Multigraph Topology Design for Cross-Silo Federated Learning

Binh X. Nguyen[1], Tuong Do[1], Hien Nguyen[1], Vuong Pham[1], Toan Tran[3],
Erman Tjiputra[1], Quang Tran[1], Anh Nguyen[2]
[1]AIOZ, Singapore
[2]University of Liverpool, UK
[3]VinAI Research
{binh.xuan.nguyen,tuong.khanh-long.do,hien.nguyen,vuong.pham,erman.tjiputra,quang.tran}
@aioz.io
anh.nguyen@liverpool.ac.uk
v.toantm3@vinai.io

Abstract

Cross-silo federated learning utilizes a few hundred reliable data silos with high-speed access links to jointly train a model. While this approach becomes a popular setting in federated learning, designing a robust topology to reduce the training time is still 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 reducing the training time. We further propose a new distributed learning algorithm to use with our multigraph topology. The intensive experiments on public datasets show that our proposed method significantly reduces the training time compared with recent state-of-the-art topologies while ensuring convergence and maintaining the model's accuracy.

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 [35]. According to [21], 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 has become a popular training setting in different federated learning applications such as healthcare [7, 76], robotics [82, 57], medical imaging [8, 41], and finance [64].

In practice, federated learning is a new promising research direction where we can utilize the effectiveness of machine learning methods while respecting the user's privacy. The key challenges in federated learning include model convergence, communication congestion, or imbalance of data distributions in different silos [21]. A popular training method in federated learning is to set up a central node that orchestrates the training process and aggregates the contributions of all clients. 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-to-peer) federated learning approach. In decentralized federated learning, the communication is done via 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 the accuracy of the model.

In federated learning, the communication topology plays an important role. A more efficient topology leads to quicker convergence and reduces the training time, quantifying by the worst-case convergence bounds in the topology design [20, 55, 73]. 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 usage [77, 57, 22]. Designing a robust topology that can reduce the training time while maintaining the model accuracy is still an open problem in federated learning [21]. This paper aims to design a new topology for cross-silo federated learning, which is one of the most common training scenarios in practice.

Recently, different topologies have been proposed for

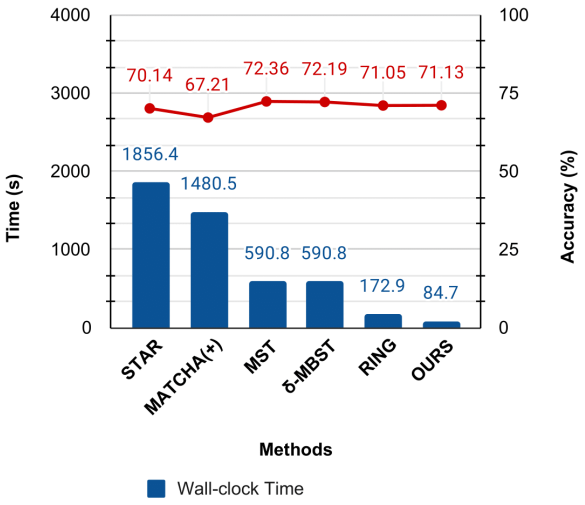


Figure 1. Comparison between different topologies on FEMNIST dataset and Exodus network [52]. The accuracy and total wall-clock training time (or overhead time) are reported after $6,400$ communication rounds.

cross-silo federated learning. In [3], the STAR topology is designed where the orchestrator averages all models throughout each communication round. The authors in [74] 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 their models. Marfoq *et al*. [47] introduces the RING topology with the largest throughput using the theory of 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 [74, 77], straggler effect [56, 60], or identical topology in all communication rounds [20, 47].

In this paper, we propose a new multigraph topology based on the recent RING topology [47] to reduce the training time for cross-silo federated learning. Our method first constructs the multigraph based on the overlay of RING topology. Then we parse this multigraph into simple graphs (i.e., graphs with only one edge between two nodes). We call each simple graph is 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. To ensure model convergence, we also adapt and propose a new distributed learning algorithm. The intensive experiments show that our proposed topology significantly reduces the training time in cross-silo federated learning (See Figure 1).

In summary, our contributions are as follows:

- We introduce a new multigraph topology that significantly reduces the training time in cross-silo federated learning.
- We adapt and propose a new distributed learning algorithm to ensure the model convergence when using our multigraph topology.
- We conduct intensive experiments to validate our results. Our code will be released for further study.

2. Literature Review

Federated Learning. Federated learning has been regarded as a system capable of safeguarding data privacy [31, 66, 78, 65, 12, 81, 33]. Contemporary federated learning has 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 Konecny *et al*. [29], as well as a widely circulated article from McMahan *et al*. [50]. Then, Yang *et al*. [79], Shalev-Shwartz *et al*. [62], Ma *et al*. [45], Jaggi *et al*. [19], and Smith *et al*. [67] extend the concept of federated learning and its related distributed optimization algorithms. Federated Averaging (FedAvg) [49] and its variations as FedSage [80] or DGA [85] 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 [14].

Decentralized Federated Learning. Decentralized (or peer-to-peer) federated learning allows each silo data to interact with its neighbors directly without a central node [14]. Due to its nature, decentralized federated learning does not have the communication congestion at the central node, however, optimizing a fully peer-to-peer network is a challenging task [54, 39, 16, 40, 74, 47, 46, 32]. Noticeably, the decentralized periodic averaging stochastic gradient descent [73] is proved to converge at a comparable rate to the centralized algorithm while allowing large-scale model training [75, 63, 58]. Recently, systematic analysis of the decentralized federated learning has been explored in [38, 9, 28].

Communication Topology. The topology has a direct impact on the complexity and convergence of federated learning [6].Many works have been introduced to improve the effectiveness of topology, including star-shaped topology [3, 31, 51, 49, 21] and optimized-shaped topology [56, 74, 47, 28, 1, 71]. Particularly, a spanning tree topology is discovered by [61] and improved by [47, 71] to reduce the training time. In [3], STAR topology is designed where an orchestrator averages model updates in each communication round. Wang *et al*. [74] introduce MATCHA to speed up the training process through decomposition sampling. Since the duration of a communication round is dic-

tated by stragglers [24, 34], Neglia *et al*. [56] explore how to choose the degree of a regular topology. Recently, Marfoq *et al*. [47] proposes RING topology for cross-silo federated learning using the theory of max-plus linear systems.

Multigraph. The definition of multigraph has been introduced as a traditional paradigm [10, 72]. A typical "graph" usually refers to a simple graph with no loops or multiple edges between two nodes. Different from a simple graph, multigraph allows multiple edges between two nodes. In deep learning, multigraph has been applied in different domains, including clustering [48, 43, 23], medical image processing [42, 83, 2], traffic flow prediction [44, 84], activity recognition [68], recommendation system [69], and cross-domain adaptation [59]. 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, we generally follow [73] to define the objective function for federated learning:

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

where $L_i(\mathbf{w}, \xi_i)$ is the loss of model $\mathbf{w} \in \mathbb{R}^d$. ξ_i is an input sample drawn from data at silo i. The coefficient $p_i > 0$ specifies the relative importance of each silo. Recently, different distributed algorithms have been proposed to optimize Eq. 1 [30, 49, 36, 74, 37, 73, 25]. In this work, the decentralized periodic averaging stochastic gradient descent (DPASGD) [73] is used since it allows local-update in each silo during the learning process. Local updating allows worker nodes to synchronize their local models on a regular basis, which paves the way for minimizing communication frequency and latency. In particular, the weight of silo i in each training round of DPASGD is updated as follow:

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

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

Connectivity and Overlay. Following [47], we consider the *connectivity* $\mathcal{G}_c = (\mathcal{V}, \mathcal{E}_c)$ as a graph that captures the possible direct communications among silos. Based on its definition, the connectivity is often a fully connected graph and is also a directed graph whenever the upload and download are set during learning. 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 [47] for more in-deep discussions.

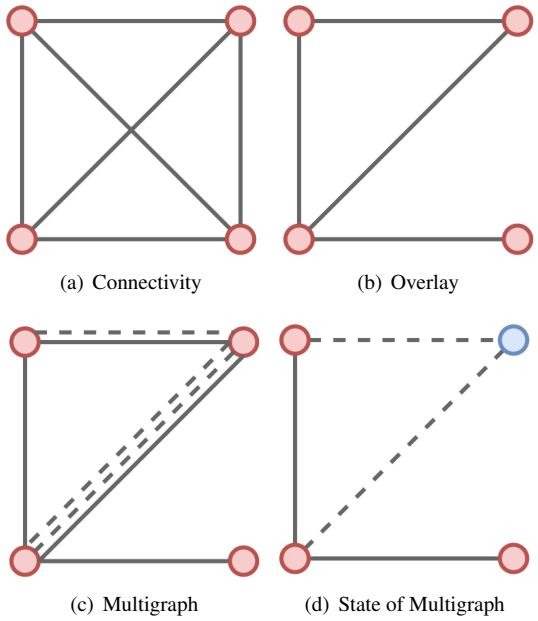


Figure 2. Example of connectivity, overlay, multigraph, and a state of our multigraph. Blue node is an isolated node. Dotted line denotes a weakly-connected edge.

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.

In the multigraph $\mathcal{G}_m$, the connection edge between two nodes has two types: strongly-connected edge and weakly-connected edge [26]. Under both strongly and weakly 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-dated 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 call each simple graph as a state $\mathcal{G}_m^s$ of the multigraph. The graph concepts are shown in Figure 2.

Isolated Node. A node is called isolated when all of its connections to other nodes are weakly-connected edges.

3.3. Delay and Cycle Time in Multigraph

Delay. Following [47], a delay to an edge $(i,j) \in \mathcal{E}_m$ is the time interval when node j receives the weight sending by node i, which can be defined by:

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

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; $A(i,j)$ 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 $A(i,j)$ is computed as:

$$A(i,j) = \min\left(\frac{C_{\mathrm{UP}}(i)}{|\mathcal{N}_i^-|}, \frac{C_{\mathrm{DN}}(j)}{|\mathcal{N}_i^+|}\right) \tag{4}$$

where C_{UP} and C_{DN} denote the upload and download link capacity. Note that the upload and download happen in parallel, and both are called access link capacity in general.

Since the 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 follow:

$$d_k(i,j) = \begin{cases} d_k(i,j), \\ \qquad \text{if } (e_k(i,j) = \mathbb{1} \text{ and } e_{k-1}(i,j) = \mathbb{1}) \text{ or } k = 0 \\ \max(u \times T_c(j), d_k(i,j) - d_{k-1}(i,j)), \\ \qquad \text{if } e_k(i,j) = \mathbb{1} \text{ and } e_{k-1}(i,j) = \mathbb{0} \\ \tau_k(\mathcal{G}_m) + d_{k-1}(i,j)), \\ \qquad \text{if } e_k(i,j) = \mathbb{0} \text{ and } e_{k-1}(i,j) = \mathbb{0} \\ \tau(\mathcal{G}_m)_k, \\ \qquad \text{otherwise} \end{cases} \tag{5}$$

where $e(i,j) = \mathbb{0}$ indicates weakly-connected edge, $e(i,j) = \mathbb{1}$ indicates strongly-connected edge; $\tau_k(\mathcal{G}_m)$ is the cycle time at the k-th computation round during the training process.

Cycle Time. The cycle time per round is the time required to complete a communication round [47]. In this work, we define that 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:

$$\tau(\mathcal{G}_m) = \frac{1}{k}\sum_{k=0}^{k-1}\left(\max_{j \in \mathcal{N}_i^{++} \cup \{i\}, \forall i \in \mathcal{V}} \left(d_k\left(j,i\right)\right)\right) \tag{6}$$

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

4. Methodology

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, our method does not choose isolated nodes randomly, but relies on the delay time. In our design, each isolated node has a long delay time in a current communication round. However, in the next round, its delay time will be updated using Eq. 5, and therefore it can become a normal node. This strategy allows us to reduce the waiting time with isolated nodes, while ensuring that isolated nodes can become normal nodes and contribute to the training in the next communication round.

4.1. Multigraph Construction

Algorithm 1 describes our methods to generate the multigraph $\mathcal{G}_m$ with multiple edges between silos. The algorithm takes the overlay $\mathcal{G}_o$ as input. As in [47], we use the Christofides algorithm [53] to obtain the overlay.

In Algorithm 1, we focus on establishing 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 $d_{\min}$ over all 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 the 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 the 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

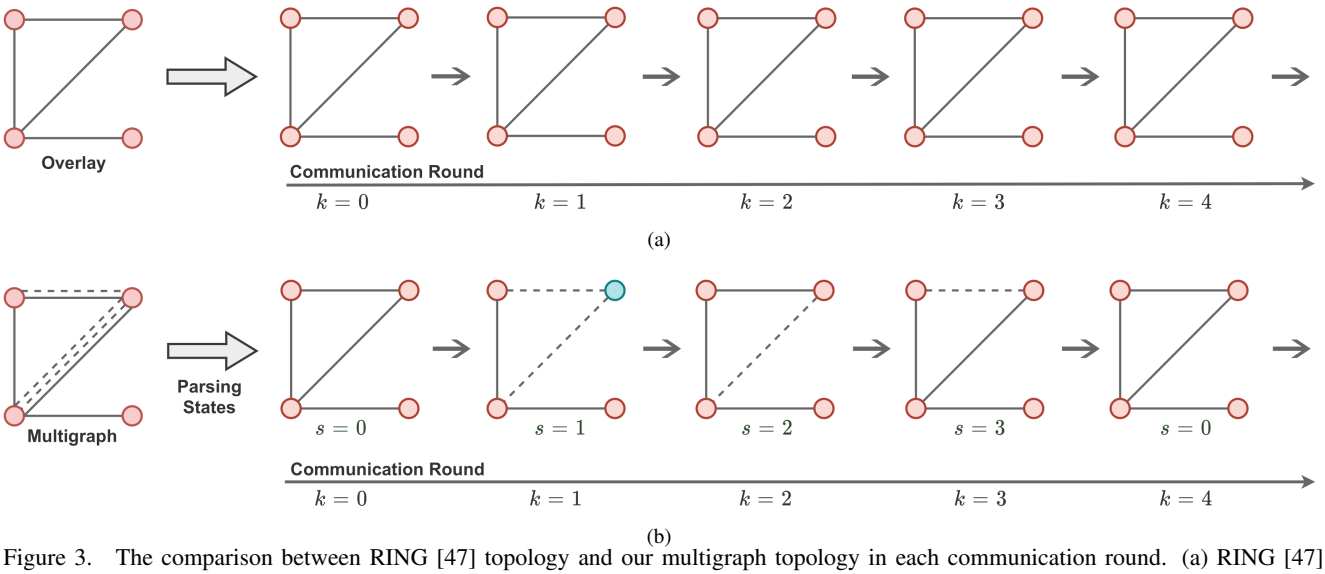


Figure 3. The comparison between RING [47] topology and our multigraph topology in each communication round. (a) RING [47] uses the same overlay in each round. (b) Our proposed multigraph is parsed into different graph states. Each graph state is used in a communication round. Lines denote strongly-connected edges, dotted lines denote weakly-connected ones, and the blue color indicates isolated nodes.

Algorithm 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 $\mathcal{L}$.

// Compute delay in overlay
$D_o \leftarrow$ NULL
foreach $(i, j) \in \mathcal{E}_o$ **do**
  $d(i, j) \leftarrow$ Using Eq. 3
  Append $d(i, j)$ into D_o
// Construct multigraph
$d_{min} = \min(D_0)$ // find smallest delay
$\mathcal{E}_m \leftarrow$ NULL // multiset of edges
$\mathcal{L}[|\mathcal{V}|, |\mathcal{V}|] \leftarrow \{0\}$
foreach $(i, j) \in \mathcal{E}_o$ **do**
  $n(i, j) = \min\left(t, \text{round}\left(\frac{d(i,j)}{d_{\min}}\right)\right)$ // find number of edges for (i,j)
  $\mathcal{E}_t \leftarrow$ NULL // temporary edge set
  Append $e(i, j) = \mathbb{1}$ into $\mathcal{E}_t$
  foreach $(n(i, j) - 1)$ **do**
    Append $e(i, j) = \mathbb{0}$ into $\mathcal{E}_t$
  Append $\mathcal{E}_t$ into $\mathcal{E}_m$
  $\mathcal{L}[i, j] \leftarrow n(i, j)$.
return $\mathcal{G}_m = (\mathcal{V}, \mathcal{E}_m)$; $\mathcal{L}$

round to perform model aggregation. In each graph state, our goal is to identify the isolated nodes. During the training, the 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) [13]. 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

The original DPASGD algorithm [73] can not be directly used with our multigraph because the learning process will be terminated when it first meets an isolated node. To overcome this problem, we introduce an upgraded version of DPASGD, namely, DPASGD++ (See Algorithm 3 for details). 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.

Formally, the weight in DPASGD++ is updated as:

Algorithm 2: Multigraph Parsing

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

$s_{max} \leftarrow$ LCM [13] $(\{ |\mathcal{E}_m(i,j)| : i,j \in \mathcal{V}\})$
$\bar{\mathcal{L}} = \mathcal{L}$; $\bar{\mathcal{E}}_m^s \leftarrow$ NULL
// Establish states
for $s = 0$ **to** s_{max} **do**
  $\mathcal{E}_t \leftarrow$ NULL // temporary edge set
  foreach $(i,j) \in \mathcal{E}_m$ **do**
    if $\bar{\mathcal{L}}[i,j] = \mathcal{L}[i,j]$ **then**
      Append $e(i,j) = \mathbb{1}$ into $\mathcal{E}_t$
    else
      Append $e(i,j) = \mathbb{0}$ into $\mathcal{E}_t$
    if $\bar{\mathcal{L}}[i,j] = 1$ **then**
      $\bar{\mathcal{L}}[i,j] = \mathcal{L}[i,j]$
    else
      $\bar{\mathcal{L}}[i,j] -= 1$
  Append $\mathcal{E}_t$ into $\bar{\mathcal{E}}_m^s$
return $\mathcal{S} = \{\mathcal{G}_m^s = (\mathcal{V}, \mathcal{E}_m^s)\}$ by using $\bar{\mathcal{E}}_m^s$.

Algorithm 3: DPASGD++ Algorithm

Input: List of multigraph states $\mathcal{S}$;
Initial weight $\mathbf{w}_i(0)$ for each silo i;
Maximum training round K.

$c = 0$ // states counting variable
for $k = 0$ **to** $K-1$ **do**
  $\mathcal{G}_{m_c}^s \leftarrow$ Select c-th $\mathcal{G}_m^s$ in $\mathcal{S}$
  $c = c + 1$
  if $c \geq$ *sizeof*$(\mathcal{S})$ **then**
    $c = 0$
  for $i = 0$ **to** N **do**
    $\mathcal{N}_i^{++} \leftarrow$ strongly-connected edges list of i using $\mathcal{G}_{m_i}^s$.
  // The loop below is parallel
  foreach *silo* $i \in N$ **do**
    for $\flat = 0$ **to** u **do**
      $m_\flat \leftarrow$ Sampling from local dataset of i
      $\mathbf{w}_i(k+1) \leftarrow$ Update model using Eq. 7.

$$\mathbf{w}_i\left(k+1\right) = \begin{cases} \sum_{j \in \mathcal{N}_i^{++} \cup \{i\}} \mathbf{A}_{i,j} \mathbf{w}_j\left(k-h\right), \\ \qquad \text{if k} \equiv 0 \left(\text{mod } u+1\right) \,\&\, \left|\mathcal{N}_i^{++}\right| > 1, \\ \mathbf{w}_i\left(k\right) - \alpha_k \frac{1}{b} \sum_{h=1}^{b} \nabla L_i\left(\mathbf{w}_i\left(k\right), \xi_i^{(h)}\left(k\right)\right), \\ \qquad \text{otherwise.} \end{cases} \tag{7}$$

where $(k-h)$ is the index of the considered weights; $\mathbf{A}_{i,j} = \frac{1}{|\mathcal{N}_i^{++}|}$; h is initialized to 0 and is changed when the condition in Eq. 8 is met, i.e.,

$$h = h + 1, \qquad \text{if } \; e_{k-h}(i,j) = \mathbb{0} \tag{8}$$

Through Eq. 7 and Eq. 8, 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. We next theoretically show in the following propositions that our proposed DPASGD++ is a general case for DPASGD [73], and will become the original DPASGD with some certain conditions.

Proposition 1. *Assuming that all states of multigraph contains only strongly-connected edge, i.e.,* $e(i,j) = \mathbb{1}, \forall (i,j) \in \mathcal{G}_m$. *Then, DPASGD++ described in Eq. 7 becomes the original DPASGD [73].*

Proof. In deed, if $e(i,j) = \mathbb{1}, \forall (i,j) \in \mathcal{G}_m$, from Eq. 8 and Eq. 7 we have $h = 0$, and then DPASGD++ becomes the original DPASGD [73]. □

Proposition 2. *Assuming that all states of multigraph contains only weakly-connected edge (all nodes are isolated), i.e.,* $e(i,j) = \mathbb{0}, \forall (i,j) \in \mathcal{G}_m$ *and* $\left|\mathcal{N}_i^{++}\right| = 1, \forall i \in \mathcal{G}_m$. *Then, DPASGD++ becomes DPASGD [73] when DPASGD has* $u \to \infty$, *and we have*

$$\mathbf{w}_i\left(k+1\right) = \mathbf{w}_i\left(k\right) - \alpha_k \frac{1}{b} \sum_{h=1}^{b} \nabla L_i\left(\mathbf{w}_i\left(k\right), \xi_i^{(h)}\left(k\right)\right) \tag{9}$$

Proof. Given the assumption that $\left|\mathcal{N}_i^{++}\right| = 1, \forall i \in \mathcal{G}_m$, by combining this with Eq. 7, and noting that the first case in Eq. 7 is violated, we obtain

$$\mathbf{w}_i\left(k+1\right) = \mathbf{w}_i\left(k\right) - \alpha_k \frac{1}{b} \sum_{h=1}^{b} \nabla L_i\left(\mathbf{w}_i\left(k\right), \xi_i^{(h)}\left(k\right)\right).$$

□

5. Experiments

5.1. Experimental Setup

Implementation. We use PyTorch with the MPI backend in our implementation. The maximum number of edges between two nodes t is set to 5 in all experiments. The training is conducted using NVIDIA Tesla P100 16Gb GPUs.

Network. Following [47], we consider five distributed networks in our experiments: Exodus, Ebone, Géant, Amazon [52] and Gaia [18]. The Exodus, Ebone, and Géant

are from the Internet Topology Zoo [27]. The Amazon and Gaia network are synthetic networks that are constructed using the geographical locations of the data centers. Table 1 shows the statistic of these networks.

Datasets. We use three standard federated datasets in our experiments to evaluate our multigraph topology: Sentiment140 [11], iNaturalist [70], and FEMNIST [4]. All datasets and the pre-processing process are conducted by following recent works [74] and [47]. Table 2 shows the dataset setups in details.

Network	#Silos	#Maximum Links
Gaia [18]	11	55
Amazon [52]	22	231
Géant [27]	40	61
Exodus [27]	79	147
Ebone [27]	87	161

Table 1. The network setups in our experiments.

Dataset	FEMNIST [4]	Sentiment140 [11]	iNaturalist [70]
#Samples	805M	1,600M	450M
Model	CNN [47]	LSTM [17]	ResNet18 [15]
#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 implementation details in our experiments. The model size is in Mbits.

Baselines. We compare our multigraph topology with recent state-of-the-art topology designs: STAR [3], MATCHA [74], MATCHA(+) [47], MST [61], δ-MBST [47], and RING [47]. MST [61] and δ-MBST [47] are topologies based on the minimum spanning tree. STAR [3] is a typical client-server setup, and RING [47] is a topology based on the overlay graph.

5.2. Results

Table 3 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 [47], 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 to reduce the cycle and training time in federated learning.

From Table 3, 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 [47].

5.3. Ablation Study

Convergence Analysis. Figure 4 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 topology converges faster than other methods while maintaining the model accuracy. This confirms that, although our method utilizes isolated nodes to reduce the cycle time, the overall model accuracy is well-preserved. We observe the same results in other datasets and network setups.

Access Link Capacities Analysis. Following [47], we analyse the effect of access link capacity on our multigraph topology. Access link capacity is related to the bandwidth when packages are transmitted between silos. Figure 5 shows the results under Exodus network and FEMNIST dataset in two scenarios: all access links have the same 1 Gbps capacity and one orchestra node has a fixed 10 Gbps access link capacity. From Figure 5, we can see that our multigraph topology slightly outperforms RING [47] when the link capacity is low. However, when the capacity between silos is high, then our method clearly improves over RING [47]. In all setups, our method archives the best cycle time and training time.

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 4 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 results with other topology designs.

Cycle Time and Accuracy Trade-off. In our method, the maximum number of edges between two nodes t mainly affects the number of isolated nodes. This leads to a trade-off between the model accuracy and cycle time. Table 5 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 [47]. When t is set higher, we can increase the number of isolated nodes, hence decreasing the 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. From Table 5, we set $t = 5$ to balance the trade-off between the cycle time

Table 3. Cycle time (ms) comparison between different typologies. (↓ ∘) indicates our reduced times compared with other methods.

Dataset	Network	Topology Design						
		STAR [3]	**MATCHA [74]**	**MATCHA(+) [47]**	**MST [61]**	**δ-MBST [47]**	**RING [47]**	**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 (↓ 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 (↓ 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**

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 4. 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.

Topology	t	**Cycle time (ms)**	**Acc(%)**
RING [47]	-	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

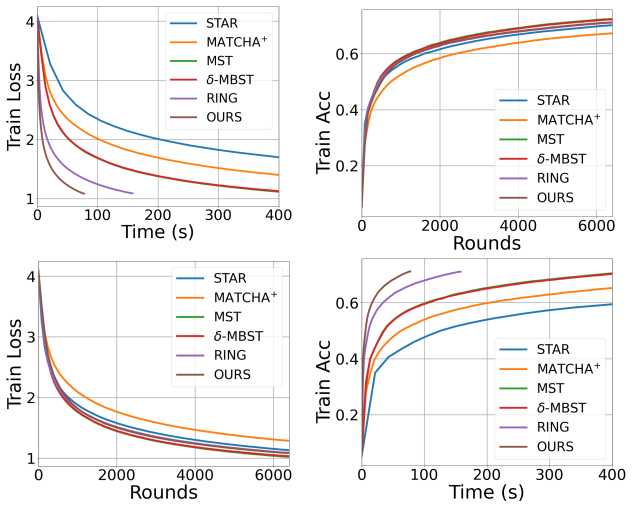
Table 5. Cycle time and accuracy trade-off with different value of t, i.e., the maximum number of edges between two nodes.

Methods	**Criteria**	**#Removed Nodes**	**Cycle Time (ms)**	**Acc (%)**
RING [47]	*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 6. The cycle time and accuracy of our multigraph vs. RING with different criteria.

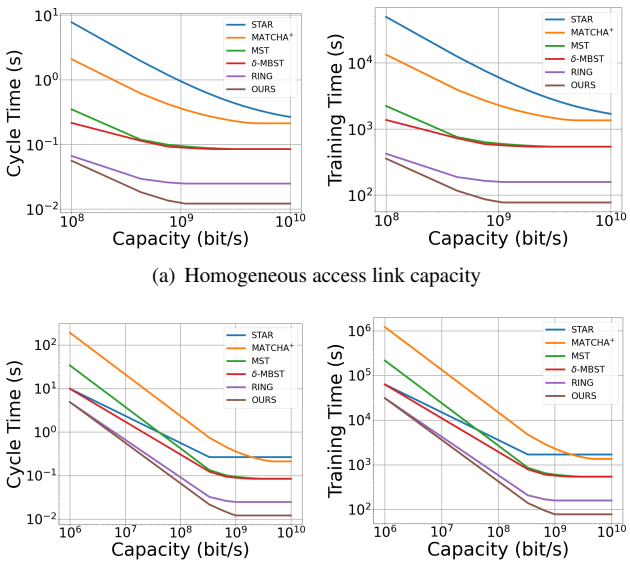
and the model accuracy.

Multigraph vs. RING vs. Random Strategy. In our proposed multigraph, the existence of isolated nodes plays an important role as we can skip the model aggregation step in the isolated nodes. The multigraph and isolated nodes are generated using our Algorithm 1 and Algorithm 2. In practice, we can have a trivial solution to create isolated nodes by randomly removing some nodes from the overlay of RING [47]. Table 6 shows the experiment results in two scenarios on FEMNIST dataset [4] and Exodus Network [27]: i) Randomly remove some silos in the overlay of RING, and ii) Remove most inefficient silos (i.e., silos with the longest delay) in the overlay of RING. Note that, in RING, one overlay is used in all communication rounds.



(a) Train Loss (b) Train Accuracy

Figure 4. Convergence analysis of our multigraph under communication rounds (top row) and wall-clock time (bottom row). Best viewed in color.



(a) Homogeneous access link capacity

(b) Orchestra node with 10 Gbps access capacity

Figure 5. The effect of access link capacity on cycle time and training time of different approaches. (a) All access links have the same 1 Gbps capacity. (b) One orchestra node has a fixed 10 Gbps access link capacity. The training time is counted until the training process of all setups reaches $6,400$ communication rounds.

For random setups, we do the training 5 times for each setup and report the average. From Table 6, we can see that when we apply two aforementioned scenarios to the overlay of RING, we can significantly reduce the cycle time. However, the accuracy of the model also significantly drops. This experiment shows that randomly removing some nodes from the overlay of RING is a trivial solution, and can not maintain accuracy. On the other hand, our multigraph can reduce the cycle time of the model, and maintain the accuracy at the same time. This is because, in our multigraph, we 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 the normal nodes and contribute to the final model.

Limitation. Since our multigraph is designed based on RING [47] overlay, our method inherits both the strength and weakness of RING. We can see that the "lower bound" of our multigraph is the overlay of RING when there are no isolated nodes. In this case, all states in our multigraph are the input overlay. Hence, there is no improvement. Furthermore, compared to RING [47], our multigraph is more sensitive to the low bandwidth capacity setup (Figure 5).

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

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

[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.

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

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

[7] Yiqiang Chen, Xin Qin, Jindong Wang, Chaohui Yu, and Wen Gao. Fedhealth: A federated transfer learning framework for wearable healthcare. *IEEE Intelligent Systems*, 2020.

[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.

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

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

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

[12] 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.

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

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

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

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

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

[18] 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.

[19] 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.

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

[21] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv*, 2019.

[22] 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.

[23] 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.

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

[25] 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.

[26] 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.

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

[28] 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.

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

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

[31] 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.

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

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

[34] 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.

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

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

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

[38] 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.

[39] 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.

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

[41] 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.

[42] 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.

[43] 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.

[44] 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.

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

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

[47] Othmane Marfoq, Chuan Xu, Giovanni Neglia, and Richard Vidal. Throughput-optimal topology design for cross-silo federated learning. In *NIPS*, 2020.

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

[49] 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.

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

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

[52] Frederic P. Miller, Agnes F. Vandome, and John McBrewster. *Amazon Web Services*. Alpha Press, 2010.

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

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

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

[56] 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.

[57] 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. In *33rd IEEE Intelligent Vehicles Symposium*, 2022.

[58] 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.

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

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

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

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

[63] 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.

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

[65] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *ACM SIGSAC conference on computer and communications security*, 2015.

[66] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. Federated multi-task learning. In *NIPS*, 2017.

[67] 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.

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

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

[70] 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.

[71] 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.

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

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

[74] 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.

[75] 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.

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

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

[78] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology*, 2019.

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

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

[81] 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.

[82] 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.

[83] 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.

[84] 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.

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