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

Tuong Do†, Binh X. Nguyen†, Vuong Pham, Toan Tran, Erman Tjiputra, Quang D.Tran, Anh Nguyen

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 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].

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-

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.
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:

\[
\min_{w \in \mathbb{R}^d} \sum_{i=1}^{N} p_i E_{\xi_i} [L_i (w, \xi_i)] ,
\]

where \( L_i (w, \xi_i) \) is the loss of model parameterized by the weight \( w \in \mathbb{R}^d \), \( \xi_i \) 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:

\[
w_i (k + 1) = \begin{cases} \sum_{j \in N_i^+ \cup \{i\}} A_{i,j} w_j (k), & \text{if } k \equiv 0 \mod u + 1, \\ w_i (k) - \alpha_k \frac{1}{b} \sum_{h=1}^{b} \nabla L_i \left( w_i (k), \xi_i^{(h)} (k) \right), & \text{otherwise.} \end{cases}
\]

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, \( A \in \mathbb{R}^{N \times N} \) is a consensus matrix with non-negative weights, and \( N_i^+ \) 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 \( \mathcal{E}_o \subseteq \mathcal{E}_c \). 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.

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 \( N_i^- \) or download models from their in-neighbours \( N_i^+ \). 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 \( \mathcal{G}_m^i \) 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:

\[
d(i, j) = u \times T_e (i) + l(i, j) + \frac{M}{O(i, j)},
\]

where \( T_e (i) \) 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 \( O(i, j) = \min \left( \frac{C_{UP}(i)}{|N_i^-|}, \frac{C_{DN}(i)}{|N_i^+|} \right) \) 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 \( d_{k+1}(i, j) \), with \( k \) is the \( k \)-th communication round during the training process, as:

\[
d_{k+1}(i, j) =
\begin{cases}
  d_k(i, j), & \text{if } e_{k+1}(i, j) = 1 \text{ and } e_k(i, j) = 1 \\
  \max(u \times T_c(i, j), d_k(i, j) - d_{k-1}(i, j)), & \text{if } e_{k+1}(i, j) = 1 \text{ and } e_k(i, j) = 0 \\
  \tau_k(G_m) + d_{k-1}(i, j)), & \text{if } e_{k+1}(i, j) = 0 \text{ and } e_k(i, j) = 0 \\
  \tau_k(G_m), & \text{otherwise}
\end{cases}
\]  

where \( e(i, j) = 0 \) is weakly-connected edge, \( e(i, j) = 1 \) is strongly-connected edge; \( \tau_k(G_m) \) is the cycle time at the \( k \)-th communication 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:

\[
\tau(G_m) = \frac{1}{k} \sum_{k=0}^{k-1} \max_{i, j \in V} \left\{ d_k(j, i) \right\},
\]

where \( N_i^+ \) 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 \( G_m \) with multiple edges between silos. The algorithm takes the overlay \( G_o \) 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 \( G_m \) into multiple graph states \( G_m^k \). Graph states are essential to identify the connection status of silos in a specific communication round.
Algorithm 2: Multigraph Parsing.
Input: Multigraph \( G_m = (V, E_m) \).
List edge numbers between silo pairs \( L \).
Output: List of multigraph states \( S = \{G_m^s = (V, E_m^s)\} \).
1 \( s_{\text{max}} \leftarrow \text{Compute maximum number of distinct state in } G_m \text{ by using Least Common Multiple} [15]\).
2 \( \hat{L} \leftarrow \text{Establish the dynamic list that tracks the number of edges between silo pairs during changing graph state and is initialized by input list edge } L \).
3 \( \hat{E}_m^s \leftarrow \text{Establish the list of all possible extracted states.} \)
4 for state \( s = 0 \) to \( s_{\text{max}} \) do
5 \( \hat{E}_t \leftarrow \text{Establish temporary edge set.} \)
6 foreach edge \( e(i, j) \in E_m \) do
7 if \( \hat{L}[i, j] = \hat{L}[i, j] \) then
8 Append strong connected-edge \( e(i, j) = 1 \) into edge set \( \hat{E}_t \).
9 else
10 Append weak-connected edge \( e(i, j) = 0 \) into edge set \( \hat{E}_t \).
11 if \( \hat{L}[i, j] = 1 \) then
12 Update the status of dynamic list \( \hat{L} \) using input \( L \) through \( \hat{L}[i, j] = \hat{L}[i, j] \).
13 else
14 Reduce the corresponding number of edges by \( \hat{L}[i, j] = 1 \).
15 Append edge set \( \hat{E}_t \) into \( \hat{E}_m^s \).
16 return List of multigraph state \( S = \{G_m^s = (V, E_m^s)\} \) by using list of possible extracted states \( \hat{E}_m^s \).

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.

4.3. Multigraph Training

In each communication round, a state graph \( G_m^s \) is selected in a sequence that identifies the topology design used for training. We then collect all strongly-connected edges in the graph state \( G_m^s \) 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]:

\[
\begin{align*}
    w_i(k+1) &= \begin{cases} 
        \sum_{j \in \mathcal{N}_i^{++} \cup \cup_i} A_{i,j} w_j(k-h) , & \text{if } k \equiv 0 \mod (u+1) \text{ & } |\mathcal{N}_i^{++}| > 1 , \\
        w_i(k) - \alpha_{k+1} \sum_{h=1}^h \nabla L_i(w_i(k); \xi_i^{(h)}(k)) , & \text{otherwise.}
    \end{cases}
\end{align*}
\]

(6)

where \((k-h)\) is the index of the considered weights; \(h\) is initialized to 0 and \(h = h + 1\), if \(e_{k-h}(i, j) = 0\). 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.
Table 1. Cycle time (ms) comparison between different topologies. (↓ o) indicates our reduced times compared with other methods.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FEMNIST</td>
<td>Gaia</td>
<td>289.8 (↓18.5)</td>
<td>166.4 (↓10.6)</td>
<td>166.4 (↓10.6)</td>
<td>77.2 (↓4.9)</td>
<td>77.2 (↓4.9)</td>
<td>57.2 (↓3.6)</td>
<td>15.7</td>
</tr>
<tr>
<td></td>
<td>Amazon</td>
<td>98.8 (↓7.3)</td>
<td>57.7 (↓4.2)</td>
<td>57.7 (↓4.2)</td>
<td>28.7 (↓2.1)</td>
<td>28.7 (↓2.1)</td>
<td>20.3 (↓1.5)</td>
<td>13.6</td>
</tr>
<tr>
<td></td>
<td>Géant</td>
<td>132.2 (↓11.0)</td>
<td>46.9 (↓3.9)</td>
<td>102.3 (↓8.5)</td>
<td>40.1 (↓3.3)</td>
<td>40.1 (↓3.3)</td>
<td>27.7 (↓2.3)</td>
<td>12.0</td>
</tr>
<tr>
<td></td>
<td>Exodus</td>
<td>265.2 (↓21.9)</td>
<td>84.7 (↓7.0)</td>
<td>211.5 (↓17.5)</td>
<td>84.4 (↓7.0)</td>
<td>84.4 (↓7.0)</td>
<td>24.7 (↓2.0)</td>
<td>12.1</td>
</tr>
<tr>
<td></td>
<td>Ebene</td>
<td>190.9 (↓15.0)</td>
<td>61.5 (↓4.8)</td>
<td>112.6 (↓8.9)</td>
<td>60.9 (↓4.8)</td>
<td>60.9 (↓4.8)</td>
<td>18.5 (↓1.5)</td>
<td>12.7</td>
</tr>
</tbody>
</table>

Table 1. Cycle time (ms) comparison between different topologies. (↓ o) 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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Samples</th>
<th>FEMNIST</th>
<th>Sentiment140</th>
<th>iNaturalist</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEMNIST</td>
<td>805M</td>
<td>1,600M</td>
<td>450M</td>
<td></td>
</tr>
<tr>
<td>Sentiment140</td>
<td>128</td>
<td>478M</td>
<td>512M</td>
<td>16</td>
</tr>
<tr>
<td>iNaturalist</td>
<td>462</td>
<td>18.38</td>
<td>42.88</td>
<td></td>
</tr>
</tbody>
</table>

### 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.
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 cannot 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.

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

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].

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

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-
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 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.