D-Cliques: Compensating for Data Heterogeneity with Topology in Decentralized Federated Learning

Aurélien Bellet*
Inria, Lille, France
aurelien.bellet@inria.fr

Anne-Marie Kermarrec
EPFL, Switzerland
anne-marie.kermarrec@epfl.ch

Erick Lavoie
EPFL, Switzerland
erick.lavoie@epfl.ch

***Abstract*—The convergence speed of machine learning models trained with Federated Learning is significantly affected by heterogeneous data partitions, even more so in a fully decentralized setting without a central server. In this paper, we show that the impact of label distribution skew, an important type of data heterogeneity, can be significantly reduced by carefully designing the underlying communication topology. We present D-Cliques, a novel topology that reduces gradient bias by grouping nodes in sparsely interconnected cliques such that the label distribution in a clique is representative of the global label distribution. We also show how to adapt the updates of decentralized SGD to obtain unbiased gradients and implement an effective momentum with D-Cliques. Our extensive empirical evaluation on MNIST and CIFAR10 validates our design and demonstrates that our approach achieves similar convergence speed as a fully-connected topology, while providing a significant reduction in the number of edges and messages. In a 1000-node topology, D-Cliques require 98% less edges and 96% less total messages, with further possible gains using a small-world topology across cliques.**

***Index Terms*—Decentralized Learning, Federated Learning, Topology, Heterogeneous Data, Stochastic Gradient Descent**

I. Introduction

Machine learning is currently shifting from a *centralized* paradigm, where training data is located on a single machine or in a data center, to *decentralized* ones in which data is processed where it was naturally produced. This shift is illustrated by the rise of Federated Learning (FL) [1]. FL allows several parties (hospitals, companies, personal devices...) to collaboratively train machine learning models on their joint data without centralizing it. Not only does FL avoid the costs of moving data, but it also mitigates privacy and confidentiality concerns [2]. Yet, working with natural data distributions introduces new challenges for learning systems, as local datasets reflect the usage and production patterns specific to each participant: in other words, they are *heterogeneous*. An important type of data heterogeneity encountered in federated classification problems, known as *label distribution skew* [2], [3], occurs when the frequency of different classes of examples varies significantly across local datasets. A key challenge in FL is to design algorithms that can efficiently deal with such heterogeneous data distributions [2], [4], [5], [3].

Federated learning algorithms can be classified into two categories depending on the underlying network topology they run on. In server-based FL, the network is organized according to a star topology: a central server orchestrates the training process by iteratively aggregating model updates received from the participants (*clients*) and sending back the aggregated model [1]. In contrast, fully decentralized FL algorithms operate over an arbitrary network topology where participants communicate only with their direct neighbors in the network. A classic example of such algorithms is Decentralized SGD (D-SGD) [6], in which participants alternate between local SGD updates and model averaging with neighboring nodes.

In this paper, we focus on fully decentralized algorithms as they can generally scale better to the large number of participants seen in "cross-device" applications [2]. Effectively, while a central server may quickly become a bottleneck as the number of participants increases, the topology used in fully decentralized algorithms can remain sparse enough such that all participants need only to communicate with a small number of other participants, i.e. nodes have small (constant or logarithmic) degree [6]. In the homogeneous setting where data is independent and identically distributed (IID) across nodes, recent work has shown both empirically [6], [7] and theoretically [8] that sparse topologies like rings or grids do not significantly affect the convergence speed compared to using denser topologies.

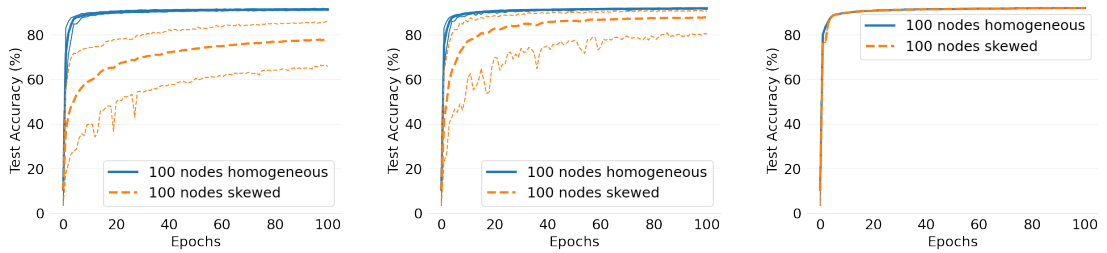
In contrast to the homogeneous case, our experiments demonstrate that *the impact of topology is extremely significant under heterogeneous data*. This phenomenon is illustrated in Figure 1: we observe that under label distribution skew, using a sparse topology clearly jeopardizes the convergence speed of decentralized SGD.[1] This strong impact of data heterogeneity and its possible interplay with the topology is not well captured by current theoretical analyses of decentralized FL, as they model heterogeneity by some (unknown) constant that bounds the variance of local gradients, independently of the topology [6], [7], [8], [9]. Motivated by the above empirical observations, in this paper we address the following question:

Can we design sparse topologies with convergence speed similar to a fully connected network for problems involving many participants with label distribution skew?

Specifically, we make the following contributions: (1) We propose D-Cliques, a sparse topology in which nodes are organized in interconnected cliques (i.e., locally fully-connected sets of nodes) such that the joint label distribution of each clique is close to that of the global distribution; (2) We design

*Authors in alphabetical order.

[1]Unlike in centralized FL [1], [5], [3], this happens even when nodes perform a single local update before averaging the model with their neighbors.



(a) Ring topology (b) Grid topology (c) Fully-connected topology

Fig. 1: Convergence speed of decentralized SGD with and without label distribution skew for different topologies. The task is logistic regression on MNIST (see Section IV-A for details on the experimental setup). Bold lines show the average test accuracy across nodes while thin lines show the minimum and maximum accuracy of individual nodes. While the effect of topology is negligible for homogeneous data, it is very significant in the heterogeneous case. On a fully-connected network, both cases converge similarly.

Greedy Swap, a randomized greedy algorithm for constructing such cliques efficiently; (3) We introduce Clique Averaging, a modified version of the standard D-SGD algorithm: Clique Averaging decouples gradient averaging, used for optimizing local models, from distributed averaging, used to ensure that all models converge, thereby reducing the bias introduced by inter-clique connections; (4) We show how Clique Averaging can be used to implement unbiased momentum that would otherwise be detrimental in the heterogeneous setting; (5) Through an extensive experimental study on decentralized learning of linear models and deep convolutional networks on MNIST and CIFAR10 datasets, we validate our various design choices and demonstrate that our approach is able to remove the effect of label distribution skew while maintaining a sparse topology; (6) Finally, we demonstrate the scalability of our approach by considering up to 1000-node networks, in contrast to most previous work on fully decentralized learning which performs empirical evaluations on networks with at most a few tens of nodes [10], [8], [11], [12], [13].

For instance, our results show that under strong label distribution skew, using D-Cliques in a 1000-node network requires 98% less edges (18.9 vs 999 edges per participant on average) to obtain a similar convergence speed as a fully-connected topology, thereby yielding a 96% reduction in the total number of required messages (37.8 messages per round per node on average instead of 999). An additional 22% improvement is possible when using a small-world inter-clique topology, with further potential gains at larger scales through a quasilinear $O(n \log n)$ scaling in the number of nodes n. We also show that D-Cliques empirically provide faster and more robust convergence than other topologies with a similar number of edges, such as random graphs and the exponential graphs recently promoted in [9].

The rest of this paper is organized as follows. We first describe the problem setting in Section II. We then present the design of D-Cliques in Section III. Section IV compares D-Cliques to different topologies and algorithmic variations to demonstrate their benefits, constructed with and without Greedy Swap in an extensive experimental study. Finally, we review some related work in Section V, and conclude with promising directions for future work in Section VI.

II. Problem Setting

a) Objective: We consider a set $N = \{1, \ldots, n\}$ of n nodes seeking to collaboratively solve a classification task with L classes. We denote a labeled data point by a tuple (x, y) where x represents the data point (e.g., a feature vector) and $y \in \{1, \ldots, L\}$ its label. Each node has access to a local dataset that follows its own local distribution D_i which may differ from that of other nodes. In this work, we tackle *label distribution skew*: formally, this means that the probability of (x, y) under the local distribution D_i of node i, denoted by $p_i(x, y)$, decomposes as $p_i(x, y) = p(x|y)p_i(y)$, where $p_i(y)$ may vary across nodes. We refer to [2], [3] for concrete examples of problems with label distribution skew.

The objective is to find the parameters θ of a global model that performs well on the union of the local distributions by minimizing the average training loss:

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{(x_i, y_i) \sim D_i} [F_i(\theta; x_i, y_i)], \tag{1}$$

where (x_i, y_i) is a data point drawn from D_i and F_i is the loss function on node i. Therefore, $\mathbb{E}_{(x_i, y_i) \sim D_i} F_i(\theta; x_i, y_i)$ denotes the expected loss of model θ over D_i.

To collaboratively solve Problem (1), each node can exchange messages with its neighbors in an undirected network graph $G = (N, E)$ where $\{i, j\} \in E$ denotes an edge (communication channel) between nodes i and j.

b) Training algorithm: In this work, we use the popular Decentralized Stochastic Gradient Descent algorithm, aka D-SGD [6]. As shown in Figure 2, a single iteration of D-SGD at node i consists in sampling a mini-batch from its local distribution D_i, updating its local model θ_i by taking a stochastic gradient descent (SGD) step according to the mini-batch, and performing a weighted average of its local model with those of its neighbors. This weighted average is defined by a mixing matrix W, in which W_{ij} corresponds to the weight of the outgoing connection from node i to j and $W_{ij} = 0$

Require: initial model $\theta_i^{(0)}$, learning rate γ, mixing weights W, mini-batch size m, number of steps K
for $k = 1, \ldots, K$ **do**
  $S_i^{(k)} \leftarrow$ mini-batch of m samples drawn from D_i
  $\theta_i^{(k-\frac{1}{2})} \leftarrow \theta_i^{(k-1)} - \gamma \nabla F(\theta_i^{(k-1)}; S_i^{(k)})$
  $\theta_i^{(k)} \leftarrow \sum_{j \in N} W_{ji}^{(k)} \theta_j^{(k-\frac{1}{2})}$

Fig. 2: D-SGD algorithm (from the perspective of node i).

for $\{i, j\} \notin E$. To ensure that the local models converge on average to a stationary point of Problem (1), W must be doubly stochastic ($\sum_{j \in N} W_{ij} = 1$ and $\sum_{j \in N} W_{ji} = 1$) and symmetric, i.e. $W_{ij} = W_{ji}$ [6]. Given a network topology $G = (N, E)$, we generate a valid W by computing standard Metropolis-Hasting weights [14]:

$$W_{ij} = \begin{cases} \frac{1}{\max(\text{degree}(i), \text{degree}(j)) + 1} & \text{if } i \neq j \text{ and } \{i, j\} \in E, \\ 1 - \sum_{j \neq i} W_{ij} & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

III. D-Cliques

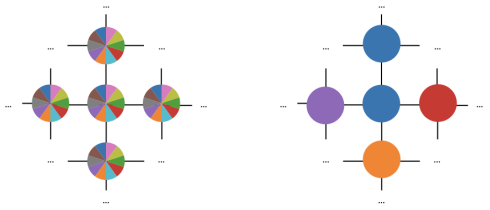
In this section, we introduce D-Cliques, a topology designed to compensate for data heterogeneity. We also present some modifications of D-SGD that leverage some properties of the proposed topology and enable the implementation of a successful momentum scheme.

A. Intuition

To give the intuition behind our approach, let us consider the neighborhood of a single node in a grid topology represented on Figure 3. Nodes are distributed randomly in the grid and the colors of a node represent the proportion of each class in its local dataset. In the homogeneous setting, the label distribution is the same across nodes: in the example shown in Figure 3a, all classes are represented in equal proportions on all nodes. This is not the case in the heterogeneous setting: Figure 3b shows an extreme case of label distribution skew where each node holds examples of a single class only.

From the point of view of the center node in Figure 3, a single training step of D-SGD is equivalent to sampling a mini-batch five times larger from the union of the local distributions of neighboring nodes. In the homogeneous case, since gradients are computed from examples of all classes, the resulting averaged gradient points in a direction that tends to reduce the loss across all classes. In contrast, in the heterogeneous case, the representation of classes in the immediate neighborhood of the node is different from the global label distribution (in Figure 3b, only a subset of classes are represented), thus the gradients will be biased. Importantly, as the distributed averaging process takes several steps to converge, this variance persists across iterations as the locally computed gradients are far from the global average.[2] This

[2]One could perform a sufficiently large number of averaging steps between each gradient step, but this is too costly in practice.



(a) Homogeneous data (b) Heterogeneous data

Fig. 3: Neighborhood in a grid.

Require: maximum clique size M, max steps K, set of all nodes $N = \{1, 2, \ldots, n\}$, procedure $\texttt{inter}(\cdot)$ to create inter-clique connections (see Sec. III-C)
$DC \leftarrow []$
while $N \neq \emptyset$ **do**
  $C \leftarrow$ sample M nodes from N at random
  $N \leftarrow N \setminus C$; DC.append(C)
for $k \in \{1, \ldots, K\}$ **do**
  $C_1, C_2 \leftarrow$ random sample of 2 elements from DC
  $s \leftarrow \mathit{skew}(C_1) + \mathit{skew}(C_2)$
  $\mathit{swaps} \leftarrow []$
  for $i \in C_1, j \in C_2$ **do**
    $s' \leftarrow \mathit{skew}(C_1 \setminus \{i\} \cup \{j\}) + \mathit{skew}(C_2 \setminus \{j\} \cup \{i\})$
    if $s' < s$ **then**
      swaps.append((i, j))
  if len(swaps) > 0 **then**
    $(i, j) \leftarrow$ random element from swaps
    $C_1 \leftarrow C_1 \setminus \{i\} \cup \{j\}; C_2 \leftarrow C_2 \setminus \{j\} \cup \{i\}$
$E \leftarrow \{(i, j) : C \in DC, i, j \in C, i \neq j\}$ **return** topology $G = (N, E \cup \texttt{inter}(DC))$

Fig. 4: Greedy-Swap algorithm for D-Cliques construction.

can significantly slow down convergence speed to the point of making decentralized optimization impractical.

With D-Cliques, we address label distribution skew by carefully designing a network topology composed of *locally representative cliques* while maintaining *sparse inter-clique connections* only.

B. Constructing Locally Representative Cliques

D-Cliques construct a topology in which each node is part of a *clique* (i.e., a subset of nodes whose induced subgraph is fully connected) such that the label distribution in each clique is close to the global label distribution. Formally, for a label y and a clique composed of nodes $C \subseteq N$, we denote by $p_C(y) = \frac{1}{|C|} \sum_{i \in C} p_i(y)$ the distribution of y in C and by $p(y) = \frac{1}{n} \sum_{i \in N} p_i(y)$ its global distribution. We measure the *skew* of C by the L1 distance, i.e. the sum of the absolute differences of $p_C(y)$ and $p(y)$ over y, which is a commonly

used measure of distance between histograms:

$$\textit{skew}(C) = \sum_{l=1}^{L} |p_C(y=l) - p(y=l)|. \quad (3)$$

To efficiently construct a set of cliques with small skew, we propose Greedy-Swap (Figure 4). The parameter M is the maximum size of cliques and controls the number of intra-clique edges. We start by initializing cliques at random. Then, for a certain number of steps K, we randomly pick two cliques and swap two of their nodes so as to decrease the sum of skews of the two cliques. The swap is chosen randomly among the ones that decrease the skew, hence this algorithm can be seen as a form of randomized greedy algorithm.

In practice, Greedy-Swap may be executed by a single party (which could be one of the nodes): in this case, each node i should send to this party its label distribution $p_i(y)$ at each node i. Alternatively, Greedy-Swap may be executed in a decentralized fashion, e.g. with a leader-based implementation, as follows. We assume that each node has a unique, randomly assigned identifier. As a pre-processing step, the global label distribution $p(y)$ is computed by running a standard decentralized averaging algorithm (e.g. [15]) so that all nodes know $p(y)$. Cliques are initialized using the random identifiers and in each clique, the node with smallest identifier becomes the clique leader. Leaders from each clique then randomly sample other leaders with peer sampling [16] to find potential swap candidates. Once a leader l_1 has found another leader l_2, l_1 sends the label distributions of all its individual clique members to l_2, who compares the distributions to its own and picks a pair of nodes to swap such that the sum of the skew of both cliques will decrease. After a swap, the leaders of each clique update their clique members with the new memberships (including the leaders if needed). Swapping continues for a predefined number of steps or until the topology stabilizes.

Remark 1 (Clique Size): In practice, the clique size is chosen such that each class can be represented in each clique, which can be inferred from the label distributions across nodes, but no bigger than the maximum communication budget desired for each node, which depends on the deployment environment and system constraints. In general, the larger the clique size, the faster the convergence (in number of rounds), but the higher the communication costs per round. The best clique size can be tuned empirically based on the trade-off between communication cost and clique skew.

Remark 2 (Privacy): Constructing cliques with Greedy-Swap only requires nodes to reveal their local label proportions (i.e., a histogram with L bins), which is compact and aggregated information. To further reduce the leakage in applications where the label membership of a data point may be sensitive information, techniques from differential privacy [17] can be readily used. In particular, given the desired privacy level $\epsilon > 0$ (the smaller, the more privacy), each node i can share a *private histogram* by adding independent centered Laplace noise with standard deviation $\sigma = 1/(m_i\epsilon)$ to each entry of its true histogram, where m_i is the number of examples at node i. This gives an unbiased estimate of the true histogram with a variance that increases with the desired privacy level. Thus, the skew of cliques, and thereby the resulting convergence speed of D-SGD, will be inversely proportional to the privacy level. In the limit of $\epsilon \to 0$ (perfect privacy), the allocation of nodes in cliques will become fully random. However, for a reasonable privacy budget (say $\epsilon = 1$, which is generally considered as a good guarantee) and $m_i \simeq 500$ as used for all experiments with 100 nodes in Section IV, the impact of privacy will be negligible as $\sigma \simeq 0.002$.

The key idea of D-Cliques, implemented by the above Greedy-Swap algorithm, is to ensure the clique-level label distribution $p_C(y)$ matches closely the global distribution $p(y)$. As a consequence, the local models of nodes across cliques will remain rather close throughout the iterations of D-SGD. Therefore, a sparse inter-clique topology can be used, significantly reducing the total number of edges without slowing down the convergence. We discuss some possible choices for this inter-clique topology in the following section.

C. Adding Sparse Inter-Clique Connections

To ensure a global consensus and convergence, we introduce *inter-clique connections* between a small number of node pairs that belong to different cliques, thereby implementing the `inter` procedure called at the end of Figure 4. We aim to ensure that the degree of each node remains low and balanced so as to make the network topology well-suited to decentralized federated learning. We consider several choices of inter-clique topology, which offer different scalings for the number of required edges and the average distance between nodes in the resulting graph.

The *ring* has (almost) the fewest possible number of edges for the graph to be connected: in this case, each clique is connected to exactly two other cliques by a single edge. This topology requires only $O(\frac{n}{M})$ inter-clique edges but suffers an $O(n)$ average distance between nodes.

The *fractal* topology provides a logarithmic bound on the average distance. In this hierarchical scheme, cliques are arranged in larger groups of M cliques that are connected internally with one edge per pair of cliques, but with only one edge between pairs of larger groups. The topology is built recursively such that M groups will themselves form a larger group at the next level up. This results in at most M edges per node if edges are evenly distributed: i.e., each group within the same level adds at most $M-1$ edges to other groups, leaving one node per group with $M-1$ edges that can receive an additional edge to connect with other groups at the next level. Since nodes have at most M edges, the total number of inter-clique edges is at most nM edges.

We can also design an inter-clique topology in which the number of edges scales in a log-linear fashion by following a small-world-like topology [18] applied on top of a ring [19]. In this scheme, cliques are first arranged in a ring. Then each clique adds symmetric edges, both clockwise and counter-clockwise on the ring, with the c closest cliques in sets of cliques that are exponentially bigger the further they are on the

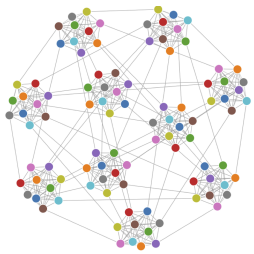


Fig. 5: D-Cliques with $n = 100$, $M = 10$ and a fully connected inter-clique topology on a problem with 1 class/node.

ring (see extended paper [20] for details on the construction). This topology ensures a good connectivity with other cliques that are close on the ring, while keeping the average distance small. This scheme uses $O(c\frac{n}{M}\log\frac{n}{M})$ edges, i.e. log-linear in n.

Finally, we can consider a *fully connected* inter-clique topology such that each clique has exactly one edge with each of the other cliques, spreading these additional edges equally among the nodes of a clique, as illustrated in Figure 5. This has the advantage of bounding the distance between any pair of nodes to 3 but requires $O(\frac{n^2}{M^2})$ inter-clique edges, i.e. quadratic in n.

D. Optimizing over D-Cliques with Clique Averaging and Momentum

While limiting the number of inter-clique connections reduces the amount of messages traveling on the network, it also introduces a form of bias. Figure 6 illustrates the problem on the simple case of two cliques connected by one inter-clique edge (here, between the green node of the left clique and the pink node of the right clique). In this example, each node holds example of a single class. Let us focus on node A. With weights computed as in (2), node A's self-weight is $\frac{12}{110}$, the weight between A and the green node connected to B is $\frac{10}{110}$, and all other neighbors of A have a weight of $\frac{11}{110}$. Therefore, the gradient at A is biased towards its own class (pink) and against the green class. A similar bias holds for all other nodes without inter-clique edges with respect to their respective classes. For node B, all its edge weights (including its self-weight) are equal to $\frac{1}{11}$. However, the green class is represented twice (once as a clique neighbor and once from the inter-clique edge), while all other classes are represented only once. This biases the gradient toward the green class. The combined effect of these two sources of bias is to increase the variance of the local models across nodes.

a) Clique Averaging: We address this problem by adding *Clique Averaging* to D-SGD (Figure 7), which essentially decouples gradient averaging from model averaging. The idea is to use only the gradients of neighbors within the same clique to compute the average gradient so as to remove the bias due to inter-clique edges. In contrast, all neighbors' models (including those in different cliques) participate in model averaging as in the original version. Adding Clique Averaging requires gradients to be sent separately from the model parameters: the number of messages exchanged between nodes is therefore twice their number of edges.

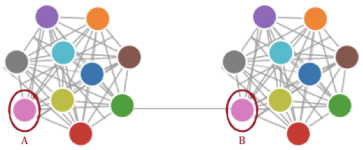


Fig. 6: Illustrating the bias induced by inter-clique connections (see main text for details).

Require initial model $\theta_i^{(0)}$, learning rate γ, mixing weights W, mini-batch size m, number of steps K
for $k = 1, \ldots, K$ **do**
  $S_i^{(k)} \leftarrow$ mini-batch of m samples drawn from D_i
  $g_i^{(k)} \leftarrow \frac{1}{|Clique(i)|}\sum_{j \in Clique(i)} \nabla F(\theta_j^{(k-1)}; S_j^{(k)})$
  $\theta_i^{(k-\frac{1}{2})} \leftarrow \theta_i^{(k-1)} - \gamma g_i^{(k)}$
  $\theta_i^{(k)} \leftarrow \sum_{j \in N} W_{ji}^{(k)} \theta_j^{(k-\frac{1}{2})}$

Fig. 7: D-SGD with Clique Averaging (perspective of node i).

b) Implementing momentum with Clique Averaging: Efficiently training high capacity models usually requires additional optimization techniques. In particular, momentum [21] increases the magnitude of the components of the gradient that are shared between several consecutive steps, and is critical for deep convolutional networks like LeNet [22], [3] to converge quickly. However, a direct application of momentum in data heterogeneous settings can actually be very detrimental and even fail to converge, as we will show in our experiments (Figure 10 in Section IV). Clique Averaging allows us to reduce the bias in the momentum by using the clique-level average gradient $g_i^{(k)}$ of Figure 7:

$$v_i^{(k)} \leftarrow m v_i^{(k-1)} + g_i^{(k)}. \tag{4}$$

It then suffices to modify the original gradient step to apply momentum:

$$\theta_i^{(k-\frac{1}{2})} \leftarrow \theta_i^{(k-1)} - \gamma v_i^{(k)}. \tag{5}$$

IV. Evaluation

To validate our design choices and demonstrate the ability of D-Cliques to remove the effect of label skew, we focus our investigation on concrete experiments. This is complementary to existing theoretical analyses of D-SGD, which hold for D-Cliques but are unable to capture its benefits. Indeed, in these formal convergence results, the effect of data heterogeneity is abstracted away with the use of unknown constants that bound the variance of local gradients, and the effect of topology is analyzed independently [6], [7], [8], [9]. In contrast, our experiments show that a careful data-dependent design of the topology significantly improves the convergence speed under data heterogeneity, suggesting that the usually overlooked constants and the interplay with topology have a major impact.

Our experiments have motivated refined analyses that can theoretically explain the benefits of our approach [23], [24].

Below, we first describe our experimental setup. We then compare D-Cliques to alternative topologies to show the benefits and relevance of our main design choices. We also evaluate different inter-clique topologies to further reduce the number of inter-clique connections so as to gracefully scale with the number of nodes. Then, we further show the impact of removing intra-clique edges. Finally, we show that Greedy Swap (Alg. 4) constructs cliques efficiently with consistently lower skew than random cliques.

A. Experimental Setup

Our main goal is to provide a fair comparison of the convergence speed across different topologies and algorithmic variations, in order to validate our design choices and show that D-Cliques can remove much of the effects of label distribution skew.

In our study, we focus our investigation on the convergence speed, rather than the final accuracy after a fixed number of iterations. Indeed, depending on when training is stopped, the relative difference in final accuracy across different algorithms may vary significantly and lead to different conclusions. Instead of relying on somewhat arbitrary stopping points, we show the convergence curves of generalization performance (i.e., the accuracy on the test set throughout training), up to a point where it is clear that the different approaches have converged, will not make significantly more progress, or behave essentially the same.

a) Datasets: We experiment with two datasets: MNIST [25] and CIFAR10 [26], which both have $L = 10$ classes. For MNIST, we use 50k and 10k examples from the original 60k training set for training and validation respectively. We use all 10k examples of the test set to measure prediction accuracy. The validation set preserves the original unbalanced ratio of the classes in the test set, and the remaining examples become the training set. For CIFAR10, classes are evenly balanced: we initially used 45k/50k images of the original training set for training, 5k/50k for validation, and all 10k examples of the test set for measuring prediction accuracy. After tuning hyper-parameters on initial experiments, we then used all 50k images of the original training set for training for all experiments, as the 45k did not split evenly in 1000 nodes with the partitioning scheme explained in the next paragraph.

For both MNIST and CIFAR10, we use the heterogeneous data partitioning scheme proposed by McMahan et al. [1] in their seminal FL work: we sort all training examples by class, then split the list into shards of equal size, and randomly assign two shards to each node. When the number of examples of one class does not divide evenly in shards, as is the case for MNIST, some shards may have examples of more than one class and therefore nodes may have examples of up to 4 classes. However, most nodes will have examples of 2 classes. The varying number of classes, as well as the varying distribution of examples within a single node, makes the task of creating cliques with low skew nontrivial.

b) Models: We use a logistic regression classifier for MNIST, which provides up to 92.5% accuracy in the centralized setting. For CIFAR10, we use a Group-Normalized variant of LeNet [3], a deep convolutional network which achieves an accuracy of 74.15% in the centralized setting. These models are thus reasonably accurate (which is sufficient to study the effect of the topology) while being sufficiently fast to train in a fully decentralized setting and simple enough to configure and analyze. Regarding hyper-parameters, we jointly optimize the learning rate and mini-batch size on the validation set for 100 nodes, obtaining respectively 0.1 and 128 for MNIST and 0.002 and 20 for CIFAR10. For CIFAR10, we additionally use a momentum of 0.9.

c) Metrics: We evaluate 100- and 1000-node networks by creating multiple models in memory and simulating the exchange of messages between nodes. To ignore the impact of distributed execution strategies and system optimization techniques, we report the test accuracy of all nodes (min, max, average) as a function of the number of times each example of the dataset has been sampled by a node, i.e. an *epoch*. This is equivalent to the classic case of a single node sampling the full distribution. To further make results comparable across different number of nodes, we lower the batch size proportionally to the number of nodes added, and inversely, e.g. on MNIST, 128 with 100 nodes vs. 13 with 1000 nodes. This ensures the same number of model updates and averaging per epoch, allowing a fair comparison.[3]

d) Baselines: We compare our results against an ideal baseline: a fully-connected network topology with the same number of nodes. All other things being equal, any other topology using less edges will converge at the same speed or slower: *this is therefore the most difficult and general baseline to compare against*. This baseline is also essentially equivalent to a centralized (single) IID node using a batch size n times bigger, where n is the number of nodes. Both a fully-connected network and a single IID node effectively optimize a single model and sample uniformly from the global distribution: both thus remove entirely the effect of label distribution skew and of the network topology on the optimization. In practice, we prefer a fully-connected network because it converges slightly faster and obtains slightly better final accuracy than a single node sampling randomly from the global distribution.[4]

We also provide comparisons against popular sparse topologies, such as random graphs and exponential graphs [9].

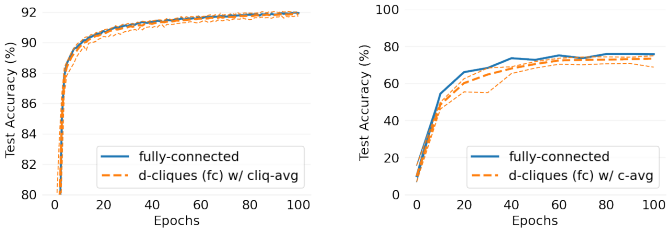
B. D-Cliques Match the Convergence Speed of Fully-Connected with a Fraction of the Edges

In this first experiment, we show that D-Cliques with Clique Averaging (and momentum when mentioned) converges al-

[3]Updating and averaging models after every example can eliminate the impact of label distribution skew. However, the resulting communication overhead is impractical.

[4]We conjecture that an heterogeneous data partition in a fully-connected network may force more balanced representation of all classes in the union of all mini-batches, leading to better convergence.

most as fast as a fully-connected network on both MNIST and CIFAR10. Figure 8 illustrates the convergence speed of D-Cliques with $n = 100$ nodes on MNIST (with Clique Averaging) and CIFAR10 (with Clique Averaging and momentum). Observe that the convergence speed is very close to that of a fully-connected topology, and significantly better than with a ring or a grid (see Figure 1). It also has less variance than both the ring and grid.



(a) MNIST (b) CIFAR10 (w/ momentum)

Fig. 8: Comparison on 100 heterogeneous nodes (2 shards/node) between a fully-connected network and D-Cliques (fully-connected) constructed with Greedy Swap (10 cliques of 10 nodes) using Clique Averaging. Bold line is the average accuracy over all nodes. Thinner upper and lower lines are maximum and minimum accuracy over all nodes.

C. Clique Averaging is Beneficial and Sometimes Necessary

In this experiment, we perform an ablation study of the effect of Clique Averaging. Figure 9 shows that Clique Averaging (Figure 7) reduces the variance of models across nodes and slightly accelerates the convergence on MNIST. Recall that Clique Averaging induces a small additional cost, as gradients and models need to be sent in two separate rounds of messages. Nonetheless, compared to fully connecting all nodes, the total number of messages per round for 100 nodes is reduced by $\approx 80\%$.

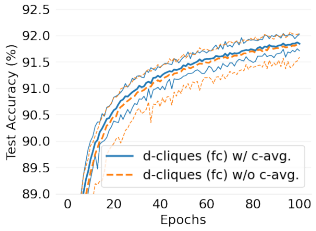
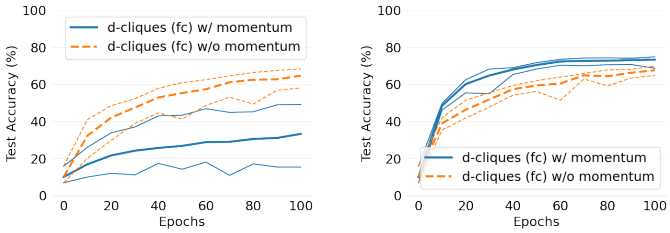


Fig. 9: MNIST: Effect of Clique Averaging on D-Cliques (fully-connected) with 10 cliques of 10 heterogeneous nodes (100 nodes). Y axis starts at 89.

The effect of Clique Averaging is much more pronounced on CIFAR10, as can be seen in Figure 10, especially when used in combination with momentum. Without Clique Averaging, the use of momentum is actually detrimental. With Clique Averaging, the situation reverses and momentum is again beneficial. The combination of both has the fastest convergence speed and the lowest variance among all four possibilities. We believe that the gains obtained with Clique Averaging are larger on CIFAR10 than on MNIST because the model we train on CIFAR10 (a deep convolutional network) has much higher capacity than the linear model used for MNIST. The resulting highly nonconvex objective increases the sensitivity of local updates to small differences in the gradients, making them point in different directions, as observed by Kong et al. [13] even in the homogeneous setting. Clique Averaging helps to reduce this effect by reducing the bias in local gradients.



(a) Without Clique Averaging (b) With Clique Averaging

Fig. 10: CIFAR10: Effect of Clique Averaging, without and with momentum, on D-Cliques (fully-connected) with 10 cliques of 10 heterogeneous nodes (100 nodes).

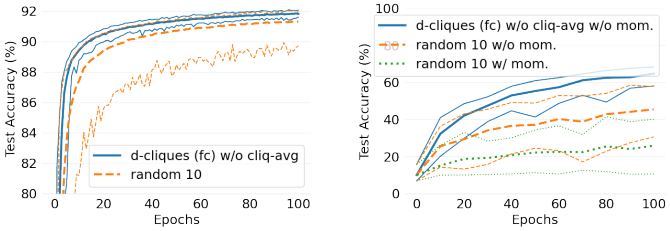
D. D-Cliques Converge Faster than Random and Exponential Graphs

In this experiment, we compare D-Cliques to a random graph and an exponential graph [9].

For the random graph, we use a similar number of edges (10) per node to determine whether a simple sparse topology could work equally well. To ensure a fair comparison, because a random graph does not support Clique Averaging, we do not use it for D-Cliques either. Figure 11 shows that even *without* Clique Averaging, D-Cliques converge faster and with lower variance. Furthermore, the use of momentum in a random graph is detrimental, similar to D-Cliques without the use of Clique Averaging (see Figure 10a). D-Cliques converge faster and with less variance even if we force the local neighborhoods of the random graph to be representative of the global label distribution (see the appendix of the extended version of this paper [20], as the experiment requires a different data partitioning scheme for a fair comparison).

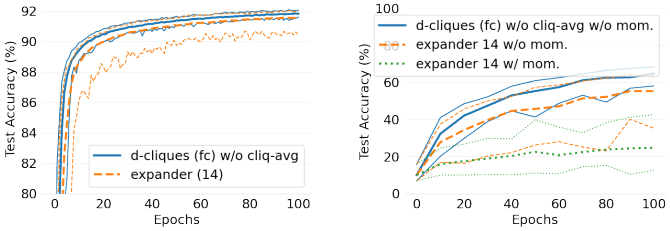
For the exponential graph, we follow the deterministic construction of [9] and consider edges to be undirected as required by D-SGD, resulting in 14 edges per node. Figure 12 shows that D-Cliques converge faster and with less variance, despite the exponential graph having 1.4 times more edges.

These experiments show that a careful design of the topology is indeed necessary, especially regarding the skew in the neighborhood of each node. In addition, it appears that the "bounded data heterogeneity" assumption, typically used in convergence theorems to compare different algorithms and topologies [6], [7], [8], [9], overlooks the major role that the choice of topology (e.g., one that ensures clustered representative neighborhoods like D-Cliques) may have on the convergence speed.



(a) MNIST (b) CIFAR10

Fig. 11: Comparison on 100 heterogeneous nodes between D-Cliques (fully-connected) with 10 cliques of size 10 and a random graph with 10 edges per node *without* Clique Averaging or momentum.



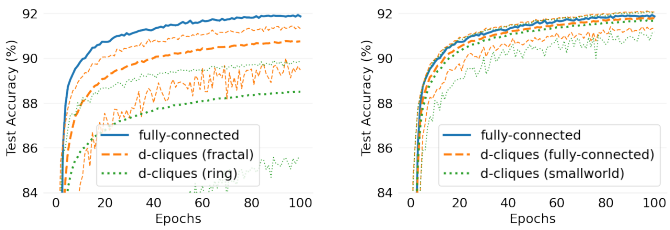
(a) MNIST (b) CIFAR10

Fig. 12: Comparison on 100 heterogeneous nodes between D-Cliques (fully-connected) with 10 cliques of size 10 and an exponential graph with 14 undirected edges per node *without* Clique Averaging or momentum.

E. D-Cliques Scale with Sparser Inter-Clique Topologies

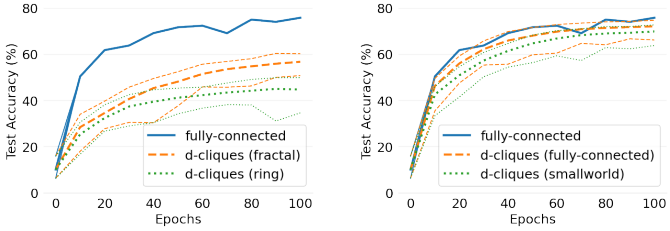
In this experiment, we explore the trade-offs between scalability and convergence speed induced by the several sparse inter-clique topologies introduced in Section III-C. Figure 13 and Figure 14 show the convergence speed respectively on MNIST and CIFAR10 on a larger network of 1000 nodes, compared to the ideal baseline of a fully-connected network representing the fastest convergence speed achievable if topology had no impact. Among the linear schemes, the ring topology converges but is much slower than our fractal scheme. Among the super-linear schemes, the small-world topology has a convergence speed that is almost the same as with a fully-connected inter-clique topology but with 22% less edges (14.5 edges on average instead of 18.9).

While the small-world inter-clique topology shows promising scaling behavior, the fully-connected inter-clique topology still offers significant benefits with 1000 nodes, as it represents a 98% reduction in the number of edges compared to fully connecting individual nodes (18.9 edges on average instead of 999) and a 96% reduction in the number of messages (37.8 messages per round per node on average instead of 999). We refer to an extended version of this paper [20] for additional results comparing the convergence speed across different number of nodes. Overall, these results show that D-Cliques can gracefully scale with the number of nodes.



(a) Linear (b) Super- and Quasi-Linear

Fig. 13: MNIST: D-Cliques convergence speed with 1000 nodes (10 nodes per clique, same number of updates per epoch as 100 nodes, i.e. batch-size 10x less per node) and different inter-clique topologies.

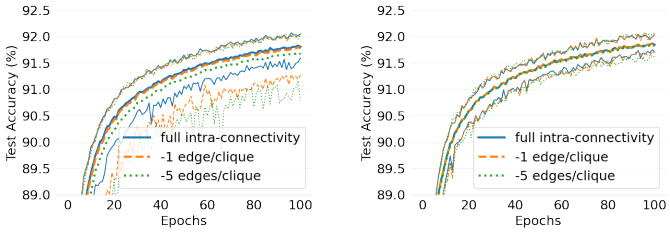


(a) Linear (b) Super- and Quasi-Linear

Fig. 14: CIFAR10: D-Cliques convergence speed with 1000 nodes (10 nodes per clique, same number of updates per epoch as 100 nodes, i.e. batch-size 10x less per node) and different inter-clique topologies.

F. Full Intra-Clique Connectivity is Necessary

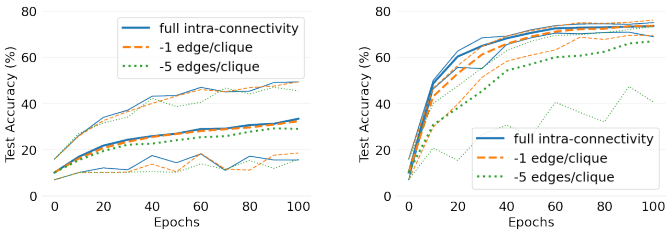
In this experiment, we measure the impact of removing intra-clique edges to assess how critical full connectivity is within cliques. We choose edges to remove among the 45 undirected edges present in cliques of size 10. The removal of an edge removes the connection in both directions. We remove 1 and 5 edges randomly, respectively 2.2% and 11% of intra-clique edges. Figure 15 shows that for MNIST, when not using Clique Averaging, removing edges decreases slightly the convergence speed and increases the variance between nodes. When using Clique Averaging, removing up to 5 edges does not noticeably affect the convergence speed and variance.



(a) Without Clique Averaging (b) With Clique Averaging

Fig. 15: MNIST: Impact of intra-clique edge removal on D-Cliques (fully-connected) with 10 cliques of 10 heterogeneous nodes (100 nodes). Y axis starts at 89.

In contrast, Figure 16 shows that for CIFAR10, the impact is stronger. We show the results with and without Clique Averaging with momentum in both cases, as momentum is critical for obtaining the best convergence speed on CIFAR10. Without Clique Averaging, removing edges has a small effect on convergence speed and variance, but the convergence speed is too slow to be practical. With Clique Averaging, removing a single edge has a small but noticeable effect. Strikingly, removing 5 edges per clique significantly damages the convergence and yields a sharp increase in the variance across nodes. Therefore, while D-Cliques can tolerate the removal of some intra-clique edges when training simple linear models and datasets as in MNIST, fast convergence speed and low variance requires full or nearly full connectivity when using high-capacity models and more difficult datasets. This is in line with the observations made in Section IV-C regarding the effect of Clique Averaging. Again, these results show the relevance of our design choices, including the choice of constructing fully connected cliques.



(a) Without Clique Averaging (b) With Clique Averaging

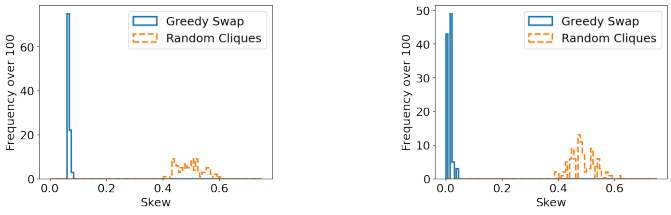
Fig. 16: CIFAR10: Impact of intra-clique edge removal (with momentum) on D-Cliques (fully-connected) with 10 cliques of 10 heterogeneous nodes (100 nodes).

G. Greedy Swap Improves Random Cliques at an Affordable Cost

In the next two sub-sections, we compare cliques built with Greedy Swap (Alg. 4) to Random Cliques, a simple and obvious baseline, on their quality (skew), the cost of their construction, and their convergence speed.

1) Cliques with Low Skew can be Constructed Efficiently with Greedy Swap: We compared the final average skew of 10 cliques with 10 nodes each (for $n = 100$) created either randomly or with Greedy Swap, over 100 experiments after 1000 steps. Figure 18, in the form of an histogram, shows that Greedy Swap generates cliques of significantly lower skew, close to 0 in a majority of cases for both MNIST and CIFAR10.

Figure 18 shows such a low skew can be achieved in less than 400 steps for both MNIST and CIFAR10. In practice it takes less than 6 seconds in Python 3.7 on a Macbook Pro 2020 for a network of 100 nodes and cliques of size 10. Greedy Swap is therefore fast and efficient. Moreover, it illustrates the fact that a global imbalance in the number of examples across classes makes the construction of cliques with low skew harder and slower.



(a) MNIST (b) CIFAR10

Fig. 17: Final quality of cliques (skew) with a maximum size of 10 over 100 experiments in a network of 100 nodes.

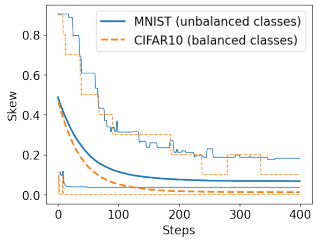
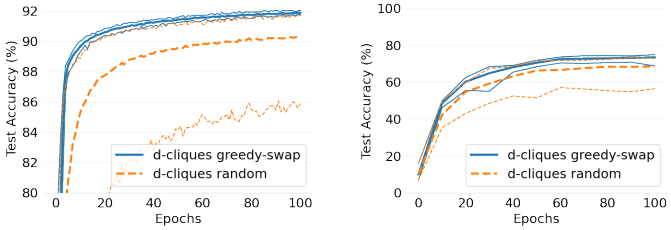


Fig. 18: Skew decrease during clique construction of 10 cliques of 10 heterogeneous nodes (100 nodes). Bold line is the average over 100 experiments. Thin lines are respectively the minimum and maximum over all experiments. In wall-clock time, 1000 steps take less than 6 seconds in Python 3.7 on a MacBook Pro 2020.

2) Cliques built with Greedy Swap Converge Faster than Random Cliques: Figure 19 compares the convergence speed of cliques optimized with Greedy Swap for 1000 steps with cliques built randomly (equivalent to Greedy Swap with 0 steps). For both MNIST and CIFAR10, convergence speed increases significantly and variance between nodes decreases dramatically. Decreasing the skew of cliques is therefore critical to convergence speed.



(a) MNIST (b) CIFAR10

Fig. 19: Convergence speed of D-Cliques constructed randomly vs Greedy Swap with 10 cliques of 10 heterogeneous nodes (100 nodes).

H. Additional Experiments on Extreme Label Distribution Skew

In an extended version of this paper [20], we replicate experimental results on an extreme case of label distribution skew where each node only has examples of a single class.

These results consistently show that our approach remains effective even for extremely skewed label distributions.

V. Related Work

In this section, we review some related work on dealing with heterogeneous data in federated learning, and on the role of topology in fully decentralized algorithms.

a) Dealing with heterogeneity in server-based FL: Data heterogeneity is not much of an issue in server-based FL if clients send their parameters to the server after each gradient update. Problems arise when participants perform multiple local updates to reduce the number of communication rounds, as in the popular FedAvg algorithm [1]. Indeed, data heterogeneity can prevent such algorithms from converging to a good solution [3], [5]. This led to the design of algorithms that are specifically designed to mitigate the impact of heterogeneity while performing multiple local updates, using adaptive client sampling [3], update corrections [5] or regularization in the local objective [4]. Another direction is to embrace the heterogeneity by learning personalized models for each client [27], [28], [29], [30], [31]. We note that recent work explores rings of server-based topologies [32], but the focus is not on dealing with heterogeneous data but to make server-based FL more scalable to a large number of clients.

b) Dealing with heterogeneity in fully decentralized FL: Data heterogeneity is known to negatively impact the convergence speed of fully decentralized FL algorithms in practice [33]. Aside from approaches that aim to learn personalized models [34], [35], this motivated the design of algorithms with modified updates based on variance reduction [10], momentum correction [11], cross-gradient aggregation [12], or multiple averaging steps between updates (see [13], and references therein). These algorithms typically require significantly more communication and/or computation, and have only been evaluated on small-scale networks with a few tens of nodes.[5] In contrast, we focus on the design of a sparse topology which compensates for the effect of heterogeneous data and scales to large networks. We do not modify the simple and efficient D-SGD algorithm [6] beyond removing some neighbor contributions that otherwise bias the gradient direction.

c) Impact of topology in fully decentralized FL: It is well known that the choice of network topology can affect the convergence of fully decentralized algorithms. In theoretical convergence rates, this is typically accounted for by a dependence on the spectral gap of the network, see for instance [36], [37], [6], [38], [9]. However, for homogeneous (IID) data, practice contradicts these classic results as fully decentralized algorithms have been observed to converge essentially as fast on sparse topologies like rings or grids as they do on a fully connected network [6], [7]. Recent work [8], [13] sheds light on this phenomenon with refined convergence analyses based on differences between gradients or parameters across nodes, which are typically smaller in the homogeneous case. However, these results do not give any clear insight regarding the role of the topology in the presence of heterogeneous data. Indeed, in all of the above analyses, the impact of data heterogeneity is abstracted away through (unknown) constants that bound the variance of local gradients, independently of the topology. We note that some work has gone into designing topologies to optimize the use of network resources (see e.g., [39]), but the topology is chosen independently of how data is distributed across nodes.

In summary, the interplay between the network topology and data heterogeneity is not well understood and we are not aware of prior work focusing on this question. This paper is the first to show that an appropriate choice of data-dependent topology can effectively compensate for heterogeneous data. We note that our work has already started to inspire refined theoretical analyses of D-SGD that now consider the effect of the neighborhood data distribution on the averaged gradient and can formally explain the benefits of D-Cliques [23], [24].

[5] We also observed that [10] is subject to numerical instabilities when run on topologies other than rings. When the rows and columns of W do not exactly sum to 1 (due to finite precision), these small differences get amplified by the proposed updates and make the algorithm diverge.

VI. Conclusion

We proposed D-Cliques, a sparse topology that obtains similar convergence speed as a fully-connected network in the presence of label distribution skew. D-Cliques is based on assembling subsets of nodes into cliques such that the clique-level class distribution is representative of the global distribution, thereby locally recovering homogeneity of data. Cliques are connected together by a sparse inter-clique topology so that they quickly converge to the same model. We proposed Clique Averaging to remove the bias in gradient computation due to non-homogeneous averaging neighborhood by averaging gradients only with other nodes within the clique. Clique Averaging can in turn be used to implement an effective momentum. Through our extensive set of experiments, we showed that the clique structure of D-Cliques is critical in obtaining these results and that a small-world inter-clique topology with only $O(n \log n)$ edges achieves a very good compromise between convergence speed and scalability with the number of nodes.

The ideas of D-Cliques can be useful to implement fully decentralized learning in a wider range of heterogeneous environments. Future works include evaluating D-Cliques in more challenging settings (node failures including adversaries, arrival and departure of nodes, non-uniform latency between pairs of nodes) and dynamically constructing the topology *while* optimizing the model. More general types of data heterogeneity could be tackled: An important example is covariate shift or feature distribution skew [2], for which local density estimates could be used as basis to construct cliques that approximately recover the global distribution.

Acknowledgements

This work was supported in part by the French National Research Agency (ANR) through grant ANR-20-CE23-0015 (Project PRIDE) and ANR-16-CE23-0016-01 (Project PAMELA).

REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017.

[2] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konecný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, H. Qi, D. Ramage, R. Raskar, M. Raykova, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[3] K. Hsieh, A. Phanishayee, O. Mutlu, and P. B. Gibbons, "The Non-IID Data Quagmire of Decentralized Machine Learning," in *ICML*, 2020.

[4] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated Optimization in Heterogeneous Networks," in *MLSys*, 2020.

[5] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic Controlled Averaging for On-Device Federated Learning," in *ICML*, 2020.

[6] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent," in *NIPS*, 2017.

[7] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous Decentralized Parallel Stochastic Gradient Descent," in *ICML*, 2018.

[8] G. Neglia, C. Xu, D. Towsley, and G. Calbi, "Decentralized gradient methods: does topology matter?" in *AISTATS*, 2020.

[9] B. Ying, K. Yuan, Y. Chen, H. Hu, P. Pan, and W. Yin, "Exponential Graph is Provably Efficient for Decentralized Deep Training," in *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021. [Online]. Available: https://openreview.net/forum?id=l2UWXn5iBQI

[10] H. Tang, X. Lian, M. Yan, C. Zhang, and J. Liu, "D^2: Decentralized Training over Decentralized Data," in *ICML*, 2018.

[11] T. Lin, S. P. Karimireddy, S. U. Stich, and M. Jaggi, "Quasi-Global Momentum: Accelerating Decentralized Deep Learning on Heterogeneous Data," arXiv:2102.04761, Tech. Rep., 2021.

[12] Y. Esfandiari, S. Y. Tan, Z. Jiang, A. Balu, E. Herron, C. Hegde, and S. Sarkar, "Cross-Gradient Aggregation for Decentralized Learning from Non-IID data," arXiv:2103.02051, Tech. Rep., 2021.

[13] L. Kong, T. Lin, A. Koloskova, M. Jaggi, and S. U. Stich, "Consensus Control for Decentralized Deep Learning," arXiv:2102.04828, Tech. Rep., 2021.

[14] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[15] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based Computation of Aggregate Information," *Foundations of Computer Science*, 2003.

[16] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. Van Steen, "Gossip-based peer sampling," *ACM Transactions on Computer Systems (TOCS)*, vol. 25, no. 3, pp. 8–es, 2007.

[17] C. Dwork and A. Roth, "The Algorithmic Foundations of Differential Privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, 2014.

[18] D. J. Watts, *Small worlds: The dynamics of networks between order and randomness*. Princeton University Press, 2000.

[19] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Transactions on networking*, vol. 11, no. 1, pp. 17–32, 2003.

[20] A. Bellet, A.-M. Kermarrec, and E. Lavoie, "D-Cliques: Compensating for Data Heterogeneity with Topology in Decentralized Federated Learning," *CoRR*, vol. abs/2104.07365, 2021. [Online]. Available: https://arxiv.org/abs/2104.07365

[21] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *ICML*, 2013.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[23] B. L. Bars, A. Bellet, M. Tommasi, and A.-M. Kermarrec, "Yes, topology matters in decentralized optimization: Refined convergence and topology learning under heterogeneous data," *arXiv preprint arXiv:2204.04452*, 2022.

[24] Y. Dandi, A. Koloskova, M. Jaggi, and S. U. Stich, "Data-heterogeneity-aware mixing for decentralized learning," *arXiv preprint arXiv:2204.06477*, 2022.

[25] Y. LeCun, C. Cortes, and C. J. Burges, "The MNIST database of handwritten digits," http://yann.lecun.com/exdb/mnist/, 2020.

[26] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," 2009.

[27] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated Multi-Task Learning," in *NIPS*, 2017.

[28] F. Hanzely, S. Hanzely, S. Horváth, and P. Richtarik, "Lower Bounds and Optimal Algorithms for Personalized Federated Learning," in *NeurIPS*, 2020.

[29] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized Federated Learning with Theoretical Guarantees: A Model-Agnostic Meta-Learning Approach," in *NeurIPS*, 2020.

[30] C. T. Dinh, N. H. Tran, and T. D. Nguyen, "Personalized Federated Learning with Moreau Envelopes," in *NeurIPS*, 2020.

[31] O. Marfoq, G. Neglia, A. Bellet, L. Kameni, and R. Vidal, "Federated Multi-Task Learning under a Mixture of Distributions," in *NeurIPS*, 2021.

[32] J.-W. Lee, J. Oh, S. Lim, S.-Y. Yun, and J.-G. Lee, "Tornadoaggregate: Accurate and scalable federated learning via the ring-based architecture," arXiv:2012.03214, Tech. Rep., 2020.

[33] I. Hegedüs, G. Danner, and M. Jelasity, "Decentralized learning works: An empirical comparison of gossip learning and federated learning," *Journal of Parallel and Distributed Computing*, vol. 148, pp. 109–124, 2021.

[34] P. Vanhaesebrouck, A. Bellet, and M. Tommasi, "Decentralized Collaborative Learning of Personalized Models over Networks," in *AISTATS*, 2017.

[35] V. Zantedeschi, A. Bellet, and M. Tommasi, "Fully Decentralized Joint Learning of Personalized Models and Collaboration Graphs," in *AISTATS*, 2020.

[36] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual Averaging for Distributed Optimization: Convergence Analysis and Network Scaling," *IEEE Transactions on Automatic Control*, vol. 57, no. 3, pp. 592–606, 2012.

[37] I. Colin, A. Bellet, J. Salmon, and S. Clémençon, "Gossip Dual Averaging for Decentralized Optimization of Pairwise Functions," in *ICML*, 2016.

[38] A. Nedić, A. Olshevsky, and M. G. Rabbat, "Network Topology and Communication-Computation Tradeoffs in Decentralized Optimization," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 953–976, 2018.

[39] O. Marfoq, C. Xu, G. Neglia, and R. Vidal, "Throughput-Optimal Topology Design for Cross-Silo Federated Learning," in *NeurIPS*, 2020.