Semester Project

Exploration of D-Cliques Variations and Edges Cases for Decentralized Federated Learning

Raphaël Attias

January 7, 2022

Project Advisor: Prof. Dr. Anne-Marie Kermarrec
Project Supervisor: Erick Lavoie

Abstract

Federated learning aims at training a machine learning algorithm, for instance, deep neural networks, on multiple local datasets contained in local nodes without explicitly exchanging data samples. The general principle consists in training local models on local data samples and exchanging the weights between these local nodes at some frequency to generate a global model shared by all nodes. A recent approach developed by the SACS laboratory at EPFL is D-Cliques (Bellet et al. 2021). This topology provides an intuitive structure, where a local node neighborhood called *Cliques* should aim to provide a close representation in terms of class distribution to the global network.

In this report, we take a closer look at Greedy Swap, an iterative algorithm that can build D-Cliques topology with impressive performance after training. First, we study the impact of the selection of the permutation on the convergence rate of both the skew and the accuracy during training. Then in the following section, we discuss the requirements and hypothetical results we could obtain with a decentralized implementation of Greedy Swap. In the next section, we look at the relationship between the sparsity of connections between Cliques and the redundancy of the data. Finally, we study the first implementation of multi-clique ownership, where a node could belong simultaneously to multiple Cliques.

Contents

List of Figures

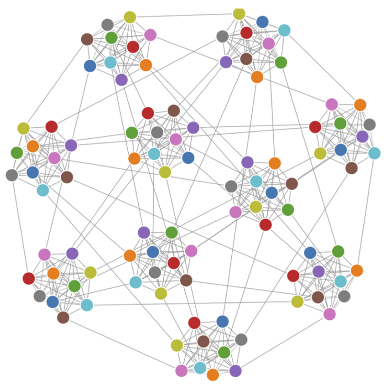
1 Introduction on D-Cliques and Greedy Swap

Standard machine learning approaches require centralizing the training data on one machine or in a data center. Federated Learning is a modern approach that trains a model across multiple decentralized devices or servers, holding local data samples, and without exchanging them. Federated Learning (FL) has met in the last 5 years a lot of interest by researchers, as it can build robust machine learning models without sharing data. This approach also addresses critical issues such as data privacy, security, and access to heterogeneous data. Federated Learning can be orchestrated either in a centralized or decentralized (DFL) setting, which differs in the coordinator being either a central server or the nodes of the network themself.

One key hyper-parameter for DFL is the network topology, which describes how the local outputs are pooled and the way the nodes communicate with each other. In particular one should seek for a balance between a fully connected topology, where all nodes communicate together, and a sparse topology, where each node is aware of only a proportionally small subsample of neighbor nodes. The main advantage of being fully connected is that such topology is equivalent to one single node entity that would be aware of all updates and data.

Decentralized Stochastic Gradient Descent (D-SGD) (Nedic et al. 2009) provides an algorithm for optimizing a set of objective functions distributed over a network. Each node maintains estimates of the optimal optimization step on information concerning his cost function, and exchanges these estimates directly with the other nodes in the network. The latter task highlights the necessity of building meaningful and relevant topology.

1.1 D-Cliques



D-Cliques (Bellet et al. 2021) is a recent approach to coordinate and structure a network for DFL. While a fully connected topology is not practical due to the number of edges increasing quadratically, D-Cliques provide an intuitive approach in providing locally fully connected neighborhoods. Each node belongs to a Clique, a set of fully connected nodes with data distribution as close as possible to the global distribution of the data through the network. Each Clique of the network provides a fair representation of the true data distribution. By fully connecting only the Cliques and the nodes within the Cliques, on a 1000-nodes network, the D-Cliques topology provides a decrease of 98% of the number of edges in the network compared to the $\frac{n(n-1)}{2}$ edges in a fully connected topology. The connection between the nodes remains sparse, expected in a most relevant neighborhood (i.e the Clique).

1.2 Greedy-Swap

In the best-case scenario, D-Cliques topology can build a model with performance close to a single centralized node. An efficient algorithm proposed by D-Cliques (Bellet et al. 2021) is Greedy-Swap. Starting from a random D-Cliques setting, this iterative algorithm greedily permutes nodes in pairs if the permutation reduces the skew between the original Cliques and the global distribution. This approach guarantees that the average skew of Cliques strictly decreases, and with enough iterations can build topology close or equal to optimum solutions. Some assumptions are made, notably that the topology is built before training, thus that we do not expect the class distribution to change within the Cliques during training. The global distribution is also assumed to be easily obtainable through pre-processing steps (Jelasity et al. 2005). This topology building algorithm has been studied and implemented in centralized settings but should be easily generalized to a decentralized setting.

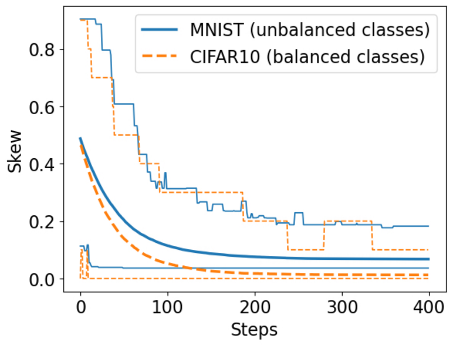


Figure 1.1: Mean skew at each iteration of Greedy Swap.

Algorithm 1 D-Cliques Construction via Greedy Swap

```
Require: maximum clique size $M$, max steps $K$, set of all nodes $N = \{1, 2, \ldots, n\}$, procedure inter($\cdot$) to create intra-clique connections
$DC \leftarrow []$
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 $DC$
    $s \leftarrow skew(C_1) + skew(C_2)$
    $swaps \leftarrow []$
    for $i \in C_1, j \in C_2$ do
        $s' \leftarrow skew(C_1 \setminus \{i\} \cup \{j\}) + skew(C_2 \setminus \{i\} \cup \{j\})$
        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 $G = (N, E \cup$ inter($DC$)$)$
```

2 Statistical Distances

The main goal of the D-Cliques topology is to gather local clusters named *Cliques*, with local distribution of the data that would be as close as possible to the data distribution at the global scale. One question among many arise, how can we qualitatively measure the distance between the global and clique distribution? Reducing this skew is synonymous with reducing the data heterogeneity between the clique and global distribution. In this section, we will compare multiple distances, and try to bring statistical insights when targeting a lower clique skew. We will discuss the impact of the choice of permutation in the greedy swap when building the final node topology. One empirical observation is that the choice of metric does not seem to have a significant impact on either the final mean test accuracy or the variance across nods.

2.1 Introduction

A **statistical distance** $d(X, Y)$ qualifies the distance between two statistical objects X and Y, such as probability distributions. Its use is crucial to assess the adequacy of a model and the similarity in the distribution of two environments. One should note that statistical distances provide a way to measure the similarity in distribution, but do not provide more insights on statistical dependence or correlation between two objects. Nonetheless, the use of such distances is useful as error functions and is used as such in the context of data heterogeneity between a clique and the global environment.

One should note that a statistical distance is not necessarily a **metric**, and as such we should avoid this misnomer. In fact, there is no rigorous definition for such distance, and compared to the definition of metric we may lack symmetry, subadditivity, or more importantly the **identity of indiscernibles**:

$$d(X, Y) = 0 \iff X = Y$$

This last property is particularly important in the context of D-Cliques, as one of the objectives is to reach a local distribution d_C close to or equal to the global distribution d_G. This section will focus on distances with this property, thus we do not expect to see any final difference in performance for topology obtained with different metrics. However, we will observe through empirical experiments if there is any difference in the rate of convergence for the test accuracy, and the variance in error through the nodes of the environment.

In the original work of D-Cliques (Bellet et al. 2021), the distance measure between any two distributions was named *skew*. More commonly called *total variation distance*, this measure is identical to the L_1 distance between the classes distributions in a clique and the global distribution.

$$\text{tvd}(C) = \sum_{l=1}^{L} |p_C(y=l) - p(y=l)|$$

2.2 Introducing different statistical distances

Through multiple experiments, we will try to have a qualitative insight into the change of test accuracy variance observed during the training, and the rate of convergence. In addition to the baseline *tdv*, we will compare 3 additional metrics that respect the indiscernibility property and study their statistical context.

2.2.1 Euclidean distance

The Euclidean norm, also known as the L_2 norm, has the advantage of being more sensitive to outliers than the skew. This sensitiveness may help in the building process to find early cliques with distributions diverging from the global distribution.

$$\text{euclidean}(C) = \sum_{l=1}^{L} |p_C(l) - p(l)|^2$$

2.2.2 Hellinger distane

The Hellinger norm is often considered as the probabilistic extension of the euclidean norm. The Hellinger distance is the canonical distance used in the space of exponential distributions functions and is also a proper metric.

$$\text{hellinger}(C) = \sum_{l=1}^{L} \left| \sqrt{p_C(l)} - \sqrt{p(l)} \right|^2$$

2.2.3 Symmetric Entropy

Essentially, what we are looking at with the KL divergence is the expectation of the log difference between the probability of data in the original distribution with the approximating distribution. If we think in terms of $\log_2$ we can interpret this as "how many bits of information we expect to lose".

$$\text{sym\_entropy}(C) = \sum_{l=1}^{L} p_C(l) \log \frac{p_C(l)}{p(l)} + p(l) \log \frac{p(l)}{p_C(l)}$$

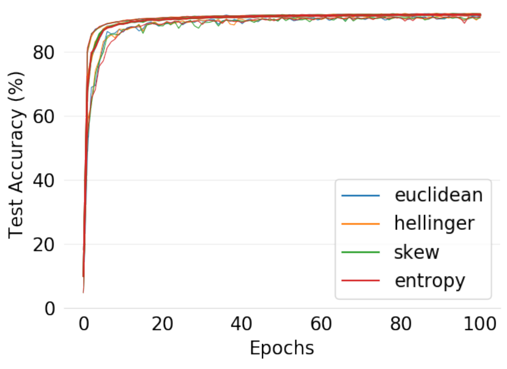
2.3 Comparing the metrics at random

In this subsection, we will compare the aforementioned metrics in the context of the greedy swaps permutations steps. As we have seen in Algorithm 1, at each subsequent step a permutation of two nodes is picked at random among the set of permutations that reduces the skew for a given metric. Can we reduce variance during training when building the topology with a different metric?

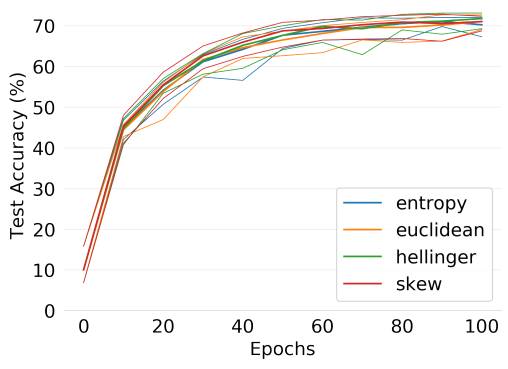
To study this question, we will subsequently compare the test accuracy of models built upon topology where Greedy Swap was used with different metrics. We will also take a closer look at the skew convergence of each of the distances. As the choice of permutation is done at random, we expect that on average the models will perform similarly due to the introduced stochasticity. In this section, we will compare the 4 aforementioned distances using the following setup: MNIST/CIFAR, using a linear/GLenet model, batch size 128/20, learning rate 0.1/0.002, momentum 0.0/0.9, and 100 epochs. The topology consists of 100 nodes over 10 cliques.

We observe on figure 2.4(a) and 2.4(b) that the choice metric does not seem to impact the average test accuracy nor its variance. We may note that the euclidean metric seems to perform just as great as the other dissimilarity metrics but with lower variance than the skew, Hellinger, and entropy metrics.

However, in figure 2.2, we have compared the average skew across Cliques with different choices of statistical distance at each iteration of Greedy Swap, when the permutation is chosen at random. We can see that from all the distances, the total variation distance both has the highest skew and is the latest to convergence, at approximately 150 iterations of Greedy Swap. All the other statistical distances improve on those two points, with notably the symmetric relative entropy which starts from the first iterations with the lowest skew, and the Euclidean distance which reaches convergence at only 100 iterations.



(a) Linear Model on MNIST



(b) GNLenet on CIFAR10

Figure 2.1: Test Accuracy convergence for topology built using Greedy Swap with different statistical distance, using random choice of permutation. We observe that in average the method reach the same test accuracy.

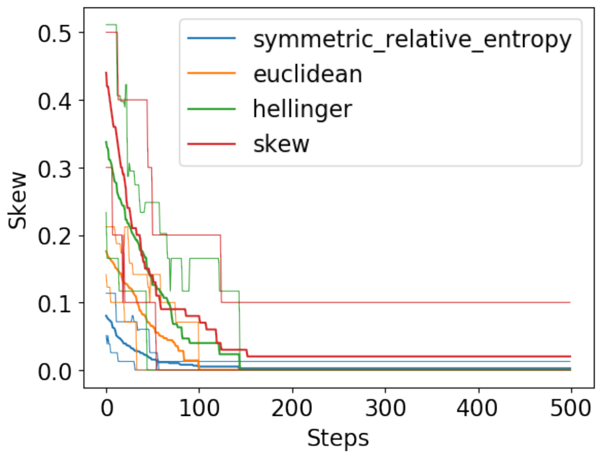


Figure 2.2: Mean skew across the Cliques of the topology during Greedy Swap for different metrics, when the permutation is chosen at random. We observe that all distances reach convergence before the skew, and with a greater rate of convergence. The euclidean distance reaches convergence with $\simeq 100$ iterations, instead of $\simeq 150$ for the skew distance.

Now one could wonder since the choice of candidates is done at random, what would happen if we picked the permutation of nodes that decrease the dissimilarity metric the most? The similar results obtained in the figures may be due to the fact that the permutation candidates are indeed similar, and the random pick may mitigate any advantage of a different metric. This idea will be explored in the following subsections, first, we will compare on a single metric how the impact of choosing a random permutation compared to the best permutation, in terms of the statistical distance.

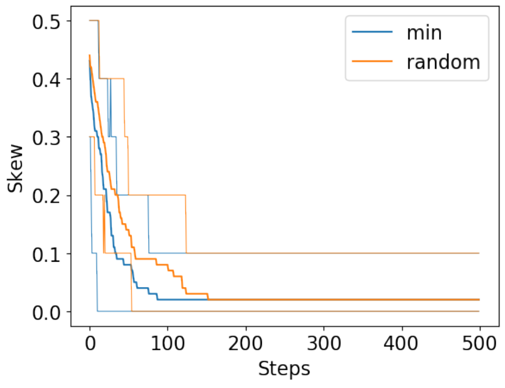
2.4 On random and best permutations in Greedy Swap

In the topology building process of Greedy Swap, the node permutation is chosen at random among the set of permutations that reduces the statistical distance of the skew compared to the global distribution. However, picking the node permutation at random from the list of possible candidates may mitigate any advantages of changing the dissimilarity metric. The intuition is again greedy since we are trying to reach the lowest skew among cliques, only choosing the best permutation should lead to equal or better skew.

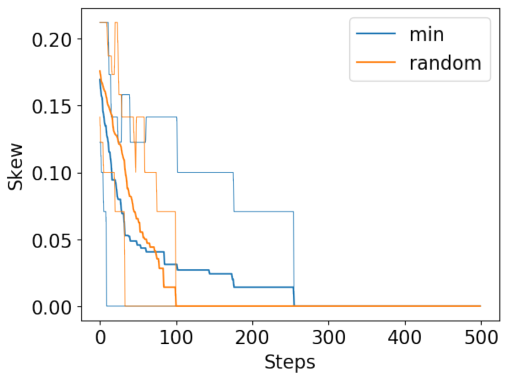
How does the choice of permutation during the Greedy Swap building process impact the final test accuracy? To proceed on this, we will compare the same initial set of nodes and dataset but by either choosing a random permutation (*random*) or the permutation that reduces the statistical distance the most (*min*). Again we will set 100 nodes on 10 cliques, with intra, and inter topology fully connected. We will look at MNIST/CIFAR, with batch size 128/20, learning rate 0.1/0.002, model Linear/GLenet, momentum 0.9, and 100 epochs.

In both the MNIST-Linear and CIFAR-GLenet experiments, we observe in Figure A.1 and A.2 that the convergence rate look appears to be extremely close when using either the minimizing permutation (*min*) or random lowering permutation (*random*). It does not appear to have an impact on the final test accuracy. The intuition that we may have here is since the set of permutations that reduces the skew at a given state is finite, with enough steps Greedy Swap with a random choice of minimizing permutation (*random*) will still end up with the same topology as we have chosen only the best permutation (*min*).

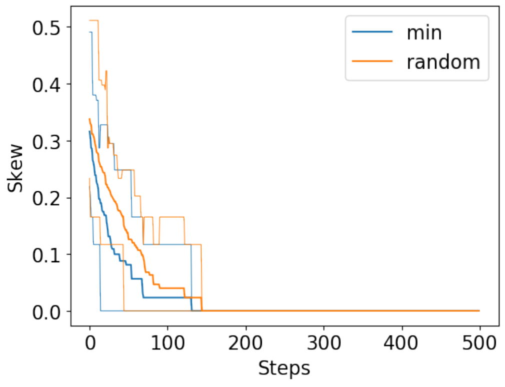
If we focus on the skew convergence rate during the iterative algorithm Greedy Swap, the results are more nuanced. From Figure 2.3, we see that choosing the minimal permutation always leads at first to a higher convergence rate and lower skew than random permutation. However, the convergence curve quickly tappers and the algorithm requires a lot more iterations to find minimal permutations, while the randomized choice has a quasi-linear decrease rate. We will note that this is not the case of the total variation distance, where choosing the minimum permutation leads to a faster convergence rate and fewer swaps than choosing at random.



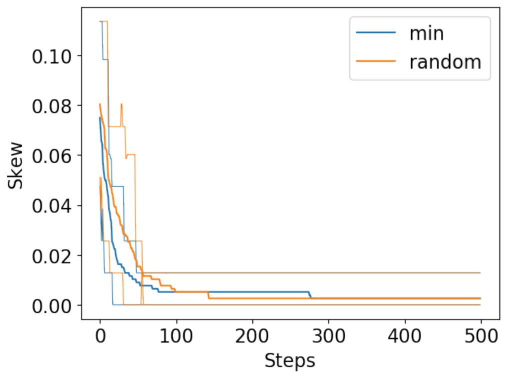
(a) Total Variation Distance



(b) Euclidean



(c) Hellinger



(d) Entropy

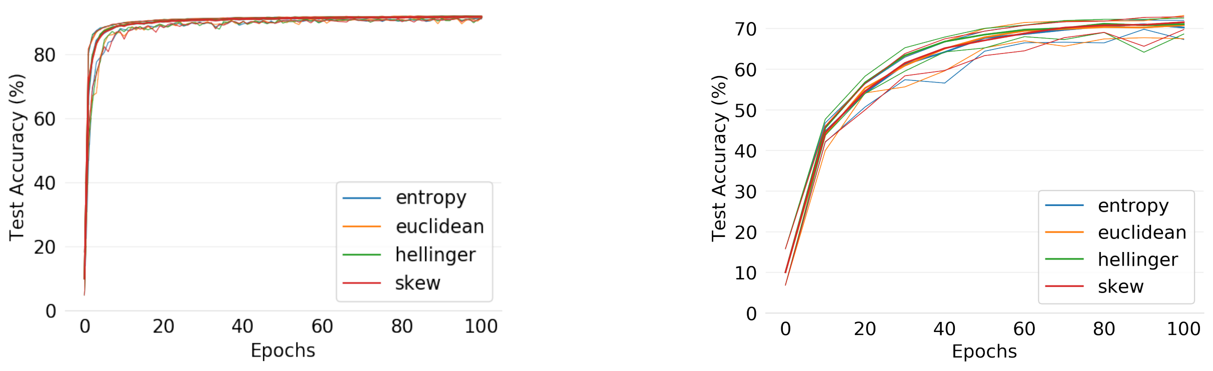
Figure 2.3: Skew convergence during Greedy Swap for different metrics

2.5 Comparing the metrics with the best permutations

In this section, we will compare the choice of metrics in greedy swap, when only the best permutation is chosen in the greedy swap algorithm. We emphasize the difference with Section 2.3, where we compared the metric but the permutation was still chosen at random in the set of permutations lowering the skew. The intuition is that since the choice of permutation was made at random in Section 2.3, we may not be able to observe any of the improvement that a more fitting metric would bring in the topology building process.

Similarly to 2.3, we observe that the choice of metrics does not seem to impact the convergence rate or the average accuracy value on the test set. However we again observe fluctuations in the variance of the model's accuracy, and the Hellinger metric seems to outperform all the other metrics until the last epoch in terms of accuracy variance for GNLenet and CIFAR10. In this experiment where the choice of the metric has much more impact on the Greedy Swap building process, we may lean toward recommending Hellinger as an appropriate metric for Greedy Swap.

When looking at the convergence rate of the skew in Figure 2.5, we observe a similar result to Figure 2.2. The total variation distance (i.e *skew*) has both the highest value and converges last to its final value. Moreover, the symmetric relative entropy reaches the lowest skew value and converges faster.



(a) Linear Model on MNIST

(b) GNLenet on CIFAR10

Figure 2.4: Test Accuracy convergence for topology built using Greedy Swap with different statistical distance, using the most decreasing permutation. We observe that in average the method reach the same test accuracy.

2.6 Conclusion

The goal of this section was to bring an understanding of the choice of the statistical distance in the Greedy Swap permutation choice, but more importantly to see how critical that choice was. Through multiple experiments, either by choosing a different distance or by selecting the permutation differently, we have seen that the expected convergence rate and accuracy variance was kept identical. As most metrics studied kept the propriety of indiscernibility, reducing the clique skew to 0 does equate to having the same class distribution. Also, the stochasticity introduced by selecting a random permutation is equivalent to enough iterations to greedily select the best permutation in Greedy Swap.

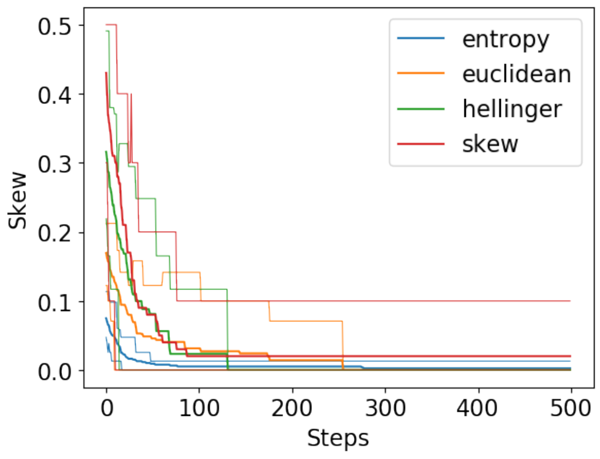


Figure 2.5: Mean skew across the Cliques of the topology during Greedy Swap for different metrics, when the permutation is chosen using the most decreasing permutation. The symmetric relative entropy reaches convergence first with $\simeq 90$ iterations.

We have also compared the choice of permutation by either selecting a random permutation or the most decreasing permutation. What we observed is that indeed, greedily choosing the permutation initially reduces the skew faster but in the long run will take more time to reach the optimal solution than random selection.

On the last note, we would recommend using the euclidean distance or the symmetric relative entropy when using greedy swap, if the number of steps is a critical parameter to minimize. While it may not have an impact on the final test accuracy, it has been shown to provide a sufficient topology with a low number of iterations.

3 | Decentralized Greedy Swap

Greedy Swap has introduced a powerful algorithm to build topology for D-Cliques in the context of Decentralized Federated Learning (DFL). DFL allows multiple participants to collaboratively train an efficient model without exposing data privacy or the need of a central supervisor. As in the current state of the project, D-Cliques has only been simulated in an experimental environment, with a global supervisor that would still manage all communication. In this section, we will study how such an algorithm would perform in the scenario where the communication of the Clique distribution was to be accounted for.

3.1 Introduction

In many large network settings, much information can be obtained from the individual nodes' training steps. However, traditional centralized approaches for computing gradient steps struggle with at least two obstacles: the data may be difficult to obtain (both due to technical reasons and because of privacy concerns), and the sheer size of the networks makes the computation expensive. A decentralized, distributed algorithm addresses both of these obstacles: it utilizes the computational power of all nodes in the network and their ability to communicate, thus speeding up the computation with the network size. And as each node knows its incident edges, the data collection problem is avoided as well.

In the context of D-Cliques, there are no such data collection issues through the topology to be avoided as each node performs its update given its local gradient, the average clique gradient, and the global model. However before we can implement and evaluate such a decentralized platform, we need to conceptualize how Cliques and nodes should communicate in a decentralized setting. In particular, we will discuss one possible way to implement a decentralized Greedy Swap algorithm, and observe empirical results on topology built in such settings.

3.2 Clique Leader

The intuition with D-Cliques was to provide a local approximation of the global distribution. We will pursue this philosophy for the Decentralized approach, we will remove the central node of the system by considering a central node for each clique, which we will call a *leader*. A leader is the analogy of the central node but only for its clique. The leader manages communication in name of the clique with the leaders of the other cliques, and has a full knowledge of the nodes of its clique.

We will use this idea to describe a first attempt for a decentralized greedy swap algorithm, which is defined in more details in 2 and 3. In a decentralized setting, two Cliques may be randomly aware of each other and perform one iteration of D-GS together. One of the two Cliques is randomly chosen as the Clique leader, we will call it C_1 and the following clique C_2. The idea is that C_1 will repetitively send each of its node n_1 to C_2, and C_2 will answer back with candidates nodes n_2. C_2 is only sending nodes whom the permutation lowers its own dissimilarity metric. From the candidates, C_1 will save in a list the permutations that also lowers its own dissimilarity metric. The leader nodes of C_1 and C_2 are in charge of the permutations in their respective clique, and of the communications with the opposite clique.

The two algorithms are similar and we provide similar performance. However in a scenario where communication to a central node would be expensive or unreliable, the fact that leaders only have to manage their own clique may be an advantage. Furthermore, as we have mentioned the following clique C_2 is answering back with a single node candidate, that can be selected using one of two criteria :

1. The node n_2 that lowers C_2 skew the most
2. One random node n_2 that lowers C_2 skew

Algorithm 2 Follow and Validate

```
1:  Require:
        the clique $C_2$ which runs this code,
        global distribution $G$,
        a node $i$ from $C_1$
2:  Define FOLLOW_AND_VALIDATE($C_2, i$)
3:  $s \leftarrow skew(C_2)$
4:  $swaps \leftarrow []$
5:  for $j \in C_2$ do
6:      $s' \leftarrow skew(C_2 \setminus \{j\} \cup \{i\}, G)$
7:      if $s' < s$ then
8:          $swaps.append(j)$
9:  if $len(swaps) > 0$ then
10:     return $swaps$
```

What would be the performance of topology built by Greedy Swap in such a decentralized setup? In the following subsection, we will compare the accuracy that could be theoretically obtained in a scenario with a Clique follower and a Clique leader. We expected the accuracy on the test set to be equivalent in average, as we have seen in the previous section that the choice of permutation (either *min* or *random*) did not have a large impact on the accuracy. More notably we expected the *centralized* method to perform with less variance, as the introduced stochasticity in the larger choice of permutations would prevent Cliques with outlier skew.

3.3 Experimental Results

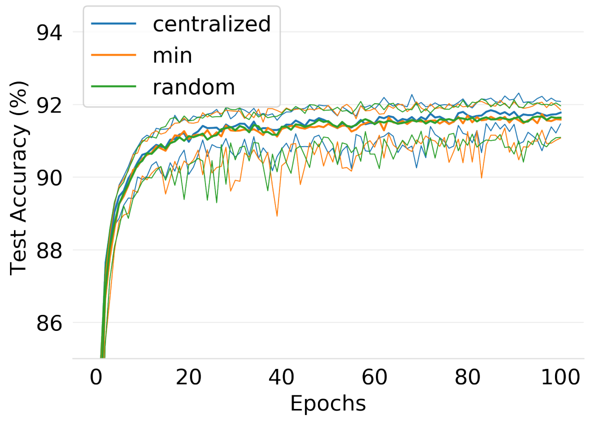
For this experiment, we compared identical training routines and setups, while only changing the initial topology builder. The initial topology (set of Cliques and Edges) is built either using the regular Greedy Swap 1 algorithm (*centralized*), the Clique Leader approach with minimal

Algorithm 3 Lead and Swap

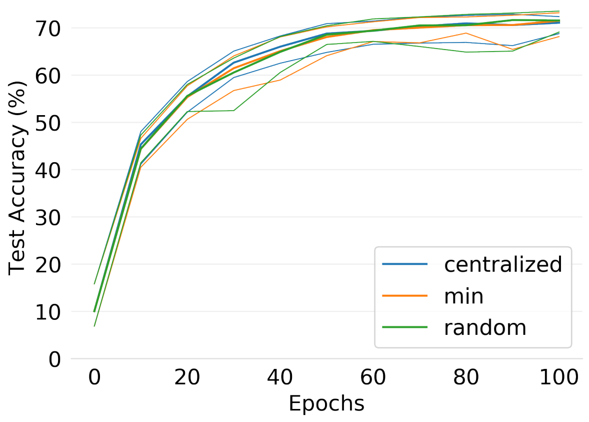
```
1: Require:
      the clique $C_1$ which runs this code,
      global distribution $G$,
      a node $i$ from $C_1$
2: Define LEAD_AND_SWAP($N_C$)
3: $C_2 \leftarrow$ sample 1 clique from $N_C$ at random
4: $s \leftarrow skew(C_1)$
5: $swaps \leftarrow []$
6: for $i \in C_1$ do
7:    for $j \in$ FOLLOW_AND_SWAP($C_2, i$) do
8:       $s' \leftarrow skew(C_1 \setminus \{i\} \cup \{j\}, G)$
9:       if $s' < s$ then
10:         $swaps.append((i, j))$
11: if $len(swaps) > 0$ then
12:    $(i, j) \leftarrow$ random element from $swaps$
13:    $C_1 \leftarrow C_1 \setminus \{i\} \cup \{j\}$
14:    $C_2 \leftarrow C_2 \setminus \{j\} \cup \{i\}$
```

node (*min*) or Clique Leader but with random node (*random*). The topology is built over 100 nodes and 10 cliques, with 2 shards/node and a shard size of 200. We used as examples MNIST/CIFAR, using a linear/GLenet model, batch size 128/20, learning rate 0.1/0.002, and 100 epochs.

On figure 3.1(a) and 3.1(b), we can observe the resulting test accuracy on these experiments, while on figure 3.2 we present the average Clique skew during the different topology building process. We note that the latter is identically the same for MNIST and CIFAR, as both are initialized with the same seed for a 10 class classification problem with 4000 samples per class.



(a) MNIST



(b) CIFAR10

Figure 3.1: Distribution of the average skew among cliques

First, we observe that for both datasets, all the three mentioned methods seem to lead to identical test accuracy on average. Another possible observation is that the variance across iterations seem to be lower for *centralized* compared to *min* and *random*. A possible explanation is that by using the total set of permutations that lower the skew, *centralized* introduces stochastic that prevent nonoptimal solution for the topology to be built. Finally, we do not observe a measurable difference between *min* and *random* in terms of either variance of mean accuracy.

In Figure 3.2, we observe similar results in concordance with the aforementioned depiction. Notably, only the *centralized* method reached an optimal solution, with skew null average skew. Not only do the other methods have a lower convergence rate, but they fail to reach the optimal solution with an average skew of approximately 0.1 and higher variance.

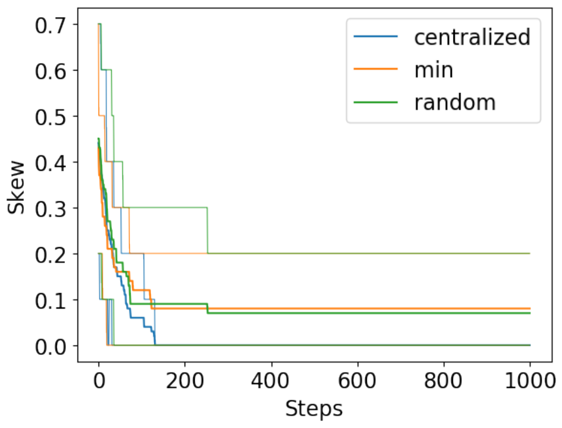


Figure 3.2: Distribution of the average skew among cliques for both CIFAR and MNIST

3.4 Conclusion

From this section, we have observed some first results on a potential approach for a decentralized Greedy Swap algorithm. Such an algorithm would perform on average the same as a centralized algorithm, but with higher observed variance. We note that this "decentralized" approach mainly focused on independently lowering the skew of the follower and leader Clique, but should not provide a reduced number of communication in practice. Furthermore, we have tested different selection criteria for the nodes of the following Clique and selecting the permutation that minimizes the leader or follower skew is a possible implementation.

4 | On the impact of the trainset on D-Cliques

4.1 Introduction

The following step consists of studying the impact of the size of the training set on the final test accuracy. On the one hand, the dataset is split into so many shares that one could think that each node needs a large enough sample, on the other hand, by being connected efficiently to other nodes one could rely on the global knowledge. The MNIST and CIFAR10 datasets appear to have many 'redundant' examples: examples that differ from one another in only minor variations. We wondered if the benefits offered by D-Cliques, in the reduction of the total number of edges, may be explained by the redundancy in the dataset. To understand whether this could be plausible, we have measured the final accuracy and convergence speed of identical networks with reduced training sets on each node, by completely removing training examples from the global dataset at random. We will also compare these experiments with a single node scenario, this will allow us to estimate when does D-Cliques provide a countermeasure for a small dataset. Furthermore, in topologies where the cliques are not fully interconnected, we will see if increasing the dataset can mitigate the sparse intercliques connections.

4.2 Experiments

In both MNIST and CIFAR, we have 10 classes that we will assume of the same size. We split this section into MNIST and CIFAR, as we will break down some insights on the computation step to keep the number of updates per node per epoch constant. We have seen in empirical tests that the total number of updates per node at each epoch has the greatest influence on the final test accuracy. We will provide some computation steps to reproduce these experiments that maintain the number of updates per node at each epoch constant.

4.2.1 MNIST Experiments

Let i be the number of samples per class, with 10 possible classes. We will try in the following experiments a range of $i = 200$ to $i = 5000$, with increments of 200, the total number of samples in the trainset will go from 2000 to 50000. If we fix the number of nodes to 100, and the number of shards to 2 per node, then the size of each shard must be $i/20$.

The number of update per node at each epoch is:

$$\text{num\_updates} = \frac{\text{shard\_size} \cdot \text{shard\_per\_node}}{\text{batch\_size}}$$

We need first to find the constant number of updates that we wish to fix for all i, and to find the appropriate batch size that will depend on both i and the number of updates. We set the batch size and the number of samples to the parameters used to obtain the best results, which is to say $i = 5000$ and $\text{batch\_size} = 128$:

$$\text{num\_updates} = \frac{250 \cdot 2}{128} \simeq 4$$

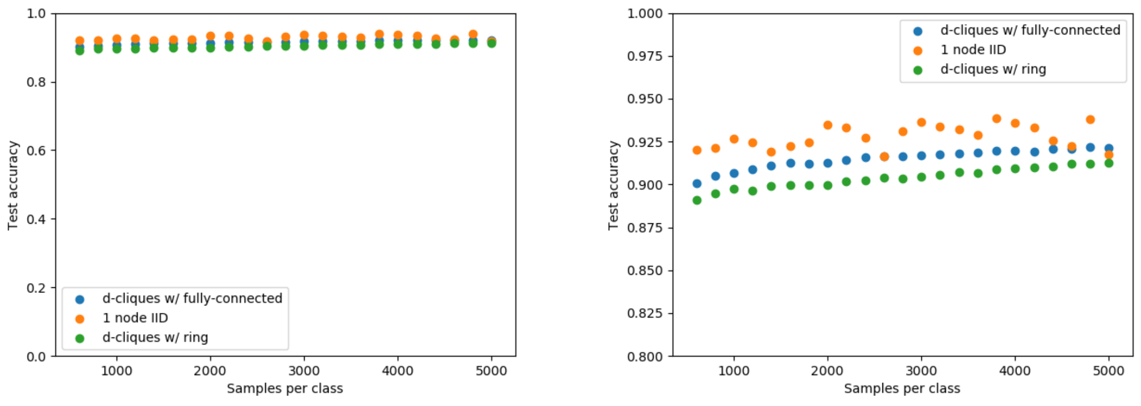
Hence, the number of updates per node at each epoch must be 4. For each trainset size i, we need to set an appropriate batch size to keep this number of update at 4.

$$\text{batch\_size}(i) = \frac{\text{shard\_size} \cdot \text{shard\_per\_node}}{\text{num\_updates}}$$

$$= \frac{(i/20) \cdot 2}{4} = \frac{i}{40}$$

Using this batch size, we have initialized topology and trained models with samples per class ranging from 200 to 5000. The learning rate was set to 0.1, with a linear model, 100 nodes through 10 Cliques. We can observe the results in figure 4.1(a) and **??**. First we see than D-Cliques topology seem to follow a much more linear increase than the simulations over one single node. This may be explained by the fact than both *d-cliques w/ fully-connected* and *d-cliques w/ ring* are the average taken over 100 nodes of the test-accuracy, while *1 node IID* is a single node and much more subject to variation. Taking the average of *1 node IID* over multiple random seed should yield to a smoother linear increase, similar to the other setups.

More interestingly we see that the rate of increase of D-Cliques fully connected or with ring does not change, and the difference between the two remains nearly constant. The hypothesis that the redundancy in examples would compensate the sparsity of intercliques connections in *d-cliques w/ ring* is not verified. If that was the case, increasing the number of samples per class should reduce the difference in accuracy observed in *ring* and *fully-connected*, but it remains the same.



(a) $[0, 1]$ y range **(b)** $[0.8, 1]$ y range

Figure 4.1: Test accuracy reached by models trained with different number of samples per class.

4.2.2 CIFAR Experiments

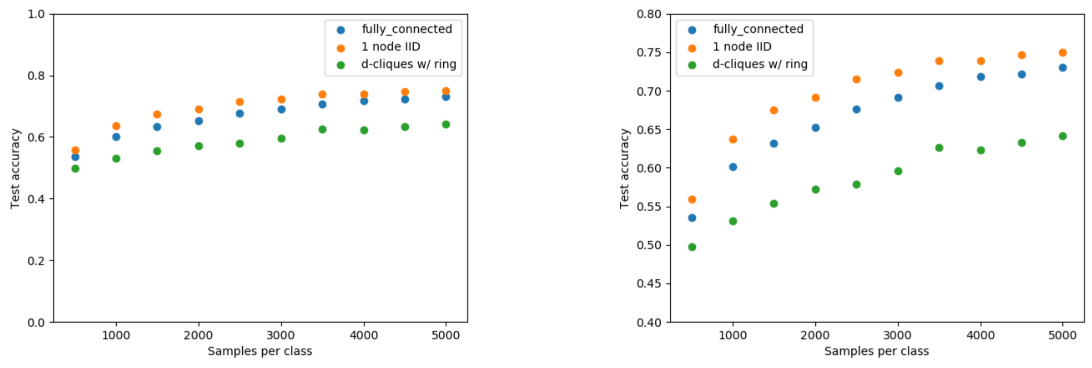
As discussed in the previous section, we will provide the experimental setup for the experiment. For the CIFAR dataset, we have experimented class size ranging from 500 to 5000, with steps of 500 samples. Keeping a size shard of $i/20$, as we have 100 nodes and 2 shards per node, since the best results where obtained with $i = 5000$ and batch_size = 20, the number of update per node at each epoch must be:

$$\text{num\_updates} = \frac{250 \cdot 2}{20} = 25$$

Then by performing similar calculations as before, we obtain that the batch size must be equal to:

$$\text{batch\_size}(i) = \frac{(i/20) \cdot 2}{25} = \frac{i}{250}$$

We can observe on figure 4.2 the final test accuracy of models trained over different amount of samples per class. We have a similar result to the previous experiment, which is to say that the 1 node IID outperforms both D-Cliques topology. The impact of the number of sample per class on the test accuracy on a D-Cliques fully connected topology is much more alike to the 1 node IID, while a ring topology does not improve as significantly.



(a) $[0, 1]$ y range **(b)** $[0.8, 1]$ y range

Figure 4.2: Test accuracy reached by models trained with different number of samples per class.

4.3 Conclusion

Through these experiments, we have attempted to see the relationship between the redundancy of the data and the topology of the network. In particular, we studied the difference in test accuracy when increasing the size of the training set. We observed that using a ring topology, the test accuracy does not increase as much as a fully connected or single node setup. Fully connected Cliques seem to be required to observe the same convergence rate and similar performance to a single node IID, and the redundant data does not help sparse D-Cliques topology improve as well.

5 On multi-clique ownership

5.1 Introduction

So far the D-Clique topology assumes that each node belongs to a single clique, with the goal that each Clique provides a similar distribution to the global distribution of the data. In real-world circumstances, we may observe a certain class distributed in fewer nodes than expected but still required to be observed in each clique. To counter that issue, we may imagine a scenario where a node belongs to multiple cliques simultaneously.

In this section, we will present an experimental setup, and discuss the results of a node belonging to multiple cliques. Consider a classification problem with 10 classes (such that MNIST or CIFAR), and 19 nodes distributed among 2 cliques. Each of the first 9-th class is represented by two nodes, thus the cliques C_1 and C_2 have an even representation of these classes. For the 10-th class, represented by the 19-th node, we will link its ownership to both C_1 and C_2.

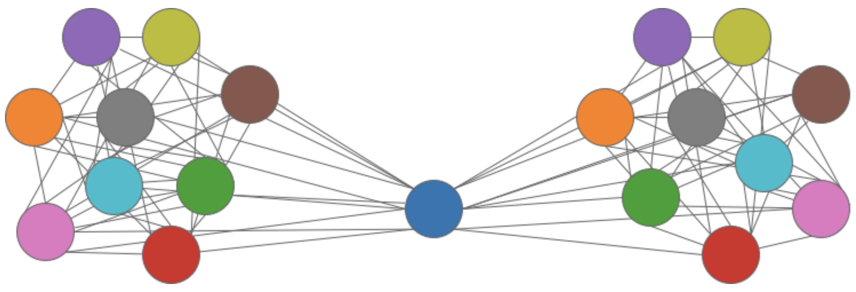
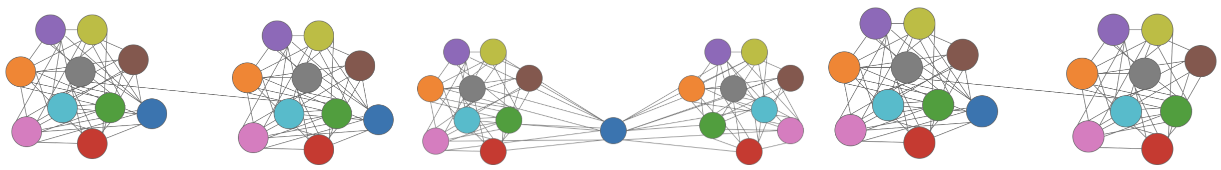


Figure 5.1: Constructed topology for this experiment. The dark blue node is owned by both clique simultaneously.

5.2 Experimental setup

In this section, we will be experimenting with two cliques of 10 nodes, one of them being shared by both cliques, thus in total 19 nodes. For this 10 classes classification problem, each of the first 9 classes will be represented by two nodes (one in each clique) with 950 samples per node. The 10-th class is under-represented with only 950 samples instead of 1900 and will be assigned to the 19-th node.

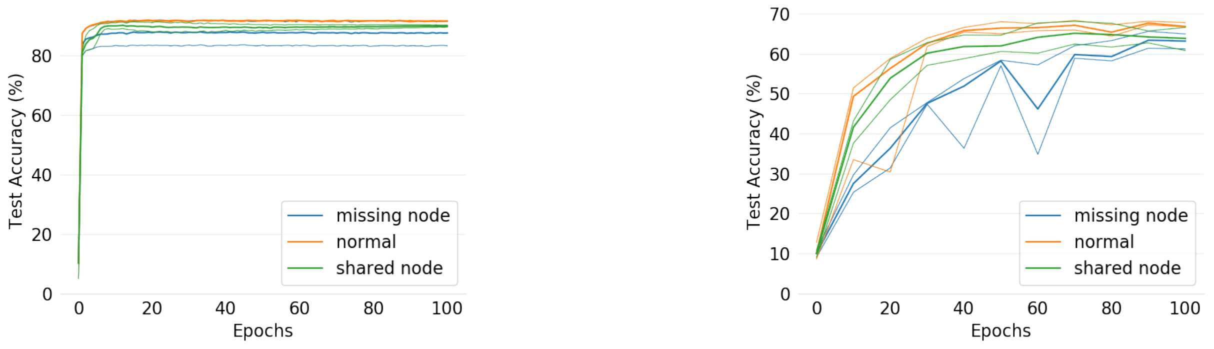
For the model and optimizers, we will use the best parameters observed in D-Cliques Bellet et al. 2021 and through this report. In other words, for MNIST a Linear model, batch size 128, momentum 0.9 and learning rate 0.1 and for CIFAR the Google Lenet, batch size 20, momentum 0.9, and learning rate 0.002. We compared a *normal* scenario where both cliques would have 10 nodes, *missing node* where one clique has 10 nodes and the other 9, and *shared node* which is the scenario described at the beginning of this section.



(a) Normal **(b)** Shared Node **(c)** Missing Node

Figure 5.2: Illustrations decrypting the different setups seen in this section. Note that the dark blue node is either present twice, or in both Cliques, or only in one Clique.

5.3 Results



(a) MNIST **(b)** CIFAR

Figure 5.3: Test accuracy reached by models, either with a *normal* topology, a *shared* node or a *missing* node.

Through empirical experiments, we observe the expected results, a *normal* topology does reach the best final test accuracy, with the highest convergence rate and lowest variance during training. This setup is the baseline where both Cliques would have the 10th node and have *iid* class.

The *shared node* topology (i.e the solution to the unbalanced class representation in our network) appears to be a great middle ground between the baseline *normal* and *missing node*, as seen in figure 5.3. The convergence rate of the former is similar to the baseline and with slightly greater variance. The final test accuracy reflects these results, as *shared node* is between *missing node* and *normal*.

5.4 Conclusion and Perspectives

As seen in this preliminary experiment, when a class of data is lacking in the network it may be an effective solution to share the node with multiple Cliques. Sharing the node would prevent the relocation of the data through the network and seems to help the model training. From a future perspective, one could think of an algorithm that would use the global knowledge distribution to estimate the number of Cliques with which one node should be shared with.

6 Conclusion

This project was an opportunity to discuss and explore variants of D-Cliques topology, and the topology building algorithm Greedy Swap. These new approaches to Decentralized Federated Learning provide not only an efficient to connect nodes for Federated Learning training on a network but also an intuitive argumentation on their performance.

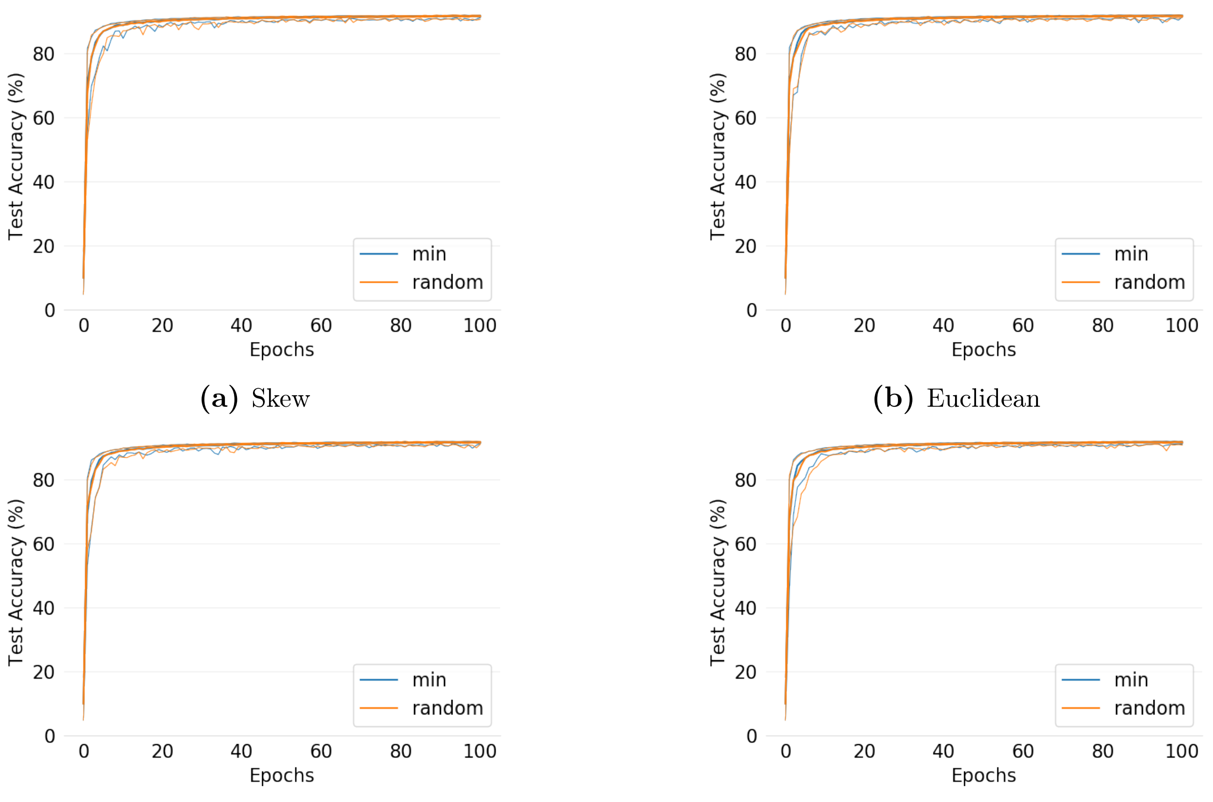
In the second section, we had a closer look at the choice of permutations in the iterative algorithm Greedy Swap. From the baseline original algorithm, where a permutation between Cliques is chosen at random using a L_1 criteria, we have compared different statistical distances and choice of permutations. We observed that while many setups could reach a zero skew topology, the number of iterations and convergence rate was greatly influenced by those parameters. One conclusion of this section is that choosing the permutation that minimizes the skew the most, using entropy or euclidian distance leads to both a faster convergence rate and fewer iterations needed for Greedy Swap.

In the third section, we discussed the imperative and requirements of a decentralized approach for Greedy Swap. The current implementation could naturally lead up to a decentralized version, but we discussed the hypothetical results that one could obtain. Notably, while the Cliques are paired during an iteration, we looked at the results one would obtain if each Clique was trying to only minimize its skew. We observed that this approach had minimal impact on the average test accuracy and with slightly greater variance. On the other hand, this approach fails to find a zero skew topology.

In the fourth section, we experimented with the size of the training set, to investigate the relationship between the redundancy of the data and the topology of the network. In particular, we wanted to see if sparse intercliques connections could be compensated by the redundancy of the data in class. When increasing the number of samples per class, we did not observe a growing difference in the test accuracy, compared to fully connected topology or a single node, as we would have expected if redundancy was an explanatory factor.

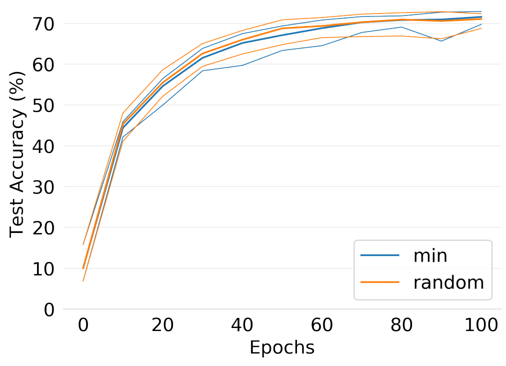
In the final section, we discussed the scenario where a class of data would be represented or only present on a smaller set of nodes. A consequence would be that some Cliques may have a larger skew since no node would have a class represent. While we could redistribute the underrepresented class to other nodes, this approach may imply too much communication. A solution implemented and discussed is multi-clique ownership, where a node could belong to multiple Cliques. As expected, this setup reaches a better test accuracy and less variance than single Clique ownership.

A Statistical Distances

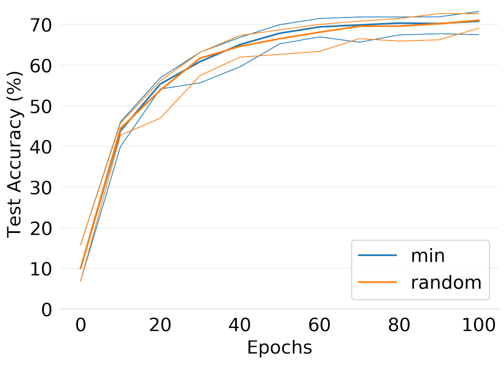


(a) Skew **(b)** Euclidean **(c)** Hellinger **(d)** Entropy

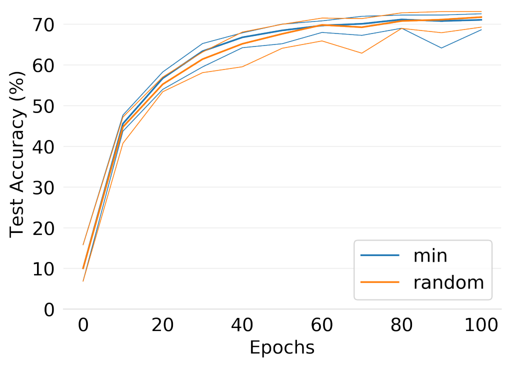
Figure A.1: Mean test accuracy for MNIST using Linear model, when building the topology with Greedy Swap with best permutation (*min*) or using any permutation that minimises the skew (*random*)



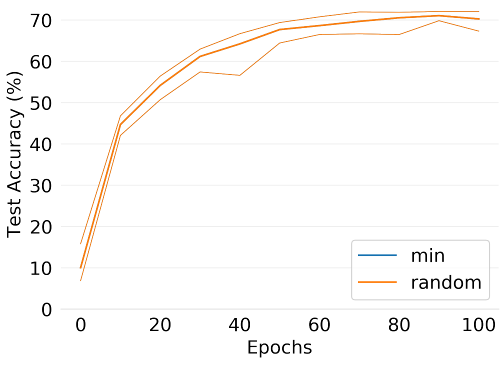
(a) Skew



(b) Euclidean



(c) Skew



(d) Entropy

Figure A.2: Mean test accuracy for CIFAR10 using GLenet, when building the topology with Greedy Swap with best permutation (*min*) or using any permutation that minimises the skew (*random*). In average, it does not affect the test accuracy of the model.

Bibliography

Bellet, Aurélien, Anne-Marie Kermarrec, and Erick Lavoie (2021). "D-Cliques: Compensating NonIIDness in Decentralized Federated Learning with Topology". In: *CoRR* abs/2104.07365. arXiv: 2104.07365. <https://arxiv.org/abs/2104.07365>.

Jelasity, Márk, Alberto Montresor, and Ozalp Babaoglu (08/2005). "Gossip-Based Aggregation in Large Dynamic Networks". In: *ACM Transactions on Computer Systems* 23, pp. 219–252. DOI: 10.1145/1082469.1082470.

Nedic, Angelia and Asuman Ozdaglar (2009). "Distributed Subgradient Methods for Multi-Agent Optimization". In: *IEEE Transactions on Automatic Control* 54.1, pp. 48–61. DOI: 10.1109/TAC.2008.2009515.