Incentive-Aware Resource Allocation for Multiple Model Owners in Federated Learning

Feng-Yang Chen and Li-Hsing Yen, *Member, IEEE*

Abstract—A user (model owner) in federated learning builds a learning model by aggregating local learning models trained by independent workers with their private datasets. A fundamental issue of federating learning is allocating resource from workers to the training task. As the allocation causes extra costs and overheads, workers are inherently reluctant to participate. Therefore, it is crucial to design an incentive-based resource allocation mechanism (incentive mechanism) that motivates workers to contribute their resources. Though some incentive mechanisms have been proposed for federating learning, none has devoted to the case when multiple users coexist and compete for worker service whereas a worker can contribute to multiple training tasks at the same time. For this scenario, this paper proposes an auction-based approach, where multiple users as buyers place bids for worker's service. We devise two algorithms attempting to find an auction result that maximizes social welfare, together with a pricing rule that ensures incentive compatibility and individual rationality. Simulation results show that one of the algorithms, which is based on the alternating direction method of multipliers (ADMM), outperforms the other greedy algorithm in terms of social welfare particularly when workers do not have adequate computing resource for all the training tasks.

Index Terms—Resource allocation, Auction theorem, Federated learning.

1 Introduction

Conventionally, machine learning takes a simple architecture where a powerful computing machine, such as a cloud server, constructs a learning model by performing a training task itself with a large dataset. The machine itself is a performance bottleneck and also a single point of failure. To enhance the performance and the reliability, we could adopt an alternative architecture called *distributed learning* [1], [2], which partitions the dataset into several parts, trains a local model for each part by an individual computing device, and then forms a global model by aggregating the locally trained models.

Similar to the conventional architecture, distributed learning implicitly assumes that the one who constructs the learning model (i.e., the *model owner*) also owns the dataset in need. This assumption may not be true for many cases. In that case, the dataset owners may be reluctant to provide their datasets for model training due to privacy or security concern. Concerning this, Google has proposed federated learning (FL) [3] for model training on privacy-sensitive datasets that are not owned by the model owner. FL enables model training under the premise that data owners need not expose the contents of their datasets, thus preserving data privacy.

FL could be vertical or horizontal [4], [5]. In this work, we assume horizontal FL, where the datasets are of the same feature set. We refer to model owners and data owners as users and workers, respectively, hereafter. A user wants to build a learning model (referred to as a global model) from the help of workers. Each worker trains a local learning model for a global model based on its own dataset with the initial global model provided by the user. The user aggregates all local models delivered by workers to update the global model. The updated global model is then disseminated to workers for another epoch of model training. The whole training process may involve several epochs: it stops only when the loss rate becomes acceptable or when the number of epochs reaches a predefined value.

Since the introduction of FL, several issues have been identified with it. First, because different local models are trained on different datasets, aggregating these models into a unified global model may be challenging. To this end, FedAvg [6], FedProx [7], and many other aggregation schemes [8], [9], [10] have been proposed. Second, malicious workers may use problematic local model to poison the global model. Consequently, the global model may be inaccurate. Several detection techniques have been proposed to detect and tackle attacks from malicious workers [11], [12]. Third, since the training process induces time and energy costs, we need to incentivize workers to offer their datasets and computing resources for local model training.

Incentive-based resource allocation mechanism or *incentive mechanism* for FL involves two main tasks. One is to configure the amounts of various types of resource (computing power, dataset, etc.) to be allocated from workers for the user's model learning. This is an optimization problem since the user's valuation on the amount of worker resource generally exhibits the effect of diminishing returns. The other task is to set a pricing rule for resource usage so as to motivate worker to contribute resource for the model training. Incentive mechanisms for FL could be designed based on contract theory [13], [14], [15], [16], Stackelberg game [17], [18], [19], [20], or auction [21], [22]. These approaches all assume a single user (i.e., model owner), so

- *F.-Y. Chen was with the Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan.*
- *L.-H. Yen is with the Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan.*

the incentive mechanisms were designed for workers who need compete with one another for possible payoffs from the user. Unlike these approaches, this study aims to design an incentive mechanism for multiple users. When multiple users coexist in an FL system, they have to compete with one another for worker resource.

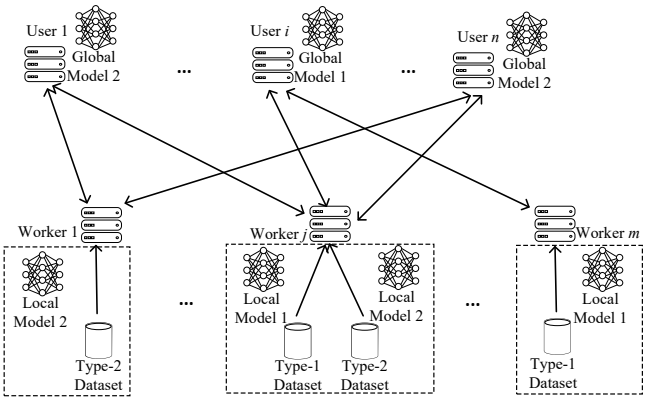


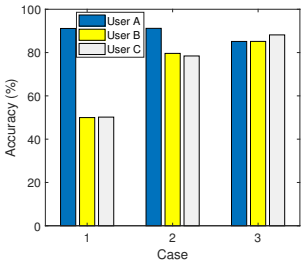
Fig. 1: FL architecture with multiple users sharing worker resource in a non-exclusive manner

In this work, we address the problem of designing an incentive mechanism for multiple users in FL. Different from prior studies, a worker in our work may posses different types of datasets and contribute its resource to multiple training tasks (using the same type of dataset or not) at the same time. Therefore, users share worker resource in a non-exclusive manner (Fig. 1). The sharing is subject to the worker's computation capability: all workers must complete their training tasks within a designated time period for a timely model aggregation. We remark that a non-exclusive sharing of worker training resource (even with training time constraint), when compared with otherwise exclusive sharing setting [15], [23], [24], [25], provides a potentially better results in terms of accuracy. Fig. 2 shows an experimental result using the MNIST dataset [26] with three different cases of worker resource sharing from three workers (X, Y, and Z) to three users (A, B, and C). When users exclusively shared worker training resource (Case 1), B and C had poor performance due to label distribution skew. When B and C equally shared the training resource of Y and Z (Case 2), their performance significantly improved. When all users equally share worker resource (Case 3), user performance as a whole further improved slightly despite a degraded performance of user A.

The optimal resource allocation (from multiple workers to multiple users) is modeled as a concave integer programming problem which seeks to maximize the social welfare of all participants. Social welfare maximization is a design goal common to auction-based approaches (e.g., [21], [22]). We tackle this problem by an auction-based approach, where a data transaction platform (DTP) acting as an auctioneer solves the optimization problem using either a greedy algorithm or an approximation algorithm based on the alternating direction method of multipliers (ADMM) [27], [28]. The algorithms are complemented by a pricing rule which charges each user the loss of social welfare due to her/his participation. We prove that this pricing rule

		Case 1			Case 2			Case 3		
		Workers			Workers			Workers		
		X	Y	Z	X	Y	Z	X	Y	Z
Users	A	1	0	0	1	0	0	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
	B	0	1	0	0	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
	C	0	0	1	0	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

(a) Ratio of resource sharing



(b) Accuracies

Fig. 2: Three different worker-resource-sharing cases and corresponding tested accuracies. Worker X had an image dataset consisting of all 10 handwritten digits, while Workers Y's and Z's datasets contained only images of the smaller and larger five digits ($[0, 4]$ and $[5, 9]$), respectively. Worker datasets were of the same size (600 images) and data labels were uniformly-distributed in each data set. Accuracies were obtained after 100-epoch training using LeNet. FedAvg was used with 10 iterations and batch size set to 60.

provides two desirable properties: incentive compatibility and individual rationality. The former demands that no user can be better off by not claiming its true valuation on the result while the latter means that no user can be worse off by participating in the auction. For performance evaluation, we set up simulations to observe the impacts of various factors on social welfare in the proposed algorithms. The factors include the number of users, the number of workers, dataset size, and the maximum training time. The results of the simulation show that, when the worker resources or the number of workers are insufficient, the ADMM-based algorithm can achieve a higher social welfare than the greedy algorithm.

The main contributions of this work are twofold:

- To the best of our knowledge, this is the first study on the incentive mechanism for FL that allows data owners to contribute their resource to multiple training tasks at the same time. This work will help designing a market which motivates multiple model owners and heterogeneous data owners to participate for possible service trading.
- We propose an auction-based approach for configuring the amount of resource allocated from each data owner to each learning task and the associated price. For the resource allocation, we propose a greedy and an ADMM-based algorithms as approximation to the maximum social welfare. For pricing, we propose a pricing rule which provides incentive compatibility and individual rationality.

The rest of this paper is organized as follows. We review existing resource allocation mechanisms in Section 2. Section 3 describes the system model and formulates the resource allocation problem in FL. Section 4 elaborates the designs of the greedy algorithm, the ADMM-based algorithm and the pricing rule. The settings and results of the simulation are presented in Section 5. Section 6 concludes this paper and presents future work.

2 Related Work

Resource allocation problem has arisen in some other related fields, for which many auction-based schemes have

been proposed. For example, allocating sensing devices to users in crowdsourcing is a resource allocation problem which involves the consideration of sensing device energy cost, sensing task computing time, transmission time, and transmission cost. Li et al. [29] used a randomized auction to select workers for user's sensing tasks. In edge computing, an edge server provides various types of computing resource to users to perform tasks such as model training and Bitcoin mining. To maximize the utility of edge server, Li et al. [30] used a double auction mechanism to resolve the resource allocation problem, which ensures that both edge users and edge server are incentive compatible. Zhong et al. [31] considered a scenario in which customers submit their requests for energy resource such as electricity, heating energy, and cooling energy to an energy hub, which then decides an optimized energy scheduling for customers. The authors designed an auction mechanism, which ensures incentive compatibility, and proposed using ADMM to find the auction result.

Resource allocation problem in FL could be treated separately without considering participant's incentive. Dinh et al. [32] aimed to find an optimal resource allocation for FL that minimize overall time and energy consumption. They proposed decomposing the problem into multiple convex sub-problems and solving the sub-problems by Lagrange multiplier. While the result might be optimal, this approach does not induce workers to participate.

In the literature, only a few auction-based resource allocation schemes have been proposed for FL. Cong et al. [33] aimed to maximize social welfare which is defined as the quality of user model minus worker cost. They used Vickrey-Clarke-Groves (VCG) payment [34], [35], [36] to make this approach incentive compatible. Kim [37] also proposed an incentive mechanism for FL, which takes the same VCG payment rule with a unique feature of preserving worker's data privacy. Le et al. [22] viewed workers as bidders who bid for user's model learning tasks. A bid submitted by a bidder consists of the resource usages, the local accuracy, and the cost corresponding to the model training. When determining the set of winning bids, the auctioneer aims to maximize the social welfare while ensuring that each worker is allocated to at most one learning task. Le et al. proposed a primal-dual based greedy algorithm to approximate the optimal solution. They also proposed a critical-valued-based pricing rule that ensures incentive compatibility, individual rationality, and computation efficiency. Jiao et al. [21] considered a similar auction-based platform for FL, where each worker places bids to get reward by model training. The bid includes the dataset size, values of accuracy-related parameters, requested communication resource, and the associated cost. The goal of the auction is also to select a set of workers that maximizes the social welfare. The authors first devised an approximation algorithm to the optimization problem, and then proposed a deep reinforcement learning based approach that further improves the social welfare and the efficiency. Both approaches in [21], [22] are proved to be incentively compatible and individually rational.

All resource allocation schemes mentioned above considered the matching between a number of resource suppliers (i.e., workers) and a single resource requester (i.e., user). Recently, some approaches have been proposed considering the existence of multiple users in FL. Lim et al. [15] considered a scenario where users recruit their respective workers using a contract-theoretic approach and multiple users collaboratively form several disjoint coalitions to maximize their payoffs. Deng et al. [23] considered allocating workers with various dataset sizes/qualities to multiple users with various budgets via a reverse auction. Their work aims to maximize the sum of estimated learning qualities, where the estimations are based on historical records. For this the auctions should be done periodically for record collections. By contrast, our approach needs no estimation on datasets and is therefore one-shot. Lim et al. [24] proposed a two-level approach where cluster heads are introduced between users and workers to help intermediate model aggregation and relaying. Two main challenges in their work are for workers to select their respective cluster heads and to assign cluster heads to users. The competition for the service provided by cluster heads among users is modeled as a multi-round single-item auction. Both the allocation result and the associated payment are determined by a deep learning approach which ensures seller revenue maximization while achieving individual rationality and incentive compatibility. Because each worker and cluster head can only participate in the training process with a single user, this approach effectively partitions all workers among a set of users. Ng et al. [25] also proposed a two-level approach which separates data owners from FL workers. FL workers are similar to cluster heads [24] in that they do not own their own datasets. The modeling and the resolution of the competition among users for the service provided FL workers are also similar to those presented in [24]. In short, all these incentive mechanisms proposed for multiple FL users [15], [23], [24], [25] constrain that a worker can participate in only one learning task, which implies that the resource allocation result effectively partitions all workers among users.

3 System Model and Problem Formulation

3.1 System Model

We assume n users $U = \{1, 2, \cdots, n\}$ and m workers $W = \{1, 2, \cdots, m\}$. We also assume l data types $\Phi = \{1, 2, \cdots, l\}$. Each worker may own several types of data while each user $i \in U$ requests a certain type of data $\phi^i \in \Phi$ for its model training. We assume that each user i demands η^i training epochs and τ^i training iterations per epoch from each worker for its model training. On the other hand, the batch size in one training iteration for the model training is individually set up for each worker (subject to constraints specific to the worker). Note that aggregation schemes can handle the variability of batch size with weighted averaging, i.e., assigning each worker a weight based on the size of the batch used for the worker's local training. Let q_j^i denote the batch size of worker j set for the model training of user i (q_j^i is preset to 0 if worker j cannot work for user i for privacy or other concern). Then, $q_j^i \tau^i$ is the data size of worker j allocated for user i's model training and $\sigma^i = \sum_{j \in W} q_j^i \tau^i$ is the size of the total data allocated by all workers for user i's model training.

The quality of a learning model can be evaluated in terms of loss rate or accuracy. Raudys and Jain [38] argued that small size of data can easily contaminate a learning

model, which implies that the larger the dataset is, the better quality (lower loss rate or higher accuracy) the trained model will be. However, the benefit from large training data also exhibits the effect of diminishing returns: after a user gets adequate data for model training, more data does not significantly improve the quality of the training model. Zhan et al. [17] reported that the test accuracy of training model can be regarded as a logarithm function with respect to the amount of training data. Therefore, we estimate the model quality of user i as

$$\textit{model quality of user } i = \ln\left(1+\gamma^i\sigma^i\right), \tag{1}$$

where $\gamma^i > 0$ is a model-specific parameter that represents the sensitivity of model quality to the size of the training data. Note that model quality depends on not only quantity but also quality of the training data. For example, Zhao et al. [39] reported that FedAvg does not exhibit the behavior indicated by (1) with highly skewed non-IID data. With contaminated data, the training accuracy neither improves with more training data. However, we do not assume prior knowledge of the statistical characteristics of worker's datasets. With this assumption, we use (1) to capture the relationship between model quality and dataset size.

Users also need to estimate the worth of their models, referred to as their *valuations* on the models, to budget their payments to workers. The valuation of a model certainly relates to the model quality, yet how closely these two metrics are related varies from user to user. For example, some user may be willing to pay more to workers for a high-quality model while some other user may only need an acceptable model or has a limited budget. To capture the variety, we introduce α^i to denote the degree of closeness between model quality and user i's valuation on it. Formally, user i's valuation on σ^i is a function

$$\begin{aligned} v^i(\sigma^i) &= \alpha^i \ln\left(1+\gamma^i\sigma^i\right) \\ &= \alpha^i \ln\left(1+\gamma^i \sum_{j\in W} q_j^i \tau^i\right). \end{aligned} \tag{2}$$

A large α^i value implies that user i's valuation on a model closely relates the quality of the model. In that case, user i is willing to pay more for high-quality model.

On the other hand, a worker has a limited supply of data for model training and no worker can allocate more data than its supply capacity to any user. We use $\sigma_{j,k}$ to denote the size of type k dataset owned by each worker j. Workers also take additional time and bear extra cost for user's training tasks, which calls for compensations from users. We argue that the cost and the time of delivering a learning model (more specifically, delivering parameters of a learning model) are considerably less than the cost and the time of model training, respectively. Therefore, we focus on the latter part. We assume that both the cost and the time of worker j for user i's model training depend on the product of how many data of j are used and how many times these data are used, which is amounted to $q_j^i\tau^i\eta^i$. Therefore, the total cost for worker j to perform model training for user i is

$$c_j^i = \xi_j^i q_j^i \tau^i \eta^i, \tag{3}$$

where ξ_j^i is worker j's training cost on a unit of data for the model training of user i. This formulation is consistent with the experimental result on energy consumption reported in [40]. The associated time is

$$t_j^i = \rho_j^i q_j^i \tau^i \eta^i, \tag{4}$$

where ρ_j^i is the time that worker j takes on a unit of data for the model training of user i. It is well known that model training in FL can be slowed down considerably by *stragglers*, i.e., workers that cannot complete their training tasks in time. Stragglers can occur without warning at any work for many unpredictable reasons, but we pay attention to workers with low computation capability and thus long response time. One way to mitigate the straggler effect is to set up a permissible upper limit for the model training time of each worker [41], [42]. Therefore, we require that the training time of a worker does not exceed an upper bound $T_{\max}$.

Table 1 summarises notation in this paper.

TABLE 1: Notation

U	Set of users
W	Set of workers
Φ	Set of data types
$T_{\max}$	Maximum training time
	User
ϕ^i	Data type needed for user i's model training
τ^i	The number of training iterations demanded by user i
η^i	The epoch count for user i's model training
p_j^i	User i's payment to worker j
$v^i(\cdot)$	Valuation function of user i
α^i, γ^i	Two user-specific parameters in $v^i(\cdot)$
	Worker
$\sigma_{j,k}$	Size of type k dataset owned by worker j.
ξ_j^i	Unit cost of worker j for user i's model training
c_j^i	Total cost of worker j for user i's model training
ρ_j^i	Unit time of worker j for user i's model training
t_j^i	Total time of worker j spent in user i's model training
q_j^i	Batch size of worker j for user i's model training

Besides users and workers, there is a data transaction platform (DTP) which acts as an auctioneer for the proposed auction-based resource allocation approach. Each user as a buyer $i \in U$ submits its request $\langle \phi^i, \tau^i, \eta^i, \alpha^i, \gamma^i \rangle$ as a bid to the DTP. Meanwhile, workers as sellers consider the DTP as a trustworthy third-party and each worker $j \in W$ independently reports $\langle \{\xi_j^i, \rho_j^i\}_{i\in U}, \{\sigma_{j,k}\}_{k\in\Phi} \rangle$ to the DTP. After receiving all the information, the DTP decides the values of q_j^i and p_j^i for each pair of user $i \in U$ and worker $j \in W$. The workers and users proceed to train the models after receiving the auction result announced by the DTP. The whole process is similar to the FL service market proposed in [21]. Fig. 3 illustrates the whole process.

3.2 Problem Formulation

For an economically efficient resource allocation, the objective of the auctioneer is to maximize social welfare defined as the total utility of all buyers and sellers. Let p_j^i be the

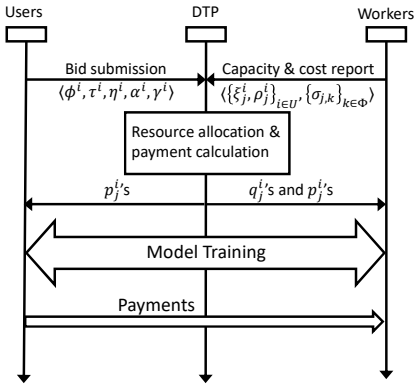


Fig. 3: Auction-based resource allocation for FL

payment of user i to worker j set by the auctioneer. Each buyer i's utility is defined as

$$u^i(\sigma^i, \{p_j^i\}_{j\in W}) = \zeta v^i(\sigma^i) - \sum_{j\in W} p_j^i = \zeta\alpha^i \ln\left(1+\gamma^i \sum_{j\in W} q_j^i\tau^i\right) - \sum_{j\in W} p_j^i, \quad (5)$$

where ζ is a factor used to transform $v^i(\cdot)$ to monetary. On the other hand, each seller j's utility is

$$u_j(\{p_j^i\}_{i\in U}, \{c_j^i\}_{i\in U}) = \sum_{i\in U} (p_j^i - c_j^i). \quad (6)$$

Therefore, the social welfare is

$$\sum_{i\in U}\left[\zeta v^i(\sigma^i) - \sum_{j\in W} p_j^i\right] + \sum_{j\in W}\sum_{i\in U}(p_j^i - c_j^i) = \sum_{i\in U}\left[\zeta\alpha^i \ln\left(1+\gamma^i \sum_{j\in W} q_j^i\tau^i\right)\right] - \sum_{j\in W}\sum_{i\in U}(\xi_j^i q_j^i \tau^i \eta^i). \quad (7)$$

The value of (7) depends on the setting of q_j^i's for each $i \in U$ and each $j \in W$. Therefore, the resource allocation problem P in FL is

$$\max_{\mathbf{q}}\left\{\zeta \sum_{i\in U}\left[\alpha^i \ln(1+\gamma^i \sum_{j\in W} q_j^i\tau^i)\right] - \sum_{j\in W}\sum_{i\in U}\xi_j^i q_j^i\tau^i\eta^i\right\} \quad (8)$$

subject to the following constraints:

$$\sum_{i\in U} \rho_j^i q_j^i \tau^i \eta^i \le T_{\max}, \forall j \in W, \quad (9)$$

$$q_j^i \tau^i \le \sigma_{j,\phi^i}, \forall i \in U, \forall j \in W, \quad (10)$$

and

$$q_j^i \in \mathbb{N}, \forall i \in U, \forall j \in W. \quad (11)$$

Though a worker may train multiple learning models at the same time, the *training time constraint* (9) limits the total training time of any worker to be a value not exceeding $T_{\max}$. Eq. (10) is the *capacity constraint* that specifies the largest possible batch size. Eq. (11) limits all batch sizes to nature numbers.

To see the complexity of P, we define

$$f^i(\mathbf{q}^i) = \zeta\alpha^i \ln\left(1+\gamma^i \sum_{j\in W} q_j^i\tau^i\right) - \sum_{j\in W} \xi_j^i q_j^i \tau^i \eta^i, \quad (12)$$

and rewrite (8) as

$$\max_{\mathbf{q}} \sum_{i\in U} f^i(\mathbf{q}^i). \quad (13)$$

Theorem 1. For each $i \in U$, $f^i(\mathbf{q}^i)$ is a concave function.

Proof: See Appendix. □

Because $f^i(\cdot)$ is concave for each $i \in U$, the objective function as a sum of $f^i(\cdot)$'s is also concave. As all batch sizes are non-negative integers, the resource allocation problem P is a concave integer programming problem.

Besides $\mathbf{q}$, the DTP also decides the payment p_j^i of each winning user $i \in B$ to each worker j, where $B \subseteq U$ is the set of all users i with $q_j^i > 0$ for some $j \in W$. Although payments seem not directly affecting the optimal value defined in (8), the pricing rule does affect the attainable social welfare in two ways. First, if the pricing rule does not incentivize users to be truthful about their valuations, the derived solution to P may not actually maximize the social welfare. We thus demand the pricing rule to be *incentive compatible*, meaning that no users can be better off by lying about their valuations. Second, the pricing rule may discourage users to participate if they may get negative utilities by the pricing rule. In that case, the social welfare may be lower than that when all users participated. Therefore, another requirement of the pricing rule is to provide *individual rationality*, which means no user can get a negative utility by participating in the auction.

4 Proposed Mechanisms

We first assume truthful bidders and propose two approaches to the resource allocation problem P. We then present a pricing rule that is incentive compatible and provides individual rationality.

4.1 Greedy Algorithm

The first approach is a greedy method which allocates worker resource to users one at a time. When allocating resource to a particular user, it first attempts allocating all the training data in need to maximize the user's utility from a worker with the lowest unit cost of training data. If the amount of allocated data is not adequate, it then attempts allocating the rest from a worker with the second lowest unit cost, and so forth. The algorithm turns to the next user when it cannot further increase the first user's utility. It repeats this process until all user's requests have been processed. Algorithm 1 elaborates the whole procedure.

There are two design issues in Algorithm 1. The first is the order to process user's requests. A straightforward idea is to rank all users by their valuation functions. However, this is not as simple as it seems. By (2), user i's valuation on σ^i depends on both α^i and γ^i. So it is possible that user i has a higher valuation than user j on data of small size but does not when the data size becomes large. Fig. 4 shows one such example. Algorithm 1 takes a simple strategy by ranking users by lexicographical order of the (α^i, γ^i) pairs. More

Algorithm 1 The Greedy Algorithm

1: $\mathbf{q} \leftarrow$ a $|U| \times |W|$ matrix with all zeros;
2: sort all users $i \in U$ by (α^i, γ^i);
3: **for** each $i \in U$ **do**
4: $\quad FW^i \leftarrow \{j \in W | \sigma_{j,\phi^i} > 0\}$; ▷ users with data
5: $\quad$ sort all $j \in FW^i$ in a non-decreasing order of ξ^i_j;
6: $\quad$ **for** each $j \in FW^i$ **do**
7: $\quad\quad$ set $q^i_{j,\text{opt}}$ by (16)
8: $\quad\quad$ set $q^i_{j,\max}$ by (17)
9: $\quad\quad$ set $q^i_{j,\text{rem}}$ by (18)
10: $\quad\quad q^i_j \leftarrow \lfloor \min\{q^i_{j,\text{opt}}, q^i_{j,\max}, q^i_{j,\text{rem}}\} \rfloor$
11: $\quad$ **end for**
12: **end for**
13: return $\mathbf{q}$;

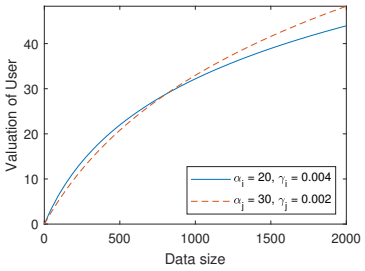


Fig. 4: Valuation functions of two users

specifically, user i ranks higher than user j if 1) $\alpha^i > \alpha^j$ or 2) $\alpha_i = \alpha_j$ and $\gamma^i > \gamma^j$.

The other design issue is the batch size of a worker allocated for the user's model training. Recall that the allocation is to maximize the user's utility subject to worker capacity and training time constraints. Therefore, three factors should be considered for this issue: the user's utility, the worker's dataset size, and the worker's training time constraint.

Suppose that we are to assign worker j for the model training of user i, and let $\sigma^i_{\text{alloc}} = \sum_{l \in W \setminus \{j\}} q^i_l \tau^i$ to be the total size of the data already allocated to user i before the allocation of worker j's dataset. If q is the batch size of worker j set for user i's model training, i's valuation will become $\alpha^i \ln(1 + \gamma^i(\sigma^i_{\text{alloc}} + q\tau^i))$. The associated payment, p^i_j, is not set at this moment but surely will be proportional to the associated cost $\xi^i_j q\tau^i\eta^i$. Letting $p^i_j = \xi^i_j q\tau^i\eta^i$, the optimal value of q that maximizes user i's utility, denoted by $q^i_{j,\text{opt}}$, is

$$q^i_{j,\text{opt}} = \arg\max_q \left\{ \zeta\alpha^i \ln\left(1 + \gamma^i\left(\sigma^i_{\text{alloc}} + q\tau^i\right)\right) - \xi^i_j q\tau^i\eta^i \right\}. \tag{14}$$

Since i's utility is a concave function of q, $q^i_{j,\text{opt}}$ can be derived by solving the following equation

$$\frac{\mathrm{d}}{\mathrm{d}q}\left(\zeta\alpha^i \ln\left(1 + \gamma^i\left(\sigma^i_{\text{alloc}} + q\tau^i\right)\right) - \xi^i_j q\tau^i\eta^i\right) = 0. \tag{15}$$

The value of $q^i_{j,\text{opt}}$ is the solution to (15), i.e.,

$$q^i_{j,\text{opt}} = q = \frac{\zeta\alpha^i\gamma^i - \xi^i_j\eta^i\left(1 + \gamma^i\sigma^i_{\text{alloc}}\right)}{\xi^i_j\eta^i\gamma^i\tau^i}. \tag{16}$$

Concerning the second factor, the setting of batch size must meet the capacity constraint as specified by (10). This constraint limits the maximum value of q^i_j to be

$$q^i_{j,\max} = \frac{\sigma_{j,\phi^i}}{\tau^i}. \tag{17}$$

For the last factor, the maximum value of q^i_j is subject to the training time constraint shown in (9), which is

$$q^i_{j,\text{rem}} = \frac{T_{\max} - T_{j,\text{alloc}}}{\rho^i_j\tau^i\eta^i}, \tag{18}$$

where $T_{j,\text{alloc}} = \sum_{l \in U \setminus \{i\}} \rho^l_j q^l_j \tau^l \eta^l$ is the amount of j's training time already allocated to other training tasks.

The algorithm sets the value of q^i_j to be the minimum of $q^i_{j,\text{opt}}$, $q^i_{j,\max}$, and $q^i_{j,\text{rem}}$, and makes a rounding on q^i_j in Line 10.

4.2 ADMM-Based Algorithm

The second approach is an approximation to the optimal solution by alternating direction method of multipliers (ADMM) [27], [28]. We target at (13) and define two functions:

$$g_{j,\text{time}}(\mathbf{q}_j) = \sum_{i \in U} \rho^i_j q^i_j \tau^i \eta^i - T_{\max}, \forall j \in W, \tag{19}$$

and

$$g^i_{j,\text{data}}(q^i_j) = q^i_j\tau^i - \sigma_{j,\phi^i}, \forall i \in U, \forall j \in W, \tag{20}$$

where $\mathbf{q}^i = [q^i_1, q^i_2, \cdots, q^i_m]$ and $\mathbf{q}_j = [q^1_j, q^2_j, \cdots, q^n_j]$. Our objective function (13) is subject to

$$g_{j,\text{time}}(\mathbf{q}_j) \le 0, \forall j \in W \tag{21}$$

and

$$g^i_{j,\text{data}}(q^i_j) \le 0, \forall i \in U, \forall j \in W. \tag{22}$$

To ensure that ADMM is applicable to our problem, $f^i(\mathbf{q}^i)$ in (13) should be concave for all $i \in U$ and all functions in (21) and (22) should be linear. Theorem 1 already proves that all $f^i(\mathbf{q}^i)$'s are indeed concave. The function linearity can be easily seen from the definitions in (19) and (20).

To apply ADMM, we first relax constraint (11) by allowing fractional solutions. We also need to define auxiliary variables $\mu_{j,\text{time}}$ for each $j \in W$ and $\mu^i_{j,\text{data}}$ for each $i \in U$ and $j \in W$. These variables are initialized to 1's in our algorithm. ADMM approximates the optimal solution by iterating the following two steps.

4.2.1 Step 1

Decompose P into unconstrained, individually solvable sub-problems. This is for each user $i \in U$ to find $\mathbf{q}^i$ such that

$$\mathbf{q}^i = \arg\max_{\mathbf{q}} \Big\{ f^i(\mathbf{q}) - \sum_{j \in W} h_1(j) - \sum_{i \in U}\sum_{j \in W} h_2(i,j) \Big\}, \tag{23}$$

where

$$h_1(j) = \begin{cases} 0, & \text{if } g_{j,\text{time}}(\mathbf{q}_j) \le 0, \\ \|g_{j,\text{time}}(\mathbf{q}_j) - \mu_{j,\text{time}}\|^2_2, & \text{otherwise,} \end{cases} \tag{24}$$

and

$$h_2(i,j) = \begin{cases} 0, & \text{if } g^i_{j,\text{data}}(q^i_j) \le 0, \\ \left\| g^i_{j,\text{data}}(q^i_j) - \mu^i_{j,\text{data}} \right\|_2^2, & \text{otherwise,} \end{cases} \tag{25}$$

are two penalties that user i will receive if the setting of $\mathbf{q}^i$ does not meet the constraints (21) and (22), respectively.

4.2.2 Step 2

Update $\mu_{j,\text{time}}$ by

$$\mu_{j,\text{time}} = \mu_{j,\text{time}} + g_{j,\text{time}}(\mathbf{q}_j) \tag{26}$$

for every $j \in W$ and update $\mu^i_{j,\text{data}}$ by

$$\mu^i_{j,\text{data}} = \mu^i_{j,\text{data}} + g^i_{j,\text{data}}(q^i_j) \tag{27}$$

for every $j \in W$ and $i \in U$. The algorithm repeats these two steps until $||(\mu_{1,\text{time}}, \cdots, \mu_{m,\text{time}}, \mu^1_{1,\text{data}}, \cdots, \mu^n_{m,\text{data}})|| \le ERR$ or $||(g_{1,\text{time}}(\mathbf{q}_1), \cdots, g_{m,\text{time}}(\mathbf{q}_m), g^1_{1,\text{data}}(q^1_1), \cdots, g^n_{m,\text{data}}(q^n_m))|| \le \epsilon$, where ERR and ϵ are two preset values standing for error tolerance and significant difference, respectively. The resulting allocation profile $\mathbf{q} = [\mathbf{q}^1, \mathbf{q}^1, \cdots, \mathbf{q}^n]$, after rounding to integers, can approximate the maximal social welfare. Algorithm 2 elaborates the process.

Algorithm 2 The ADMM-Based Algorithm

$\mu_{j,\text{time}} \leftarrow 1$ for each $j \in W$
$\mu^i_{j,\text{data}} \leftarrow 1$ for each $i \in U$ and $j \in W$
repeat
  for each $i \in U$ **do**
    find $\mathbf{q}^i$ by (23) ▷ update $\mathbf{q}^i$
  end for
  for each $j \in W$ **do**
    $\mu_{j,\text{time}} \leftarrow \mu_{j,\text{time}} + g_{j,\text{time}}(\mathbf{q}_j)$ ▷ update $\mu_{j,\text{time}}$
    for each $i \in U$ **do**
      $\mu^i_{j,\text{data}} \leftarrow \mu^i_{j,\text{data}} + g^i_{j,\text{data}}(q^i_j)$ ▷ update $\mu^i_{j,\text{data}}$
    end for
  end for
  $\delta_1 \leftarrow ||(\mu_{1,\text{time}}, \cdots, \mu_{m,\text{time}}, \mu^1_{1,\text{data}}, \cdots, \mu^n_{m,\text{data}})||$
  $\delta_2 \leftarrow ||(g_{1,\text{time}}(\mathbf{q}_1), \cdots, g_{m,\text{time}}(\mathbf{q}_m), g^1_{1,\text{data}}(q^1_1), \cdots, g^n_{m,\text{data}}(q^n_m))||$
until $\delta_1 \le ERR$ or $\delta_2 \le \epsilon$
$q^i_j \leftarrow \lfloor q^i_j \rfloor$ for each $q^i_j \in \mathbf{q}$ ▷ rounding

4.3 The Pricing Rule

Recall that a pricing rule should provide incentive compatibility and individual rationality to users. We hereby propose a pricing rule that follows the principle of VCG, which charges each winning user i of the loss of the social welfare of all other users due to the participation of user i. More explicitly, define P_{-i} to be a sub-problem that is identical to P except that user i does not participate. That is,

$$\max_{\mathbf{q}} \sum_{k \in U \setminus \{i\}} f^k\left(\mathbf{q}^k\right) \tag{28}$$

subject to

$$g_{j,\text{time}}(\mathbf{q}_j) \le 0, \forall j \in W \tag{29}$$

and

$$g^i_{j,\text{data}}(q^k_j) \le 0, \forall k \in U \setminus \{i\}, \forall j \in W. \tag{30}$$

Let $\hat{\mathbf{q}} = [\hat{\mathbf{q}}^1, \hat{\mathbf{q}}^2, \cdots, \hat{\mathbf{q}}^n]$ and $\bar{\mathbf{q}}_{-i} = [\bar{\mathbf{q}}^1, \bar{\mathbf{q}}^2, \cdots, \bar{\mathbf{q}}^{i-1}, \bar{\mathbf{q}}^{i+1}, \cdots, \bar{\mathbf{q}}^n]$ be the solutions to P and P_{-i}, respectively. Our pricing rule demands each winning user $i \in U$ to pay

$$p^i = \sum_{k \in U \setminus \{i\}} f^k\left(\bar{\mathbf{q}}^k\right) - \sum_{k \in U \setminus \{i\}} f^k\left(\hat{\mathbf{q}}^k\right) + \sum_{j \in W} \xi^i_j \hat{q}^i_j \tau^i \eta^i. \tag{31}$$

The last term in (31) represents the additional cost borne by all workers for user i's model training. With this payment, each winning user i's utility is

$$u^i(\hat{\mathbf{q}}) = \zeta v^i(\hat{\mathbf{q}}^i) - p^i, \tag{32}$$

which accounts for its valuation on $\hat{\mathbf{q}}^i$ minus the associated payment.

Before presenting the properties of this payment rule, we need the following definition.

Definition 1. An algorithm designed for the resource allocation problem is *monotonic* if its solution to P provides a social welfare that is at least as high as the social welfare of its solution to P_{-i} for any $i \in U$. Formally, such an algorithm is monotonic if

$$\sum_{k \in U} f^k\left(\hat{\mathbf{q}}^k\right) \ge \sum_{k \in U \setminus \{i\}} f^k\left(\bar{\mathbf{q}}^k\right) \tag{33}$$

for all $i \in U$.

A monotonic resource allocation algorithm provides a desirable feature that more user participants cannot lower the social welfare.

Theorem 2. Any algorithm that optimally solves the resource allocation problem P is monotonic.

Proof: Let $\hat{\mathbf{q}} = [\hat{\mathbf{q}}^1, \hat{\mathbf{q}}^2, \cdots, \hat{\mathbf{q}}^n]$ and $\bar{\mathbf{q}}_{-i} = [\bar{\mathbf{q}}^1, \bar{\mathbf{q}}^2, \cdots, \bar{\mathbf{q}}^{i-1}, \bar{\mathbf{q}}^{i+1}, \cdots, \bar{\mathbf{q}}^n]$ be the optimal solutions to P and P_{-i}, respectively, generated by the algorithm. For any $i \in U$, if $\hat{\mathbf{q}}^i = \mathbf{0}$, then $f^i(\hat{\mathbf{q}}^i) = 0$ and

$$\sum_{k \in U} f^k\left(\hat{\mathbf{q}}^k\right) = \sum_{k \in U \setminus \{i\}} f^k\left(\bar{\mathbf{q}}^k\right). \tag{34}$$

Otherwise, let $Q = \{\tilde{\mathbf{q}} \mid \tilde{\mathbf{q}}^i = \mathbf{0} \text{ s.t. (29) and (30)}\}$ be a set of suboptimal solutions to P. Because

$$\max_{\tilde{\mathbf{q}} \in Q} \sum_{k \in U} f^k\left(\tilde{\mathbf{q}}^k\right) = \sum_{k \in U \setminus \{i\}} f^k\left(\bar{\mathbf{q}}^k\right),$$

we have

$$\sum_{k \in U} f^k\left(\hat{\mathbf{q}}^k\right) > \sum_{k \in U \setminus \{i\}} f^k\left(\bar{\mathbf{q}}^k\right). \tag{35}$$

By (34) and (35), the algorithm is monotonic. □

The greedy algorithm does not guarantee monotonicity. The ADMM-based algorithm can be monotonic if the approximation and rounding errors can be controlled such that (33) holds for all $i \in U$ even after rounding. It is nontrivial to bound the errors of the ADMM-based algorithm. However, if the error of the ADMM-based solution can be modeled as a Gaussian random variable X, then the social welfares of the ADMM-based solutions to P and P_{-i} are

$$\sum_{k \in U} f^k\left(\hat{\mathbf{q}}^k\right) + X \tag{36}$$

and

$$\sum_{k\in U\setminus\{i\}} f^k\left(\bar{\mathbf{q}}^k\right) + X, \tag{37}$$

respectively. By Theorem 2, the expected value of (36) is larger than or equal to the expected value of (37). Therefore, the ADMM-based algorithm is monotonic in terms of expected social welfare. In fact, the ADMM-based algorithm exhibited monotonicity in our simulation results presented in Sec. 5.3.

For the pricing rule to provide individual rationality, each user must have a nonnegative utility as indicated by Theorem 3.

Theorem 3. For a resource allocation algorithm that is monotonic, the pricing rule shown in (31) ensures that

$$u^i(\hat{\mathbf{q}}) = \zeta v^i(\hat{\mathbf{q}}^i) - p^i \geq 0$$

for every user $i \in U$.

Proof: If user i does not win the auction, i.e., $\hat{\mathbf{q}}^i = \mathbf{0}$, then $v^i(\hat{\mathbf{q}}^i) = 0$. Also, $\sum_{k\in U\setminus\{i\}} f^k\left(\bar{\mathbf{q}}^k\right) = \sum_{k\in U\setminus\{i\}} f^k\left(\hat{\mathbf{q}}^k\right)$ so $p^i = 0$ by (31). Therefore, its utility is 0 by (32). If user i is a winning user,

$$\begin{aligned}
u^i(\hat{\mathbf{q}}^i) =& \zeta v^i(\hat{\mathbf{q}}^i) - p^i \\
=& \zeta v^i(\hat{\mathbf{q}}^i) - \sum_{k\in U\setminus\{i\}} f^k\left(\bar{\mathbf{q}}^k\right) + \sum_{k\in U\setminus\{i\}} f^k\left(\hat{\mathbf{q}}^k\right) \\
& - \sum_{j\in W} \xi_j^i \hat{q}_j^i \tau^i \eta^i && (38) \\
=& \sum_{k\in U\setminus\{i\}} f^k\left(\hat{\mathbf{q}}^k\right) + \zeta v^i(\hat{\mathbf{q}}^i) - \sum_{j\in W} \xi_j^i \hat{q}_j^i \tau^i \eta^i \\
& - \sum_{k\in U\setminus\{i\}} f^k\left(\bar{\mathbf{q}}^k\right) && (39) \\
=& \sum_{k\in U\setminus\{i\}} f^k\left(\hat{\mathbf{q}}^k\right) + f^i\left(\hat{\mathbf{q}}^k\right) - \sum_{k\in U\setminus\{i\}} f^k\left(\bar{\mathbf{q}}^k\right) && (40) \\
=& \sum_{k\in U} f^k\left(\hat{\mathbf{q}}^k\right) - \sum_{k\in U\setminus\{i\}} f^k\left(\bar{\mathbf{q}}^k\right). && (41)
\end{aligned}$$

The former and the latter terms in (41) are the social welfares of P and P_{-i}, respectively. Therefore, for a resource allocation algorithm that is monotonic, the utility of each winning user i is

$$u^i\left(\hat{\mathbf{q}}\right) = \sum_{k\in U} f^k\left(\hat{\mathbf{q}}^k\right) - \sum_{k\in U\setminus\{i\}} f^k\left(\bar{\mathbf{q}}^k\right) \geq 0. \tag{42}$$

Therefore, the pricing rule (31) provides individual rationality. □

For the pricing rule to be incentive compatible, no user $i \in U$ can be better off by declaring a valuation function $\tilde{v}^i(\cdot)$ that deviates from $v^i(\cdot)$. This property is formally stated as Theorem 4.

Theorem 4. Let $\tilde{\mathbf{q}}$ be a solution to P generated by an optimal resource allocation algorithm when $i \in U$ declares a valuation function $\tilde{v}^i(\cdot) \not\equiv v^i(\cdot)$. The pricing rule shown in (31) ensures $u^i(\hat{\mathbf{q}}) \geq u^i(\tilde{\mathbf{q}})$ for this algorithm.

Proof: Note that regardless of what $\tilde{v}^i(\cdot)$ is, the social welfare in P_{-i} remains to be $\sum_{k\in U\setminus\{i\}} f^k\left(\bar{\mathbf{q}}^k\right)$. By (41),

$$\begin{aligned}
u^i\left(\hat{\mathbf{q}}\right) - u^i\left(\tilde{\mathbf{q}}\right) =& \sum_{k\in U} f^k\left(\hat{\mathbf{q}}^k\right) - \sum_{k\in U\setminus\{i\}} f^k\left(\bar{\mathbf{q}}^k\right) \\
& - \sum_{k\in U} f^k\left(\tilde{\mathbf{q}}^k\right) + \sum_{k\in U\setminus\{i\}} f^k\left(\bar{\mathbf{q}}^k\right) && (43) \\
=& \sum_{k\in U} f^k\left(\hat{\mathbf{q}}^k\right) - \sum_{k\in U} f^k\left(\tilde{\mathbf{q}}^k\right). && (44)
\end{aligned}$$

For any optimal resource allocation algorithm, we have

$$u^i\left(\hat{\mathbf{q}}\right) - u^i\left(\tilde{\mathbf{q}}\right) = \sum_{k\in U} f^k\left(\hat{\mathbf{q}}^k\right) - \sum_{k\in U} f^k\left(\tilde{\mathbf{q}}^k\right) \geq 0. \tag{45}$$

□

5 Numerical Results

5.1 Simulation Settings

We conducted extensive simulations to evaluate the performance of the proposed methods. We assumed three types of data. For each worker, we first determined the number of data types owned by the worker using a uniform distribution over the set $\{A, B, C\}$. We then randomly determined the set of data types owned by the worker. We varied the number of users (from 2 to 11), the number of workers (from 2 to 11), the number of training samples (i.e., the dataset size) owned by workers, and the maximum training time to study the impact of these factors on the social welfare. All the other parameters are shown in Table 2. For the ADMM-based algorithm, we used SciPy library [43] to solve the subproblem in (23). All results are averages over 800 trials.

Besides the greedy and the ADMM-based algorithms, we also considered a variant of the greedy algorithm for comparison purpose. This variant differs from the greedy algorithm in three points.

- It ranks users randomly.
- For a particular user, it randomly picks up workers for resource allocation.
- After finding the best batch size q_j^i (Line 10 in Algorithm 1), it randomly sets a batch size between 0 and q_j^i for worker j to train the model of user i.

This variant is referred to as Random in all the simulation results.

5.2 Model Validation

It is difficult to perform an experimental trial based on real traces due to the lack of an existing target system. More specifically, we do have open datasets and aggregation algorithm for FL. However, concerning the incentive mechanism part, it is not feasible to acquire user-dependent and worker-dependent private information such as user's valuation function $v_i(\cdot)$ (and associated coefficients α^i and γ^i) and worker's unit cost for training user j's model ξ_j^i. Nevertheless, we performed a simple experiment to testify the validness of the basic assumptions for the proposed approach.

As we assumed three data types, we divided the MNIST dataset into three subsets (one for each data type) A, B, and

TABLE 2: Simulation Settings

Name	Description	Value Setting
l	Number of data types	3
n	Number of users	Default: 4. Range: 2 to 11 w/ step size 1
m	Number of workers	Default: 4. Range: 2 to 11 w/ step size 1
$T_{\max}$	Maximum training time (s.)	Default: 3600. Range: 1200 to 12000 w/ step size 1200
ζ	Parameter used in (5)	5
$\sigma_{j,k}$	Number of type-k training samples owned by worker j	$\mathcal{U}[\sigma_{\min}, \sigma_{\max}]$
$\sigma_{\min}$	Minimum dataset size	Default: 0. Range: 0 to 25000 w/ step size 2500
$\sigma_{\max}$	Maximum dataset size	Default: 10000. Range: 10000 to 35000 w/ step size 2500
τ^i	Iteration count of user i	$\mathcal{U}[300, 450]$
η^i	Epoch count of user i	$\mathcal{U}[10, 30]$
α^i	Parameter in $v^i(\cdot)$ definition	$\mathcal{U}[100, 500]$
γ^i	Parameter in $v^i(\cdot)$ definition	$\mathcal{U}[30, 50]$
C_i	User-dependent coefficient for user i	$\mathcal{U}[0.1, 1]$
ξ_j	Worker-dependent unit cost of worker j	$\mathcal{U}[10^{-4}, 10^{-3}]$
ξ_j^i	Unit cost of worker j for training user i's model	$\xi_j \times C_i$
ρ_j	Worker-dependent unit time of worker j	$\mathcal{U}[0.005, 0.01]$
ρ_j^i	Unit time of worker j for training user i's model (s.)	$\rho_j \times C_i$
ERR	Parameter in Algorithm 2	0.1
ϵ	Parameter in Algorithm 2	0.01

TABLE 3: The setup of each user and worker

(a) User's demands and valuation coefficients

User ID	Data type	#Iteration	#Epoch	α^i	γ^i
1	B	424	23	170.2	35.1
2	A	418	15	399.4	41.2

(b) Worker's dataset

Worker ID	Type A	Type B	Type C
1	✓		
2	✓	✓	✓
3		✓	
4		✓	✓

(c) Value of ρ_j^i

j	$i=1$	$i=2$
1	-	0.0011
2	0.0057	0.0009
3	0.0065	-
4	0.0042	-

(d) Value of $\xi_j^i \times 10^5$

j	$i=1$	$i=2$
1	-	9.95
2	49.08	8.20
3	36.79	-
4	17.23	-

C, each consisting of images of three sequential handwritten digits ($A : [1, 3]$, $B : [4, 6]$, $C : [7, 9]$). We took the same simulation setting used in the next section to generate 1) the demanded data type and associated iteration and epoch counts for each user and 2) the set of data types owned by each worker and parameters for unit time and cost. Table 3 shows the setting generated for two users and four workers. We then ran the ADMM-based and greedy algorithms to derive the batch sizes for workers to train their local models (Table 4). We used weighted FedAvg to aggregate local models, which assigned a weight to each model that represents the batch size for that model. We measured each worker's training time for each user, and tested the accuracy of each user's model.

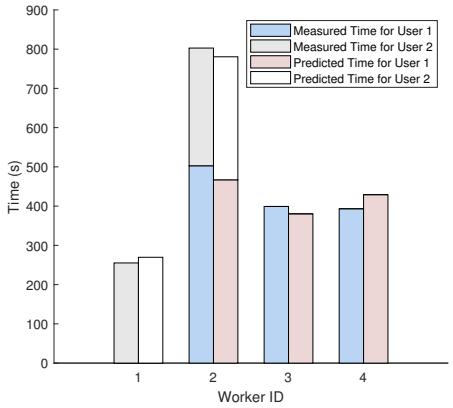


Fig. 5: Measured and predicted training time. All the training tasks were done by a PC with Ryzen 5600X@3.7GHz CPU and 16 GB RAM.

Fig. 5 shows the measured training time together with the predicted working time of each worker. All the training tasks were done by a single computer, whereas the working time was predicted by (4). Because all workers actually had identical computing power in the measurements, we replaced ρ_j^i in (4) with a constant 0.001 in the corresponding predictions for a fair comparison. The result clearly shows the validness of the working time prediction.

The model accuracies of the two users were measured to be 99.54% and 99.68%, respectively. On the other hand, the model qualities as estimated by (1) are 14.48 and 14.29, respectively. Clearly, our definition of model quality does capture model accuracy in this trial. .

5.3 Number of Users

We first study the changes of the social welfare by varying the number of users. We initially set up four workers and two users. We then added one user at a time and measured the average social welfare. Figure 6 shows how the number of users affects the average social welfare. In all algorithms, the average social welfare increased as the number of users increased. This is exactly the reason why we need to guarantee individual rationality: more user participation implies higher social welfare. The result also supports the conjecture that these algorithms are monotonic.

The ADMM-based algorithm has the highest increasing rate among all the algorithms, followed by Random and then the greedy algorithm. When only two users were requesting model training, the workers could collectively offer the optimal batch size as indicated by (16) to each user. The associated training time was also lower than $T_{\max}$. Consequently, as shown in Table 4, the greedy algorithm and the ADMM-based algorithm yielded identical result. However, when 11 users were requesting model training from only four workers, both the capacity and training time

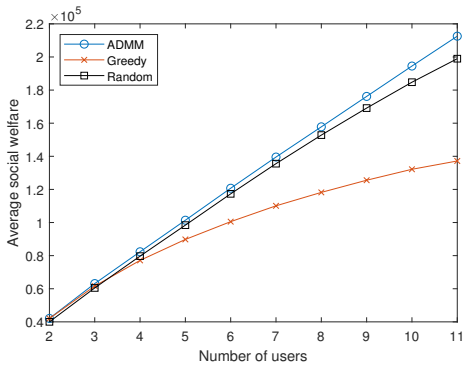


Fig. 6: Average social welfare with respect to the number of users

constraints were obstacles to an optimal allocation to all users. In this case, the ADMM-based algorithm achieved a higher social welfare than the greedy algorithm by providing a finer granularity of resource allocation. Refer to Table 5 for the allocation results of both algorithms. As demonstrated by the ADMM-based algorithm, the way to maximize the social welfare in this case is to distribute worker training resource among a relatively large number of users. The greedy algorithm failed to do so because it tends to allocate resource to fewer but more "valuable" users. In fact, a worker was assigned to four to six users by the ADMM-based algorithm but only two to three users by the greedy algorithm. Consequently, six users did not have their models trained by any worker in the greedy algorithm, which explains the low social welfare of the greedy algorithm in that case.

TABLE 4: The batch size of each worker for each user (two users and four workers)

(a) ADMM-based algorithm

User ID	Worker ID 1	2	3	4
1	0	48	39	44
2	43	50	0	0

(b) Greedy algorithm

User ID	Worker ID 1	2	3	4
1	0	48	39	44
2	43	50	0	0

TABLE 5: The batch size of each worker for each user (11 users and four workers)

(a) ADMM-based algorithm

User ID	Worker ID 1	2	3	4
1	0	11	10	13
2	43	50	0	0
3	16	47	0	0
4	7	29	0	0
5	0	0	0	42
6	0	0	9	9
7	0	0	0	10
8	0	0	15	16
9	39	1	0	0
10	0	0	46	5
11	5	0	0	0

(b) Greedy algorithm

User ID	Worker ID 1	2	3	4
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	46	53	0	0
5	0	0	0	42
6	0	0	0	0
7	0	0	0	0
8	0	0	43	48
9	28	41	0	0
10	0	0	38	31
11	0	0	0	0

The superiority of Random over the greedy algorithm can be justified by the following feature of Random. When setting up the batch size for a worker to train a user model, Random does not allocate the largest possible batch size (as indicated by q_j^i in Algorithm 1) from worker j to user i. Consequently, Random generally assigns a worker to more users than the greedy algorithm. This helps increasing the social welfare especially when workers do not have adequate training resource for all the model training tasks.

5.4 Number of Workers

We next measured the impact of the worker population on social welfare. We fixed four users and tested 2 to 11 workers by adding one worker at a time. Figure 7 shows the result.

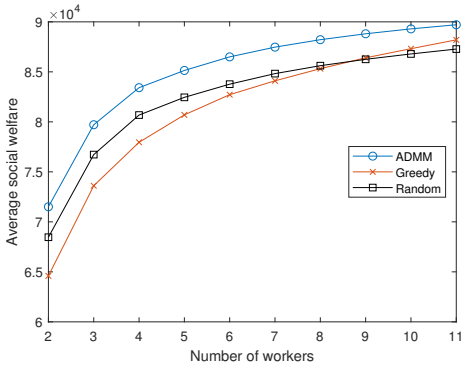


Fig. 7: Average social welfare with four users.

We can observe that the social welfare increased in all algorithms as the number of workers increased. This can be explained as each new worker brings in additional data and computing power for model training. However, the model training cost increases linearly with the batch size whereas the user valuation increases only sub-linearly with the batch size (due to the effect of diminishing returns). Therefore, as shown in Figure 7, when the number of workers increased, the social welfare increased but the increasing rate decreased. Although the greedy algorithm performed worse than the ADMM-based algorithm in all settings, the performance gap becomes smaller when more workers participated. This is consistent with the finding in the previous subsection that the performance of the greedy algorithm is acceptable when workers have adequate resource for the model training.

Note that the number of workers can easily exceed a hundred in practice. The result here does not exclude the need for tens or hundreds of workers to maximize social welfare in other settings (e.g., when workers have few training samples). However, we also note that, more workers (i.e., sellers) leads to more supply of training resource (goods) and thus lessens resource competition among users (i.e., buyers). Consequently, it becomes a buyer's market where each user can hire any desired set of workers with a payment to each worker just a bit higher than the worker's training cost. Our study targets at a balanced market where users may need to compete for worker's resources.

5.5 Effects of The Dataset Size

We investigated the effects of the capacity constraint by fixing the set of users and workers but expanding the dataset size. The mean number of training samples of each worker was set to 5000 initially. We then increased the mean by 2500 at a time and measured the average social welfare. Figure 8 shows how the dataset size affects the result.

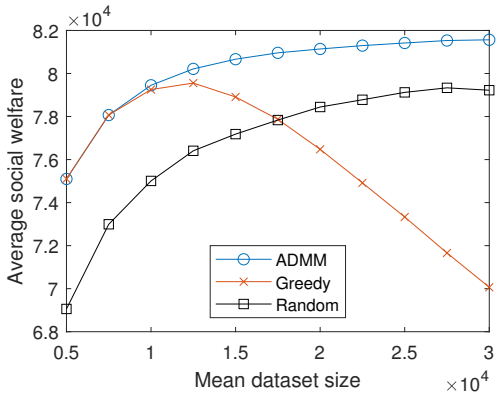


Fig. 8: The average social welfare versus the dataset size

Both Random and the ADMM-based algorithm achieved a growth in social welfare with increase in dataset size. For the greedy algorithm, the social welfare increased with the dataset size when the mean dataset size did not exceed 12500. After that point, enlarging datasets turned to decrease the social welfare of the greedy algorithm.

TABLE 6: The batch size of each worker for each user (mean dataset size: 5000)

(a) ADMM-based algorithm

User ID	Worker ID 1	2	3	4
1	0	25	26	18
2	15	12	16	0
3	0	29	24	0
4	0	29	25	0

(b) Greedy algorithm

User ID	Worker ID 1	2	3	4
1	0	25	26	18
2	15	12	16	0
3	0	29	24	0
4	0	29	25	0

The reason behind this phenomenon is that the worker computing power remained unchanged. With small datasets, the resource allocation is only subject to the capacity constraint. So setting up the largest possible batch size for a pair of worker and user does not deviate significantly from the optimal setting. Refer to Table 6 for the allocation results with the mean dataset size set to 5000. Here two to three workers were needed for the model training of a single user because no single worker had sufficiently large dataset to maximize a user's utility. Furthermore, even if the greedy algorithm allocates the whole dataset of a worker to a particular user, the worker still has spare time to train other user's learning models. This explains the result why some worker was able to train models for all the four users at the same time. In short, the training time constraint did not prevent a worker from serving as many users as possible.

TABLE 7: The batch size of each worker for each user (mean dataset size: 30000)

(a) ADMM-based algorithm

User ID	Worker ID 1	2	3	4
1	0	74	73	69
2	25	22	22	0
3	0	9	8	0
4	0	0	60	0

(b) Greedy algorithm

User ID	Worker ID 1	2	3	4
1	0	80	82	74
2	65	0	0	0
3	0	26	34	0
4	0	0	0	0

However, with large datasets, the resource allocation is no longer bounded by the capacity constraint. When the average dataset size was larger than 12500, allocating the whole dataset of a worker to a particular user for model training takes considerable time from the worker. As a result, the training time constraint prevents the worker from serving too many users at the same time. Because user's valuation on dataset size exhibits the effect of diminishing returns, distributing worker computation power to more users can generally achieve higher social welfare than the alternative that allocates worker resource to few users. Refer to Table 7 for the detailed allocations with mean dataset size set to 30000.

5.6 Effects of The Training Time Constraint

The last factor we examined is the training time constraint. The maximum training time $T_{\max}$ was ranged from 1200 to 12000 seconds with a step size of 1200 seconds. Figure 9 shows how the average social welfare changed with respect to different settings of $T_{\max}$.

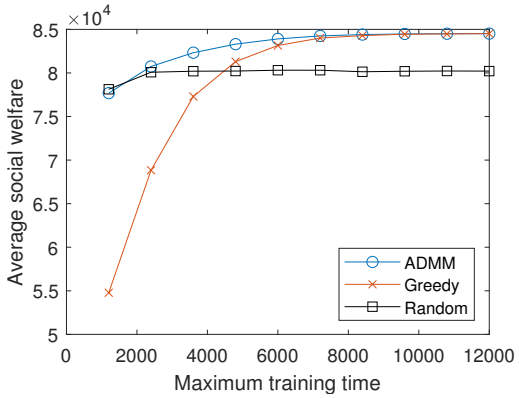


Fig. 9: The average social welfare versus the maximum training time.

When $T_{\max}$ is small, a worker can only serve a few users if we always allocate the largest possible batch size for a training task. This is exactly what the greedy algorithm did in our simulations. In fact, a worker could have served more users if the batch size had not been set to the maximum. This allocation strategy were actually taken by Random and the ADMM-based algorithm, which explains their superiority over the greedy algorithm when $T_{\max}$ was small. Refer to Table 8 for the allocation results with $T_{\max} = 1200$. As the maximum training time increased, both the ADMM-based and the greedy algorithms had increased average social welfare. This can be justified as the training time constraint is no longer an obstacle to the resource allocation when $T_{\max}$ is sufficiently large. In that case, the allocation is only bounded by the capacity constraint. This explains not only the trend of the increasing social welfare but also the convergence of both algorithms. Refer to Table 9 for the allocation results with $T_{\max} = 12000$.

The performance of Random seems not improving as $T_{\max}$ increased. This is because Random did not allocate the maximum batch size even if doing so could maximize the social welfare without violating the training time constraint.

5.7 Heterogeneity of Users

We simulated heterogeneous users by varying the diversity in user's valuations on dataset size. More specifically, we

TABLE 8: The batch size of each worker for each user ($T_{\max} = 1200$)

(a) ADMM-based algorithm

User ID	Worker ID 1	2	3	4
1	0	47	0	0
2	0	38	53	33
3	70	34	51	31
4	57	3	13	24

(b) Greedy algorithm

User ID	Worker ID 1	2	3	4
1	0	0	0	0
2	0	41	45	35
3	70	0	0	4
4	57	40	48	36

TABLE 9: The batch size of each worker for each user ($T_{\max} = 12000$)

(a) ADMM-based algorithm

User ID	Worker ID 1	2	3	4
1	0	47	0	0
2	0	43	57	35
3	70	48	58	44
4	57	40	48	36

(b) Greedy algorithm

User ID	Worker ID 1	2	3	4
1	0	47	0	0
2	0	43	57	35
3	70	48	58	44
4	57	40	48	36

used a uniform distribution $\mathcal{U}[300-\epsilon, 300+\epsilon]$ to set up the value of α^i for each user i's valuation, where the value of ϵ ranged form 0 to 135 with a step size 15. A large ϵ implies a large range of the α^i values and thus highly diverse user valuations. Fig. 10 shows the results by applying the ADMM-based algorithm.

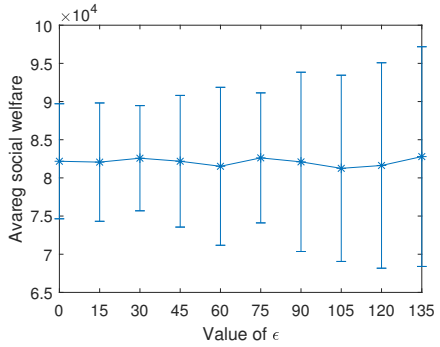


Fig. 10: Average social welfare versus the diversity of user valuations on datasets

It can be seen that the average value of the social welfare did not vary significantly with ϵ. The reason is that the mean value of α^i remained to be 300 regardless of the value of ϵ. Nevertheless, the standard deviation did increase as ϵ increased.

6 Conclusions

In this work, we have formally modeled the incentive-aware resource allocation problem in multi-worker multi-user FL and proposed an auction-based solution framework. In the framework, the DTP as an auctioneer allocates source from each workers to each user for the user's model training. We have proposed two methods for the DTP to approximate the optimal result of the auction: a greedy algorithm and an ADMM-based algorithm. We have also proposed a pricing rule that provides incentive compatibility and individual rationality. For performance evaluation, we have conducted simulations to study how the social welfare is affected by various factors including the number of users, the number of workers, the dataset size, and the maximum training time. The simulation results show that the ADMM-based algorithm achieves higher social welfare than the greedy algorithm due to its ability to allocate worker resource to users in a more elaborate manner. The performance gap between these two approaches is significant when resource collectively supplied by workers does not suffice to train all user's models but not when sufficiently many workers participate or when the training time constraint is appropriately loosened.

In short, this study demonstrates the feasibility of auction-based incentive mechanism for multi-user FL. Our future work is to perform optimality analyses of the proposed algorithms and evaluate them with a considerable number of workers that have weak computation capability such as Internet of Things (IoT) devices.

References

[1] T. Kraska, A. Talwalkar, J. Duchi, R. Griffith, M. J. Franklin, and M. Jordan, "MLbase: A distributed machine-learning system," in *The 6th Biennial Conf. on Innovative Data Systems Research*, Jan. 2013.

[2] M. Li, D. G. Andersen, A. Smola, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *Proc. 11th USENIX Symp. on Operating Systems Design and Implementation*, Oct. 2014, pp. 583–598.

[3] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated Learning: Strategies for Improving Communication Efficiency," *CoRR*, vol. abs/1610.05492, 2016.

[4] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. on Intelligent Systems and Technology*, vol. 10, no. 12, pp. 1–19, 2019.

[5] O. Wahab, A. Mourad, H. Otrok, and T. Taleb, "Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1342–1397, 2021.

[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Stat.*, 2017, pp. 1273–1282.

[7] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Machine Learning and Systems*, vol. 2, 2020, pp. 429–450.

[8] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," *CoRR*, vol. abs/1912.00818, 2019.

[9] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," *CoRR*, vol. abs/2002.06440, 2020.

[10] X. Gong, A. Sharma, S. Karanam, Z. Wu, T. Chen, D. Doermann, and A. Innanje, "Ensemble attention distillation for privacy-preserving federated learning," in *Proc. IEEE/CVF Int'l Conf. on Computer Vision*, Oct. 2021, pp. 15 076–15 086.

[11] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 634–643.

[12] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. Int. Conf. Artif. Intell. Stat.*, 2020, pp. 2938–2948.

[13] 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. 10 700–10 714, Dec. 2019.

[14] Y. M. Saputra, D. N. Nguyen, D. T. Hoang, T. X. Vu, E. Dutkiewicz, and S. Chatzinotas, "Federated learning meets contract theory: energy-efficient framework for electric vehicle networks," *CoRR*, vol. abs/2004.01828, 2020.

[15] W. Y. B. Lim, Z. Xiong, C. Miao, D. Niyato, Q. Yang, C. Leung, and H. V. Poor, "Hierarchical incentive mechanism design for federated machine learning in mobile networks," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9575–9588, Jul. 2020.

[16] 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.

[17] Y. Zhan, P. Li, 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.

[18] L. U. Khan, S. R. Pandey, N. H. Tran, W. Saad, Z. Han, M. N. H. Nguyen, and C. S. Hong, "Federated learning for edge networks: Resource optimization and incentive mechanism," *IEEE Commun. Mag.*, vol. 58, no. 10, pp. 88–93, 2020.

[19] S. R. Pandey, N. H. Tran, M. Bennis, Y. K. Tun, A. Manzoor, and C. S. Hong, "A crowdsourcing framework for on-device federated learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3241–3256, May 2020.

[20] G. Xiao, M. Xiao, G. Gao, S. Zhang, H. Zhao, and X. Zou, "Incentive mechanism design for federated learning: a two-stage Stackelberg game approach," in *IEEE 26th Int'l Conf. on Parallel and Distrib. Syst.*, Dec. 2020.

[21] Y. Jiao, P. Wang, D. Niyato, B. Lin, and D. I. Kim, "Toward an automated auction framework for wireless federated learning services market," *IEEE Trans. Mobile Comput.*, vol. 20, no. 10, pp. 3034–3048, Oct. 2021.

[22] T. Le, N. H. Tran, Y. K. Tun, M. N. H. Nguyen, S. R. Pandey, Z. Han, and C. S. Hong, "An incentive mechanism for federated learning in wireless cellular networks: An auction approach," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 4874–4887, Aug. 2021.

[23] Y. Deng, F. Lyu, J. Ren, Y.-C. Chen, P. Yang, Y. Zhou, and Y. Zhang, "FAIR: Quality-aware federated learning with precise user incentive and model aggregation," in *Proc. IEEE INFOCOM*, May 2021.

[24] W. Y. B. Lim, J. S. Ng, Z. Xiong, J. Jin, Y. Zhang, D. Niyato, C. Leung, and C. Miao, "Decentralized edge intelligence: a dynamic resource allocation framework for hierarchical federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 536–550, Mar. 2022.

[25] J. S. Ng, W. Y. B. Lim, Z. Xiong, X. Cao, D. Niyato, C. Leung, and D. I. Kim, "A hierarchical incentive design toward motivating participation in coded federated learning," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 1, pp. 359–375, Jan. 2022.

[26] L. Deng, "The MNIST database of handwritten digit images for machine learning research," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, Nov. 2012.

[27] G. Banjac, P. Goulart, B. Stellato, and S. Boyd, "Infeasibility detection in the alternating direction method of multipliers for convex optimization," *J. Optim. Theory Appl.*, vol. 183, pp. 490–519, 2019.

[28] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[29] J. Li, Y. Zhu, Y. Hua, and J. Yu, "Crowdsourcing sensing to smartphones: A randomized auction approach," *IEEE Trans. Mobile Comput.*, vol. 16, no. 10, pp. 2764–2777, 2017.

[30] Q. Li, H. Yao, T. Mai, C. Jiang, and Y. Zhang, "Reinforcement-learning- and belief-learning-based double auction mechanism for edge computing resource allocation," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5976–5985, 2020.

[31] W. Zhong, C. Yang, K. Xie, S. Xie, and Y. Zhang, "ADMM-based distributed auction mechanism for energy hub scheduling in smart buildings," *IEEE Access*, vol. 6, pp. 45 635–45 645, 2018.

[32] C. T. Dinh, N. H. Tran, M. N. H. Nguyen, C. S. Hong, W. Bao, A. Y. Zomaya, and V. Gramoli, "Federated learning over wireless networks: Convergence analysis and resource allocation," *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 398–409, Feb. 2021.

[33] M. Cong, H. Yu, X. Weng, J. Qu, Y. Liu, and S.-M. Yiu, "A VCG-based fair incentive mechanism for federated learning," *CoRR*, vol. abs/2008.06680, 2020.

[34] W. Vickrey, "Counterspeculation, auctions and competitive sealed tenders," *The Journal of Finance*, vol. 16, no. 1, pp. 8–37, Mar. 1961.

[35] E. Clarke, "Multipart pricing of public goods," *Public Choice*, vol. 11, no. 1, pp. 17–33, 1971.

[36] T. Groves, "Incentives in teams," *Econometrica*, vol. 41, no. 4, pp. 617–631, Jul. 1973.

[37] S. Kim, "Incentive design and differential privacy based federated learning: a mechanism design perspective," *IEEE Access*, vol. 8, pp. 187 317–187 325, 2020.

[38] S. J. Raudys and A. K. Jain, "Small sample size effects in statistical pattern recognition: Recommendations for practitioners," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 3, pp. 252–264, Mar. 1991.

[39] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," *CoRR*, vol. abs/1806.00582v2, 2022.

[40] Y. Zou, S. Feng, D. Niyato, Y. Jiao, S. Gong, and W. Cheng, "Mobile device training strategies in federated learning: an evolutionary game approach," in *Proc. Int. Conf. Internet Things, IEEE Green Comput. Commun., IEEE Cyber, Phy. Social Comput., IEEE Smart Data*, 2019, pp. 874–879.

[41] N. Ferdinand, H. Al-Lawati, and S. C. Draper, "Anytime Minibatch: Exploiting stragglers in online distributed optimization," in *Proc. of the 7th Int. Conf. on Learning Representations*, May 2019.

[42] A. Reisizadeh, H. Taheri, A. Mokhtari, H. Hassani, and R. Pedarsani, "Robust and communication-efficient collaborative learning," in *Proc. Advances in Neural Information Processing Systems*, Dec. 2019, pp. 8386–8397.

[43] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright *et al.*, "SciPy 1.0: fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, no. 3, pp. 261–272, 2020.

Feng-Yang Chen received the M.S. degree in computer science from National Yang Ming Chiao Tung University, Hsinchu, Taiwan, in 2021. He is currently an engineer with Chunghwa Telecom, Taiwan. His research interests include resource optimization and game theory.

Li-Hsing Yen received the B.S., M.S., and Ph.D. degrees in computer science from National Chiao Tung University, Hsinchu, Taiwan, 1989, 1991, and 1997, respectively. He has been a professor of National Chiao Tung University, Taiwan, since 2016. His current research interests include distributed computing, wireless networking, and game theory. He has been in the Editor Boards of the Springer's Wireless Networks and is the recipient of the Best Paper Awards of the IEEE WCNC 2013, ISPA 2015, and APNOMS 2021.