FEDGS: Federated Graph-based Sampling with Arbitrary Client Availability

Zheng Wang,[1] **Xiaoliang Fan,** [1,*] **Jianzhong Qi,** [2] **Haibing Jin,**[1] **Peizhen Yang,** [1]
Siqi Shen,[1] **Cheng Wang**[1]

[1]Fujian Key Laboratory of Sensing and Computing for Smart Cities, School of Informatics, Xiamen University, Xiamen, China
[2]School of Computing and Information Systems, University of Melbourne, Melbourne, Australia
zwang@stu.xmu.edu.cn,
fanxiaoliang@xmu.edu.cn, jianzhong.qi@unimelb.edu.au, {jinhaibing, yangpz}@stu.xmu.edu.cn,
{siqishen,cwang}@xmu.edu.cn

Abstract

While federated learning has shown strong results in optimizing a machine learning model without direct access to the original data, its performance may be hindered by intermittent client availability which slows down the convergence and biases the final learned model. There are significant challenges to achieve both stable and bias-free training under arbitrary client availability. To address these challenges, we propose a framework named Federated Graph-based Sampling (FEDGS), to stabilize the global model update and mitigate the long-term bias given arbitrary client availability simultaneously. First, we model the data correlations of clients with a Data-Distribution-Dependency Graph (3DG) that helps keep the sampled clients data apart from each other, which is theoretically shown to improve the approximation to the optimal model update. Second, constrained by the far-distance in data distribution of the sampled clients, we further minimize the variance of the numbers of times that the clients are sampled, to mitigate long-term bias. To validate the effectiveness of FEDGS, we conduct experiments on three datasets under a comprehensive set of seven client availability modes. Our experimental results confirm FEDGS's advantage in both enabling a fair client-sampling scheme and improving the model performance under arbitrary client availability. Our code is available at https://github.com/WwZzz/FedGS.

Introduction

Federated learning (FL) enables various data owners to collaboratively train a model without sharing their own data (McMahan et al. 2017). In a FL system, there is a server that broadcasts a global model to clients and then aggregates the local models from them to update the global model. Such a distributed optimization may cause prohibitive communication costs due to the unavailability of clients (Gu et al. 2021).

As an early solution to this problem, (McMahan et al. 2017) propose to uniformly sample a random subset of clients without replacement to join the training process. (Li et al. 2020) sample clients in proportion to their data sizes with replacement to obtain an unbiased estimator of update. More recently, some works take the client availability into account when sampling clients (Yan et al. 2020; Gu et al. 2021; Balakrishnan et al. 2021; Cho, Wang, and Joshi 2020; Huang et al. 2020). They show that selecting clients without considering whether the clients are active will lead to unbounded waiting times and poor responding rates. As a result, they sample only the active clients to guarantee immediate client availability (Gu et al. 2021; Cho, Wang, and Joshi 2020). For example, in Fig. 1, the server will not sample the inactive Client A and Client B at *Round* $t-1$.

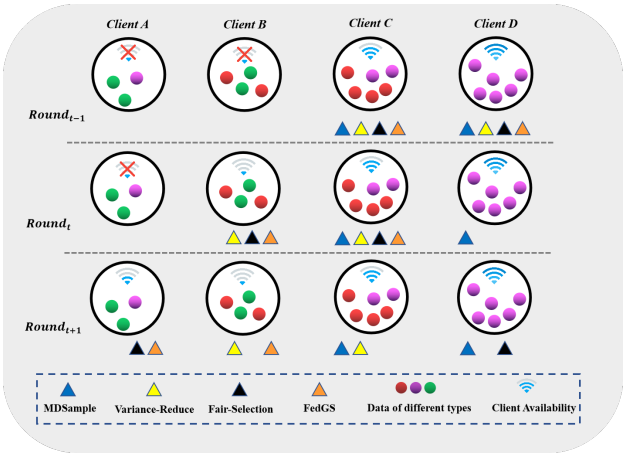


Figure 1: An motivating example: there are significant challenging to achieve both stable model updates and bias-free training with the intermittent client availability in FL.

However, enabling both stable model updates and bias-free training under *arbitrary client availability* (i.e., without any assumption on when each client will become available) poses significant challenges which have not been addressed. **On one hand**, the model is improved on the data distributions of the sampled clients at each round, which might lead to the detriment of the data specificity of non-sampled clients (Fraboni et al. 2021). For example, in Fig. 1, fair-selection (Huang et al. 2020) tries to guarantee the least sampled times for each client (and hence it is "fair"). While mitigating the long-term bias, it will ignore the data of the red type at *Round* $t+1$, since it only considers the balance of the sampled frequency of clients. This fails to observe the data heterogeneity across the clients and leads to instability of the

*Corresponding Author

model update due to the absence of the gradients computed on the "red" data. **On the other hand**, the global models trained by FL may also be biased towards clients with higher availability in a long run. Also in Fig. 1, the MDSample (Li et al. 2020) and Variance-Reduce (Fraboni et al. 2021; Balakrishnan et al. 2021) methods, which do not consider the difference in client availability, introduce bias towards the clients with higher availability (i.e. Client A is overlooked at *Round* $t+1$ regardless of not sampled in the previous rounds). In summary, there are significant challenges to address two competitive issues (i.e. stable model updates and bias-free training) that limit the FL training performance under the arbitrary client availability.

To address the issues above, we propose a novel FL framework named *Federated Graph-based Sampling* (FEDGS) to tackle the arbitrary client availability problem. We first model the data correlations of clients with a Data-Distribution-Dependency Graph (3DG) that helps keep the sampled clients data far from each other. We further minimize the variance of the numbers of times that the clients are sampled to mitigate long-term bias. Extensive experiments on three datasets under different client availability modes confirm FEDGS's advantage in both enabling a fair client-sampling scheme and improving the model performance under arbitrary client availability.

The contributions of this work are summarized as follow:

- We propose FEDGS that could both stabilize the model update and mitigate long-term bias under arbitrary client availability. To the best of our knowledge, this is the first work that tackles the two issues simultaneously.
- We propose the data correlations of clients with a Data-Distribution-Dependency Graph (3DG), which helps keep sampled clients apart from each other, and is also dedicate to mitigate long-term bias.
- We design a comprehensive set of seven client availability modes, on which we evaluate the effectiveness of FEDGS on three datasets. We observe that FEDGS outperforms existing methods in both client-selection fairness and model performance.

Background and Problem Formulation

Given N clients where the kth client has a local data size of n_k and a local objective function $F_k(\cdot)$, we study the standard FL optimization problem as:

$$\min_{\theta} F(\theta) = \sum_{k=1}^{N} \frac{n_k}{n} F_k(\theta) \tag{1}$$

where θ is the shared model parameter and $n = \sum_{k=1}^{N} n_k$ is the total data size. A common approach to optimize this objective is to iteratively broadcast the global model (i.e., its learned parameter values) θ^t to all clients at each training round t and aggregate local models $\{\theta_1^{t+1}, ..., \theta_N^{t+1}\}$ that are locally trained by clients using SGD with fixed steps:

$$\theta^{t+1} = \sum_{k=1}^{N} \frac{n_k}{n} \theta_k^{t+1} \tag{2}$$

When the number of clients is large, it is infeasible to update θ^{t+1} with θ_k^{t+1} from each client k, due to communication constraints. Sampling a random client subset $S_t \subset [N]$ to obtain an estimator of the full model update at each round becomes an attractive in this case, which is shown to enjoy convergence guarantee when the following unbiasedness condition is met (Li et al. 2019; Fraboni et al. 2021):

$$\mathbb{E}_{S_t}\left[\theta^{t+1}\right] = \sum_{k=1}^{N} \frac{n_k}{n} \theta_k^{t+1} \tag{3}$$

Further, Fraboni et al. and Balakrishnan et al. propose to reduce the variance of the estimator as follows to enable faster and more stable training:

$$Var(\nabla F_\theta) = \|Var(\nabla F_\theta)\|_1 \tag{4}$$

$$= \mathbb{E}_t \|\nabla F_{\theta^t} - \mathbb{E}[\nabla F_{\theta^t}]\|_2^2 \tag{5}$$

However, the effectiveness of these variance-reducing methods is still limited by the long-term bias caused by the arbitrary client availability as discussed earlier.

Mitigating long-term bias

We first propose an objective that could mitigate the long-term bias without any assumption on the client availability. We denote the set of available clients at the round t as $A_t \subseteq [N]$. Then, sampled clients should satisfy $S_t \subseteq A_t$ and $|S_t| \leq M$, where M is the maximum sample size limited by the server's capacity. To mitigate the impact of unexpected client availability on the sampled subset from a long-term view, we sample clients by minimizing the variance of the sampling counts of clients (i.e., the numbers of times that the clients are sampled after t rounds). Let the sampling counts of N clients after t rounds be $\mathbf{v}^t = [v_1^t, ..., v_N^t]$ where $v_k^t = \sum_{\tau=1}^{t} \mathbb{I}(k \in S_\tau) = v_k^{t-1} + \mathbb{I}(k \in S_t)$. Then, the variance of the client sampling counts after round t is:

$$Var(\mathbf{v}^t) = \frac{1}{N-1} \sum_{k=1}^{N} (v_k^t - \bar{v}^t)^2 \tag{6}$$

$$= \frac{1}{N-1} \sum_{k=1}^{N} \left(v_k^{t-1} + \mathbb{I}(k \in S_t) - (\bar{v}^{t-1} + M/N)\right)^2 \tag{7}$$

As discussed earlier, only balancing participating rates for clients may introduce large variance of model updates that slows down the model convergence (Fraboni et al. 2021). To enable a stable training, we introduce low-variance model updates as a constraint on the feasible space when minimizing the variance of sampling counts of the clients. We thus formulate our sampling optimization problem as:

$$\min_{|S_t| \leq M, S_t \subseteq A_t} Var(\mathbf{v}^t) \tag{8}$$

$$\text{s.t.} \qquad Var(\nabla F_{S_t}(\theta^t)) \leq \sigma^2 \tag{9}$$

where $\sigma^2 \geq 0$ is a coefficient that allows to search for a trade-off between the two objectives of stable model updates

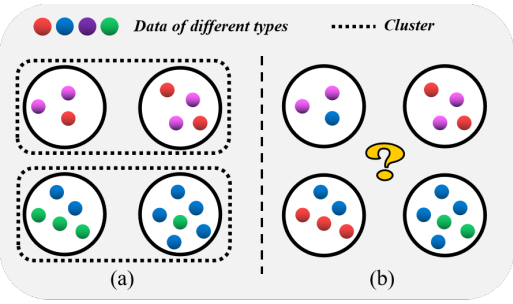


Figure 2: An example of two types of local data distributions: (a) a simple cluster; and (b) a complex cluster.

and balanced client sampling counts. When $\sigma^2 \to \infty$, the optimal solution will select the currently available clients with the lowest sampling counts. On the other hand, a small σ^2 will limit the sampled clients to those with adequately small variance of the corresponding model updates.

Methodology

This section presents our solutions to the optimization problem above (Eq. 8 and 9). The main challenge lies in converting the constraint on the variance of the global model update into a solvable one. For this purpose, we utilize the data similarity between clients to increase the data diversity of the sampled subset.

Variance Reduction Based On 3DG

Before illustrating our method, we briefly review previous works that address the Gradient Variance Reduction problem in FL. Li et al. add an proximal term to the local objectives to prevent the model from overfitting on the local data. Karimireddy et al. use control variate to dynamically correct the biased updates during local training. These two methods can avoid large model update variance by debiasing the local training procedure, which is orthogonal to the sampling strategy. Fraboni et al. group the clients into M clusters and then sample clients from these clusters without replacement to increase the chance for each client to be selected while still promising the unbiasedness of the model updates. Similarly, Balakrishnan et al. approximates the full model updates by enlarging the diversity of the selected client set, which is done by minimizing a relaxed upper bound of the difference between the optimal full update and the approximated one. Such a relaxation aims to achieve that for each client $k \in [N]$ there exists an adequately similar client $i \in S_t$ in the sampled subset.

The existing methods work well when there are obvious clusters of clients based on their local data distributions in Fig. 2(a). However, when the local data distributions are too complex to cluster like Fig. 2(b), clustering the clients cannot accurately capture the implicit data correlations between clients, which may lead to performance degradation. Meanwhile, minimizing the relaxed upper is not the only means to enlarge the diversity of the sampled clients. We can achieve the same purpose without such minimization.

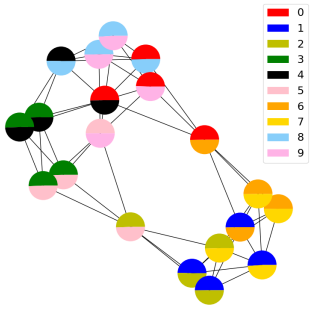


Figure 3: A visualized example of the oracle 3DG generated on CIFAR10 partitioned by 20 clients and each of them owns data with only two labels. The different color means the ratio of corresponding labels in their local dataset.

To better describe the correlations among clients' local data distributions, we model the local data distribution similarities with a Data-Distribution-Dependency Graph (3DG) instead of grouping the clients into discrete clusters, as shown in Fig. 3. Then, we show that keeping a large average shortest-path distance between the sampled nodes (i.e. clients) on the 3DG helps approximate the full model update. Intuitively, encouraging the sampled nodes to spread as far as possible helps differentiate the sampled local data distributions, which brings a higher probability to yield good balanced approximations for the full model update. This is proven as Theorem 1.

Theorem 1. *Suppose that there are C types of data (i.e., C types of labels) over all datasets, where each data type's ratio is p_i such that the dataset can be represented by the vector $\mathbf{p}^* = [p_1^*, ..., p_C^*], \mathbf{1}^\top \mathbf{p}^* = 1$. Without losing generality, consider $\mathbf{p}^*$ to be uniformly distributed in the simplex in $\mathbb{R}^C$, the number of local updates to be 1, and 3DG is a complete graph. A larger distance of sampled clients on the 3DG leads to a more approximate full model update.*

Proof. See Appendix A. □

Although the proof is based on that 3DG is a complete graph, we empirically show that keeping the clients far away from each others on the 3DG can benefit FL training even when this assumption is broken.

Construction of 3DG in FL

Now we discuss how to construct the 3DG. A straightforward approach is to directly calculate the distance (e.g., KL divergence) between different local data distributions, which is infeasible in FL because the clients do not share their local data. Without loss of generality, we assume that there is a feature vector $\mathbf{u}_k \in \mathbb{R}^d$ that can well represent the information about the local data distribution of each client c_k. We argue that this is achievable in practice. For example, training an ML model for tasks of supervised learning (e.g., classification) usually face severe data heterogeneity in FL, where there may exist label skewness in clients' local data (e.g. each client only owns data with a subset of

labels). In this case, the label distribution vectors (i.e. the number of items of each label) can well reflect the bias of each client's local data. Once given the feature vectors $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_N]$, we can easily calculate the similarity between any two clients c_i and c_j with a similarity function $f_{sim} : \mathbb{R}^d \times \mathbb{R}^d \to [0, 1]$ as:

$$\mathbf{V} = [V_{ij}]_{N\times N}, V_{ij} = f_{sim}(\mathbf{u}_i, \mathbf{u}_j) \tag{10}$$

Then, the similarity matrix $\mathbf{V}$ can be converted into an adjacent matrix $\mathbf{R}$ for the 3DG over the clients for the 3DG over the clients by:

$$R_{ij} = \begin{cases} 0, & i = j \\ e^{-V_{ij}/\sigma^2} & i \neq j, V_{ij} \geq \epsilon \\ \infty, & i \neq j, V_{ij} < \epsilon \end{cases}$$

where $\epsilon > 0$ is a positive threshold used to control the sparsity of the adjacent matrix and σ controls the diversity of the edge weight. A large value of σ will lead to small difference between the edge weights.

The feature vector $\mathbf{u}_k$ can also leak sensitive information about the clients, and it may not be exposed to the other clients or the server. It is necessary for the server to obtain the similarities between clients in privacy-preserving manner to reconstruct or accurately approximate the *oracle 3DG* (i.e., the 3DG corresponding to the true features $\mathbf{U}$). To achieve this goal, we present two methods that can help the server construct the 3DG.

The first is to use techniques based on Secure Scalar Product Protocols (SSPP) (Wang et al. 2009; Shundong, Mengyu, and Wenting 2021), which aims to compute the dot product of private vectors of parties. We argue that any existing solutions of SSPP can be used to reconstruct the similarity matrix $\mathbf{V}$ of clients in our settings, and we detail one of the existing solutions for scalar product protocal (Du and Zhan 2002) with a discussion on how to adapt it to build the 3DG in the Appendix D.

Although the SSPP-based methods can construct the oracle 3DG without any error, it cannot be adapted to the case where the feature vectors are difficult or impossible to obtain. In addition, the SSPP-based method only applies when the similarity function f_{sim} is a simple dot product.

We propose a second method that computes the similarity between clients based on their uploaded model parameters. Given the locally trained models θ_i^{t+1} and θ_j^{t+1}, a straightforward way to calculate the similarity between the two clients is to compute the cosine similarity of their model updates (Xu and Lyu 2020; Xu et al. 2021)

$$V_{ij} = \max(\frac{\Delta\theta_i^{t\top}\Delta\theta_j^t}{\|\Delta\theta_i^t\|\|\Delta\theta_j^t\|}, 0) \tag{11}$$

where $\Delta\theta_i^{t\top} = \theta_i^{t+1} - \theta^t$. However, since the model parameters' update is usually of extremely high dimensions but low rank (Azam et al. 2021), the direct cos similarity may contain too much noise, causing inaccuracy when constructing the 3DG. Motivated by (Baek et al. 2022), we instead compute the functional similarity based on the model parameters to overcome the problem above. We first feed a batch of random Gaussian noise $\epsilon \sim \mathcal{N}(\mu, \Sigma)$ to all the locally trained models, where μ and Σ are respectively the mean and covariance of a small validation dataset owned by the server (Zhao et al. 2018). Then, we take the average of the lth layer's network embedding on this batch for each client to obtain e_i, and we compute the similarity as:

$$e_i = \theta_i(\epsilon)[l], V_{ij} = \max(\frac{e_i^\top e_j}{\|e_i\|\|e_j\|}, 0) \tag{12}$$

where we set l as the output layer in practice.

Another concerning issue is that the server may not have access to all the clients' feature vectors during the initial phase. As a result, the adjacent matrix of clients may need to be dynamically built and polished round by round. Nevertheless, we emphasize that we are not trying to answer how to optimally capture the correlations between clients' local data distributions in FL. Instead, we aim at showing that the topological correlations of clients' local data can be utilized to improve the training process of FL, and we key how to build the optimal 3DG for as our future work.

For convenience, we simply assume that all the clients are available at the initial phase, by which the server can obtain the 3DG just before training starts. We empirically show the effectiveness of the approximated 3DG in Sec. 4.4.

FEDGS

We now present our proposed Federated Graph-based Sampling (FEDGS) method. As mentioned in Sec. 3.1, we bound the variance of the global model update at each round by keeping a larger average shortest-path distance between each pair of sampled clients. Given a 3DG, we first use the Floyd–Warshall algorithm to compute the shortest-path distance matrix $\mathbf{H} = [h_{ij}]_{N\times N}$ for all pairs of clients. Let $s_k^t \in \{0, 1\}$ be a binary variable, where $s_k^t = 1$ means client c_k is selected to participate training round t and $s_k^t = 0$ otherwise. Then, the sampling result in round t can be $\mathbf{s}_t = [s_1^t, ..., s_N^t] \in \{0, 1\}^N$, where the average shortest-path distance between sampled clients is written as:

$$g(S_t) = \frac{2}{N(N-1)} \sum_{i,j\in S_t, i\neq j} h_{ij} s_i^t s_j^t = \frac{\mathbf{s}_t^\top \mathbf{H} \mathbf{s}_t}{N(N-1)} \tag{13}$$

Accordingly, we replace the constraint in Equation (9) with $g(S_t) \geq \alpha$, and we convert this constraint into a penalty term added into the objective. After rewriting the equation (8) to be a maximization problem based on s_t, we obtain:

$$\max_{\mathbf{s}_t \leq \mathbf{a}^t, s_k^t \in \{0,1\}} \frac{\alpha \mathbf{s}_t^\top \mathbf{H} \mathbf{s}_t}{N(N-1)} - \frac{1}{N-1} \mathbf{z}^\top \mathbf{s}_t \tag{14}$$

$$\text{s.t.} \qquad \mathbf{1}^\top \mathbf{s}_t = min(M, |A_t|) \tag{15}$$

where $z_k = 2(v_k^{t-1} - \bar{v}^{t-1} - M/N) + 1$, $\mathbf{a}^t = \{a_1^t, ..., a_N^t\} \in \{0, 1\}^N$ and $a_k^t = 1$ means client c_k is available in round t. Note that $s_k^t = 0$ for the clients unavailable clients in round t and $s_k^{t2} = s_k^t$. Thus, Equation (14) can be

reduced to:

$$\max_{\tilde{s}_k^t \in \{0,1\}} \tilde{\mathbf{s}}_t^\top \left(\frac{\alpha}{N} \tilde{\mathbf{H}} - \mathbf{diag}(\tilde{\mathbf{z}}) \right) \tilde{\mathbf{s}}_t \tag{16}$$

$$\text{s.t.} \quad \mathbf{1}^\top \tilde{\mathbf{s}}_t = min(M, |A_t|) \tag{17}$$

$\tilde{\mathbf{s}}_t = [s_{i1}^t, ..., s_{i|A_t|}^t] \in \{0,1\}^{|A_t|}$ and $s_{ij}^t = 1$ represents that the jth client in the available set is selected. $\tilde{\mathbf{z}} \in \mathbb{R}^{|A_t|}$ and $\tilde{\mathbf{H}} \in \mathbb{R}^{|A_t| \times |A_t|}$ also only contains the element where the corresponding clients are available in round t.

This rewritten problem is a constrained mixed integer quadratic problem, which is a variety of an NP-hard problem, *p-dispersion* (Pisinger 1999), with a non-zero diagonal. We optimize it to select clients within a fixed upper bound of wall-clock time. We empirically show that a local optimal can already bring non-trivial improvement when the client availability varies.

Aggregation Weight. Instead of directly averaging the uploaded model parameters like (Balakrishnan et al. 2021; Li et al. 2020), We normalize the ratio of the local data size of selected clients as weights of the model aggregation:

$$\theta^{t+1} = \sum_{k \in S_t} \frac{n_k}{\sum_{i \in S_t} n_i} \theta_k^{t+1} \tag{18}$$

We argue that this is reasonable in our sampling scheme. Firstly, FEDGS forces to balance the sampling counts of all the clients regardless of their availability. Thus, for convenience, we simply assume that all the clients will be uniformly sampled with the same frequency $\frac{MT_c}{N}$ in every T_c rounds, and that the size of the set of available clients $|A_t|$ is always larger than the sample size limit M in each round t. By treating the frequency $\frac{MT_c}{N}/T_c = M/N$ as the probability of each client being selected without replacement in each round $T_0 + \tau$, $(\tau \leq T_c)$, we obtain:

$$\mathbb{E}_{S_t}[\theta_{t+1}] = \mathbb{E}_{S_t} \left[\frac{M}{N} \sum_{k=1}^{N} \frac{n_k}{n_k + \sigma(S_t, k)} \theta_k^{t+1} \right] \tag{19}$$

$$= \frac{M}{N} \sum_{k=1}^{N} \frac{n_k}{n_k + \sigma_k} \theta_k^{t+1} \tag{20}$$

where $\sigma_k = \mathbb{E}_{S_t}[\sigma(S_t, k)] = \mathbb{E}_{S_t}[\sum_{j \in S_t, j \neq k} n_j] = \frac{M-1}{N-1}(n - n_k)$. Therefore, the expected updated model of the next round follows:

$$\mathbb{E}_{S_t}[\theta_{t+1}] = \frac{M}{N} \sum_{k=1}^{N} \frac{n_k}{n_k + \frac{M-1}{N-1}(n - n_k)} \theta_k^{t+1} \tag{21}$$

$$= \sum_{k=1}^{N} \frac{n_k}{n} \frac{1}{1 + \frac{1}{M} \frac{N-M}{N-1} \frac{n_k - \bar{n}}{\bar{n}}} \theta_k^{t+1} \tag{22}$$

$$= \sum_{k=1}^{N} \frac{n_k}{n} \gamma_k \theta_k^{t+1} \tag{23}$$

From Equation (22), we see that the degree of data imbalance will impact the unbiasedness of the estimation. When the data size is balanced as $n_k = \bar{n}, \forall k \in [N]$, the estimation is unbiased since $\gamma_k = 1, \forall k \in [N]$. If the data size is imbalanced, the degree of data imbalance will only have a controllable influence on the unbiasedness with the ratio of each client's local data's size to the average data size $\frac{\|n_k - \bar{n}\|}{\bar{n}}$. This impact can be immediately reduced by increasing the number of sampled clients M at each round.

The analysis above is based on the assumption that our proposed FEDGS can well approximate the results obtained by uniform sampling without replacement in ideal settings. Generally speaking, a small $Var(\mathbf{v}^t)$ will limit the difference between the sampling counts, which is also empirically verified by our experimental results. The pseudo codes in Algorithm 1 summarizes the main steps of FEDGS.

Algorithm 1: Federated Graph-Based Sampling

Input:The global model θ, the feature matrix of clients' data distribution $\mathbf{U}$, the maximum wall-clock time of the solver τ_{max}, the sizes of clients' local data n_k, the number of local updating steps E, and the learning rate η_t

```
Initialize the global model parameters θ_0 and the sampling counts of clients v^0 = [0, ..., 0] ∈ N^N.
Create the 3DG G based on the techniques in Sec. 3.2.
Compute the shortest-path distance of each pair of nodes on 3DG by Floyd Algorithm to obtain H.
for communication round t = 0, 1, ..., T − 1 do
    The server checks the set of available clients A_t.
    The server uses v^t and H to solve equation (16) within the maximum wall-clock time τ_max to obtain the sampled client set S_t ⊆ A_t
    The server broadcasts the model θ^t to clients in S_t.
    for each client k ∈ S_t do
        for each iteration i = 0, 1, ..., E − 1 do
            θ^t_{k,i+1} ← θ^t_{k,i} − η_t ∇F_k(θ^t_{k,i})
        end for
        Client k send the model parameters θ^{t+1}_k = θ^t_{k,E} to the server.
    end for
    The server aggregates the received local model parameters θ^{t+1} = Σ_{k∈S_t} n_k / (Σ_{i∈S_t} n_i) θ^{t+1}_k
    The server updates the sampling counts of clients v^{t+1}[k] ← v^t[k] + I(k ∈ S_t)
end for
```

Experiment

Experimental Setup

Datasets and models to be trained We validate FEDGS on three commonly used federated datasets: **Synthetic (0.5, 0.5)** (Li et al. 2020), **CIFAR10** (Krizhevsky, Hinton et al. 2009) and **FashionMNIST** (Xiao, Rasul, and Vollgraf 2017). For Syntetic dataset, we follow the settings use by (Li et al. 2020) to generate imbalance and non-i.i.d. dataset with 30 clients. For CIFAR10, we unequally partition the dataset into 100 clients following the label distribution $Y_k \sim Dir(\alpha p)$ (Hsu, Qi, and Brown 2019) (p is

Name	Description	Dependency			Active Rate
		Time	Data	Other	
IDeaL	Full client availability	-	-	-	1
More**D**ata**F**irst (Ours)	More data, higher availability	-	data size n_k	-	$\frac{n_k^\beta}{\max_i n_i^\beta}$
Less**D**ata**F**irst (Ours)	Less data, higher availability	-	data size n_k	-	$\frac{n_k^{-\beta}}{\max_i n_i^{-\beta}}$
YMax**F**irst (Gu et al. 2021)	Larger value of label, higher availability	-	value of label set $\{y_{ki}\}$	-	$\beta\frac{\min_i\{y_{ki}\}}{\max_{c,i}\{y_{ci}\}}+(1-\beta)$
YCycle (Ours)	Periodic availability with label values	round t	value of label set $\{y_{ki}\}$	-	$\beta\mathbb{I}\left(\cup_{y_{ki}}\frac{1+(t\%T_p)}{T_p}\in[\frac{y_{ki}}{numY},\frac{y_{ki}+1}{numY})\right)+(1-\beta)$
Log **N**ormal (Ribero, Vikalo, and De Veciana 2022)	Independent availability obeying *lognormal*	-	-	$c_k\sim\text{lognormal}(0,ln\frac{1}{1-\beta})$	$\frac{c_k}{\max_i c_i}$
Sin **L**og **N**ormal (Ribero, Vikalo, and De Veciana 2022)	*Sin*-like intermittent availability of with **LN**	round t	-	$c_k\sim\text{lognormal}(0,ln\frac{1}{1-\beta})$	$\frac{c_k}{\max_i c_i}\left(0.4sin\left(\frac{1+(t\%T_p)}{T_p}2\pi\right)+0.5\right)$

Table 1: An Overview of Different Client Availability Modes.

Dataset	Synthetic(0.5, 0.5)					CIFAR10					FashionMNIST		
Availability	**IDL**	**LN**	**SLN**	**LDF**	**MDF**	**IDL**	**LN**	**SLN**	**LDF**	**MDF**	**IDL**	**YMF**	**YC**
UniformSample	0.302	0.320	0.324	0.330	0.362	0.975	1.042	1.038	1.049	0.999	0.315	0.331	0.333
MDSample	0.302	0.322	0.328	0.328	0.326	0.971	1.037	1.048	1.051	0.991	0.315	0.333	0.338
Power-of-Choice	0.691	0.362	0.352	0.557	**0.301**	1.287	1.108	1.078	1.267	1.026	0.345	0.326	0.311
FEDPROX $\mu = 0.01$	**0.301**	0.346	0.376	0.319	0.410	0.972	1.056	1.162	1.039	0.995	0.315	0.331	0.374
FEDGS $\alpha = 0.0$	0.307	**0.305**	0.320	0.309	0.310	1.006	0.976	1.002	0.974	0.967	0.302	0.310	0.324
FEDGS $\alpha = 0.5$	0.311	0.306	0.319	**0.308**	0.311	0.977	0.973	1.000	**0.966**	0.974	**0.299**	0.312	0.312
FEDGS $\alpha = 1.0$	0.328	0.306	**0.318**	**0.308**	0.311	**0.968**	**0.972**	**0.996**	0.971	**0.963**	0.300	0.308	0.310
FEDGS $\alpha = 2.0$	0.306	0.307	0.320	**0.308**	0.311	0.970	0.975	1.001	0.973	0.969	0.310	**0.303**	0.312
FEDGS $\alpha = 5.0$	0.317	0.307	0.321	0.309	0.311	0.976	0.974	0.996	0.972	0.972	0.307	**0.303**	**0.307**

Table 2: The optimal testing loss of methods running under different client availability modes on three datasets. Each result in the table is averaged over 3 different random seeds.

the global label distribution). For FashionMNIST, we balance the data sizes for 100 clients, each of whom owns data of only two labels. We train a logistic regression model for Synthetic(0.5, 0,5) and CNN models for CIFAR10 and FashionMNIST. More details on datasets are in Appendix C.

Client Availability We first review the client availability settings discussed in existing FL literature (Ribero, Vikalo, and De Veciana 2022; Gu et al. 2021), a common way is to allocate an active probability to each client at each round. We observe that the client's active probability may depend on data distribution or time (Ribero, Vikalo, and De Veciana 2022; Gu et al. 2021). We mainly conclude the existing client availability modes and propose a comprehensive set of seven client availability modes in Table 1 to conduct experiments under arbitrary availability. For each mode, we set a coefficient $\beta \in [0, 1]$ to control the degree of the global unavailability, where a large value of β suggests a small chance for most devices to be active. A further explanation about these availability modes is provided in Appendix C, where we also visualize the active state of clients at each round.

Baselines We compare our method FEDGS with: (1) UNIFORMSAMPLE (McMahan et al. 2017), which samples available clients uniformly without replacement, (2) FEDPROX/MDSAMPLE (Li et al. 2020), which samples available clients with a probability proportion to their local data size and trains w/wo proximal term, (3) POWER-OF-CHOICE (Cho, Wang, and Joshi 2020), which samples available clients with top-M highest loss on local data and is robust to the client unavailability. Particularly, all the reported results of our FEDGS are directly based on the oracle 3DG. We put results obtained by running FEDGS on the constructed 3DG in the Appendix B.

Hyper-parameters For each dataset, we tune the hyper-parameters by grid search with FedAvg, and we adopt the optimal parameters on the validation dataset of FedAvg to all the methods. The batch size is $B = 10$ for Synthetic and $B = 32$ for both CIFAR10 and FashionMNIST. The optimal parameters for Synthetic, Cifar10, FashionMNIST are resepctively $\eta = 0.1, E = 10$, $E = 10, \eta = 0.03$ and $E = 10, \eta = 0.1$. We round-wisely decay the learning srate by a factor of 0.998 for all the datasets. More details about the hyper-parameters are put in Appendix C.

Implementation All our experiments are run on a 64 GB-RAM Ubuntu 18.04.6 server with Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz and 4 NVidia(R) 2080Ti GPUs. All code is implemented in PyTorch 1.12.0.

Results of Impact of Client Availability

We run experiments under different client availability modes to study the impact of arbitrary client availability, where the main results are shown in Table 2. Overall, the optimal model performance measured by the test loss is impacted by the client availability modes for all methods and over all three datasets. On Synthetic, UniformSample suffers the worst model performance degradation, 19.8% (i.e. **IDL** v.s. **MDF**), while that for MDSample is 8.6% (i.e. **IDL** v.s. **LDF**). FEDGS retrains a strong model performance, with a degradation no more than 5% for all the values of α. Further, our proposed FEDGS achieves the best performance under all the availability modes except for **IDL** and **MDF**. For **IDL**, our FEDGS still yields competitive results comparing with UniformSample and MDSample (0.306 v.s. 0.302).

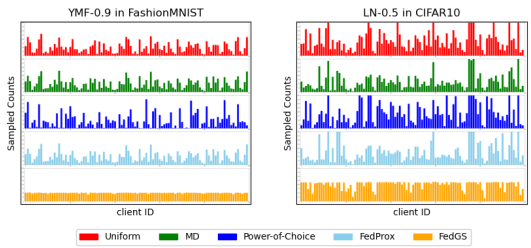


Figure 4: The results of final client sampling counts on FashionMNIST-YMF-0.9 and Cifar10-LN-0.5.

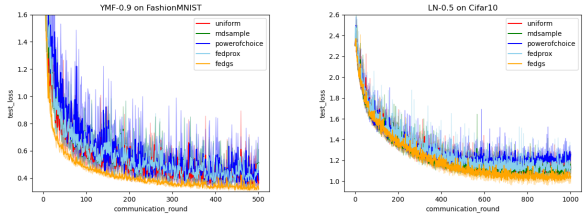


Figure 5: The testing loss curve respectively on FashionMNIST-YMF-0.9 and CIFAR10-LN-0.5.

For **MDF**, although Power-of-Choice achieves the optimal result, its model performance is extremely unstable across different availability modes, e.g., its model performance is almost 200% worse than the others for **IDL**. Meanwhile, FEDGS still achieves 4.9% improvement over MDSample and 14.3% over UniformSample in this case. For CIFAR10 and FashionMNIST, all the optimal results fall into the region runned with FEDGS. Using $\alpha > 0$ brings a non-trivial improvement over using $\alpha = 0$ in most of the client availability modes, which suggests that the variance reduction of FEDGS can also benefit FL training.

Results of Client Selection Fairness

As shown in Fig. 4, FEDGS$_{\alpha=1}$ can substantially enhance fairness in client selection, which results in a uniform distribution of client sample counts both on FashionMNIST with YMF-0.9 and CIFAR10 with LN-0.5. We also show the corresponding curves of test loss in Fig. 5, which show that FEDGS can stabilize FL training and find a better solution.

Effectiveness of 3DG Construction

To validate the effectiveness of our method to reconstruct the oracle 3DG based on the functional similarity of model parameters, we use the F1-score of predicting the edges in the oracle 3DG to measure the quality of the construction, and we compare results with those obtained using the cosine similarity. The oracle 3DG is generated with $\epsilon = 0.1$ and $\sigma^2 = 0.01$. Considering the difference in the feature space of the oracle and those of the model-based methods, we vary the value of $\epsilon \in \{0, 0.01, 0.05, 0.1, 0.5\}$ and report the results with the highest F1-score for each method. Results in Table 3 confirm the effectiveness of the proposed method to approximate the oracle 3DG, where the functional-similarity method achieves a higher F1-score than the cos-similarity method on both datasets. The results of FEDGS running on the model-based 3DG are included in Appendix B.

Dataset	Method	Precision	Recall	F1-Score
CIFAR10	*functional similarity*	0.8789	**0.8700**	**0.8744**
	cosine similarity	**1.0000**	0.1316	0.2327
FashionMNIST	*functional similarity*	**0.9761**	0.7097	**0.8218**
	cosine similarity	0.3765	**0.9853**	0.5448

Table 3: The effectiveness of how to construct 3DG.

Related Works

Client Sampling in FL

Client sampling is proven to has a significant impact on the stability of the learned model (Cho, Wang, and Joshi 2020). (McMahan et al. 2017) uniformly samples clients without replacement to save communication efficiency. (Li et al. 2020) samples clients proportion to their local data size and uniformly aggregate the models with full client availability. (Fraboni et al. 2021) reduces the variance of model to accelerate the training process. Nevertheless, these works ignored the long-term bias introduced by arbitrary client availability, which will result in the model overfitting on a particular data subset. Recent works (Ribero, Vikalo, and De Veciana 2022; Gu et al. 2021; Huang et al. 2020) are aware of such long-term bias from the arbitrary availability of clients. However, these two competitive issues (e.g. stable model updates and bias-free training) have not been considered simultaneously.

Graph Construction in FL

When it is probable to define the topology structure among clients in FL, several works directly utilized underlying correlations among different clients according to their social relations (He et al. 2021). Other works proposed to connect the clients with their spatial-temporal relations as a graph (Meng, Rambhatla, and Liu 2021; Zhang et al. 2021). However, those works are used to conduct explicit correlations between clients (e.g. social relation, spatial relation), which were not able to uncover the important and implicit connections among clients. In short, we are the first to construct Data-Distribution-Dependency Graph (3DG) to learn the potential data dependency of sampled clients, which is proven to both guarantee a fair client sampling scheme and improve the model performance under arbitrary client availability.

Conclusion

We addressed the long-term bias and the stability of model updates simultaneously to enable faster and more stable FL under arbitrary client availability. To this end, we proposed the FEDGS framework that models clients' data correlations with a Data-Distribution-Dependency Graph (3DG) and utilizes the graph to stabilize the model updates. To mitigate the long-term bias, we minimize the variance of the numbers of times that clients are sampled under the far-distance-on-3DG constraint. Our experimental results on three real datasets under a comprehensive set of seven client availability modes confirm the robustness of FEDGS on arbitrary client availability modes. In the future, we plan to study how to define and construct 3DG across various ML tasks.

Acknowledgements

The research was supported by Natural Science Foundation of China (62272403, 61872306), Fundamental Research Funds for the Central Universities (20720200031), FuXiaQuan National Independent Innovation Demonstration Zone Collaborative Innovation Platform (No.3502ZCQXT2021003), and Open Fund of PDL (WDZC20215250113).

References

Azam, S. S.; Hosseinalipour, S.; Qiu, Q.; and Brinton, C. 2021. Recycling Model Updates in Federated Learning: Are Gradient Subspaces Low-Rank? In *International Conference on Learning Representations*.

Baek, J.; Jeong, W.; Jin, J.; Yoon, J.; and Hwang, S. J. 2022. Personalized Subgraph Federated Learning. arXiv:2206.10206.

Balakrishnan, R.; Li, T.; Zhou, T.; Himayat, N.; Smith, V.; and Bilmes, J. 2021. Diverse client selection for federated learning via submodular maximization. In *International Conference on Learning Representations*.

Cho, Y. J.; Wang, J.; and Joshi, G. 2020. Client Selection in Federated Learning: Convergence Analysis and Power-of-Choice Selection Strategies. arXiv:2010.01243.

Du, W.; and Zhan, Z. 2002. Building decision tree classifier on private data. *Electrical Engineering and Computer Science*.

Fraboni, Y.; Vidal, R.; Kameni, L.; and Lorenzi, M. 2021. Clustered sampling: Low-variance and improved representativity for clients selection in federated learning. In *International Conference on Machine Learning*, 3407–3416. PMLR.

Gu, X.; Huang, K.; Zhang, J.; and Huang, L. 2021. Fast federated learning in the presence of arbitrary device unavailability. *Advances in Neural Information Processing Systems*, 34: 12052–12064.

He, C.; Balasubramanian, K.; Ceyani, E.; Yang, C.; Xie, H.; Sun, L.; He, L.; Yang, L.; Yu, P. S.; Rong, Y.; Zhao, P.; Huang, J.; Annavaram, M.; and Avestimehr, S. 2021. FedGraphNN: A Federated Learning System and Benchmark for Graph Neural Networks. arXiv:2104.07145.

Hsu, T.-M. H.; Qi, H.; and Brown, M. 2019. Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification. arXiv:1909.06335.

Huang, T.; Lin, W.; Wu, W.; He, L.; Li, K.; and Zomaya, A. Y. 2020. An efficiency-boosting client selection scheme for federated learning with fairness guarantee. *IEEE Transactions on Parallel and Distributed Systems*, 32(7): 1552–1564.

Karimireddy, S. P.; Kale, S.; Mohri, M.; Reddi, S.; Stich, S.; and Suresh, A. T. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, 5132–5143. PMLR.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Li, T.; Sahu, A. K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; and Smith, V. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2: 429–450.

Li, X.; Huang, K.; Yang, W.; Wang, S.; and Zhang, Z. 2019. On the Convergence of FedAvg on Non-IID Data. arXiv:1907.02189.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, 1273–1282. PMLR.

Meng, C.; Rambhatla, S.; and Liu, Y. 2021. Cross-node federated graph neural network for spatio-temporal data modeling. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 1202–1211.

Peng, B. 2007. The determinant: A means to calculate volume. *Recall*, 21: a22.

Pisinger, D. 1999. *Exact solution of p-dispersion problems*. DIKU.

Ribando, J. M. 2006. Measuring solid angles beyond dimension three. *Discrete & Computational Geometry*, 36(3): 479–487.

Ribero, M.; Vikalo, H.; and De Veciana, G. 2022. Federated Learning Under Intermittent Client Availability and Time-Varying Communication Constraints. arXiv:2205.06730.

Shundong, L.; Mengyu, Z.; and Wenting, X. 2021. Secure Scalar Product Protocols. *Chinese Journal of Electronics*, 30(6): 1059–1068.

Wang, I.-C.; Shen, C.-H.; Zhan, J.; Hsu, T.-s.; Liau, C.-J.; and Wang, D.-W. 2009. Toward empirical aspects of secure scalar product. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 39(4): 440–447.

Wang, J.; Liu, Q.; Liang, H.; Joshi, G.; and Poor, H. V. 2020. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33: 7611–7623.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv:1708.07747.

Xu, X.; and Lyu, L. 2020. A Reputation Mechanism Is All You Need: Collaborative Fairness and Adversarial Robustness in Federated Learning. arXiv:2011.10464.

Xu, X.; Lyu, L.; Ma, X.; Miao, C.; Foo, C. S.; and Low, B. K. H. 2021. Gradient driven rewards to guarantee fairness in collaborative machine learning. *Advances in Neural Information Processing Systems*, 34: 16104–16117.

Yan, Y.; Niu, C.; Ding, Y.; Zheng, Z.; Wu, F.; Chen, G.; Tang, S.; and Wu, Z. 2020. Distributed Non-Convex Optimization with Sublinear Speedup under Intermittent Client Availability. arXiv:2002.07399.

Zhang, C.; Zhang, S.; James, J.; and Yu, S. 2021. FASTGNN: A topological information protected federated learning approach for traffic speed forecasting. *IEEE Transactions on Industrial Informatics*, 17(12): 8464–8474.

Zhao, Y.; Li, M.; Lai, L.; Suda, N.; Civin, D.; and Chandra, V. 2018. Federated Learning with Non-IID Data. arXiv:1806.00582.

A. Proof of Theorem 1

Proof. Supposing there are C types of data in the global dataset, the global dataset can be represented by the data ratio vector $\mathbf{p}^* = [p_1^*, p_2^*, ..., p_C^*]^\top$, where each data type's ratio is $p_i^* > 0$ and $\mathbf{1}^\top \mathbf{p}^* = 1$. Then, each client c_k's local data distribution can also be represented by $\mathbf{p}_k = [p_{k1}, p_{k2}, ..., p_{kC}]^\top, \mathbf{1}^\top \mathbf{p}_k = 1$. We slightly modify the way of constructing 3DG as

$$R_{ij} = \begin{cases} 0, & i = j \\ \|e_i - e_j\|_2^2 & i \neq j \end{cases}$$

where $e_i = \frac{p_i}{\|p_i\|}, e_j = \frac{p_j}{\|p_j\|}$ and the principle of the smaller similarity corresponding to the larger distance still holds. Given that 3DG is a complete graph, we demonstrate that the shortest-path distance matrix $\mathbf{H}$ is the same to $\mathbf{R}$, since $distance(i,k) + distance(k,j) \geq distance(i,j)$ is always established for any k.

In each communication round t, the server samples M clients to participate. By denoting the sampled clients' normalized data distribution as $\mathbf{P}_t = [\mathbf{e}_{i_1}, ..., \mathbf{e}_{i_M}] \in \mathbb{R}_+^{C \times M}$, the average shortest-path distance of these selected clients is

$$f(S_t) = \frac{1}{M(M-1)} \sum_{i,j \in S_t} H_{ij} \tag{24}$$

$$= \frac{1}{M(M-1)} \left(\sum_{i,j \in S_t} (2 - 2e_i^\top e_j) \right) \tag{25}$$

$$= \frac{2M}{M-1} - \frac{2}{M(M-1)} \|\mathbf{P}_t^\top \mathbf{P}_t\|_{m1} \tag{26}$$

After receiving the local models, the server aggregates their uploaded models to obtain

$$\theta_{t+1} = \sum_{k \in S_t} w_k \theta_k^{t+1} = \theta_t + \sum_{k \in S_t} w_k \Delta\theta_k^t \tag{27}$$

where w_k is the aggregation weight of the selected client c_k and $\mathbf{1}^\top \mathbf{w} = 1$. Now, the problem is how possible can the aggregated model update recover the true model update $\Delta\theta_t$ that is computed on all the clients?

To answer this problem, we first simply consider the model update computed on all the ith type of data to be $\Delta\bar{\theta}_{ti} \in \mathbb{R}^d$ and $\mathbf{\Delta}_t = [\Delta\bar{\theta}_{t1}, ..., \Delta\bar{\theta}_{tC}] \in \mathbb{R}^{d \times C}$, then we can approximate the true model update by

$$\Delta\theta_t^* = \sum_{i=1}^{C} p_i^* \Delta\bar{\theta}_{ti} \tag{28}$$

$$= \mathbf{\Delta}_t \mathbf{p}* \tag{29}$$

Similarly, each client c_k's model update can also be represented by

$$\Delta\theta_k^t = \sum_{i=1}^{C} p_{ki} \Delta\bar{\theta}_{ti} = \mathbf{\Delta}_t \mathbf{p}_k \tag{30}$$

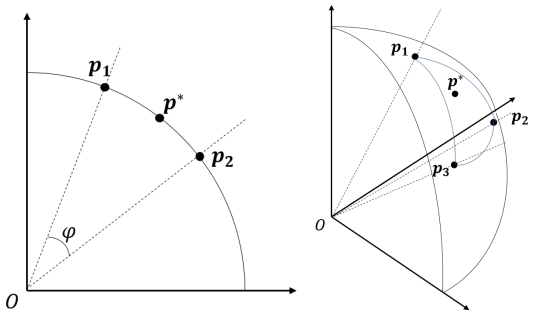


Figure 6: The example of the *angle* of n vectors in n-dimension space, where $n = 2$ for the left and $n = 3$ for the right.

Then, the aggregated model update is

$$\sum_{k \in S_t} w_k \Delta\theta_k^t = \sum_{k \in S_t} \mathbf{\Delta}_t \frac{\mathbf{p}_k}{\|\mathbf{p}_k\|} w_k \|\mathbf{p}_k\| \tag{31}$$

$$= \mathbf{\Delta}_t \mathbf{P}_t (\mathbf{w} \odot [\|\mathbf{p}_1\|, ..., \|\mathbf{p}_M\|]^\top) \tag{32}$$

$$= \mathbf{\Delta}_t \mathbf{P}_t \tilde{\mathbf{w}} \tag{33}$$

When there exists an optimal weight vector $\mathbf{w}^*$ such that $\exists \mu > 0, \mu \mathbf{P}_t \mathbf{w}^* = \mathbf{p}^*$, we demonstrate that the sampled subset can well approximate the true model update, which requires the true weight $\mathbf{p}^*$ to fall into the region of the convex cone $\mu \mathbf{P}_t \mathbf{w}^*$. Given the assumption that p^* is uniformly distributed in the simplex of $\mathbb{R}^C$, the problem becomes how to compare the probability $Pr\left(\mathbf{p}^* \in \{\mu \mathbf{P}_t \mathbf{w}^* | \mathbf{1}^\top \mathbf{w} = 1, \mathbf{w} \geq 0, \mu > 0\}\right)$ for different $\mathbf{P}_t$.

Let's first consider the 2-dimension case (i.e. the left one of Fig.6). Given two vectors p_1 and p_2, the probability can be measured by the ratio of the angle between p_1 and p_2 as $\frac{\phi}{\pi/2}$. For the cases where the dimension is higher than $n = 2$ (i.e. the right one in Fig.6.), (Ribando 2006) defines the normalized measurement of *solid angle*

$$\tilde{V}_\Omega = \frac{\text{vol}_C(\Omega \cap B_C)}{\text{vol}_C(B_C)} = \frac{\text{vol}_{C-1}(\Omega \cap S_{C-1})}{\text{vol}_{C-1} S_{C-1}} \tag{34}$$

where vol_C is the usual volume form in $\mathbb{R}^C$, B_C is the unit ball in $\mathbb{R}^C$ and S_C is the unit C-sphere. Further, they provide a way to compute the solid angle for C unit positive vector

$$\tilde{V}_\Omega = \frac{1}{\pi^{C/2}} \int_{\mathbb{R}^C_{\geq 0}} e^{-\mathbf{u}^\top \mathbf{P}_t^\top \mathbf{P}_t \mathbf{u}} |\det \mathbf{P}_t| d\mathbf{u} \tag{35}$$

which represents the probability of the optimal weight in the convex cone by projecting p^* and each ray (i.e. column) in $\mathbf{P}_t$ onto the surface of a unit sphere centered at O in $\mathbb{R}^C$.

For the case where the number of sampled clients M is smaller than C or $M = C$ but $|\det \mathbf{P}_t| = 0$, we have $\tilde{V}_\Omega = 0$ (i.e. simply repeating one of the vectors in P_t when $M < C$ until the number of vectors reaches C), which means it's nearly impossible to recover the uniformly distributed optimal weight by directly modifying the aggregation weights of the sampled clients. One way to approximate

the global model update in this case is to choose clients with higher data quality in priority (i.e. local data distribution similar to the global one), which needs $\mathbf{p}^*$ and $\mathbf{p}_k$ are already known and is out of the scope of our discussion. Therefore, we limit our discussion to the case where $|\det \mathbf{P}_t| \neq 0$ and $M = C$ (i.e. the solid angle for $M > C$ can be computed by first dissecting $\mathbf{P}_t$ into simplicial cones (Ribando 2006)).

Based on the equation (35), we demonstrate that the large average shortest-path distance will encourage a large $\tilde{V}_\Omega$. We rearrang the equation (35) as

$$\tilde{V}_\Omega = \left(\frac{|\det \mathbf{P}_t|}{\pi^{C/2}}\right)\left(\int_{\mathbb{R}^C_{\geq 0}} e^{-\sum_{i,j\in[C]} u_i u_j \mathbf{e}_i^\top \mathbf{e}_j} d\mathbf{u}\right) \quad (36)$$

Now we respectively show how enlarging the average shortest-path distance (i.e. the equation (24)) leads to the increasing of the two terms on the right hand side of the equation (36).

The Impact on The First Term. To study how the determinant is impacted by the average shortest-path distance of the sampled clients, we compute the volume of the parallelepiped with the columns in $\mathbf{P}_t$ (i.e. Definition 1 and Definition 2) to obtain $|\det \mathbf{P}_t|$ according to Lemma 1 (Peng 2007).

Definition 1. *Let $\mathbf{e}_1, ..., \mathbf{e}_C \in \mathbb{R}^C$. A parallelepiped $P = P(\mathbf{e}_1, ..., e_C)$ is the set*

$$P = \{\sum_{i=1}^{C} t_i \mathbf{e}_i | 0 \leq t_i \leq 1 \textit{ for } i \textit{ from } 1 \textit{ to } C\} \quad (37)$$

Definition 2. *The n-dimensional volume of a parallelepiped is*

$$Vol_k[P(\mathbf{e}_1, ..., \mathbf{e}_k)] = \begin{cases} \|\mathbf{e}_1\|, & k = 1 \\ Vol_{k-1}[P(\mathbf{e}_1, ..., \mathbf{e}_{k-1})]\|\tilde{\mathbf{e}}_k\|, & k > 1 \end{cases}$$

where $\tilde{\mathbf{e}}_k = \mathbf{e}_k + (\mathbf{e}_1, ..., \mathbf{e}_{k-1})\mathbf{a} = \mathbf{e}_k + \mathbf{E}_{k-1}\mathbf{a}_k$ and the unique chosen of $\mathbf{a}_k$ satisfies $\tilde{\mathbf{e}}_k^\top \mathbf{e}_i = 0, \forall i \in [k-1]$.

Lemma 1. *Given a C-dimensional parallelepiped P in C-dimensional space defined by columns in $\mathbf{P}_t$, we have $Vol_C(P) = |\det \mathbf{P}_t|$*

By denoting $P_k = P(\mathbf{e}_1, ..., \mathbf{e}_k)$, the absolute value of the determinant of $\mathbf{P}_t$ is

$$|\det \mathbf{P}_t| = \text{Vol}_C(P_C) = ||\tilde{\mathbf{e}}_C|| \text{Vol}_{C-1}(P_{C-1}) \quad (38)$$

$$= \sqrt{(\mathbf{e}_C + \mathbf{E}_{C-1}\mathbf{a}_C)^\top (\mathbf{e}_C + \mathbf{E}_{C-1}\mathbf{a}_C)} \text{Vol}_{C-1}(P_{C-1}) \quad (39)$$

Since $\tilde{\mathbf{e}}_C^\top \mathbf{e}_i = 0, \forall i \in [C-1]$, we have

$$\mathbf{E}_{C-1}^\top (\mathbf{e}_C + \mathbf{E}_{C-1}\mathbf{a}_C) = 0 \quad (40)$$

$$\Rightarrow \mathbf{a}_C = -(\mathbf{E}_{C-1}^\top \mathbf{E}_{C-1})^{-1} \mathbf{E}_{C-1}^\top \mathbf{e}_C \quad (41)$$

Thus, the inner product of $\tilde{\mathbf{e}}_C$ with itself is

$$\tilde{\mathbf{e}}_C^\top \tilde{\mathbf{e}}_C = \|\mathbf{e}_C - \mathbf{E}_{C-1}(\mathbf{E}_{C-1}^\top \mathbf{E}_{C-1})^{-1} \mathbf{E}_{C-1}^\top \mathbf{e}_C\|_2^2 \quad (42)$$

$$= 1 - [\mathbf{e}_C^\top \mathbf{e}_i]_{i\neq C}^\top (\mathbf{E}_{C-1}^\top \mathbf{E}_{C-1})^{-1} [\mathbf{e}_C^\top \mathbf{e}_i]_{i\neq C} \quad (43)$$

Given $|\det P| \neq 0 \rightarrow rank(\mathbf{E}_{C-1}) = C - 1$, we have that $\mathbf{E}_{C-1}^\top \mathbf{E}_{C-1}$ is a real symmetric matrix and full rank, which indicates that it is orthogonally diagonalizable. Thus, we orthogonally diagonalize $\mathbf{E}_{C-1}^\top \mathbf{E}_{C-1}$ into $Q\Lambda Q^\top$ to obtain its inverse $(\mathbf{E}_{C-1}^\top \mathbf{E}_{C-1})^{-1} = Q\Lambda^{-1}Q^\top$ where $\lambda_i > 0$. And we rewrite the equation (43) with $\mathbf{h}_C = [\mathbf{e}_C^\top \mathbf{e}_1, ..., \mathbf{e}_C^\top \mathbf{e}_{C-1}]$ as:

$$1 - \mathbf{h}_C^\top \tilde{Q}\tilde{Q}^\top \mathbf{h}_C = 1 - \|\tilde{Q}\mathbf{h}_C\|_2^2 \quad (44)$$

$$= 1 - \sum_{i=1}^{C-1} \frac{h_i^2}{\lambda_i^2} \quad (45)$$

$$= 1 - \sum_{i=1}^{C-1} \frac{1}{\lambda_i^2} (\mathbf{e}_C^\top \mathbf{e}_i)^2 \quad (46)$$

From the equation (46), we can see that keeping $\mathbf{e}_C$ less similar with all the other vectors (i.e. $\mathbf{e}_C^\top \mathbf{e}_i, \forall i \neq C$) will increase $|\det \mathbf{P}_t|$. In addition, the analysis doesn't specify the Cth vector in $\mathbf{P}_t$ to be any particular client. Therefore, enlarging $f(S_t)$ will also enlarge the absolute value of the determinant of $\mathbf{P}_t$.

The Impact on The Second Term. According to Cauchy–Schwarz inequality,

$$\int_{\mathbb{R}^C_{\geq 0}} e^{-\sum_{i,j\in[C]} u_i u_j \mathbf{e}_i^\top \mathbf{e}_j} d\mathbf{u} \quad (47)$$

$$\geq \int_{\mathbb{R}^C_{\geq 0}} e^{-\sqrt{\sum_{i,j\in[C]} (u_i u_j)^2} \sqrt{\sum_{i,j\in[C]} (\mathbf{e}_i^\top \mathbf{e}_j)^2}} d\mathbf{u} \quad (48)$$

$$\geq \int_{\mathbb{R}^C_{\geq 0}} e^{-\sqrt{\sum_{i,j\in[C]} (u_i u_j)^2} \sqrt{\sum_{i,j\in[C]} \mathbf{e}_i^\top \mathbf{e}_j}} d\mathbf{u} \quad (49)$$

$$= \int_{\mathbb{R}^C_{\geq 0}} e^{-\sqrt{\sum_{i,j\in[C]} (u_i u_j)^2} \|\mathbf{P}_t^\top \mathbf{P}_t\|_{m1}^{\frac{1}{2}}} d\mathbf{u} \quad (50)$$

which indicates that enlarging $f(S_t)$ in the equation (24) can improve the lower bound of the second term of the solid angle.

Therefore, we conclude that keeping the average shortest-path distance of the sampled subset to be large will increase the chance to find a proper aggregation weight to well approximate the true model update.

□

B. Results on Constructed 3DG

We also run FedGS on the 3DG constructed by using the proposed functional similarity and cosine similarity of model parameters. We vary the same $\alpha \in \{0, 0.5, 1, 2, 5\}$ for FedGS$_{func}$ and FedGS$_{cos}$, and list the optimal results of them with the same settings of Table 2, as is listed in Table 4. For Synthetic dataset, FedGS$_{cos}$ outperforms **FedGS**and**FedGS**$_{func}$ under most of the client availability, which suggests that our definition of the oracle graph

Setting		FedGS	FedGS$_{func}$	FedGS$_{cos}$
Synthetic	**IDL**	0.306	**0.304**	**0.304**
	LN	0.305	0.306	**0.304**
	SLN	**0.318**	0.319	**0.318**
	LDF	0.308	**0.306**	0.307
	MDF	0.310	0.310	**0.309**
CIFAR10	**IDL**	0.968	**0.963**	0.971
	LN	**0.972**	**0.972**	0.973
	SLN	0.996	**0.991**	0.994
	LDF	**0.966**	0.975	0.967
	MDF	0.963	**0.962**	0.965
Fashion	**IDL**	**0.299**	0.305	0.304
	YMF	**0.303**	0.307	0.307
	YC	**0.307**	0.314	0.313

Table 4: The comparison of testing loss of FedGS running on the Oracle/Constructed 3DG.

on Synthetic dataset is not the true oracle one. For CIFAR10, FedGS$_{func}$'s performance is competitive with the results obtained by **FedGS**, and the two methods consistently outperform **FedGS**$_{cos}$ in most cases. For FashionMNIST, FedGS$_{func}$ and FedGS$_{cos}$ suffer more performance reduction than CIFAR10. However, they still outperform MDSample and UniformSample when client availability changes (0.313 v.s. 0.333 in **YC**).

C. Experimental Details

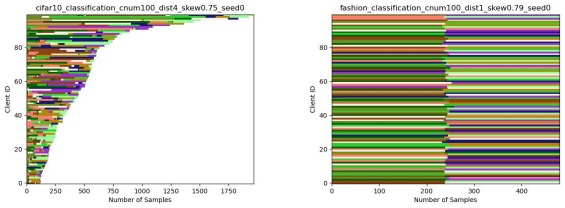
Datasets

Synthetic. We follow the setting in (Li et al. 2020) to generate this dataset by

$$y_{k,i} = \text{argmax}\{softmax(\mathbf{W}_k \mathbf{x}_{k,i} + \mathbf{b}_k)\} \quad (51)$$

where $(\mathbf{x}_{k,i}, y_{k,i})$ is the ith example in the local data D_k of client c_k. For each client c_k, its local optimal model parameter $(\mathbf{W}_k, \mathbf{b}_k)$ is generated by $\mu_k \sim \mathcal{N}(0, \alpha) \in \mathbb{R}, \mathbf{W}_k[i,j] \sim \mathcal{N}(\mu_k, 1), \mathbf{W}_k \in \mathbb{R}^{10\times 60}, \mathbf{b}_k[i] \sim \mathcal{N}(\mu_k, 1), \mathbf{b}_k \in \mathbb{R}^{10}$, and its local data distribution is generated by $B_k \sim \mathcal{N}(0, \beta), \mathbf{v}_k[i] \sim \mathcal{N}(B_k, 1), \mathbf{v}_k \in \mathbb{R}^{60}, \mathbf{x}_{k,i} \sim \mathcal{N}(\mathbf{v}_k, \mathbf{\Sigma}) \in \mathbb{R}^{60}, \mathbf{\Sigma} = \text{diag}(\{i^{-1.2}\}_{i=1}^{60})$. The local data size for each client is $n_k \sim lognormal(4, 2)$. In our experiments, we generate this dataset for 30 clients with $\alpha = \beta = 0.5$.

CIFAR10. The CIFAR10 dataset (Krizhevsky, Hinton et al. 2009) consists of totally 60000 32x32 colour images in 10 classes (i.e. 50000 training images and 10000 test images). We partition this dataset into 100 clients with both data size imbalance and data heterogeneity. To create the data imbalance, we set each client's local data size $n_k \sim lognormal(log(\frac{n}{N}) - 0.5, 1)$ to keep the mean data size is $\bar{n} = \frac{n}{N}$. Then, we generate the local label distribution $\mathbf{p}_k \sim Dirichlet(\alpha \mathbf{p}*)$ for each client, where $\mathbf{p}*$is the label distribution in the original dataset. Particularly, we loop replacing the local label distribution of clients with the new generated one from the same distribution until there exists no conflict with the allocated local data sizes, which allows the coexisting of controllable data imbalance and data heterogeneity. We use $\alpha = 1.75$ in our experiments and provide the visualized partition in Fig.7(a).



(a) CIFAR10 (b) FashionMNIST

Figure 7: The visualization of data partition for CIFAR10 (a) and FashionMNIST (b). Each bar in the figures represents a client's local dataset and each label is assign to one color. The length of each bar reflects the size of the local data.

FashionMNIST. The dataset (Xiao, Rasul, and Vollgraf 2017) consists of a training set of 60,000 examples and a test set of 10,000 examples, where each example is a 28×28 size image of fashion and associated with a label from 10 classes. In this work, we partition this dataset into 100 clients and equally allocate the same number of examples to each one, where each client owns data from two labels according to (McMahan et al. 2017). A direct visualization of the partitioned result is provided in Fig.7(b).

Models. For CIFAR10 and FashionMNIST, we use CNNs that are respectively used for CIFAR10 and MNIST in (McMahan et al. 2017). For Synthetic dataset, we use the same logistical regression model as (Li et al. 2020).

Hyperparameters

For each dataset, we partition each client's local data into training and validating parts. Then, we tune the hyperparameters with FedAvg(McMahan et al. 2017) by grid search on the validation datasets under the ideal client availability. The batch size is $B = 10$ for Synthetic and $B = 32$ for both CIFAR10 and FashionMNIST. Specifically, we fixed the local updating steps instead of epochs to avoid unexpected bias caused by imbalanced data (Wang et al. 2020). We search the number of local update steps in $E \in \{10, 50, 100\}$ for all the datasets and the learning rate $\eta_{Synthetic} \in \{0.01, 0.05, 0.1, 0.3\}, \eta_{CIFAR10,Fashion} \in \{0.003, 0.01, 0.03, 0.1\}$. For CIFAR10, the optimal parameters are $E = 10, \eta = 0.03$ and we train the model for 1000 rounds. For Synthetic, we train the model for 1000 round using $\eta = 0.1, E = 10$. For FashionMNIST, we train the model for 500 rounds with the optimal parameters $E = 10, \eta = 0.1$. We round-wisely decay the learning srate by a factor of 0.998 for all the datasets. To simulate the communication constraint of the server, we fixed the proportion of selected clients to be 0.1 for CIFAR10 and FashionMNIST, 0.2 for Synthetic dataset.

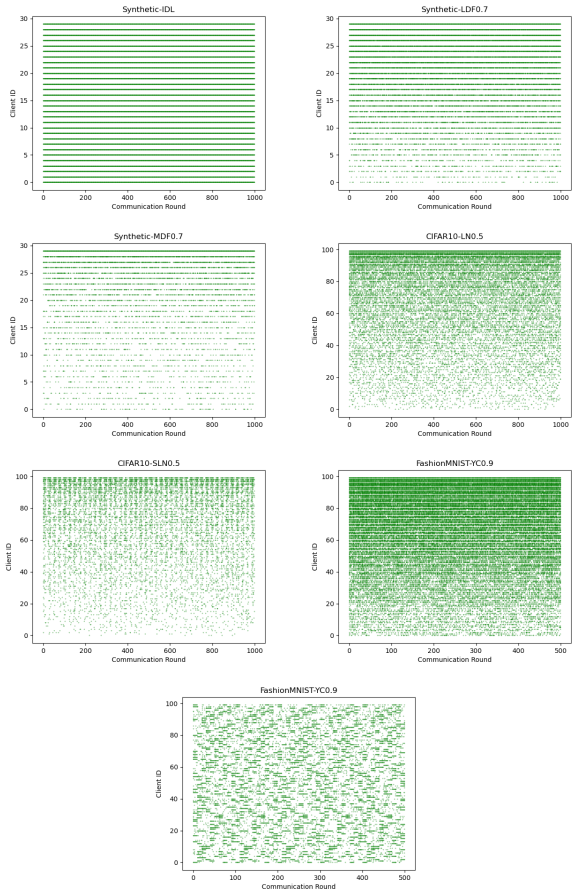


Figure 8: The active states of clients in each communication round under different availability modes.

Client Availability Modes

To obtain the results in Table 2, we conduct experiments under the client availability modes of **IDL**, **LN0.5**, **SLN0.5**, **LDF0.7**, **MDF0.7**, **YMF0.9** and **YC0.9** across different datasets, where the float number at the end of the name of these settings is the coefficient β that controls the degree of the global unavailability of clients. For each client availability mode, we visualize the active states of clients in each communication round, which provides an intuitive way to distinguish the difference between the client availability modes. In Fig.8, we respectively visualize the **IDL**, **LDF0.7**, **MDF0.7** in Synthetic, **LN0.5**, **SLN0.5** in CIFAR10 and **YMF0.9**, **YC0.9** in FashionMNIST. For a fair comparison of different methods, we use an independent random seed (i.e. independent to the other random seeds used to optimizing the model) to control the active states of clients in each communication round, which promise the active states of clients will stay the same when running different methods.

Oracle 3DG For Datasets

For FashionMNIST and CIFAR10, we use the local data distribution vectors as the features to calculate the similarities among clients. For Synthetic, since the difference between clients' local data mainly locates in the optimal model, we directly use the local optimal model parameters $(\mathbf{W}_k, \mathbf{b}_k)$ as the feature vectors. Then, we use the inner dot as the similarity function to compute the similarity matrix $\mathbf{V}$. We further normalize the similarity value V_{ij} to $[0, 1]$ by computing $V'_{ij} = \frac{V_{ij} - \min_{i,j} V_{ij}}{\max_{i,j} V_{ij} - \min_{i,j} V_{ij}}$, and finally use the normalized similarity matrices to construst the oracle 3DGs as mentioned in Sec.3.2.

Algorithm 2: Scalar Product

Input:The client A's feature vector $\mathbf{A} \in \mathbb{R}^d$, and the client B's feature vector $\mathbf{B} \in \mathbb{R}^d$

1: Server generates two random vectors $\mathbf{R}_a \in \mathbb{R}^d$, $\mathbf{R}_b \in \mathbb{R}^d$ and two scalars r_a and r_b such that $r_a + r_b = \mathbf{R}_a \cdot \mathbf{R}_b$, where either r_a or r_b is randomly generated.
2: Server sends $\{\mathbf{R}_a, r_a\}$ to client A and $\{\mathbf{R}_b, r_b\}$ to client B.
3: Client A sends $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{R}_a$ to server, and client B sends $\hat{\mathbf{B}} = \mathbf{B} + \mathbf{R}_b$ to server.
4: Server sends $\hat{\mathbf{A}}$ to client B, and $\hat{\mathbf{B}}$ to client A.
5: Client B generates a random number v_2 and computes $u = \hat{\mathbf{A}} \cdot \mathbf{B} + r_b - v_2$, then sends u and v_2 to server.
6: Server sends u to client A.
7: Client A computes $u - (\mathbf{R_a} \cdot \hat{\mathbf{B}}) + r_a = \mathbf{A} \cdot \mathbf{B} - v_2 = v_1$ and sends v_1 to the server.
8: Server calculates the scalar product $v_1 + v_2 = \mathbf{A} \cdot \mathbf{B}$.

D. Scalar Product Protocal

In our first method to construct the 3DG, we use techniques based on SSPP to compute the dot product of private vectors of parties. We argue that any existing solutions of SSPP can be used in our settings. Here we introduce (Du and Zhan 2002) to our FL scenario. The original protocol works as follows. Alice and Bob have different features on the same individuals and want to calculate the scalar product of their private vectors $\mathbf{A}$ and $\mathbf{B}$, both of size m where m is the sample size of the datasets. They will do this with the help of a commodity server we have named Merlin. The protocol consists of the following steps. First, Merlin generates two random vectors $\mathbf{R}_a$, $\mathbf{R}_b$ of size m and two scalars r_a and r_b such that $r_a + r_b = \mathbf{R}_a \cdot \mathbf{R}_b$, where either r_a or r_b is randomly generated. Merlin than sends $\{\mathbf{R}_a, r_a\}$ to Alice and $\{\mathbf{R}_b, r_b\}$ to Bob. Second, Alice sends $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{R}_a$ to Bob, and Bob sends $\hat{\mathbf{B}} = \mathbf{B} + \mathbf{R}_b$ to Alice. Third, Bob generates a random number v_2 and computes $u = \hat{\mathbf{A}} \cdot \mathbf{B} + r_b - v_2$, then sends the result to Alice. Fourth, Alice computes $u - (\mathbf{R_a} \cdot \hat{\mathbf{B}}) + r_a = \mathbf{A} \cdot \mathbf{B} - v_2 = v_1$ and sends the result to Bob. Finally, Bob then calculates the final result $v_1 + v_2 = \mathbf{A} \cdot \mathbf{B}$. In our FL settings, since there are no communications between clients, we pass intermediate variables between clients through the server. The pseudo codes in Algorithm 2 summarizes the steps of scalar product in our FL settings.