leftarrow$ Establish a list of all delays to each silo pair.
foreach *edge* $e(i,j) \in \mathcal{E}_o$ **do**
  $d(i,j) \leftarrow$ Compute delay in overlay using Eq. 3.
  Append the computed delay $d(i,j)$ into D_o.

```
// Multigraph Establishment
```
$d_{\min} \leftarrow$ Compute the smallest delay by $\min(D_0)$.
$\mathcal{E}_m \leftarrow$ Establish the multiset containing all edges.
$\mathcal{L}[|\mathcal{V}|, |\mathcal{V}|] \leftarrow$ Initialize the full-zero list for tracking the number of edges between each silo pair.
foreach *edge* $e(i,j) \in \mathcal{E}_o$ **do**
  $n(i,j) \leftarrow$ Find the number of edges for (i,j) pair by computing $\min\left(t, \text{round}\left(\frac{d(i,j)}{d_{\min}}\right)\right)$
  $\mathcal{E}_t \leftarrow$ Establish the set for each edge $e(i,j)$ that has marked with connection status. $\mathbb{1}$ labeled as strong-connected edge while $\mathbb{0}$ represented for weak-connected edge.
  Append strong-connected edge $e(i,j) = \mathbb{1}$ into $\mathcal{E}_t$.
  foreach $(n(i,j) - 1)$ **do**
    Append weak-connected edge $e(i,j) = \mathbb{0}$ into $\mathcal{E}_t$.
  Append established edge set $\mathcal{E}_t$ into multiset $\mathcal{E}_m$.
  Update the number of edges between (i,j) pair $n(i,j)$ into the corresponding position $\mathcal{L}[i,j]$ of track list $\mathcal{L}$.
return Multigraph $\mathcal{G}_m = (\mathcal{V}, \mathcal{E}_m)$; Track list $\mathcal{L}$

takes the overlay $\mathcal{G}_o$ as input. Similar to [58], we use the Christofides algorithm [64] to obtain the overlay. In Algorithm 1, we establish multiple edges that indicate different statuses (strongly-connected or weakly-connected). To identify the total edges between a silo pair, we divide the delay $d(i,j)$ by the smallest delay $d_{\min}$ overall silo pairs, and compare it with the maximum number of edges parameter t ($t = 5$ in our experiments). *We assume that the silo pairs with longer delay will have more weakly-connected edges, hence potentially becoming isolated nodes*. Overall, we aim to increase the number of weakly-connected edges, which generate more isolated nodes to speed up the training process. Note that, from Algorithm 1, each silo pair in the multigraph should have one strongly-connected edge and multiple weakly-connected edges. The role of the strongly-connected edge is to make sure that two silos have a good connection in at least one communication round.

4.2. Multigraph Parsing

In Algorithm 2, we parse multigraph $\mathcal{G}_m$ into multiple graph states $\mathcal{G}_m^s$. Graph states are essential to identify the connection status of silos in a specific communication round

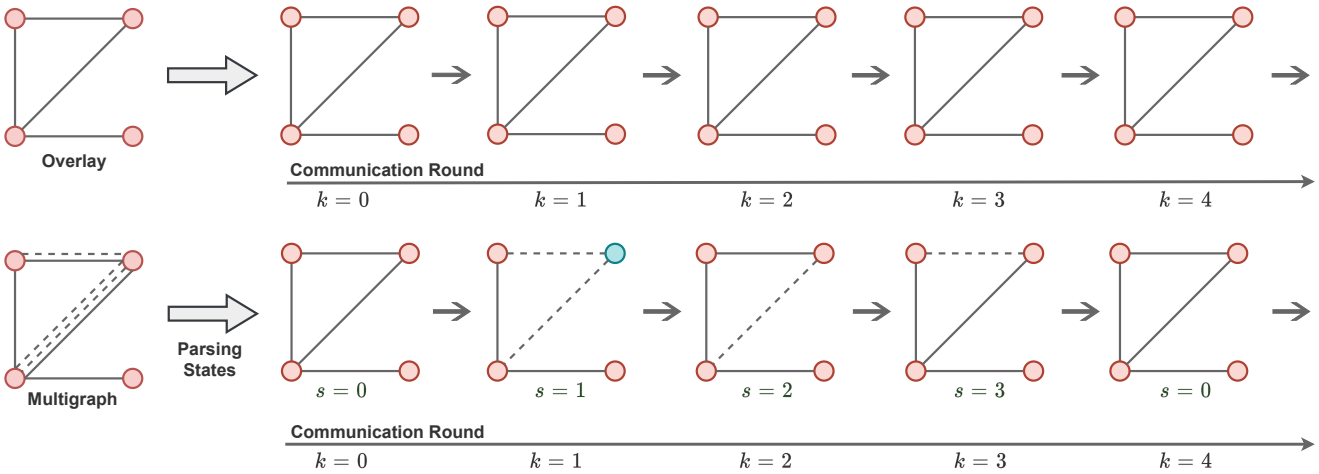


Figure 3. The comparison between RING [58] topology and our multigraph topology in each communication round. (a) RING uses the same overlay in each round. (b) Our proposed multigraph is parsed into different graph states. Each graph state is used in a communication round. Lines denote strongly-connected edges, dotted lines denote weakly-connected ones, and the blue color indicates isolated nodes.

Algorithm 2: Multigraph Parsing.

Input: Multigraph $\mathcal{G}_m = (\mathcal{V}, \mathcal{E}_m)$;
List edge numbers between silo pairs $\mathcal{L}$.
Output: List of multigraph states $\mathcal{S} = \{\mathcal{G}_m^s = (\mathcal{V}, \mathcal{E}_m^s)\}$.

$s_{\max} \leftarrow$ Compute maximum number of distinct state in $\mathcal{G}_m$ by using Least Common Multiple [15]
$\bar{\mathcal{L}} \leftarrow$ Establish the dynamic list that tracks the number of edges between silo pairs during changing graph state and is initialized by input list edge $\mathcal{L}$.
$\bar{\mathcal{E}}_m^s \leftarrow$ Establish the list of all possible extracted states.
`// States of Multigraph Establishment`
for *state* $s = 0$ **to** $s_{\max}$ **do**
  $\mathcal{E}_t \leftarrow$ Establish temporary edge set.
  foreach *edge* $e(i,j) \in \mathcal{E}_m$ **do**
    if $\bar{\mathcal{L}}[i,j] = \mathcal{L}[i,j]$ **then**
      Append strong connected-edge $e(i,j) = \mathbb{1}$ into edge set $\mathcal{E}_t$.
    else
      Append weak-connected edge $e(i,j) = \mathbb{0}$ into edge set $\mathcal{E}_t$.
    if $\bar{\mathcal{L}}[i,j] = 1$ **then**
      Update the status of dynamic list $\bar{\mathcal{L}}$ using input $\mathcal{L}$ through $\bar{\mathcal{L}}[i,j] = \mathcal{L}[i,j]$.
    else
      Reduce the corresponding number of edges by $\bar{\mathcal{L}}[i,j] -= 1$.
  Append edge set $\mathcal{E}_t$ into $\bar{\mathcal{E}}_m^s$
return List of multigraph state $\mathcal{S} = \{\mathcal{G}_m^s = (\mathcal{V}, \mathcal{E}_m^s)\}$ by using list of possible extracted states $\bar{\mathcal{E}}_m^s$.

to perform model aggregation. In each graph state, our goal is to identify the isolated nodes. During the training, isolated nodes update their weights internally and ignore all weakly-connected edges that connect to them.

To parse the multigraph into graph states, we first identify the maximum of states in a multigraph $s_{\max}$ by using the least common multiple (LCM) [15]. We then parse the multigraph into $s_{\max}$ states. The first state is always the overlay since we want to make sure all silos have a reliable topology at the beginning to ease the training. The reminding states are parsed so there is only one connection between two nodes. Using our algorithm, some states will contain isolated nodes. During the training process, only one graph state is used in a communication round. Figure 3 illustrates the training process in each communication round using multiple graph states.

4.3. Multigraph Training

In each communication round, a state graph $\mathcal{G}_m^s$ is selected in a sequence that identifies the topology design used for training. We then collect all strongly-connected edges in the graph state $\mathcal{G}_m^s$ in such a way that nodes with strongly-connected edges need to wait for neighbors, while the isolated ones can update their models. We train our multigraph with DPASGD algorithm [84]:

$$\mathbf{w}_i(k+1) = \begin{cases} \sum_{j \in \mathcal{N}_i^{++} \cup \{i\}} \mathbf{A}_{i,j} \mathbf{w}_j (k-h), \\ \qquad \text{if k} \equiv 0 \,(\text{mod } u+1) \,\&\, \left|\mathcal{N}_i^{++}\right| > 1, \\ \mathbf{w}_i(k) - \alpha_k \frac{1}{b} \sum_{h=1}^{b} \nabla L_i \left(\mathbf{w}_i(k), \xi_i^{(h)}(k)\right), \\ \qquad \text{otherwise.} \end{cases} \tag{6}$$

where $(k-h)$ is the index of the considered weights; h is initialized to 0 and $h = h+1$, if $e_{k-h}(i,j) = \mathbb{0}$. Through Eq. 6, at each state, if a silo is not an isolated node, it must wait for the model from its neighbor to update its weight. If a silo is an isolated node, it can use the model in its neighbor from the $(k-h)$ round to update its weight immediately.

Dataset	Network	Topology Design						
		STAR [3]	MATCHA [85]	MATCHA(+) [58]	MST [72]	δ-MBST [58]	RING [58]	Ours
FEMNIST	Gaia	289.8 (↓ 18.5)	166.4 (↓ 10.6)	166.4 (↓ 10.6)	77.2 (↓ 4.9)	77.2 (↓ 4.9)	57.2 (↓ 3.6)	**15.7**
	Amazon	98.8 (↓ 7.3)	57.7 (↓ 4.2)	57.7 (↓ 4.2)	28.7 (↓ 2.1)	28.7 (↓ 2.1)	20.3 (↓ 1.5)	**13.6**
	Géant	132.2 (↓ 11.0)	46.9 (↓ 3.9)	102.3 (↓ 8.5)	40.1 (↓ 3.3)	40.1 (↓ 3.3)	27.7 (↓ 2.3)	**12.0**
	Exodus	265.2 (↓ 21.9)	84.7 (↓ 7.0)	211.5 (↓ 17.5)	84.4 (↓ 7.0)	84.4 (↓ 7.0)	24.7 (↓ 2.0)	**12.1**
	Ebone	190.9 (↓ 15.0)	61.5 (↓ 4.8)	112.6 (↓ 8.9)	60.9 (↓ 4.8)	60.9 (↓ 4.8)	18.5 (↓ 1.5)	**12.7**
iNaturalist	Gaia	390.9 (↓ 5.7)	227.4 (↓ 3.3)	227.4 (↓ 3.3)	138.1 (↓ 2.0)	138.1 (↓ 2.0)	118.1 (↓ 1.7)	**68.6**
	Amazon	288.1 (↓ 3.5)	123.9 (↓ 1.5)	123.9 (↓ 1.5)	89.7 (↓ 1.1)	89.7 (↓ 1.1)	**81.3** (↓ 1.0)	**81.3**
	Géant	622.3 (↓ 9.1)	107.9 (↓ 1.6)	452.5 (↓ 6.6)	101 (↓ 1.5)	101 (↓ 1.5)	109.0 ↓ 1.6)	**68.1**
	Exodus	911.9 (↓ 14.6)	145.7 (↓ 2.3)	593.2 (↓ 9.5)	145.3 (↓ 2.3)	145.3 (↓ 2.3)	103.9 (↓ 1.7)	**62.6**
	Ebone	901.7 (↓ 13.9)	122.5 (↓ 1.9)	579.9 (↓ 8.9)	121.8 (↓ 1.9)	121.8 (↓ 1.9)	95.3 (↓ 1.5)	**64.9**
Sentiment140	Gaia	323.8 (↓ 10.5)	186.0 (↓ 6.0)	186 (↓ 6.0)	96.8 (↓ 3.1)	96.8 (↓ 3.1)	76.8 (↓ 2.5)	**31.0**
	Amazon	164.6 (↓ 4.6)	79.2 (↓ 2.2)	79.2 (↓ 2.2)	48.4 (↓ 1.4)	48.4 (↓ 1.4)	40.0 (↓ 1.1)	**35.8**
	Géant	310.5 (↓ 10.3)	66.6 (↓ 2.2)	222.6 (↓ 7.4)	59.7 (↓ 2.0)	59.7 (↓ 2.0)	54.9 (↓ 1.8)	**30.3**
	Exodus	495.4 (↓ 17.7)	104.3 (↓ 3.7)	346.3 (↓ 12.4)	104.1 (↓ 3.7)	104.1 (↓ 3.7)	50.6 (↓ 1.8)	**28.0**
	Ebone	444.2 (↓ 15.3)	81.1 (↓ 2.8)	262.2 (↓ 9.0)	80.5 (↓ 2.8)	80.5 (↓ 2.8)	43.9 (↓ 1.5)	**29.1**

Table 1. Cycle time (ms) comparison between different typologies. (↓ ∘) indicates our reduced times compared with other methods.

5. Experiments

5.1. Experimental Setup

Datasets. We use three datasets in our experiments: Sentiment140 [13], iNaturalist [80], and FEMNIST [4]. All datasets and the pre-processing process are conducted by following recent works [85] and [58]. Table 2 shows the dataset setups in detail.

Dataset	FEMNIST	Sentiment140	iNaturalist
#Samples	805M	1,600M	450M
Model	CNN [58]	LSTM [19]	ResNet [17]
#Params	1,2M	4,8M	11,2M
Batch size	128	512	16
Model size	4.62	18.38	42.88

Table 2. Dataset statistic and model details in our experiments. The model size is in Mbits.

Network. Following [58], we consider five distributed networks in our experiments: Exodus, Ebone, Géant, Amazon [63] and Gaia [22]. The Exodus, Ebone, and Géant are from the Internet Topology Zoo [35]. The Amazon and Gaia network are synthetic and are constructed using the geographical locations of the data centers.

Baselines. We compare our multigraph topology with recent state-of-the-art topology designs for federated learning: STAR [3], MATCHA [85], MATCHA(+) [58], MST [72], δ-MBST [58], and RING [58].

Hardware Setup. Since measuring the cycle time is crucial to compare the effectiveness of all topologies in practice, we test and report the cycle time of all baselines and our method on the same NVIDIA Tesla P100 16Gb GPUs. No overclocking is used.

Time Simulator. Similar to [58], we adapted PyTorch with the MPI backend. We take advantage of the network simulator and the timing simulator as in Marfod *et al*. [58]. More details of our experimental setup and time simulator for computing wall-clock time can be found in our Supplementary Material.

5.2. Cycle Time Comparison

Table 1 shows the cycle time of our method in comparison with other recent approaches. This table illustrates that our proposed method significantly reduces the cycle time in all setups with different networks and datasets. In particular, compared to the state-of-the-art RING [58], our method reduces the cycle time by 2.18, 1.5, 1.74 times in average in the FEMNIST, iNaturalist, Sentiment140 dataset, respectively. Our method also clearly outperforms MACHA, MACHA(+), and MST by a large margin. The results confirm that our multigraph with isolated nodes helps reduce the cycle time in federated learning.

From Table 1, our multigraph achieves the minimum improvement under the Amazon network in all three datasets. This can be explained that, under the Amazon network, our proposed topology does not generate many isolated nodes. Hence, the improvement is limited. Intuitively, when there are no isolated nodes, our multigraph will become the overlay, and the cycle time of our multigraph will be equal to the cycle time of the overlay in RING.

Network	Total silos	#Rounds	#States	Cycle Time (ms)
Gaia	11	4693/6400	44/60 (73.3%)	15.7 (↓3.6)
Amazon	22	2133/6400	2/6 (33.3%)	13.6 (↓1.5)
Géant	40	4266/6400	8/12 (66.7%)	12.0 (↓2.3)
Exodus	79	3306/6400	31/60 (51.7%)	12.1 (↓2.0)
Ebone	87	2346/6400	11/30 (36.7%)	12.7 (↓1.5)

Table 3. The effectiveness of isolated nodes under different network configurations. All experiments are trained with 6400 communication rounds on FEMNIST dataset.We then record the number of states and rounds that have the appearance of isolated nodes and compare our cycle time with RING [58].

5.3. Isolated Node Analysis

Isolated Nodes vs. Network Configuration. The numbers of isolated nodes vary based on the network configuration (Amazon, Gaia, Exodus, etc.). The parameter t (maximum number of edges between two nodes), and the delay time which is identified by many factors (geometry distance, model size, computational cost based on tasks, bandwidth, etc. - please Eq. 3) also affect the process of generating isolated nodes. Table 3 illustrates the effectiveness of isolated nodes in different network configurations. Specifically, we conduct experiments on the FEMNIST dataset using five network configurations (Gaia, Amazon, Geant, Exodus, Ebone). We can see that our cycle time compared with RING [58] is reduced significantly when more communication rounds or graph states have isolated nodes.

Isolated Nodes vs. RING vs. Random Strategy. Isolated nodes play an important role in our method as we can skip the model aggregation step in the isolated nodes. In practice, we can have a trivial solution to create isolated nodes by randomly removing some nodes from the overlay of RING. Table 4 shows the experiment results in two scenarios on FEMNIST dataset and Exodus Network: *i)* Randomly remove some silos in the overlay of RING, and *ii)* Remove the most inefficient silos (i.e., silos with the longest delay) in the overlay of RING. From Table 4, the cycle time reduces significantly when two aforementioned scenarios are applied. However, the accuracy of the model also drops significantly. This experiment shows that although randomly removing some nodes from the overlay of RING is a trivial solution, it can not maintain model accuracy. On the other hand, our multigraph not only reduces the cycle time of the model but also preserves the accuracy. This is because our multigraph can skip the aggregation step of the isolated nodes in a communication round. However, in the next round, the delay time of these isolated nodes will be updated, and they can become normal nodes and contribute to the final model.

Methods	Criteria	#Removed Nodes	Cycle Time (ms)	Acc (%)
RING [58]	*Baseline*	-	24.7	71.05
	Randomly remove silos in overlay	1	23.1	70.63
		5	21.7	68.57
		10	18.8	64.23
		20	13.0	61.2
	Remove most inefficient silos	1	22.5	70.71
		5	19.5	68.37
		10	15.8	63.13
		20	11.2	61.48
Multigraph (ours)	-	-	**12.1**	**71.13**

Table 4. The cycle time and accuracy of our multigraph vs. RING with different criteria.

Isolated Nodes Illustration. Figure 4 shows a detailed illustration of our algorithm with the isolated nodes in a real-world training scenario. The experiment is conducted on Gaia network [22] geometry and their corresponding hardware for supporting link latency computation. The image classification task is chosen for this benchmarking by using FEMNIST dataset [4] and CNN backbone provided by Marfod *et al*. [58]. Hence, we keep the model transmitted size at 4.62 Mb, all access links have 10 Gbps traffic capacity, the number of local updates is set to 1, and the maximum number of edges t is set to 3. As shown in Figure 4, although there are no isolated nodes in the initialized state, the number of isolated nodes increases in the next consequence states with a vast number (4 nodes). This circumstance leads to a $\sim$ 4 times reduction in cycle time compared to the initialized state. The appearance of isolated nodes also greatly reduces the connection between silos by $\sim$ 3.6 times, from 11 down to 3 connections, and also discarded ones all have high latency.

5.4. Multigraph Ablation Study

Accuracy Analysis. In federated learning, improving the model accuracy is not the main focus of topology designing methods. However, preserving the accuracy is also important to ensure model convergence. Table 5 shows the accuracy of different topologies after $6,400$ communication training rounds on the FEMNIST dataset. This table illustrates that our proposed method achieves competitive accuracy with other topology designs. This confirms that our topology can maintain the accuracy of the model, while significantly reducing the training time.

Convergence Analysis. Figure 5 shows the training loss versus the number of communication rounds and the wall-clock time under Exodus network using the FEMNIST dataset. This figure illustrates that our proposed topol-

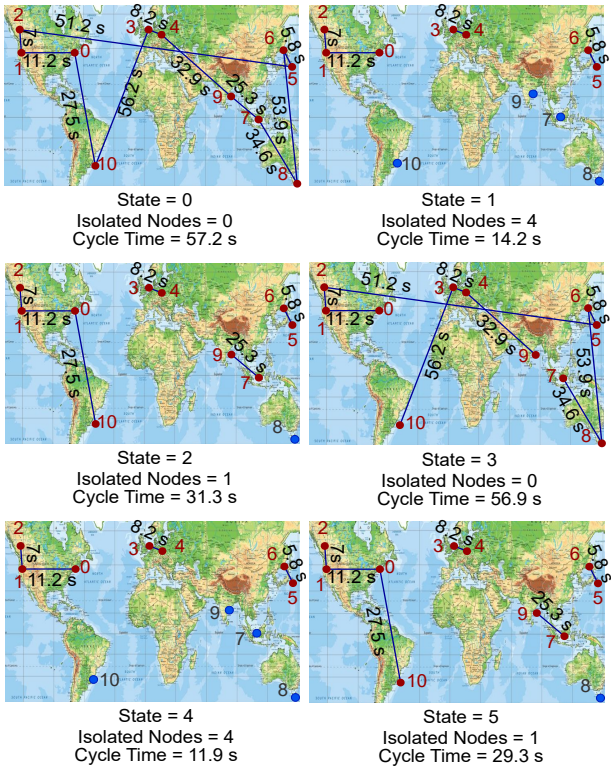


Figure 4. Isolated nodes illustration. Red nodes indicate normal nodes. Blue nodes are isolated nodes.

Network	Topology					
	STAR	MATCHA(+)	MST	δ-MBST	RING	Ours
Gaia	69.09	68.43	68.86	68.95	68.2	68.45
Amazon	69.59	69.06	69.65	70.37	69.78	69.63
Géant	68.91	65.57	69.44	68.94	69.3	68.98
Ebone	69.66	64.48	71.91	70.62	70.29	70.23
Exodus	70.14	67.21	72.36	72.19	71.05	71.13

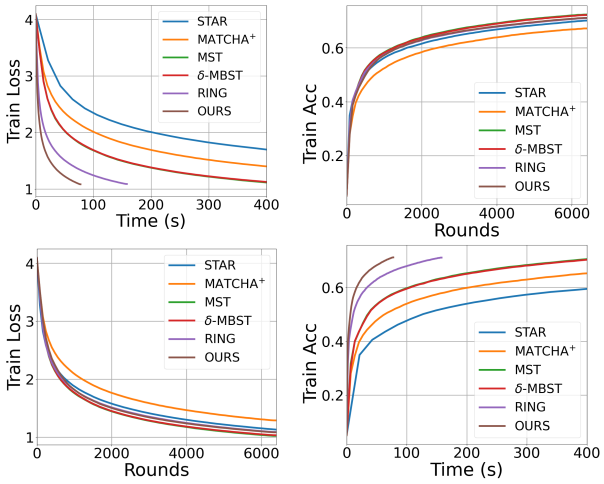
Table 5. Accuracy comparison between different topologies. The experiment is conducted using the FEMNIST dataset. The accuracy is reported after $6,400$ communication rounds in all methods.

ogy converges faster than other methods while maintaining the model accuracy. We observe the same results in other datasets and network setups.

Cycle Time and Accuracy Trade-off. In our method, the maximum number of edges between two nodes t in Algorithm 1 mainly affects the number of isolated nodes. This leads to a trade-off between the model accuracy and cycle time. Table 6 illustrates the effectiveness of this parameter. When $t = 1$, we technically consider there are no weak connections and isolated nodes. Therefore, our method uses the original overlay from RING. When t is set higher, we can increase the number of isolated nodes, hence decreasing the

Topology	t	Cycle time (ms)	Acc(%)
RING [58]	-	24.7	71.05
Multigraph (ours)	1	24.7	71.05
	3	13.5	71.08
	5	12.1	71.13
	8	11.9	69.27
	10	11.9	69.27
	20	11.9	69.27
	30	11.9	69.27

Table 6. Cycle time and accuracy trade-off with different value of t, i.e., the maximum number of edges between two nodes.



(a) Train Loss (b) Train Accuracy

Figure 5. Convergence analysis of our multigraph under communication rounds (top row) and wall-clock time (bottom row). All access links have a 10 Gbps capacity. The training time is counted after all setups finish $6,400$ communication rounds.

cycle time. In practice, too many isolated nodes will limit the model weights to be exchanged between silos. Therefore, models at isolated nodes are biased to their local data and consequently affect the final accuracy.

6. Conclusion

We proposed a new multigraph topology for cross-silo federated learning. Our method first constructs the multigraph using the overlay. Different graph states are then parsed from the multigraph and used in each communication round. Our method significantly reduces the cycle time by allowing the isolated nodes in the multigraph to do model aggregation without waiting for other nodes. The intensive experiments on three datasets show that our proposed topology achieves new state-of-the-art results in all network and dataset setups.

References

[1] Aurélien Bellet, Anne-Marie Kermarrec, and Erick Lavoie. D-cliques: Compensating noniidness in decentralized federated learning with topology. *arXiv*, 2021. 2

[2] Alaa Bessadok, Mohamed Ali Mahjoub, and Islem Rekik. Brain multigraph prediction using topology-aware adversarial graph neural network. *Medical Image Analysis*, 2021. 3

[3] Ulrik Brandes. On variants of shortest-path betweenness centrality and their generic computation. *Social Networks*, 2008. 2, 6

[4] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv*, 2018. 6, 7

[5] Gary Chartrand and Ping Zhang. *A first course in graph theory*. Courier Corporation, 2013. 3

[6] Junming Chen, Meirui Jiang, Qi Dou, and Qifeng Chen. Federated domain generalization for image recognition via cross-client style transfer. In *WACV*, 2023. 2

[7] Mingzhe Chen, H Vincent Poor, Walid Saad, and Shuguang Cui. Wireless communications for collaborative federated learning. *IEEE Communications Magazine*, 2020. 2

[8] Pierre Courtiol, Charles Maussion, Matahi Moarii, Elodie Pronier, Samuel Pilcer, Meriem Sefta, Pierre Manceron, Sylvain Toldo, Mikhail Zaslavskiy, Nolwenn Le Stang, et al. Deep learning-based classification of mesothelioma improves prediction of patient outcome. *Nature medicine*, 2019. 1

[9] Anis Elgabli, Chaouki Ben Issaid, Amrit Singh Bedi, Ketan Rajawat, Mehdi Bennis, and Vaneet Aggarwal. Fednew: A communication-efficient and privacy-preserving newton-type method for federated learning. In *ICML*, 2022. 2

[10] Liang Gao, Huazhu Fu, Li Li, Yingwen Chen, Ming Xu, and Cheng-Zhong Xu. Feddc: Federated learning with non-iid data via local drift decoupling and correction. In *CVPR*, 2022. 2

[11] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. In *NIPS*, 2020. 2

[12] Alan Gibbons. *Algorithmic graph theory*. Cambridge university press, 1985. 2

[13] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*, 2009. 6

[14] Xuan Gong, Abhishek Sharma, Srikrishna Karanam, Ziyan Wu, Terrence Chen, David Doermann, and Arun Innanje. Ensemble attention distillation for privacy-preserving federated learning. In *ICCV*, 2021. 2

[15] Godfrey Harold Hardy, Edward Maitland Wright, et al. *An introduction to the theory of numbers*. Oxford university press, 1979. 5

[16] Chaoyang He, Conghui Tan, Hanlin Tang, Shuang Qiu, and Ji Liu. Central server free federated learning over single-sided trust social networks. *arXiv*, 2019. 2

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6

[18] Lie He, An Bian, and Martin Jaggi. Cola: Decentralized linear learning. *NIPS*, 2018. 2

[19] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997. 6

[20] Junyuan Hong, Haotao Wang, Zhangyang Wang, and Jiayu Zhou. Efficient split-mix federated learning for on-demand and in-situ customization. In *ICLR*, 2021. 2

[21] S Maryam Hosseini, Milad Sikaroudi, Morteza Babaie, and HR Tizhoosh. Proportionally fair hospital collaborations in federated learning of histopathology images. *TMI*, 2023. 1

[22] Kevin Hsieh, Aaron Harlap, Nandita Vijaykumar, Dimitris Konomis, Gregory R Ganger, Phillip B Gibbons, and Onur Mutlu. Gaia: Geo-distributed machine learning approaching lan speeds. In *USENIX Symposium on Networked Systems Design and Implementation*, 2017. 6, 7

[23] Yan Huang, Ying Sun, Zehan Zhu, Changzhi Yan, and Jinming Xu. Tackling data heterogeneity: A new unified framework for decentralized sgd with sample-induced topology. In *ICML*, 2022. 2

[24] Martin Jaggi, Virginia Smith, Martin Takác, Jonathan Terhorst, Sanjay Krishnan, Thomas Hofmann, and Michael I Jordan. Communication-efficient distributed dual coordinate ascent. In *NIPS*, 2014. 2

[25] Shreyansh Jain and Koteswar Rao Jerripothula. Federated learning for commercial image sources. In *WACV*, 2023. 2

[26] Eunjeong Jeong and Marios Kountouris. Personalized decentralized federated learning with knowledge distillation. *arXiv*, 2023. 2

[27] Zhanhong Jiang, Aditya Balu, Chinmay Hegde, and Soumik Sarkar. Collaborative deep learning in fixed topology networks. *NIPS*, 2017. 2

[28] Madhura Joshi, Ankit Pal, and Malaikannan Sankarasubbu. Federated learning for healthcare domain-pipeline, applications and challenges. *ACM Transactions on Computing for Healthcare*, 2022. 1

[29] Peter Kairouz, H Brendan McMahan, et al. Advances and open problems in federated learning. *arXiv*, 2019. 1, 2

[30] Jiawen Kang, Zehui Xiong, Dusit Niyato, Han Yu, Ying-Chang Liang, and Dong In Kim. Incentive design for efficient federated learning in mobile networks: A contract theory approach. In *IEEE VTS Asia Pacific Wireless Communications Symposium*, 2019. 2

[31] Zhao Kang, Guoxin Shi, Shudong Huang, Wenyu Chen, Xiaorong Pu, Joey Tianyi Zhou, and Zenglin Xu. Multi-graph fusion for multi-view spectral clustering. *Knowledge-Based Systems*, 2020. 3

[32] Can Karakus, Yifan Sun, Suhas Diggavi, and Wotao Yin. Straggler mitigation in distributed optimization through data encoding. *NIPS*, 2017. 2

[33] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *ICML*, 2020. 3

[34] Cao Ke-xing, Li Zhao-xing, Li Xin, and Lv Zhi-han. Weak connection edges independent discriminant of rapid spanning tree recommendation of social network community. *International Journal of Multimedia and Ubiquitous Engineering*, 2016. 3

[35] Simon Knight, Hung X Nguyen, Nickolas Falkner, Rhys Bowden, and Matthew Roughan. The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, 2011. 6

[36] Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian Stich. A unified theory of decentralized sgd with changing topology and local updates. In *ICML*, 2020. 2

[37] Jakub Konečnỳ, Brendan McMahan, and Daniel Ramage. Federated optimization: Distributed optimization beyond the datacenter. *arXiv*, 2015. 2

[38] Jakub Konecnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *CoRR*, 2016. 3

[39] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv*, 2016. 2

[40] Chengxi Li, Gang Li, and Pramod K Varshney. Decentralized federated learning via mutual knowledge transfer. *IEEE Internet of Things Journal*, 2021. 2

[41] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *CVPR*, 2021. 2

[42] Songze Li, Seyed Mohammadreza Mousavi Kalan, A Salman Avestimehr, and Mahdi Soltanolkotabi. Near-optimal straggler mitigation for distributed gradient methods. In *IEEE International Parallel and Distributed Processing Symposium Workshops*, 2018. 2

[43] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 2020. 1

[44] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *MLSys Conference*, 2020. 3

[45] Xiang Li, Wenhao Yang, Shusen Wang, and Zhihua Zhang. Communication-efficient local decentralized sgd methods. *arXiv*, 2019. 3

[46] Youjie Li, Mingchao Yu, Songze Li, Salman Avestimehr, Nam Sung Kim, and Alexander Schwing. Pipe-sgd: A decentralized pipelined sgd framework for distributed deep net training. *NIPS*, 2018. 2

[47] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *arXiv*, 2017. 2

[48] Xiangru Lian, Wei Zhang, Ce Zhang, and Ji Liu. Asynchronous decentralized parallel stochastic gradient descent. In *ICML*, 2018. 2

[49] Chang Liu, Chenfei Lou, Runzhong Wang, Alan Yuhan Xi, Li Shen, and Junchi Yan. Deep neural network fusion via graph matching with applications to model ensemble and federated learning. In *ICML*, 2022. 2

[50] Hongbin Liu, Han Zhou, Hao Chen, Yong Yan, Jianping Huang, Ao Xiong, Shaojie Yang, Jiewei Chen, and Shaoyong Guo. A federated learning multi-task scheduling mechanism based on trusted computing sandbox. *Sensors*, 2023. 1

[51] Quande Liu, Cheng Chen, Jing Qin, Qi Dou, and Pheng-Ann Heng. Feddg: Federated domain generalization on medical image segmentation via episodic learning in continuous frequency space. In *CVPR*, 2021. 1

[52] Ye Liu, Lifang He, Bokai Cao, S Yu Philip, Ann B Ragin, and Alex D Leow. Multi-view multi-graph embedding for brain network clustering analysis. In *AAAI*, 2018. 3

[53] Dongsheng Luo, Jingchao Ni, Suhang Wang, Yuchen Bian, Xiong Yu, and Xiang Zhang. Deep multi-graph clustering via attentive cross-graph association. In *International Conference on Web Search and Data Mining*, 2020. 3

[54] Mingqi Lv, Zhaoxiong Hong, Ling Chen, Tieming Chen, Tiantian Zhu, and Shouling Ji. Temporal multi-graph convolutional network for traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems*, 2020. 3

[55] Chenxin Ma, Virginia Smith, Martin Jaggi, Michael Jordan, Peter Richtárik, and Martin Takác. Adding vs. averaging in distributed primal-dual optimization. In *ICML*, 2015. 2

[56] Xiaosong Ma, Jie Zhang, Song Guo, and Wenchao Xu. Layer-wised model aggregation for personalized federated learning. In *CVPR*, 2022. 2

[57] Othmane Marfoq, Giovanni Neglia, Aurélien Bellet, Laetitia Kameni, and Richard Vidal. Federated multi-task learning under a mixture of distributions. In *NIPS*, 2021. 2

[58] Othmane Marfoq, Chuan Xu, Giovanni Neglia, and Richard Vidal. Throughput-optimal topology design for cross-silo federated learning. In *NIPS*, 2020. 2, 3, 4, 5, 6, 7, 8

[59] Sebastian Martschat. Multigraph clustering for unsupervised coreference resolution. In *Association for Computational Linguistics Proceedings of the Student Research Workshop*, 2013. 3

[60] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, 2017. 1, 2, 3

[61] B McMahan and D Ramage. Google ai blog: Federated learning: Collaborative machine learning without centralized training data, 2017. 2

[62] H Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *arXiv*, 2016. 2

[63] Frederic P. Miller, Agnes F. Vandome, and John McBrewster. Amazon web services. *Retrieved November*, 2011. 1, 6

[64] Jérôme Monnot, Vangelis Th Paschos, and Sophie Toulouse. Approximation algorithms for the traveling salesman problem. *Mathematical methods of operations research*, 2003. 4

[65] Angelia Nedić and Alex Olshevsky. Distributed optimization over time-varying directed graphs. *IEEE Transactions on Automatic Control*, 2014. 2

[66] Angelia Nedić, Alex Olshevsky, and Michael G Rabbat. Network topology and communication-computation tradeoffs in decentralized optimization. *IEEE*, 2018. 2

[67] Giovanni Neglia, Gianmarco Calbi, Don Towsley, and Gayane Vardoyan. The role of network topology for distributed machine learning. In *IEEE INFOCOM Conference on Computer Communications*, 2019. 2

[68] Anh Nguyen, Tuong Do, Minh Tran, Binh X Nguyen, Chien Duong, Tu Phan, Erman Tjiputra, and Quang D Tran. Deep federated learning for autonomous driving. *IV*, 2022. 1, 2

[69] Olusola T Odeyomi and Gergely Zaruba. Differentially-private federated learning with long-term budget constraints using online lagrangian descent. In *IEEE World AI IoT Congress*, 2021. 2

[70] Yi Ouyang, Bin Guo, Xing Tang, Xiuqiang He, Jian Xiong, and Zhiwen Yu. Learning cross-domain representation with multi-graph neural network. *arXiv*, 2019. 3

[71] Jung Wuk Park, Dong-Jun Han, Minseok Choi, and Jaekyun Moon. Sageflow: Robust federated learning against both stragglers and adversaries. *NIPS*, 2021. 2

[72] Robert Clay Prim. Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 1957. 2, 6

[73] Liangqiong Qu, Yuyin Zhou, Paul Pu Liang, Yingda Xia, Feifei Wang, Ehsan Adeli, Li Fei-Fei, and Daniel Rubin. Rethinking architecture design for tackling data heterogeneity in federated learning. In *CVPR*, 2022. 2

[74] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 2013. 2

[75] Zebang Shen, Aryan Mokhtari, Tengfei Zhou, Peilin Zhao, and Hui Qian. Towards more efficient stochastic decentralized learning: Faster convergence and sparse communication. In *ICML*, 2018. 2

[76] Geet Shingi. A federated learning based approach for loan defaults prediction. In *ICDMW*, 2020. 1

[77] Virginia Smith, Simone Forte, Ma Chenxin, Martin Takáč, Michael I Jordan, and Martin Jaggi. Cocoa: A general framework for communication-efficient distributed optimization. *Journal of Machine Learning Research*, 2018. 2

[78] Maja Stikic, Diane Larlus, and Bernt Schiele. Multi-graph based semi-supervised learning for activity recognition. In *International Symposium on Wearable Computers*, 2009. 3

[79] Hao Tang, Guoshuai Zhao, Xuxiao Bu, and Xueming Qian. Dynamic evolution of multi-graph based collaborative filtering for recommendation systems. *Knowledge-Based Systems*, 2021. 3

[80] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *CVPR*, 2018. 6

[81] Thijs Vogels, Lie He, Anastasiia Koloskova, Sai Praneeth Karimireddy, Tao Lin, Sebastian U Stich, and Martin Jaggi. Relaysum for decentralized deep learning on heterogeneous data. *NIPS*, 2021. 2

[82] Richard Walker. Implementing discrete mathematics: combinatorics and graph theory with mathematica. *The Mathematical Gazette*, 1992. 2

[83] Chunnan Wang, Xiang Chen, Junzhe Wang, and Hongzhi Wang. Atpfl: Automatic trajectory prediction model design under federated learning framework. In *CVPR*, 2022. 1

[84] Jianyu Wang and Gauri Joshi. Cooperative sgd: A unified framework for the design and analysis of communication-efficient sgd algorithms. In *ICLRW*, 2018. 2, 3, 5

[85] Jianyu Wang, Anit Kumar Sahu, Zhouyi Yang, Gauri Joshi, and Soummya Kar. Matcha: Speeding up decentralized sgd via matching decomposition sampling. In *Indian Control Conference*, 2019. 2, 3, 6

[86] Jiajun Wu, Steve Drew, Fan Dong, Zhuangdi Zhu, and Jiayu Zhou. Topology-aware federated learning in edge computing: A comprehensive survey. *arXiv*, 2023. 2

[87] Tianyu Wu, Kun Yuan, Qing Ling, Wotao Yin, and Ali H Sayed. Decentralized consensus optimization with asynchrony and delays. *IEEE Transactions on Signal and Information Processing over Networks*, 2017. 2

[88] Jie Xu, Benjamin S Glicksberg, Chang Su, Peter Walker, Jiang Bian, and Fei Wang. Federated learning for healthcare informatics. *Healthcare Informatics Research*, 2021. 1

[89] Haibo Yang, Minghong Fang, and Jia Liu. Achieving linear speedup with partial worker participation in non-iid federated learning. *ICLR*, 2021. 2

[90] Tianbao Yang, Shenghuo Zhu, Rong Jin, and Yuanqing Lin. Analysis of distributed stochastic dual coordinate ascent. *arXiv*, 2013. 2

[91] Ke Zhang, Carl Yang, Xiaoxiao Li, Lichao Sun, and Siu Ming Yiu. Subgraph federated learning with missing neighbor generation. *NIPS*, 2021. 2

[92] Lin Zhang, Yong Luo, Yan Bai, Bo Du, and Ling-Yu Duan. Federated learning for non-iid data via unified feature learning and optimization objective alignment. In *ICCV*, 2021. 2

[93] Lin Zhang, Li Shen, Liang Ding, Dacheng Tao, and Ling-Yu Duan. Fine-tuning global model via data-free knowledge distillation for non-iid federated learning. In *CVPR*, 2022. 2

[94] Zijian Zhang, Shuai Wang, Yuncong Hong, Liangkai Zhou, and Qi Hao. Distributed dynamic map fusion via federated learning for intelligent networked vehicles. In *ICRA*, 2021. 1

[95] Bo-Wei Zhao, Zhu-Hong You, Lun Hu, Leon Wong, Bo-Ya Ji, and Ping Zhang. A multi-graph deep learning model for predicting drug-disease associations. In *International Conference on Intelligent Computing*, 2021. 3

[96] Kun Zhu, Shuai Zhang, Jiusheng Li, Di Zhou, Hua Dai, and Zeqian Hu. Spatiotemporal multi-graph convolutional networks with synthetic data for traffic volume forecasting. *Expert Systems with Applications*, 2021. 3

[97] Ligeng Zhu, Hongzhou Lin, Yao Lu, Yujun Lin, and Song Han. Delayed gradient averaging: Tolerate the communication latency for federated learning. *NIPS*, 2021. 2