Collaboration in Federated Learning With Differential Privacy: A Stackelberg Game Analysis

Guangjing Huang, *Graduate Student Member, IEEE*, Qiong Wu, *Member, IEEE*, Peng Sun, *Member, IEEE*, Qian Ma, *Member, IEEE*, and Xu Chen

***Abstract*—As a privacy-preserving distributed learning paradigm, federated learning (FL) enables multiple client devices to train a shared model without uploading their local data. To further enhance the privacy protection performance of FL, differential privacy (DP) has been successfully incorporated into FL systems to defend against privacy attacks from adversaries. In FL with DP, how to stimulate efficient client collaboration is vital for the FL server due to the privacy-preserving nature of DP and the heterogeneity of various costs (e.g., computation cost) of the participating clients. However, this kind of collaboration remains largely unexplored in existing works. To fill in this gap, we propose a novel analytical framework based on Stackelberg game to model the collaboration behaviors among clients and the server with reward allocation as incentive in FL with DP. We first conduct rigorous convergence analysis of FL with DP and reveal how clients' multidimensional attributes would affect the convergence performance of FL model. Accordingly, we solve the Stackelberg game and derive the collaboration strategies for both clients and the server. We further devise an approximately optimal algorithm for the server to efficiently conduct the joint optimization of the client set selection, the number of global iterations, and the reward payment for the clients. Numerical evaluations using real-world datasets validate our theoretical analysis and corroborate the superior performance of the proposed solution.**

***Index Terms*—Federated learning, difference privacy, stackelberg game, discrimination rule.**

Manuscript received 18 December 2022; revised 25 September 2023; accepted 9 January 2024. Date of publication 16 January 2024; date of current version 30 January 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFB3102700, in part by the National Science Foundation of China under Grant U20A20159, in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2021B151520008, in part by Guangzhou Basic and Applied Basic Research Program under Grant 2024A04J6367, in part by the National Natural Science Foundation of China under Grant 62372481, in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2023A1515011853, in part by the National Natural Science Foundation of China under Grant 62102337, and in part by Natural Science Foundation of Hunan Province under Grant 2023JJ40174. Recommended for acceptance by D. Grosu. *(Corresponding author: Xu Chen.)*

Guangjing Huang, Qiong Wu, and Xu Chen are with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510006, China (e-mail: huanggj27@mail2.sysu.edu.cn; wuqiong23@mail2.sysu.edu.cn; chenxu35@mail.sysu.edu.cn).

Peng Sun is with the College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China (e-mail: psun@hnu.edu.cn).

Qian Ma is with the School of Intelligent Systems Engineering, Sun Yat-sen University, Shenzhen 518107, China (e-mail: maqian25@mail.sysu.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/TPDS.2024.3354713, provided by the authors.

Digital Object Identifier 10.1109/TPDS.2024.3354713

I. Introduction

With the proliferation of Internet of Things (IoT) and mobile applications, the unprecedented valuable data is generated in end devices and scattered over the networks. Due to privacy concerns from the device clients, uploading clients' raw data to a remote cloud for model trainingis infeasible and unrealistic. In light of this, federated learning (FL) has been embraced as a privacy-preserving framework for enabling clients to train a shared model collaboratively, where each client trains its local model over local dataset and then uploads the model parameters to the server for model aggregation.

Although FL succeeds in protecting clients' data privacy to some extent by keeping their data local, clients' data privacy can still be compromised by powerful adversaries through analyzing the uploaded models (e.g., reconstruction attack [1] and membership inference attack [2]). Moreover, clients participating in federated learning incur large amounts of costs on multiple resource dimensions (e.g., computation, memory). Considering the privacy issue and excessive resource consumption, clients would be reluctant to participate in FL model training without sufficient economic compensation. Hence, the server needs to encourage clients to contribute their data and resources for FL model training while minimizing its cost (e.g., incentive reward for client recruiting, model accuracy loss) in a privacy-preserving manner. Meanwhile, clients may not voluntarily participate in FL training unless provided with sufficient incentive reward and controllable privacy leakage. Therefore, in the presence of clients' privacy-preserving demands and resource consumption, it is essential for the server to design an efficient collaboration stimulation mechanism to characterize the relational behavior of clients, stimulate clients' participation, determine collaboration strategy, and finally enhance training efficiency.

To provide reliable privacy guarantees, differential privacy (DP) techniques are successfully incorporated into FL systems [3], [4]. Particularly, the local counterpart of DP (LDP) is a typical technique to provide privacy protection at client level, which enables clients to perturb their uploaded models by adding noises based on their individual privacy demands [5], [6]. Nevertheless, when incorporating DP into FL, client's noise perturbation policy can be affected by not only its privacy demand (which is reflected by privacy budget in DP) and contributed data batchsize, but also the global iteration number predetermined by the server. The reason is that a large number of global iterations

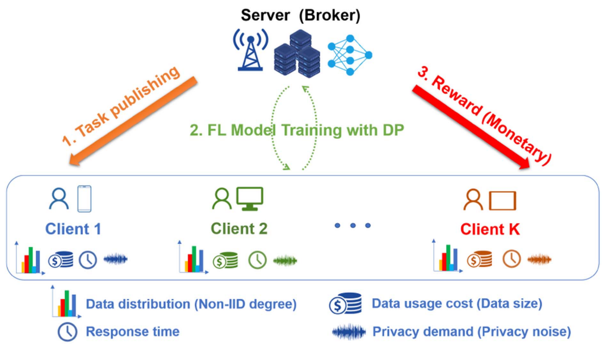

Server (Broker)
1. Task publishing
2. FL Model Training with DP
3. Reward (Monetary)
Client 1
Client 2
Client K
Data distribution (Non-IID degree)
Data usage cost (Data size)
Response time
Privacy demand (Privacy noise)

Fig. 1. FL with DP under clients' multi-dimensional heterogeneous attributes.

in FL would increase the risk of privacy leakage due to the composition property of DP. In response, clients would inject larger-scale noises. Such coupling relationship between clients' noise perturbation policy and server's FL model training setting further complicates the collaboration stimulation mechanism design in FL with DP.

For collaboration stimulation mechanisms in traditional FL settings, existing works predominantly take consideration of only one or two dimensions of clients' attributes (e.g., data usage cost, non-iid degree, response time, etc.), which may not be directly applicable in FL with DP [7], [8], [9], [10]. Conversely, in the context of introducing DP to FL, existing collaboration stimulation mechanisms focus on incentivizing clients to inject less noise into their updated models in a relatively simplified FL setting [11], [12], which largely neglects the heterogeneity of clients' attributes inherent in FL. In realistic FL schemes, the final model performance is influenced by multi-dimensional attributes (e.g., non-IID degrees of data, privacy demands, data usage costs, and response time) of clients, which should be jointly considered in the collaboration stimulation mechanism design. Specifically, to attain a satisfactory FL model, the server aims to recruit high-quality clients with lower incentive payment which is allocated reasonably to the selected clients based on their multi-dimensional attributes. Besides, the number of global iteratons, which is predefined by the server according to the given training time budget and the response time of clients, also has a significant impact on the global model performance. Intuitively, a high-usability model necessitates a large number of global iterations, which increases the risk of privacy leakage. This enlargers clients' noise perturbation and incentive payment of the server in turn.

Motivated by the above discussions, in this article, we aim to analyze the interactions between the server and the clients for collaboration stimulation mechanism design in FL with DP, by taking into account clients' multidimensional attributes (e.g., non-IID degree, privacy budget, computation cost and response time). Specifically, as illustrated in Fig. 1, the server first acts to announce its FL task and payment allocation scheme to solicit the participating clients. Then the participating clients choose their data batchsize for FL training and add noises to their local models for aggregations according to their privacy budgets and the number of global iterations. To fully understand the collaboration interactions among the server and the clients, we need to address the following three key issues:

First of all, to derive an efficient collaboration stimulation strategy for the server, evaluating the converged model performance before FL training is an essential step. Intuitively, the training performance can be jointly influenced by multiple dimensions of attributes such as clients' non-independent and identically distributed (non-IID) data distributions, contributed data volume, and privacy noise from DP. Also, the client's privacy noise scale is also affected by its contributed data batchsize, the number of global iterations, and the privacy demand. Such multi-dimensional and coupling attributes cause significant difficulty in estimating the FL model performance [13].

Second, client selection for collaboration is non-trivial for the server. On the one hand, client's data quality (determined by its non-IID degree and privacy noise) and contributed data size (resulting in data usage cost) will influence model performance [14], [15]. On the other hand, due to the heterogeneity of clients' computation capability and network conditions, clients' diverse response latency can be another major concerning factor [8], [16].

Last but not least, for FL with DP, as the final model performance is affected by multi-dimentional attributes, it is critical to minimize the server's cost in terms of model accuracy loss and incentive payment by properly setting the number of global FL iterations and the payment for collaboration stimulation.

To tackle the key challenges above, we propose a novel Stackelberg game for FL with DP to model the interactions among clients and the server. We solve the Stackelberg game by backward induction, and accordingly derive the equilibrium strategies for both the server and clients. The main results of this article are summarized as follows:

- *Analysis of FL training performance under the DP setting in presence of clients' multi-dimensional attributes:* We derive the expected convergence performance upper-bound of the gap between the model training loss of FL with DP and the optimal training loss value. This performance bound can well capture the joint impacts of clients' multi-dimensional attributes.
- *Stackelberg game analysis of collaboration strategies in FL with DP:* As for the collaboration stimulation for clients, we leverage the classic economic framework of Tullock contest for reward allocation to incentivize clients. Accordingly, we solve the Stackelberg game by backward induction to obtain the collaboration equilibrium strategies for clients. Due to complicated server's objectives, the pursuit of the Stakelberg equilibrium necessitates substantial computational resources, which is unaffordable for server. Consequently, we also devise an approximately optimal algorithm for the server to efficiently conduct the joint optimization of the client set selection, the number of global iterations, and the incentive reward payment for clients. We further reveal the efficiency of the algorithm by deriving its approximation gap from the optimal solution.
- *Extensive performance evaluation:* Extensive numerical results using popular FL datasets and models reveal that

the proposed Stackelberg game based solution is highly-efficient and achieves a gain of up to 75.86% in terms of the server's cost reduction over other schemes.

The rest of this article is organized as follows. In Section II, we introduce FL algorithm with DP and formulate the server's cost minimization problem through the convergence analysis of FL. In Section III, we characterize equilibrium strategies between server and clients, and then devise an approximately optimal algorithm for the server. Numerical experiments are shown in Section IV. The related work is elaborated in Section V followed by conclusion in Section VI.

II. System Model

In this section, we first introduce the FL paradigm with DP. We then capture clients' multi-dimensional attributes and define their utility functions, followed by the characterization of the server's cost. We further formulate the interactions between the server and clients as a Stackelberg game.

A. Federated Learning With Differential Privacy

As illustrated in Fig. 1, we consider a general FL system consisting of one server which wants to recruit a set $\mathcal{K} = \{1, \ldots, K\}$ of clients to participate in FL. Each client holds a dataset denoted by $\mathcal{D}_k = \{(\boldsymbol{x}_k^1, y_k^1), \ldots, (\boldsymbol{x}_k^{N_k}, y_k^{N_k})\}$. Here, $\boldsymbol{x}_k^i$ and y_k^i represent the data features and the label of the ith data instance of client k, respectively. Considering a general classification task, we use the loss function $\ell(\boldsymbol{w}; \boldsymbol{x}_k^i, y_k^i)$ to capture the gap between the prediction label $\hat{y}(\boldsymbol{x}_k^i; \boldsymbol{w})$ and the ground-truth label y_k^i, where $\boldsymbol{w} \in \mathbb{R}^d$ are model parameters. Formally, a client's local loss function is defined as

$$F_k(\boldsymbol{w}) = \frac{1}{|\mathcal{D}_k|} \sum_{(\boldsymbol{x}_k^i, y_k^i) \in \mathcal{D}_k} \ell(\boldsymbol{w}; \boldsymbol{x}_k^i, y_k^i). \tag{1}$$

The global loss function is a weighted average of all local loss functions:

$$F(\boldsymbol{w}) = \sum_{k \in \mathcal{K}} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} F_k(\boldsymbol{w}), \tag{2}$$

where $|\mathcal{D}| = \sum_{k \in \mathcal{K}} |\mathcal{D}_k|$ is the total number of data samples from all clients' local datasets. The goal of FL task is to learn a parameter vector $\boldsymbol{w}$ by minimizing the global loss function, i.e., $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} F(\boldsymbol{w})$.

We consider the widely-adopted synchronous FL paradigm, which involves multiple global iterations between the server and clients. Each global iteration involves the local training process and global model aggregation process. The server and clients repeat multiple global iterations until reaching a predetermined global iteration number T. We next detail these two processes in one global iteration and introduce the DP technique to derive clients' noise perturbation policy.

1) Local Training With Model Perturbation: In a global iteration τ, each client conducts the local model updating based on the global model $\boldsymbol{w}^{\tau-1}$ in the last iteration received from the server. We adopt the mini-batch stochastic gradient descent (SGD) algorithm for one-step of local model update. Specifically, each client randomly selects a subset of the local dataset $\boldsymbol{\xi}_k$ with B_k data samples as one batch to calculate local gradient in one global iteration. We denote the local gradient for client k in global iteration τ as

$$\boldsymbol{g}_k^\tau = \nabla F_k(\boldsymbol{w}^{\tau-1}; \boldsymbol{\xi}_k^\tau) = \frac{1}{B_k} \sum_{(\boldsymbol{x}_k^i, y_k^i) \in \boldsymbol{\xi}_k^\tau} \nabla \ell(\boldsymbol{w}^{\tau-1}; \boldsymbol{\xi}_k^\tau). \tag{3}$$

To provide a standard to add DP noise, we introduce the gradient clipping technique to clip the stochastic gradient. Formally, we have $\tilde{\boldsymbol{g}}_k^\tau = \boldsymbol{g}_k^\tau / \max(1, \frac{\|\boldsymbol{g}_k^\tau\|}{C})$, which ensures $\|\tilde{\boldsymbol{g}}_k^\tau\| \leq C$, and C is the clipping threshold [3]. Then, each client performs one-step update using the clipped gradient as:

$$\boldsymbol{w}_k^\tau(\boldsymbol{\xi}_k^\tau) = \boldsymbol{w}^{\tau-1} - \eta \tilde{\boldsymbol{g}}_k^\tau, \tag{4}$$

where η is the learning rate for local training. After local model updating, each client performs model perturbation based on DP technique by adding noise:

$$\tilde{\boldsymbol{w}}_k^\tau(\boldsymbol{\xi}_k^\tau) = \boldsymbol{w}_k^\tau(\boldsymbol{\xi}_k^\tau) + \boldsymbol{n}_k, \tag{5}$$

where $\boldsymbol{n}_k \in \mathbb{R}^d$ is DP noise and d is the number of model parameters. Then, each client uploads the perturbed local model $\tilde{\boldsymbol{w}}_k^\tau$ to the server.

2) Global Model Aggregation: After receiving all involved clients' perturbed local models, the server aggregates them to obtain a new global model:

$$\boldsymbol{w}^\tau = \sum_{k \in \mathcal{K}} \frac{B_k}{B} \tilde{\boldsymbol{w}}_k^\tau = \sum_{k \in \mathcal{K}} p_k \tilde{\boldsymbol{w}}_k^\tau, \tag{6}$$

where $B = \sum_{k \in \mathcal{K}} B_k$ is the total batchsize of all involved clients in global iteration τ, and $p_k = \frac{B_k}{B}$ is the weight of client k in local model aggregation.

3) Differential Privacy and Noise Perturbation: We next elaborate client's noise perturbation policy $\boldsymbol{n}_k$ based on DP. For clarity, we first provide the concept of DP. DP is a rigorous mathematical technique for preserving privacy [17]. With probability at least $1 - \delta$ $(0 < \delta < 1)$, (ϵ, δ)-DP provides a distinguishable bound ϵ of all outputs for two neighboring input datasets $\mathcal{X}$ and $\mathcal{X}'$ which only differ in a single instance. The formal definition of DP is as follows:

Definition 1 ((ϵ, δ)-DP): A randomized algorithm $\mathcal{M}: \mathcal{D} \to \mathcal{R}$ satisfies (ϵ, δ)-DP if for any two neighboring datasets $\mathcal{X}, \mathcal{X}' \in \mathcal{D}$, $\mathcal{S} \subseteq \mathcal{R}$ such that:

$$Pr[\mathcal{M}(\mathcal{X}) \in \mathcal{S}] \leq e^\epsilon Pr[\mathcal{M}(\mathcal{X}') \in \mathcal{S}] + \delta. \tag{7}$$

We use the privacy budget ϵ to measure privacy demand of clients and adopt the Gaussian mechanism to achieve local DP at clients [3], [4]. To ensure that the Gaussian noises $\boldsymbol{n}_k = \mathcal{N}(0, \sigma_k \boldsymbol{I}_d)$ added to client's local model update can guarantee (ϵ, δ)-DP, according to [17], we need to choose the noise scale $\sigma_k \geq c\Delta_k/\epsilon$ for $c \geq \sqrt{2\ln(1.25/\delta)}$. Here, Δ_k is L_2 sensitivity of $\mathcal{M}$ given by $\Delta_k = \max_{\mathcal{X}, \mathcal{X}'} \|\mathcal{M}(\mathcal{X}) - \mathcal{M}(\mathcal{X}')\|$. For simplicity, we consider that δ is a constant in this article.

According to Definition 1, we next derive the noise perturbation policy. Specifically, given two neighboring datasets $\boldsymbol{\xi}_k^\tau$ and $\boldsymbol{\xi}_k^{\tau'}$ (only differ on data samples $(\boldsymbol{x}_k^i, y_k^i)$ and $(\boldsymbol{x}_k^{i'}, y_k^{i'})$), we can derive the L_2 sensitivity of local model updating at a

Algorithm 1: Synchronous Federated Learning With DP.

```
Input: Number of global iteration $T$
Output: Parameter $\boldsymbol{w}^T$
1: Initialize global model $\boldsymbol{w}^0$
2: Server: Selects a set of clients $\mathcal{N}^* \subseteq \mathcal{K}$.
3: for Global iteration $\tau = 1$ to $T$ do
4:     Server:
5:     Broadcasts the current global parameter to clients $\mathcal{N}^*$.
6:     Each client:
7:     Calculates local gradients: $\boldsymbol{g}_k^\tau = \nabla F_k(\boldsymbol{w}^{\tau-1}; \boldsymbol{\xi}_k^\tau)$.
8:     Clips the gradients: $\boldsymbol{g}_k^\tau \leftarrow \boldsymbol{g}_k^\tau / \max(1, \frac{\|\boldsymbol{g}_k^\tau\|}{C})$.
9:     Updates local model: $\boldsymbol{w}_k^\tau \leftarrow \boldsymbol{w}^{\tau-1} - \eta \boldsymbol{g}_k^\tau$.
10:      Adds noise: $\tilde{\boldsymbol{w}}_k^\tau. \leftarrow \boldsymbol{w}_k^\tau + \boldsymbol{n}_k^\tau$.
11:      Sends the parameter $\tilde{\boldsymbol{w}}_k^\tau$ to the server.
12:      Server:
13:      Aggregates clients' parameters:
         $\boldsymbol{w}^\tau \leftarrow \sum_{k \in \mathcal{N}^*} \frac{B_k}{B} \tilde{\boldsymbol{w}}_k^\tau$.
14: end for
15: return $\boldsymbol{w}^T$.
```

client: $\Delta_k = \max_{\boldsymbol{\xi}_k^\tau, \boldsymbol{\xi}_k^{\tau'}} \|\boldsymbol{w}_k^\tau(\boldsymbol{\xi}_k^\tau) - \boldsymbol{w}_k^\tau(\boldsymbol{\xi}_k^{\tau'})\| \leq \frac{2\eta C}{B_k}$. Accordingly, given a client's privacy budget ϵ_k, we can derive the client's noise perturbation policy as:

Theorem 1: Given the number of global iterations T, to ensure (ϵ_k, δ)-DP for a client k, it needs to add Gaussian noises at each global iteration with noise scale satisfying:

$$\sigma_k = \frac{2\sqrt{2\ln(1.25/\delta)}\eta C T}{B_k \epsilon_k}. \tag{8}$$

The proof is given Section A in the separate supplementary file, available online.

Remark 1: Theorem 1 indicates that a lower privacy budget ϵ_k implies a larger noise perturbation at the client. Theorem 1 also implies that a larger number of global iterations set by the server leads to a higher risk of privacy leakage, due to the growing information exposure by more model aggregations. Thus, clients need to add the noise with a larger scale to globally satisfy the same privacy demand, which may further deteriorate the converged model accuracy. Such effect brings in new challenges for studying the FL with DP.

Finally, the procedure of synchronous FL with DP is summarized in Algorithm 1.

B. Client's Multi-Dimensional Attributes

In this article, we consider a client's multi-dimensional attributes including non-IID degree, privacy budget, data-usage cost and response time. Each client k can be identified by a tuple of $\langle \alpha_k, \nu_k, s_k, t_k \rangle$, where α_k, ν_k are discrimination parameters related to the non-IID degree of local data and the privacy budget. The unit data usage cost and the response time are denoted by s_k and t_k, respectively. In what follows, we sequentially introduce each dimensional of clients' attributes, and then define each client's utility function.

1) Non-IID Degree: We use α_k to represent a client's non-IID degree in data distribution. For example, for each client $k \in \mathcal{K}$ and arbitrary model parameters $\boldsymbol{w}$, we have $\|\nabla F_k(\boldsymbol{w}) - \nabla F(\boldsymbol{w})\| \leq \lambda_k$, where the upper bound λ_k of the difference between local gradient and global gradient can depict the non-IID degree of client k's data distribution using the global data distribution as the benchmark as revealed by the studies in [3], [18][1]. Based on this, we normalize the client's non-IID degree as:

$$\alpha_k = 1 - \left(\frac{\lambda_k}{\lambda_{\max}}\right)^2, \tag{9}$$

where $\lambda_{\max}$ is the maximum non-IID degree tolerated by the server. Here, $\alpha_k \in [0, 1]$ is a discrimination parameter and characterizes client k's non-IID degree. A larger α_k implies that client k possesses a dataset with lower non-IID degree. Note that in (9), we adopt a concave quadratic function as our discrimination rule to characterize the impact of non-IID degree on the client's utility, and we will further discuss other discrimination rules in the section of performance evaluation.

2) Privacy Budget: Based on previous discussions on DP, the privacy budget ϵ_k depicts the privacy demand of client k. Similar to the definition of α_k, we use a concave quadratic function ν_k to characterize its impact on the client's utility, which is defined as:

$$\nu_k = 1 - \left(1 - \frac{\epsilon_k}{\epsilon_{\max}}\right)^2, \tag{10}$$

where $\epsilon_{\max}$ is the maximum privacy budget tolerated by all clients. ν_k is the discrimination parameter corresponding to privacy budget. A larger ν_k implies a higher privacy budget, i.e., the client k will add a smaller noise to its uploaded local model.

3) Data Usage Cost: We denote data usage cost (i.e., local model training cost such as energy consumption) per unit data sample by s_k for client k, which directly influences the client's contributed batchsize in training process. Then, the client's data usage cost in one global iteration can be calculated as $s_k B_k$, where B_k is the chosen batchsize of the mini-batches at client k. As for the impact of the batchsize on SGD algorithms, according to [19], we assume that the stochastic gradient is unbiased and has a bounded variance, i.e., for the mini-batch $\boldsymbol{\xi}_k$, we have $\mathbb{E}[\nabla F_k(\boldsymbol{w}; \boldsymbol{\xi}_k)] = \nabla F_k(\boldsymbol{w})$ and $\|\nabla F_k(\boldsymbol{w}; \boldsymbol{\xi}_k) - \nabla F_k(\boldsymbol{w})\|^2 \leq \frac{\psi^2}{B_k}$, where ψ is the sample variance for clients[2]. Intuitively, a larger batchsize B_k can reduce the error of the trained local gradient. For simplicity, ψ is considered as a constant for all clients in this article. Note that we can also add a client-specific constant to capture the communication cost of a client due to the fixed size of the uploaded model parameters, but we omit it for simplicity since it will not affect the analysis results.

4) Response Time: During one global iteration, a client's response time consists of the computation time of local model training and communication time of model parameter exchange. Due to the fact that many devices in FL have strong parallel

[1]In practice, the server can estimate λ_k for each client with a lightweight training procedure before FL training by recording the maximum value of $\|\nabla F_k(\boldsymbol{w}) - \nabla F(\boldsymbol{w})\|$ from all global iterations [18]. Note that, some errors in estimation procedure can be tolerated in parameter estimation because of perturbed local model from clients.

[2]This assumption $\mathbb{E}\{\nabla F_k(\boldsymbol{w}; \boldsymbol{\xi}_k)\} = \nabla F_k(\boldsymbol{w})$ and $\mathbb{E}\{\|\nabla F_k(\boldsymbol{w}; \boldsymbol{\xi}_k) - \nabla F_k(\boldsymbol{w})\|^2\} \leq \frac{\sigma^2}{B_k}$, is widely-used in studies [4], [20]. σ is the sample variance for client k. Different from the previous works, we use the upper bound of $\|\nabla F_k(\boldsymbol{w}; \boldsymbol{\xi}_k) - \nabla F_k(\boldsymbol{w})\|^2$ in this article.

computing capability (e.g., with on-device GPU and NPU) and the small-scale nature of the mini-batch based SGD, we consider that the computation time is client-specific and does not depend on the batchsize for simplicity. Also, since the size of the model parameters in exchange with the server is fixed, the response time is client-specific, depending on each client's network bandwidth. Thus, in this study we use a client-specific constant t_k to model the response time for simplicity.

C. Client's Utility Function

According to clients' multi-dimensional attributes, we next define each client's utility function. As for the incentive mechanism for rewarding clients' participations, we leverage the classical economic mechanism of Tullock contest [21], which are widely adopted in many scenarios including FL (without DP) [22] and Crowd-sourcing [23], [24], [25]. Specifically, the server publishes a total reward R for all involved clients in one global iteration. Based on the principle of Tullock contest, the benchmarking reward received by a client is proportional to its data contribution and inversely proportional to others' data contribution, i.e., $\frac{B_k}{B}R$. Incorporating discrimination rules, the benchmarking reward should be discounted by the discrimination parameters α_k and ν_k as:

$$r_k = \alpha_k \nu_k \frac{B_k}{B} R. \tag{11}$$

Then, each client's utility in one global iteration is determined by the reward from the server and the incurred training cost:

$$U_k(s_k, \alpha_k, \nu_k) = r_k - B_k s_k. \tag{12}$$

Note that we do not directly characterize the response time in client' utility in (12), but we will capture its impact on the server's cost as discussed below, since it will impact the total number of global iterations, which will further impact the server's decisions in the setting of reward R.

D. Server's Cost

We consider that the server would like to obtain a high-quality FL model in a timely manner, i.e., it wishes to complete the FL within a fixed total training time budget D_{total}. On the one hand, the server needs to incentivize more clients to participate in FL to achieve high-quality training. On the other hand, the server also desires to control the monetary cost in collaboration stimulation. Consequently, the server's cost consists of the payment to the participating clients and the FL model accuracy loss.

In terms of model accuracy loss, it is challenging to predict the final model performance, due to the heterogeneity of non-IID degree, privacy budget and contributed batchsize among participating clients. To address this challenge, we propose to employ the convergence bound of the expected difference between the training loss and the optimal loss value to capture the model performance. Before presenting the convergence analysis results, we first introduce some widely used assumptions on local loss function $F_k(\boldsymbol{w})$ [3].

Assumption 1: The training loss function satisfies the following properties:

- 1) $F_k(\boldsymbol{w})$ is β-Lipschitz, i.e., $\|F_k(\boldsymbol{w}) - F_k(\boldsymbol{w}')\| \le \beta\|\boldsymbol{w} - \boldsymbol{w}'\|$ for any $\boldsymbol{w}$ and $\boldsymbol{w}'$;
- 2) $F_k(\boldsymbol{w})$ is ρ-Lipschitz smooth, i.e., $\|\nabla F_k(\boldsymbol{w}) - \nabla F_k(\boldsymbol{w}')\| \le \rho\|\boldsymbol{w} - \boldsymbol{w}'\|$ for any $\boldsymbol{w}$ and $\boldsymbol{w}'$;
- 3) $F_k(\boldsymbol{w})$ satisfies μ-strong convex. Thus, $F_k(\boldsymbol{w})$ also satisfies Polyak-Lojasiewicz condition with parameter μ, i.e., $F_k(\boldsymbol{w}) - F_k(\boldsymbol{w}^*) \le \frac{1}{2\mu}\|\nabla F_k(\boldsymbol{w})\|$ for any $\boldsymbol{w}$. Here, $\boldsymbol{w}^*$ is the optimal solution.

Due to the limited space, we summarize the main result in Theorem 2 and omit the proof.

Theorem 2: Based on Assumption 1, given the number of global iterations T, $\eta < \frac{1}{\rho}$, the set of participating clients $\mathcal{N}^* \subseteq \mathcal{K}$, and the total batchsize as $B(\mathcal{N}^*) = \sum_{k\in\mathcal{N}^*} B_k$, the expected convergence upper bound is given by:

$$\begin{aligned}&\mathbb{E}\left[F(\boldsymbol{w}^T)\right] - F(\boldsymbol{w}^*) \le \phi^T\Theta \\ &+ (1-\phi^T)\left(\kappa_1 \sum_{k\in\mathcal{N}^*} \frac{T^2}{[B(\mathcal{N}^*)]^2\epsilon_k^2} + \kappa_2 \frac{|\mathcal{N}^*|\psi^2}{B(\mathcal{N}^*)} + \kappa_3\Lambda\right),\end{aligned} \tag{13}$$

where $\Theta = F(\boldsymbol{w}^0) - F(\boldsymbol{w}^*)$, $\phi = 1 - 2\mu\eta + 2\mu\rho\eta^2$, $\kappa_1 = \frac{2\rho\eta dC^2 ln(1.25/\delta)}{\mu(1-\rho\eta)}$, $\kappa_2 = \frac{\rho\mu}{\mu(1-\rho\eta)}$ and $\kappa_3 = \frac{1}{2\mu\eta(1-\rho\eta)}$. Also,

$$\Lambda = \frac{\eta}{2}\sum_{k\in\mathcal{N}^*} p_k\lambda_k^2 + \eta\beta\sum_{k\in\mathcal{N}^*} p_k\lambda_k + \rho\eta^2\left(\sum_{k\in\mathcal{N}^*} p_k\lambda_k\right)^2, \tag{14}$$

where $p_k = \frac{B_k}{B(\mathcal{N}^*)}$, and Λ captures the impact of non-IID degree on convergence bound.

The proof is given in Section B in the separate supplementary file, available online.

Remark 2: By observing (13), the expected convergence upper bound decreases with clients' privacy budget ϵ_k. Given the participating clients $\mathcal{N}^*$, the negative impact from privacy noise can be mitigated by a lower number of global iterations T or a larger total batchsize. Moreover, a larger number of global iterations T can generally improve the performance if clients' privacy noise scales are small enough. Thus, there is an intrinsic trade-off when selecting an appropriate number of global iteration T in FL with DP.

Remark 3: In terms of the impact of non-IID item Λ, given the participating clients $\mathcal{N}^*$, clients with high data quality contribute more data (with a large proportion p_k), which is conducive to the convergence performance.

By observing the term Λ in (14), the dominated term is $\rho\eta^2(\sum_{k\in\mathcal{N}^*} p_k\lambda_k)^2$. Accordingly, for simplicity we further bound the term Λ by $\varsigma\rho\eta^2(\sum_{k\in\mathcal{N}^*} p_k\lambda_k)^2$, where $\varsigma \ge 3$ is a constant. According to Theorem 2, we use the right-hand side of (13) to capture the server's cost in terms of model accuracy loss which is denoted as:

$$\begin{aligned}M_{cost} =& \gamma_1\phi^T + \gamma_2(1-\phi^T)\sum_{k\in\mathcal{N}^*} \frac{T^2}{[B(\mathcal{N}^*)]^2\epsilon_k^2} \\ &+ \gamma_3(1-\phi^T)\frac{|\mathcal{N}^*|}{B(\mathcal{N}^*)} + \gamma_4\left(1-\phi^T\right)\left(\sum_{k\in\mathcal{N}^*} p_k\lambda_k\right)^2.\end{aligned} \tag{15}$$

Note that, in (15), we focus on the dominating terms and simplify the constant terms (given a fixed FL task) as γ_1, γ_2, γ_3 and γ_4. The total payment to participanting clients in the whole training process is $R_{pay} = TR\sum_{k\in\mathcal{N}^*}\alpha_k\nu_k p_k$. Thus, the server's cost is:

$$\mathcal{C}_{server} = \gamma M_{cost} + \gamma_5 R_{pay}, \tag{16}$$

where γ and γ_5 characterize the server's valuation on model accuracy loss and payment, respectively. Here, given the total training time constraint D_{total}, the number of global iterations satisfies $T \in [1, \lfloor \frac{D_{total}}{t^{\max}_{\mathcal{N}^*}} \rfloor]$, where $t^{\max}_{\mathcal{N}^*} = \max_{k\in\mathcal{N}^*}(t_k)$, i.e., the time duration in one global iteration depends on the slowest client.

E. Stackelberg Game Formulation

Prior to commencing with FL, it is imperative for the server to define its collaboration strategy (client selection $\mathcal{N}$, the reward R, and the number of global iterations T). Subsequently, the clients respond to the server's collaboration strategy with the aim of maximizing their individual utilities. The Stackelberg leadership model serves as an apt representation of this inherent characteristic within FL, whereby clients' actions follows with the server's collaboration strategy.

In light of this, we next model the collaborative interactions between clients and the server in FL with DP as a two-stage Stackelberg game, which consists of the following two sequential stages:

Stage I: Given the total training time D_{total}, the server selects the clients $\mathcal{N}$ from the candidate client set $\mathcal{K}$ ($\mathcal{N} \subseteq \mathcal{K}$). We denote the participating client set (with non-zero contributed batchsize) as $\mathcal{N}^* \subseteq \mathcal{N}$. Next, the server chooses the global iteration number $T \in [1, \lfloor \frac{D_{total}}{t^{\max}_{\mathcal{N}^*}} \rfloor]$ and publishes the total payment R with the reward allocation rule in (11) during one global iteration to minimize its cost defined in (16).

Stage II: According to T and R announced by the server, each client $k \in \mathcal{N}$ chooses the batchsize B_k in one global iteration to maximize his utility defined in (12).

Note that as initial trust for collaboration stimulation issue in FL with DP, similar to many existing studies [8], [26], we consider that clients are selfish but honest, such that a client is willing to report its utility parameters to the server. This can enable us focus on tractable analysis for deriving useful insights for practical collaboration mechanism implementation (e.g., using homomorphic encryption for private parameter protection) for FL with DP in the future study. Alternatively, instead of relying on clients' reports, before incentivizing clients to participate in FL, in practice the server also can estimate some parameters (e.g., ρ, μ and λ_k) through a simple pilot training procedure [18] with all candidate clients. Besides, the data usage cost can be estimated based on the client's device type and the response time can efficiently measured at the server side.

III. Equilibrium Strategy for Clients and Approximately Optimal Strategy for Server

In this section, we aim to solve the above two-stage Stackelberg game. In the context of rational client behaviors, it becomes essential for the server to first anticipate the potential strategies that clients might adopt in response to its actions (Stage II). This anticipation facilitates the server in formulating its own strategy, strategically aimed at minimizing its associated costs (Stage I). Consequently, we solve the above two-stage Stackelberg game by the principle of backward induction to characterize the clients' equilibrium strategies and server's collaboration strategy [27]. Specifically, we first derive the Nash equilibrium of the game in Stage II by clients and identify several useful structural properties. Then, based on these properties to tackle the challenging combinatorial problem of minimizing the server's cost in Stage I, we propose an approximately optimal algorithm and further derive its performance gap from the exact optimal strategy.

A. Stage II: Clients' Equilibrium Strategies

In this subsection, we first study clients' competition strategies at equilibrium. In Stage II, each client in $\mathcal{N}$ competes for the reward R published by the server in each global iteration. We model their interactions as a strategic game as follows:

Game 1 (Client Game):

- *Players: The set $\mathcal{N} = \{1, \ldots, N\} \subseteq \mathcal{K}$ of clients selected by the server.*
- *Strategies: The chosen batchsize B_k for each $k \in \mathcal{N}$.*[3]
- *Utilities: The utility function U_k defined in (12) for each $k \in \mathcal{N}$.*

The client game reaches the Nash equilibrium if and only if none of clients can improve its utility through unilaterally changing the strategy [27]. The Nash equilibrium solution for client's game is a strategy profile $(B_1^*, \ldots, B_N^*)$ such that $\forall k \in \mathcal{N}$, the following inequality holds:

$$U_k(B_1^*, \ldots, B_k, \ldots, B_N^*) \leq U_k(B_1^*, \ldots, B_k^*, \ldots, B_N^*), \forall B_k. \tag{17}$$

The Nash equilibrium implies that all clients take the mutually best response strategy simultaneously, i.e.,

$$B_k^* = \arg\max_{B_k} U_k(B_1^*, \ldots, B_k, \ldots, B_N^*). \tag{18}$$

Since U_k in (12) is a concave function with respect to B_k, based on the first-order derivatives of U_k, we can obtain the best response for client k as:

$$B_k^* = \begin{cases} \sqrt{\frac{R\alpha_k\nu_k B_{-k}}{s_k}} - B_{-k}, & \text{if } R > \frac{s_k B_{-k}}{\alpha_k\nu_k}. \\ 0, & \text{if } R \leq \frac{s_k B_{-k}}{\alpha_k\nu_k}. \end{cases} \tag{19}$$

Here, $B_{-k} = \sum_{i\in\mathcal{N}\setminus\{k\}} B_i$. At equilibrium, whether a client participates in FL is determined by the relationship between the published reward R and $B_{-k}\frac{s_k}{\alpha_k\nu_k}$. For simplicity, we define

[3]To obtain a closed-form result at equilibrium, we assume that each client has sufficient data samples in its local dataset.

client's normalized cost as $L_k = \frac{s_k}{\alpha_k \nu_k}$. We denote the participating client set at equilibrium as $\mathcal{N}^* \subseteq \mathcal{N}$, i.e., $(\forall k \in \mathcal{N}^*, B_k > 0)$. For multi-player client game (i.e., when $|\mathcal{N}| \geq 2$), we can further show the following properties:

Theorem 3: Given the announced R and the selected set $\mathcal{N}$, the client's game admits an Nash equilibrium with the following properties:

- *Property 1:* $|\mathcal{N}^*| \geq 2$.
- *Property 2:* The total batchsize is

$$B(\mathcal{N}^*) = \frac{R(|\mathcal{N}^*| - 1)}{\sum_{k \in \mathcal{N}^*} L_k}. \tag{20}$$

The equilibrium strategy of client k is

$$B_k^* = \begin{cases} B(\mathcal{N}^*) \left[1 - \frac{L_k(|\mathcal{N}^*|-1)}{\sum_{k \in \mathcal{N}^*} L_k}\right], & \text{if } k \in \mathcal{N}^*. \\ 0, & \text{if } k \notin \mathcal{N}^*. \end{cases} \tag{21}$$

- *Property 3:* $\forall k \in \mathcal{N}^*$, we have $L_k \leq \max_{k \in \mathcal{N}^*}(L_k) < \frac{\sum_{k \in \mathcal{N}^*} L_k}{|\mathcal{N}^*|-1}$.
- *Property 4:* $\forall k \in \mathcal{N}, L_k \leq \max_{k \in \mathcal{N}^*}(L_k)$, then $k \in \mathcal{N}^*$.
- *Property 5:* Nash equilibrium and $\mathcal{N}^*$ are unique.

The proof is given in Section C in the separate supplementary file, available online. Theorem 3 reveals that if client k participates in FL, the clients with a normalized cost lower than L_k must participate in FL. At Nash equilibrium, the proportion of a client's data contribution is determined by the normalized cost. We rewrite (20) as $B(\mathcal{N}^*) = RY(\mathcal{N}^*)$, where $Y(\mathcal{N}^*) = \frac{(|\mathcal{N}^*|-1)}{\sum_{k \in \mathcal{N}^*} L_k}$ means conversion rate of payment to data (one data sample per unit of money) corresponding to participating client set $\mathcal{N}^*$. For convenience, we refer to $Y(\mathcal{N}^*)$ as conversion rate in the rest of article. Interestingly, given a selected set $\mathcal{N}$, Theorem 3 indicates that there exists an unique corresponding participant set $\mathcal{N}^*$, which is independent of $R > 0$ (but R can impact on the total contributed batch size of the participants). According to Theorem 3, we design the Nash equilibrium computing algorithm, which is summarized in Algorithm 2. The key idea of Algorithm 2 is to identify the participating client set $\mathcal{N}^*$ based on Properties 3 and 4 (Line 3 to Line 9), and then determine each client's equilibrium strategy using Property 2 (Line 10 to Line 15). We can show the convergence of Algorithm 2 as follows:

Theorem 4: The Nash equilibrium computing algorithm in Algorithm 2 achieves the unique Nash equilibrium of the client game with a complexity of $O(|\mathcal{N}|)$.

The proof is given in Subsection D in the separate supplementary file, available online.

B. Stage I: Approximately Optimal Strategy for the Server

According to (16), the server aims to solve the following problem, meanwhile taking into account the Nash equilibrium of the client game:

$$\underset{\mathcal{N},T,R}{MIN}\ \mathcal{C}_{server}, \tag{22}$$

$$\text{s.t.}\quad \mathcal{N} \subseteq \mathcal{K}, \tag{22a}$$

Algorithm 2: Client Game Nash Equilibrium Computing Algorithm.

```
Input: The selected client set $\mathcal{N}$. Total reward $R$.
Output: Strategy profile $(B_1, B_2, \ldots, B_K)$.
 1: Sort the client in ascending order by normalized cost $L_k$,
    i.e., $L_1 < \cdots < L_N$.
 2: $\mathcal{N}^* \leftarrow \{1, 2\}$.
 3: for $i = 3$ to $|\mathcal{N}|$ do
 4:   if $L_i < \frac{L_i + \sum_{k \in \mathcal{N}^*} L_k}{|\mathcal{N}^*|}$ then
 5:     $\mathcal{N}^* \leftarrow \mathcal{N}^* \cup L_i$.
 6:   else
 7:     break.
 8:   end if
 9: end for
10: for each $i \in \mathcal{N}$ do
11:   if $i \in \mathcal{N}^*$ then
12:     $B_i = \frac{R(|\mathcal{N}^*|-1)}{\sum_{k \in \mathcal{N}^*} L_k}[1 - \frac{L_k(|\mathcal{N}^*|-1)}{\sum_{k \in \mathcal{N}^*} L_k}]$.
13:   else
14:     $B_i = 0$.
15:   end if
16: end for
17: return strategy profile $(B_1, B_2, \ldots, B_K)$.
```

$$T \in \left[1, \lfloor \frac{D_{total}}{t_{\mathcal{N}^*}^{\max}} \rfloor \right]. \tag{22b}$$

To achieve the Stackelberg equilibrium, we aim to obtain the optimal solution to minimize the server's cost. (server's best response to the equilibrium of client's game). However, it is challenging to solve problem (22) optimally for the following reasons. First, the client selection problem (i.e., selecting $\mathcal{N}$ from $\mathcal{K}$) couples with the number of global iterations T which depends on the heterogeneity of response time t_k and clients' attributes (non-IID degree, privacy noise, data usage cost). Second, given the selected set $\mathcal{N}$, how to determine (T, R) is still non-trivial due to the complexity of $\mathcal{C}_{server}$ in (16). Hence, the pursuit of the Stackelberg equilibrium necessitates substantial computational resources, which is inadvisable for the server.

To overcome these challenges, we propose an approximately optimal approach to solve problem (22). The key idea of our approach is to propose an efficient standard for client selection. Then for a given selected set, we need to derive the optimal strategy (T, R). In practice, T belongs a limited integer set, which inspires us to first determine R given a fixed T, and then globally select the best T accordingly.

1) Optimal Global Iterations T and Total Reward R Given a Fixed Selected Client Set $\mathcal{N}$: We first derive the optimal strategy (T, R) given selected client set $\mathcal{N}$. Due to the complexity of $\mathcal{C}_{server}$, it is hard to obtain a closed-form solution directly, and we can derive an implicit solution as follows.

Proposition 1: Given a selected client set $\mathcal{N}$ and a fixed T, we have an unique corresponding participating client set $\mathcal{N}^*$ at equilibrium. $\mathcal{C}_{server}$ is convex function with respect to R when $R > 0$. By setting the first-order derivative $\frac{\partial \mathcal{C}_{server}}{\partial R}$ to 0, the optimal R^* is equivalent to finding the root of the following equation:

$$\kappa_4 R^3 + \kappa_5 R + \kappa_6 = 0, \tag{23}$$

where $\kappa_4 = \gamma_5 T \sum_{k\in\mathcal{N}^*} \alpha_k \nu_k [1 - \frac{L_k(|\mathcal{N}^*|-1)}{\sum_{k\in\mathcal{N}^*} L_k}]$, $\kappa_5 = -\gamma\gamma_3(1-\phi^T)\frac{|\mathcal{N}^*|}{Y(\mathcal{N}^*)}$, $\kappa_6 = -\gamma\gamma_2(1-\phi^T)\sum_{k\in\mathcal{N}^*}\frac{2T^2}{[Y(\mathcal{N}^*)]^2\epsilon_k^2}$.

The cubic equations in (23) can be solved by Cardano formula [28]. Next, given $\mathcal{N}$ ($\mathcal{N}^*$), since the global iteration number T is an integer, we can search the optimal global iteration number $T = 1, 2, \ldots, \lfloor \frac{D_{total}}{t^{\max}_{\mathcal{N}^*}} \rfloor$ to minimize the server's cost.

2) Approximately Optimal Client Selection: It is challenging to determine the optimal client selection strategy since the server needs to jointly consider clients' attributes (non-IID degree, privacy noise, data usage cost), response time t_k as well as the properties of Nash equilibrium. In what follows, we propose an approximate client selection scheme.

By observing (16), a well-trained model requires a large global iteration T, while the negative impact from the privacy term will be dominated. Based on our analysis in Remark 2, a large total batchsize can mitigate the negative impact of both errors from data sampling and privacy noises. Thus, the key idea of our proposed method is to obtain the largest total batchsize after clients' competition under the constraint that each chosen client's response time is less than a given $t^{\max}$.

Based on the properties of the Nash equilibrium of the client game, we know that given R, a larger conversion rate $Y(\mathcal{N}^*)$ means a larger total batchsize $B(\mathcal{N}^*)$. Formally, we have the following result:

Theorem 5: Given the selected client sets $\mathcal{N}$ and $\tilde{\mathcal{N}}$ with $\tilde{\mathcal{N}} \subseteq \mathcal{N}$, and the corresponding participating client sets at equilibrium are $\mathcal{N}^*$ and $\tilde{\mathcal{N}}^*$, respectively, we have $Y(\tilde{\mathcal{N}}^*) \leq Y(\mathcal{N}^*)$.

The proof is given in Section E in the separate supplementary file, available online. Theorem 5 implies that expanding the selected client set is always beneficial for increasing total batchsize at equilibrium of the client game if the time constraint holds. This motivates us to enlarge the selected client set $\mathcal{N}$ to achieve a higher $Y(\mathcal{N}^*)$, which may increase the maximum response time instead. In light of these two conflicting objectives (higher conversion rate and lower response time), we leverage the principle of Pareto optimality to solve this issue. Accordingly, we define the concept of Pareto selection as follows:

Definition 2: Given the selected client sets $\mathcal{N}$ and $\tilde{\mathcal{N}}$, the corresponding participating client sets at equilibrium are $\mathcal{N}^*$ and $\tilde{\mathcal{N}}^*$, respectively. $\tilde{\mathcal{N}}^*$ Pareto dominates $\mathcal{N}^*$ if and only if $Y(\tilde{\mathcal{N}}^*) \geq Y(\mathcal{N}^*)$, $t^{\max}_{\tilde{\mathcal{N}}^*} \leq t^{\max}_{\mathcal{N}^*}$ and there is a case where $Y(\tilde{\mathcal{N}}^*) > Y(\mathcal{N}^*)$ or $t^{\max}_{\tilde{\mathcal{N}}^*} < t^{\max}_{\mathcal{N}^*}$ holds.

$\mathcal{N}^*$ is a Pareto selection if and only if there dose not exist $\tilde{\mathcal{N}}^* \subseteq \mathcal{K}$ such that $\tilde{\mathcal{N}}^*$ Pareto dominates $\mathcal{N}^*$. Then, we aim to search for the Pareto set H consisting of all Pareto selections. We summarize the procedure to obtain the approximately optimal strategy to minimize the server's cost as follows:

- *Initialization:* We set $H = \emptyset$ initially. Selecting all candidate clients $\mathcal{K}$ as the selected client set, we calculate the Nash equilibrium and obtain the participating client set $\mathcal{N}_1^*$ and corresponding maximum response time $t_1^{\max} = \max_{k\in\mathcal{N}_1^*}(t_k)$. Based on Theorem 5, $Y(\mathcal{N}_1^*)$ is the largest conversion rate for arbitrary $\mathcal{N} \subseteq \mathcal{K}$. Obviously, $\mathcal{N}_1^*$ is the Pareto selections. Then, we remove the clients with response time larger than $t_1^{\max}$ in $\mathcal{K}$, i.e., $\mathcal{K} \leftarrow \mathcal{K} \setminus \{i | t_i > t_1^{\max}\}$. This is because $\exists i, t_i > t_1^{\max}, i \in \tilde{\mathcal{N}}$, $\mathcal{N}_1^*$ must dominate $\tilde{\mathcal{N}}$. We record the Pareto selection $\mathcal{N}_1^*$ as $\langle \mathcal{N}_1^*, Y(\mathcal{N}_1^*), t_1^{\max} \rangle$, and H is updated as $H \leftarrow H \cup \{\langle \mathcal{N}_1^*, Y(\mathcal{N}_1^*), t_1^{\max} \rangle\}$.
- *Pareto selection searching:* To find the next Pareto selection, we relax the maximum response time. Specifically, we remove the client i with maximum response time in $\mathcal{N}_1^*$, i.e., $\mathcal{K} \leftarrow \mathcal{K} \setminus \{i\}, i = \arg\max_{k\in\mathcal{N}_1^*} t_k$. Based on selected client set $\mathcal{K}$, we calculate the Nash equilibrium and obtain the participating client set $\mathcal{N}_2^*$, $Y(\mathcal{N}_2^*)$ and $t_2^{\max} = \max_{k\in\mathcal{N}_2^*}(t_k)$. Then, we record the Pareto selection $H \leftarrow H \cup \{\langle \mathcal{N}_2^*, Y(\mathcal{N}_2^*), t_2^{\max} \rangle\}$. Repeat the above process until $|\mathcal{K}| < 2$. Based on Theorem 5, we can prove that each selection is Parato selection, since we always achieve the largest Y given a maximum response time $t^{\max}$ (See Fig. 2).
- *Comparison of Pareto selections:* For each Parate selection in the Pareto selection set H (a fixed selected client set $\mathcal{N}$), we can calculate the optimal T and R to minimize the server's cost. We compare the minimum server's cost of each Parate selection to obtain the best Parate selection.

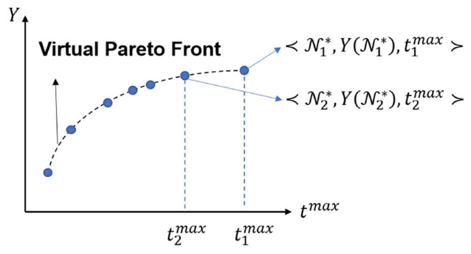


Fig. 2. Pareto selections searched by Algorithm 3.

Algorithm 3 summarizes the approximately optimal approach to minimize the server's cost. The complexity of Algorithm 3 is $O(K^2 \log K + K\frac{D_{total}}{t^{min}})$, where $t^{min} = \min\limits_{\forall k\in\mathcal{K}}(t_k)$. In what follows, we can theoretically derive the approximation performance of the solution by Algorithm 3 using the optimal solution as the benchmark.

Theorem 6: The upper bound of the server's cost by Algorithm 3 is:

$$C^*_{server} < \max\{1, \kappa_7, \kappa_8, \kappa_9\} C^{opt}_{server}, \tag{24}$$

where $\kappa_7 = |\mathcal{K}| \frac{1-\sqrt{1-\max\limits_{\forall k\in\mathcal{K}}(\nu_k)}}{1-\sqrt{1-\min\limits_{\forall k\in\mathcal{K}}(\nu_k)}}$, $\kappa_8 = \frac{1-\min\limits_{\forall k\in\mathcal{K}}(\alpha_k)}{1-\max\limits_{\forall k\in\mathcal{K}}(\alpha_k)}$, $\kappa_9 = |\mathcal{K}| \frac{\max_{k\in\mathcal{K}}(\alpha_k\nu_k)}{\min_{k\in\mathcal{K}}(\alpha_k\nu_k)} \frac{\max\limits_{\forall k\in\mathcal{K}} L_k}{\min\limits_{\forall k\in\mathcal{K}} L_k}$. Here, C^*_{server} is calculated by the proposed algorithm and C^{opt}_{server} is the globally optimal server's cost.

The proof is given in Section F in the separate supplementary file, available online.

Remark 4: Theorem 6 reveals that the cost upper-bound by Algorithm 2 would reduce when the clients' attributes are more

Algorithm 3: Server's Cost Approximate Minimization.

```
Input: Candidate client set $\mathcal{K}$.
Output: $\langle \mathcal{N}_o^*, Y(\mathcal{N}_o^*), t_o^{\max} \rangle$, global iteration $T_o$, total
  reward $R_o$.
1: Calculate the participating clients $\mathcal{N}_1^*$ based on the
   selected client $\mathcal{K}$ using Algorithm 2.
2: $H \leftarrow H \cup \{\langle \mathcal{N}_1^*, Y(\mathcal{N}_1^*), t_1^{\max} \rangle\}$, $t_{tag} \leftarrow t_1^{\max}$, $i \leftarrow 2$.
3: while $|\mathcal{K}| \geq 2$ do
4:   Remove all clients $k$ in $\mathcal{K}$ satisfying $t_k \geq t_{tag}$.
5:   Calculate the participating clients $\mathcal{N}_i^*$ based on the
     selected client $\mathcal{K}$ using Algorithm 2.
6:   $H \leftarrow H \cup \{\langle \mathcal{N}_i^*, Y(\mathcal{N}_i^*), t_i^{\max} \rangle\}$.
7:   $t_{tag} \leftarrow \max_{k \in \mathcal{N}_i^*} t_k$, $i \leftarrow i + 1$.
8:   end while
9:   for each $\langle \mathcal{N}_i^*, Y(\mathcal{N}_i^*), t_i^{\max} \rangle \in H$ do
10:    for $T = 1$ to $T \in [1, \lfloor \frac{D_{total}}{t_i^{\max}} \rfloor]$ do
11:      Solve the cubic (23) and obtain the optimal solution
         $R^*$.
12:      Calculate the corresponding server's cost
         $\mathcal{C}_{server}(T, R^*, \mathcal{N}_i^*)$.
13:    end for
14:    Obtain the optimal $T$ and $R$ for $\langle \mathcal{N}_i^*, Y(\mathcal{N}_i^*), t_i^{\max} \rangle$ by
       comparing $\mathcal{C}_{server}(T, R^*, \mathcal{N}_i^*)$, $\forall T \in [1, \lfloor \frac{D_{total}}{t_i^{\max}} \rfloor]$.
15: end for
16: Search the optimal item based on the optimal server's
    cost of each item, which is denoted as item
    $\langle \mathcal{N}_o^*, Y(\mathcal{N}_o^*), t_o^{\max} \rangle$, the global iteration $T_o$ and the total
    reward $R_o$.
17: return $\langle \mathcal{N}_o^*, Y(\mathcal{N}_o^*), t_o^{\max} \rangle$, $T_o$, $R_o$
```

homogeneous. Also, the cost upper-bound increases linearly with candidate client size $|\mathcal{K}|$, which verifies the efficiency of Algorithm 3 since the optimization space generally grows exponentially with candidate client size.

IV. Discussion on Differential Privacy

For a better presentation, we follow parameter settings from existing studies and adopt a large privacy budgets $\epsilon_k \geq 1$ in our experiments to clearly show the tendency of each evaluation [3], [29]. We need to point out that the Gaussian mechanism adopted in our article is only applicable to a scenario with lower privacy budget (i.e., $\epsilon_k < 1$).

To properly formulate noise for ϵ_k in the low privacy regime ($\epsilon_k \geq 1$), we have further extended our analysis framework to DP with an improved Gaussian mechanisms (See Theorem 4 in [30]). Specifically, to guarantee (ϵ_k, δ)-DP, client k needs to inject the noise into local model with scale $\sigma_k = \frac{\sqrt{2}\eta C}{B_k \sqrt{\epsilon_k}}$ for one gradient query. According to Theorem 1, we can rewrite the client's noise perturbation policy in FL process as: $\sigma_k = \frac{\sqrt{2}\eta CT}{B_k \sqrt{\epsilon_k}}$. Then, the server's cost in terms of model accuracy loss can be reformulated as:

$$\hat{M}_{cost} = \gamma_1 \phi^T + \gamma_6 (1 - \phi^T) \sum_{k \in \mathcal{N}^*} \frac{T^2}{[B(\mathcal{N}^*)]^2 \epsilon_k}$$

$$+ \gamma_3 (1 - \phi^T) \frac{|\mathcal{N}^*|}{B(\mathcal{N}^*)} + \gamma_4 (1 - \phi^T) \left(\sum_{k \in \mathcal{N}^*} p_k \lambda_k \right)^2, \quad (25)$$

where $\gamma_6 = \frac{\rho \eta d C^2}{2\mu(1 - \rho\eta)}$. Hence, the server's cost is formulated as: $\mathcal{C}_{server} = \gamma \hat{M}_{cost} + \gamma_5 R_{pay}$. Due to space limit, we conduct simulations in Section G in the separate supplementary file, available online to validate the effectiveness of our analysis framework in FL with DP based on improving Gaussian mechanism.

V. Numerical Results

In this section, we investigate the impact of discrimination rules on the proposed scheme. We then compare our scheme with other schemes on commonly-used FL learning models and real datasets.

A. Experimental Settings

The device for simulation in this article is equipped with a 8-core Intel(R) Core(TM) i7-8650 U CPU and 1 NVIDIA GeForce GTX 1060 GPU. The device for FL model training is based on Ubuntu 18.04.05, CUDA v11.6 and Intel(R) Xeon(R) CPU (E5-2678 v3). We use the following popular FL datasets and models in literature for performance evaluations.

1) Datasets and Models:

- *MNIST [31]:* The standard MNIST consists of 60,000 training samples and 10,000 test samples for handwritten digit recognition. As for each non-IID degree, each client is randomly assigned 3,000 data samples with 1, 2 and 10 labels (IID case), respectively. A multi-layer perception (MLP) network only with one hidden layer (256 hidden units) is used in our experiment ($d = 203,530$).
- *CIFAR10 [32]:* CIFAR10 dataset possesses 50,000 training samples and 10,000 test samples. As for each non-IID degree, each client is randomly assigned 2500 data samples with 2 labels and 10 labels (IID case), respectively. We use LeNet consisting of two sets of convolution and pooling layers, then two fully-connected layers with ReLU activation ($d = 60,260$).

2) Parameter Estimation: The values of ρ, μ and λ_k are determined by a specific loss function. In practice, we can estimate these values with a simple FL training procedure [18]. Specifically, the server performs a simple FL tasks between all clients and estimates parameters ρ, μ and λ_k based on the uploaded gradients with privacy noise.

- *MNIST settings:* We set $\Theta = 2.3$ and the number of candidate clients $K = 20$. The privacy budget of each client ϵ_k is set to 100 and the clipping threshold C is 20. For parameters estimation, we set learning rate $\eta = 0.01$ and the global iteration T is 50 (one local epoch). We run the simple training in IID case and obtain $\rho = 3.61$ and $\mu = 2.18$. As for non-IID degree, we run the simple training for each non-IID case. The clients' data with 1 label, 2 labels and 10 labels roughly correspond to $\lambda_k \in [2.1542, 2.5242]$, $\lambda_k \in [1.1567, 1.6910]$ and $\lambda_k \in [0.0837, 0.0911]$, respectively.

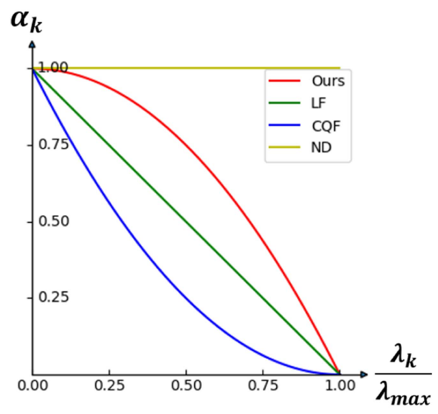


Fig. 3. Discrimination function α_k.

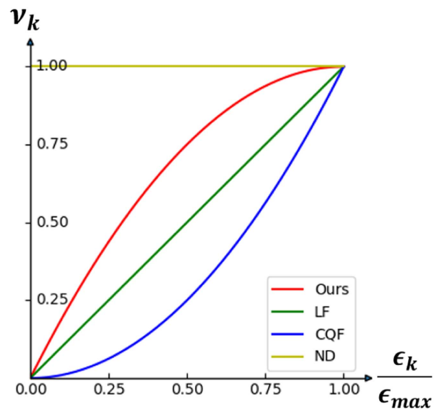


Fig. 4. Discrimination function ν_k.

- *CIFAR10 settings:* The settings of other parameters and training process for parameter estimation are the same as those on MNIST dataset. We have $\rho = 0.4933$ and $\mu = 0.3417$. The clients' data with 2 labels and 10 labels roughly correspond to $\lambda_k \in [0.7797, 0.9712]$ and $\lambda_k \in [0.0130, 0.0174]$, respectively.

B. Discrimination Rule

In this subsection, we study the impact of discrimination rules on our proposed scheme.

Specifically, the discrimination function $\alpha_k(\lambda_k)$ needs to be a monotonically decreasing function respect to λ_k and satisfies $\alpha_k(0) = 1, \alpha_k(\lambda_{\max}) = 0$, since a large λ_k means a large non-iid degree. In contrast, $\nu_k(\epsilon_k)$ is a monotonically increasing function respect to ϵ_k and satisfies $\nu_k(0) = 0, \nu_k(\epsilon_{\max}) = 1$. As shown in Figs. 3 and 4, we consider the following discrimination rules:

- *No discrimination (ND):* $\alpha_k = 1, \nu_k = 1$, i.e., this scheme pays for clients regardless of their non-IID degree and privacy noises;
- *Concave quadratic function (Ours):* α_k and ν_k are defined in (9) and (10), respectively;
- *Linear function (LF):* $\alpha_k = 1 - \frac{\lambda_k}{\lambda_{\max}}, \nu_k = \frac{\epsilon_k}{\epsilon_{\max}}$; *(iv)*

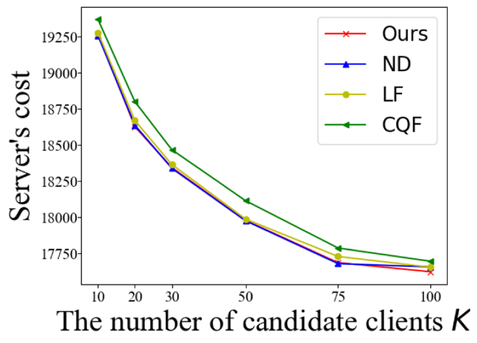


Fig. 5. Average server's cost in high quality group.

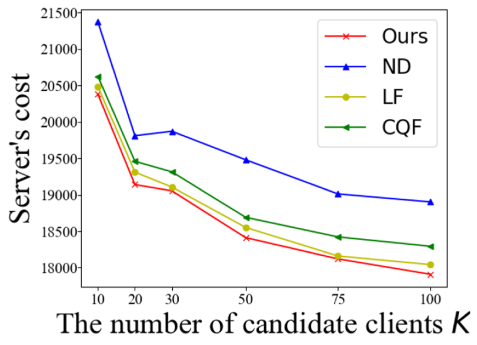


Fig. 6. Average server's cost in uniform quality group.

- *Convex quadratic function (CQF):* $\alpha_k = (1 - \frac{\lambda_k}{\lambda_{\max}})^2, \nu_k = (\frac{\epsilon_k}{\epsilon_{\max}})^2$.

Unless otherwise specified, regarding the server's parameters, we choose $\gamma = 10000, \gamma_5 = 1, \lambda_{\max} = 3, \epsilon_{\max} = 100, D_{total} = 2000\,s$. We set four types of clients with parameters $s_k \in [0.1, 1]$ and $t_k \in [1, 100]$. To mimic different data quality of clients, we set the range of parameters (λ_k, ϵ_k) for *Type-1*, *Type-2*, *Type-3* and *Type-4* clients to ([0,0.75],[75,100]), ([0.75,1.5],[50,75]), ($[1.5, 2.25], [25, 50]$) and ([2.25,3],[0,25]), respectively. We consider three group types: *High quality* group only consisting of only *Type-1* clients; *Low quality* group formed by only *Type-4* clients; *Uniform quality* group including 4 client types, and each type accounting for 25%. We plot the server's cost under different discrimination rules and group types in Figs. 5, 6, and 7. Each result is obtained by averaging 50 independent simulation runs.

In *high quality* group, the effectiveness of the schemes with discrimination rules is not significant compared with ND, since client's data usage cost becomes the server's dominant consideration when all clients have high-quality data. In *uniform* and *low quality* groups, ND has the highest server' cost compared with other schemes. These results show that the discrimination rules are efficient in selecting the high-quality clients by reducing the payment to the clients with lower data quality. In our proposed discrimination rule, the discrimination parameters decrease quickly with the data quality worsening. Compared with LF and CQF, our proposed rule induces clients with lower data quality to lose more competitiveness and achieves

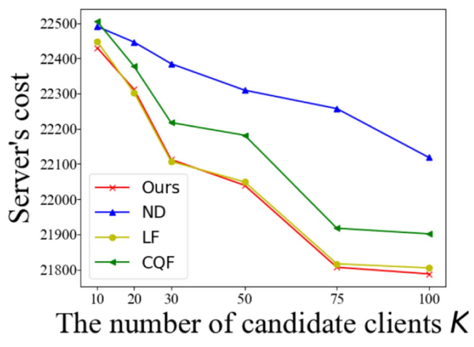


Fig. 7. Average server's cost in low quality group.

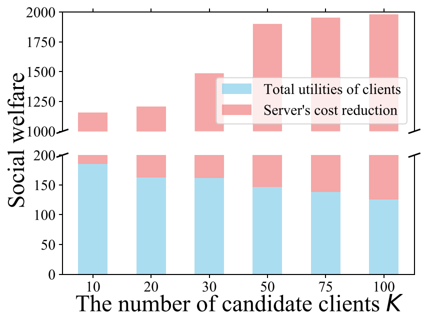


Fig. 10. Average social welfare of the proposed scheme in low quality group.

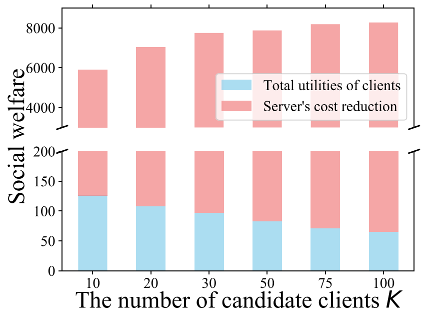


Fig. 8. Average social welfare of the proposed scheme in high quality group.

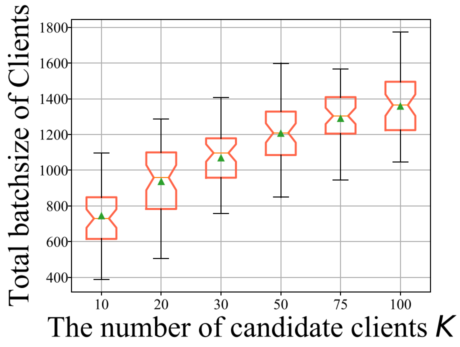


Fig. 11. Total batchsize of clients in high quality group.

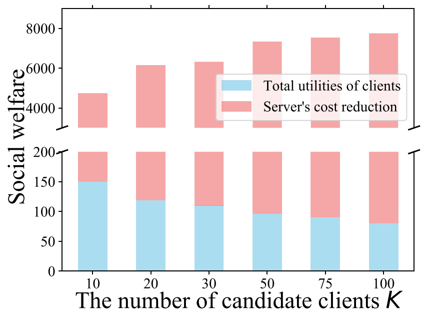


Fig. 9. Average social welfare of the proposed scheme in uniform quality group.

the lowest server's cost. This result indicates that we should enlarge discrimination in the low quality region but decrease discrimination in the high data quality region in discrimination rule design.

C. The Performance of the Proposed Scheme

In this subsection, we study the several performances of proposed scheme in different structures of client groups.

1) The Social Welfare Evaluation: For experimental setting, we consider three group types: *high quality* group, *uniform quality* and *low quality* group (See in Section IV-B). We study the social welfare of the proposed scheme in different type of groups under different number of clients in Figs. 8, 9, and 10. Here, social welfare is defined as the sum of total client utilities and server's cost reduction.

The social welfares increase with the number of clients in all three types of groups. Due to the increase of the number of clients, the intensification of client competition contributes to heightened server efficacy in selecting high-quality clients, thereby leads to a higher social welfare. This indicates that our scheme will be efficient for large-scale FL systems. In contrast, the total utilities of clients decreases with the number of clients. This observation is substantiated by Theorem 5, wherein it is demonstrated that the conversion rates of payment to data exhibit a propensity to increase when the selected client set expands (a larger number of clients). In simpler terms, the server can acquire a greater volume of data with lower payment, resulting in a reduction in the total utilities of clients. Besides, *Low quality* groups exhibit the lowest social welfare compared with *high* and *uniform quality* groups, due to low training efficiency among all low quality clients. Interestingly, when $K = 100$, the social welfare in *uniform quality* group is close to that of high quality. This is because with a large number of clients, only high quality (*Type-1*) clients will choose to participate in FL under the pronounced intensity of competition.

2) The Impact of Number of Candidate Clients on the Strategies of Clients and the Server: We conduct 50 times experiments over three types of groups under different number of clients. For each running result, we adopt boxplot to graphically show the total batchsize contributed by clients in Figs. 11, 12, and 13. Correspondingly, we plot the total payment of the server to each type of group in Fig. 14, where each result is averaged over 50 simulation runs.

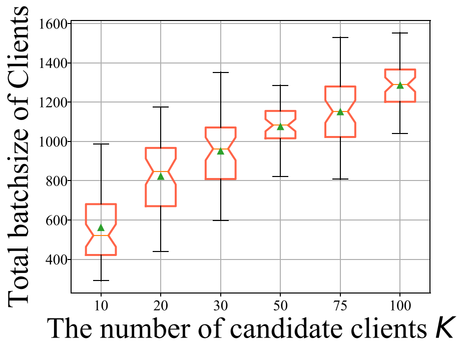


Fig. 12. Total batchsize of clients in uniform quality group.

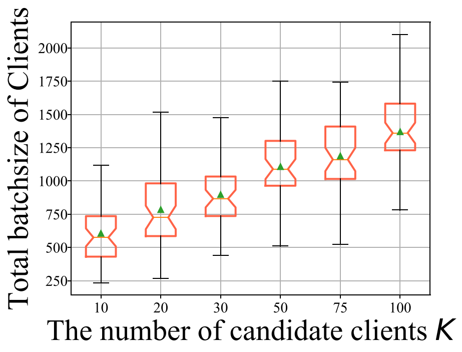


Fig. 13. Total batchsize of clients in low quality group.

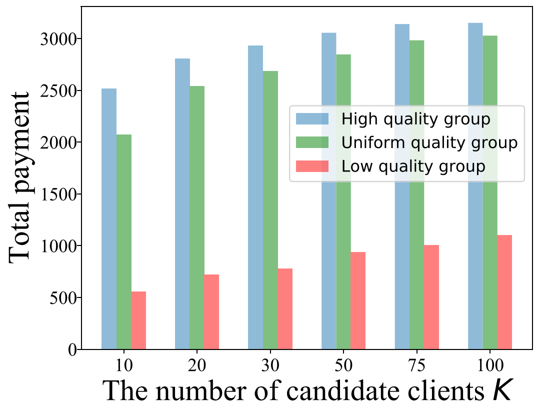


Fig. 14. Average total payment of the server in different type of groups.

As the number of clients increases, a heightened competitive environment among clients emerges, leading to only high-quality clients recruited by server. Further, from Theorem 5, it becomes apparent that the conversion rates of payment to data exhibit the potential for augmentation when expanding the selected client set (confronting a larger client population). Consequently, these implies an augmentation in both client quality and total batchsize for a fixed payment, indicative of enhanced cost-effectiveness in server expenditures. This, in turn, encourages the server to allocate higher payment for improved FL performance. In a parallel fashion, clients are motivated to contribute an increased volume of data to the FL training process.

TABLE I
CLIENT'S PARAMETERS

Type	Parameters' values $(\lambda_k, \epsilon_k, s_k, t_k)$
Type-5	$([\mathbf{2.5}, \mathbf{3}], [95, 100], [0.01, 0.1], [1, 10])$
Type-6	$([0, 0.08], [\mathbf{0}, \mathbf{1}], [0.01, 0.1], [1, 10])$
Type-7	$([0, 0.08], [95, 100], [\mathbf{0.1}, \mathbf{0.2}], [1, 10])$
Type-8	$([0, 0.08], [95, 100], [0.01, 0.1], [\mathbf{10}, \mathbf{20}])$

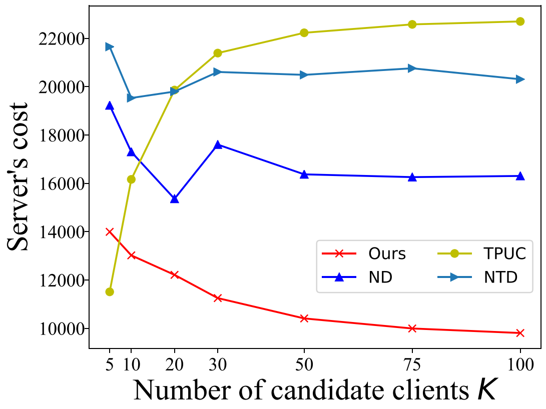


Fig. 15. Average server's cost under each scheme.

D. Server's Cost Comparison

In this subsection, we compare the performance in terms of server's cost of the proposed scheme with other schemes.

We set four client types (*Type-5–8*) as Table I, and each type of clients accounts for 25% of the total number of candidate clients. The other schemes are as follows:

- *No discrimination (ND):* ND set $\alpha_k = 1$ and $\nu_k = 1$ for each client.
- *No discrimination and training time (NDT):* NDT always chooses the candidate client set as the selected set, i.e., $\mathcal{N} = \mathcal{K}$.
- *Two Part Uniform Contract (TPUC) [10]:* TPUC is a contract-theoretic mechanism in FL scenario where clients possess only two dimensional attributes (e.g., data usage cost and response time). TPUC divide clients into multiple types, which fails to characterize the heterogeneity of clients' attributes in the same type. Specifically, TPUC provides two kinds of contract items for candidate clients. One is positive uniform payment for some types of clients and the other is zero for the rest types. For ease of comparison, We extends TPUC to a scenario with four dimensional clients' attributes. Due to continuous parameter space, there are four super-types in our setting. Obviously, TPUC has 2^4 options. We choose the optimal one for the server's cost defined in (22) through an exhaustive search.

To reduce the error from parameters' randomness, we average the results (repeated 50 times) for each type of group. As shown in Fig. 15, each result is obtained by averaging 50 independent simulation runs. Our proposed scheme achieves better performance in terms of server's cost than other schemes in most cases. Particularly, in the group ($K = 100$), the server's cost under our proposed scheme is 39.85%, 51.70%, 56.79%, 75.86% lower

TABLE II
CLIENT'S PARAMETERS

Type	Parameters' values ($\lambda_k, \epsilon_k, s_k, t_k$)
Type-9	$([1.1, 1.7], [95, 100], [0.75, 1] * 10^{-3}, [1, 5])$
Type-10	$([1.1, 1.7], [95, 100], [0.2, 0.6] * 10^{-3}, [1, 5])$
Type-11	$([0, 0.08], [95, 100], [0.2, 0.6] * 10^{-3}, [5, 10])$
Type-12	$([1.1, 1.7], [0, 1], [0.1, 0.2] * 10^{-3}, [60, 90])$
Type-13	$([0.8, 1], [95, 100], [1.25, 1.5] * 10^{-3}, [1, 5])$
Type-14	$([0, 0.02], [0, 1], [0.75, 1.25] * 10^{-3}, [1, 5])$
Type-15	$([0, 0.02], [95, 100], [0.75, 1.25] * 10^{-3}, [5, 10])$
Type-16	$([0.8, 1], [95, 100], [0.075, 0.15] * 10^{-3}, [60, 90])$

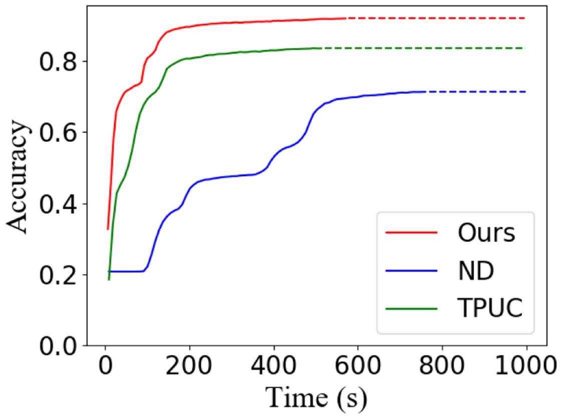


Fig. 16. Accuracy of each scheme on MNIST dataset.

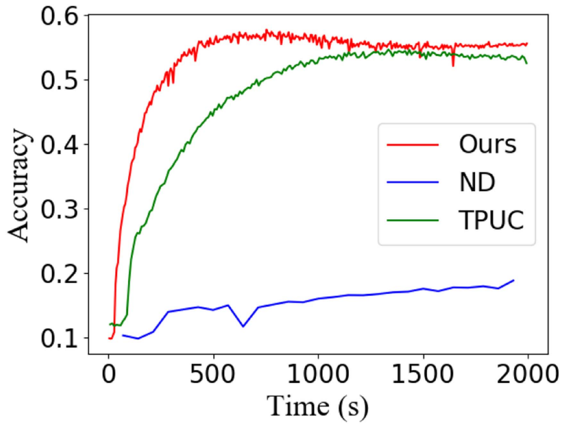


Fig. 17. Accuracy of each scheme on CIFAR10 dataset.

TABLE III
DETAILED RESULTS FOR EACH SCHEME

Scheme	Total batchsize	T	t^{max}	Total payment	Accuracy
Ours **(MNIST)**	14688.23	86	6.64s	976.65	91.83%
ND **(MNIST)**	19182.26	83	9.16s	1076.89	71.24%
TPUC **(MNIST)**	4934.89	72	9.16s	2356.92	83.44%
Ours **(CIFAR10)**	23990.54	341	5.85s	13903.70	55.57%
ND **(CIFAR10)**	11807.89	27	71.49s	69.03	18.83%
TPUC **(CIFAR10)**	17203.27	200	9.96s	19260.18	52.52%

than ND, NTD, TPUC and UC, respectively. With the increase of K, the server's cost decreases in our proposed scheme, which demonstrates that our scheme can be more efficient for large-scale FL systems. ND is superior to NDT because ND considers both training cost and response times of each client, whereas NDT only considers the training cost. UC attempts to provide a uniform contract for all clients including those with low data quality, higher training cost and response time, which leads to higher server's cost. Interestingly, the TPUC achieves the lowest server's cost when $K = 5$. In practice, TPUC can save the training cost by significantly reducing clients' utilities. However, roughly treating a large number of clients as a super-type will incur more server's payment in a large-scale FL system under TPUC scheme.

E. Training Performance Comparison

We next compare the FL training performance of our proposed scheme with other schemes on real-world datasets.

MNIST settings: The server's parameters are as follows: $\lambda_{\max} = 1.7, D_{total} = 1000\,s$. The number of candidate clients is 20. The number of data labels for each client is 2 or 10. We set 4 client types (*Type-9-12*) as shown in Table II.

CIFAR10 settings: The server's parameters are as follows: $\gamma = 10000, \gamma_5 = 0.1, \lambda_{\max} = 1, \epsilon_{\max} = 100, D_{total} = 2000\,s$. Other settings are the same as that on MNIST dataset. We set four client types (*Type-13–16*) as shown in Table II.

The detailed results of three schemes (Ours, ND, TPUC) on MNIST and CIFAR10 datasets are summarized in Figs. 16, 17, and Table III. The dotted parts of the curves in Fig. 16 represent that the server has stopped the training process. Fig. 16 and Table III show that our proposed scheme can achieve the highest accuracy and the lowest total payment. ND selects the clients mainly based on unit data usage cost regardless of data quality (non-IID degree or privacy noise), which degrades the model performance. To mitigate the negative impact of low data quality, the server has to pay more for obtaining a larger data batchsize under ND scheme. The accuracy of the model in TPUC is higher than that in ND, but TPUC incurs higher payment due to roughly treating a group of clients as a super type. On CIFAR10 dataset, our proposed scheme consistently achieves the highest accuracy and a lower payment than TPUC. ND selects clients mainly based on data usage cost and focuses on *Type-16* clients. Due to *Type-16* clients' large response time, ND can not undergo enough global iterations in 2000 seconds ($T = 27$), which leads to the lowest accuracy and payment. In summary, our proposed scheme achieves a better performance on balancing the multi-factor impacts on the FL training performance.

VI. RELATED WORK

Federated learning [33] has receiving substantial interest in researchers since it was proposed in 2017. Most of existing works focus on model performance improvement and resources optimization. Tran et al. formulate a optimization problem considering the trade-off among FL model performance, communication latency and energy consumption in FL [34]. Wang et al. design a control algorithm to balance number of steps of the local updates and the number of global iterations in FL to improve the model performance in a resource-constraint scenario [18].

From theoretical perspective, Li et al. analyze the convergence performance of FL algorithm based on non-IID settings [14].

Federated Learning With DP: The conventional FL algorithm [33] suffers from potential privacy leakage due to probable attacks from the adversaries [1], [2]. To defend against these privacy threats, DP [17] has been widely applied in FL systems to protect clients' data privacy [9], [35], [36]. Since DP noises may deteriorate the model performance, many existing works focus on analyzing the convergence performance of FL with DP, and develop some useful insights of training settings for enhancing training performance [3], [19]. However, most existing works assume that each client is willing to participate in FL.

Incentive Mechanism in FL: Due to various costs incurred in FL process, clients would be reluctant to participate in FL without a well-designed incentive mechanism. A large number of techniques based on contract [10], [37], auction [7], game theory [8] and Tullock contest [36] are applied in incentive mechanism design for FL. Different from auction mechanisms which are perfectly discriminatory, Tullock contest is partially discriminatory so that it guarantees that each participant can obtain a reward if it contributes in crowdsourcing tasks [23], [24]. However, the conventional Tullock contest only focuses on one dimension of attribute [36], which cannot be applied directly in FL systems with multi-dimensional client attributes. Different from the previous works, we extend the Tullock contest with a discrimination rule in responding to challenge caused by multiple attributes in FL and derive a collaboration strategy involving both incentive and training setting design.

VII. Conclusion

In this article, we address the novel issue of collaboration stimulation in FL with DP. We study the joint impacts of multi-dimensional client attributes on model convergence performance, and model the collaboration interactions between the server and clients as a Stackelberg game to derive their equilibrium strategies. We further propose a near-optimal approximate algorithm to minimize the server's cost in terms of model accuracy loss and rewarding cost. For the future work, we will explore how to design a practical DP enhanced FL system based on the derived Stackelberg game analysis results, by leveraging the useful implementation tools such as homomorphic encryption and trusted Execution Environment.

References

[1] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 2512–2520.

[2] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 3–18.

[3] K. Wei et al., "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 3454–3469, 2020.

[4] N. Mohammadi, J. Bai, Q. Fan, Y. Song, Y. Yi, and L. Liu, "Differential privacy meets federated learning under communication constraints," *IEEE Internet Things J.*, vol. 9, no. 22, pp. 22204–22219, Nov. 2022.

[5] M. Seif, R. Tandon, and M. Li, "Wireless federated learning with local differential privacy," in *Proc. IEEE Int. Symp. Inf. Theory*, 2020, pp. 2604–2609.

[6] Y. Zhao et al., "Local differential privacy-based federated learning for Internet of Things," *IEEE Internet Things J.*, vol. 8, no. 11, pp. 8836–8853, Jun. 2021.

[7] T. Huong et al., "An incentive mechanism for federated learning in wireless cellular network: An auction approach," *IEEE Trans. Wireless Commun.*, vol. 20, no. 8, pp. 4874–4887, Aug. 2021.

[8] Y. Sarikaya and O. Ercetin, "Motivating workers in federated learning: A stackelberg game perspective," *IEEE Netw. Lett.*, vol. 2, no. 1, pp. 23–27, Mar. 2020.

[9] L. Zhang, T. Zhu, P. Xiong, W. Zhou, and P. Yu, "A robust game-theoretical federated learning framework with joint differential privacy," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 3333–3346, Apr. 2023.

[10] N. Ding, Z. Fang, and J. Huang, "Optimal contract design for efficient federated learning with multi-dimensional private information," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 186–200, Jan. 2021.

[11] Z. Yi, Y. Jiao, W. Dai, G. Li, H. Wang, and Y. Xu, "A stackelberg incentive mechanism for wireless federated learning with differential privacy," *IEEE Wireless Commun. Lett.*, vol. 11, no. 9, pp. 1805–1809, Sep. 2022.

[12] Y. Xu, M. Xiao, H. Tan, A. Liu, G. Gao, and Z. Yan, "Incentive mechanism for differentially private federated learning in industrial Internet of Things," *IEEE Trans. Ind. Inform.*, vol. 18, no. 10, pp. 6927–6939, Oct. 2022.

[13] Y. Zhan, J. Zhang, Z. Hong, L. Wu, P. Li, and S. Guo, "A survey of incentive mechanism design for federated learning," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 2, pp. 1035–1044, Apr.-Jun. 2022.

[14] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-IID data," 2019, *arXiv: 1907.02189*.

[15] M. Li, T. Zhang, Y. Chen, and A. J. Smola, "Efficient mini-batch training for stochastic optimization," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2014, pp. 661–670.

[16] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, "Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10700–10714, Dec. 2019.

[17] C. Dwork et al., "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, no. 3/4, pp. 211–407, 2014.

[18] S. Wang et al., "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Jun. 2019.

[19] N. Mohammadi, J. Bai, Q. Fan, Y. Song, Y. Yi, and L. Liu, "Differential privacy meets federated learning under communication constraints," *IEEE Internet Things J.*, vol. 9, no. 22, pp. 22204–22219, Nov. 2022.

[20] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao, "Optimal distributed online prediction using mini-batches," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 165–202, 2012.

[21] G. Tullock, "Efficient rent seeking. in jm buchanan, rd tollison, G. tullock," in *Toward a Theory of the Rent Seeking Society*. College Station, TX, USA: A&M Univ. Press, 1980.

[22] Y. Zhan, P. Li, and Z. Qu, D. Zeng, and S. Guo, "A learning-based incentive mechanism for federated learning," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6360–6368, Jul. 2020.

[23] D. Yang, G. Xue, X. Fang, and J. Tang, "Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing," in *Proc. 18th Annu. Int. Conf. Mobile Comput. Netw.*, 2012, pp. 173–184.

[24] T. Luo, S. Salil, H. -P. Kanhere, F. TanWu, and H. Wu, "Crowdsourcing with tullock contests: A new perspective," in *Proc. IEEE Conf. Comput. Commun.*, 2015, pp. 2515–2523.

[25] Y. Mao, X. Chen, X. Li, Y. Xu, and Y. Zhou, "Can early joining participants contribute more? - timeliness sensitive incentivization for crowdsensing," 2017, *arXiv: 1710.01918*.

[26] S. R. Pandey, N. H. Tran, M. Bennis, Y. K. Tun, Z. Han, and C. S. Hong, "Incentivize to build: A crowdsourcing framework for federated learning," in *Proc. IEEE Glob. Commun. Conf.*, 2019, pp. 1–6.

[27] W. James, *N. Game Theory*. Berlin, Germany: Springer, 2006.

[28] G. Cardano, *Ars Magna or the Rules of Algebra*. New York, NY, USA: Dover, 1993.

[29] L. Zhang, T. Zhu, P. Xiong, W. Zhou, and P. S. Yu, "A game-theoretic federated learning framework for data quality improvement," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 11, pp. 10952–10966, Nov. 2023.

[30] B. Balle and Y. -X. Wang, "Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 394–403.

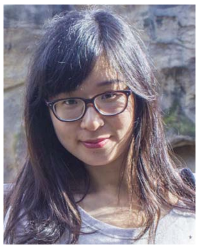
[31] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[32] A. Krizhevsky et al., “Learning multiple layers of features from tiny images,” Univ. Toronto, Tech. Rep., 2009. [Online]. Available: http://www.cs.utoronto.ca/∼kriz/learning-features-2009-TR.pdf
[33] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.
[34] N. H. Tran, W. Bao, A. Zomaya, M. N. H. Nguyen, and C. S. Hong, “Federated learning over wireless networks: Optimization model design and analysis,” in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1387–1395.
[35] M. Zhang, E. Wei, and R. Berry, “Faithful edge federated learning: Scalability and privacy,” *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3790–3804, Dec. 2021.
[36] R. Hu and Y. Gong, “Trading data for learning: Incentive mechanism for on-device federated learning,” in *Proc. IEEE Glob. Commun. Conf.*, 2020, pp. 1–6.
[37] J. Kang, Z. Xiong, D. Niyato, H. Yu, Y. Liang, and D. I. Kim, “Incentive design for efficient federated learning in mobile networks: A contract theory approach,” in *Proc. IEEE VTS Asia Pacific Wireless Commun. Symp.*, 2019, pp. 1–5.

Guangjing Huang (Graduate Student Member, IEEE) received the BS degree from the School of Computer Science, South China Normal University, Guangzhou, China, in 2019, and ME degree from the School of Computer Science and Engineering, Sun Yat-sen University (SYSU), Guangzhou, China, in 2021. He is currently working toward the PhD degree in the School of Computer Science and Engineering, SYSU. His research interests include federated learning, game theory and mechanism design.

Qiong Wu (Member, IEEE) received the BS and ME degrees from the School of Data and Computer Science, Sun Yat-sen University (SYSU), Guangzhou, China, in 2017 and 2019, respectively. She is currently working toward the PhD degree in the School of Data and Computer Science, SYSU. Her primary research interests include social data analysis, mobile edge computing and federated learning.

Peng Sun (Member, IEEE) received the BE degree in automation from Tianjin University, China, in 2015, and the PhD degree in control science and engineering from Zhejiang University, China, in 2020. From 2020 to 2022, he worked as a postdoctoral researcher with the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen. He is currently an associate professor with the College of Computer Science and Electronic Engineering, Hunan University, China. His research interests include Internet of Things, mobile crowdsensing, and federated learning.



Qian Ma (Member, IEEE) received the BS degree from the Beijing University of Posts and Telecommunications (China), in 2012 and the PhD degree in the Department of Information Engineering from the Chinese University of Hong Kong, in 2017. She is an associate professor of School of Intelligent Systems Engineering, Sun Yat-sen University. She worked as a postdoc research associate with Northeastern University during 2018-2019. Her research interests lie in the field of network optimziation and economics. She is the recipient of the Best Article Award from the IEEE International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), in 2021 and the recipient of the Best Student Article Award from the IEEE International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), in 2015.

Xu Chen received the PhD degree in information engineering from the Chinese University of Hong Kong, in 2012. He is a full professor with Sun Yat-sen University, Guangzhou, China, and the vice director of National and Local Joint Engineering Laboratory of Digital Home Interactive Applications, and worked as a postdoctoral research associate at Arizona State University, Tempe, USA, from 2012 to 2014, and a Humboldt scholar fellow at the Institute of Computer Science of the University of Goettingen, Germany from 2014 to 2016. He received the prestigious Humboldt research fellowship awarded by the Alexander von Humboldt Foundation of Germany, 2014 Hong Kong Young Scientist Runner-up Award, 2017 IEEE Communication Society Asia-Pacific Outstanding Young Researcher Award, 2017 IEEE ComSoc Young Professional Best Article Award, Honorable Mention Award of 2010 IEEE international conference on Intelligence and Security Informatics (ISI), Best Article Runner-up Award of 2014 IEEE International Conference on Computer Communications (INFOCOM), and Best Article Award of 2017 IEEE Intranational Conference on Communications (ICC). He is currently an Area Editor of the IEEE Open Journal of the Communications Society, an associate editor of *IEEE Transactions Wireless Communications*, *IEEE Internet of Things Journal* and *IEEE Journal on Selected Areas in Communications* (JSAC) *Series on Network Softwarization and Enablers*.