Incentive-Aware Resource Allocation Using Double Auction for Many-to-Many Federated Learning

Jing-Xuan Wang and Li-Hsing Yen, *Member, IEEE*

Abstract—Federated learning (FL) is a distributed learning framework that enables multiple client devices to collaborate with a central server for model training while protecting against data leakage risks. A critical issue in FL is optimally allocating training resources such as training data and client computing power. The server aims at 1) a higher model quality by allocating more training data and 2) a shorter training time by allocating a higher CPU-cycle frequency. However, more training resources also imply higher energy costs for clients and thus call for higher rewards from the server. We study incentive-based resource allocation for FL to maximize social welfare. This differs from prior work in assuming multiple servers and joint consideration of model accuracy, training time, and energy cost. To address the challenges of asymmetric information, we propose an iterative double auction algorithm that approximates maximal social welfare, accompanied by a pricing rule that ensures incentives for all participants. We provide rigorous mathematical proofs showing the convergence and economic properties of the proposed approach. We also conducted simulations to study the performance of the proposed approach and the impacts of various factors.

1 Introduction

The rapid development of edge devices, such as smart mobile devices, has led to an exponential growth of intelligent applications. A common way to utilize massive data generated by these applications is to upload the data to the cloud for model training. However, this way of data utilization has become increasingly challenging with growing privacy concerns such as those outlined in the General Data Protection Regulation (GDPR).

Federated Learning (FL) [1] is a new decentralized learning framework that allows multiple client devices to train a shared global model without compromising their data privacy. The FL learning process consists of four main steps: download, local update, upload, and aggregation (Fig. 1). Each participating client device gets the latest global model from the central server in the download step. Taking the downloaded model as its local model, each client then trains its local model using its own data and computing power in the local update step. The updated model is uploaded to the central server during the upload step. Finally, the central server aggregates the uploaded model updates to form a new global model. This process is repeated until the global model achieves the desired quality.

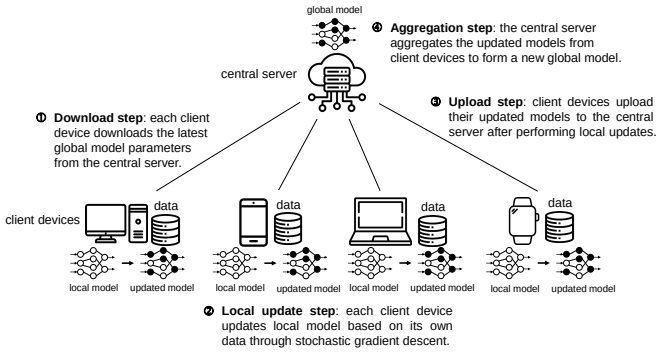


Fig. 1: Schematic Diagram of Federated Learning Process

Because FL demands training resources (e.g., training data and computing power) from clients, it is crucial to design effective incentive schemes [2] that motivate clients to participate in the training tasks actively. This challenge is especially acute in real-world deployments such as mobile crowdsourcing, edge computing with Internet of Things (IoT) devices, and cross-silo enterprise FL, where participating devices are highly heterogeneous in computational capability, communication bandwidth, and energy budget. For example, a battery-constrained IoT sensor has very different cost constraints and willingness to participate than a well-powered workstation. Without a carefully designed incentive mechanism, rational self-interested clients may withhold data or computing resources, leading to degraded model quality. A typical design introduces a reward system that offers monetary compensation to clients based on their training contributions and costs. This study uses model quality and training performance as contributions and energy consumption as cost metrics. Poor model quality can result from inadequate training data or heterogeneity among training datasets, where non-independent and identically distributed (non-IID) data may not converge to a certain accuracy [3]. Client devices can drag down

- *J.-X. Wang was with the Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan. E-mail: jxwang.cs10@nycu.edu.tw.*
- *L.-H. Yen (corresponding author) has been with the Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan. E-mail: lhyen@nycu.edu.tw.*

training performance with inferior computational capabilities, poor network conditions, or excessive training data, a phenomenon known as the straggler effect [4]. Moreover, energy consumption incurs a cost, which is an issue, particularly for energy-constrained edge devices in the IoT scenario. Incorporating these metrics in the incentive scheme is crucial to motivating clients' contributions to the FL process while ensuring high-accuracy global models.

There have been three main approaches to designing incentive mechanisms for FL: contract theory [5], [6], [7], [8], Stackelberg game [9], [10], [11], [12], and auction theory [13], [14], [15], [16]. Compared with other alternatives, the auction-based approach offers several advantages, such as maximizing social welfare and selecting the most qualified and motivated participants in the sealed market [2]. However, most existing auction-based approaches [13], [14], [15] were designed for a one-to-many (one server versus multiple clients) FL setup, where a single central server acts as the seller and multiple client devices act as buyers. When multiple servers coexist, many prior auction-based approaches [17], [18], [19] constrained a client's participation in only one learning task, which effectively partitions all clients into exclusive clusters. This setting leads to underutilization of the client's training capacities. Similar to [16], our setting of many-to-many FL allows multiple clients to contribute to multiple global models *simultaneously*, which creates an indirect interplay among servers through their common use of some clients. This type of interplay had been overlooked previously. This study differs from [16] and all other approaches in jointly considering model quality, training time, and energy cost.

In practice, FL deployments often exhibit a many-to-many interaction pattern between model owners and clients: multiple servers concurrently engage a shared pool of client devices, and each client may participate in multiple training processes. A representative example is a healthcare network in which several research institutions each train their own diagnostic models using data held by cooperating medical providers. In such many-to-many settings, a one-to-one assignment of clients to servers underutilizes client resources and ignores the resource competition introduced by sharing. Furthermore, the inputs required for computing the optimal allocation, including server utility functions and client cost functions, are inherently private due to competitive interests and confidentiality requirements. Consequently, a centralized optimizer such as a genetic algorithm (GA) is infeasible: it has no direct access to the private information on which the optimization depends. Auction theory addresses these challenges by design: the pricing rule ensures that each participant's rational self-interest leads to the socially optimal outcome, with provable guarantees on individual rationality, budget balance, incentive compatibility, and economic efficiency. These practical considerations motivate the adoption of an auction-based, decentralized mechanism that handles asymmetric information while jointly optimizing model accuracy, training time, and energy consumption.

We summarize the main contributions of this paper as follows.

- We introduce a system model incorporating two training resource types (training data and CPU-clock frequency) and three performance metrics (model quality, training time, and energy cost). No prior work has jointly considered all these factors.
- We assume the presence of multiple FL servers and allow them to share the training resources of a single client. Only a few studies [20], [15], [16] considered the same setting.
- We formulate the problem of social welfare maximization (SWM) by allocating the training resources from clients to servers. We propose a solution to SWM based on iterative double auction (IDA), where servers declare their resource demands and clients declare their resource provisions through bids in each round of the auction. Accompanied by a pricing rule, this approach is proven to converge to the optimal solution while each participant sets up its bid only to maximize its payoff.
- We provide rigorous proof to show the proposed approach's convergence and economic properties, individual rationality, budget balance, incentive compatibility, and economic efficiency.

The remainder of this paper is organized as follows. Sec. 2 reviews previous studies relevant to our work. Sec. 3 presents the system model and formulates the problem. We detail the proposed approach to resource allocation and pricing rule in Sec. 4. Several important properties of the proposed approach are analyzed in Sec. 5. We conducted a series of experiments with the results presented in Sec. 6. The last section concludes this work and lists some future work.

2 Related Work

2.1 Challenges of Federated Learning

FL presents unique challenges due to its utilization of non-IID data from distributed devices, posing a high risk of data contamination to the global model. Several studies [21], [3] have investigated the impact of data heterogeneity on model accuracy. Several studies [22], [23], [24] have addressed the straggler effect, which may be rooted in the diversity of the client's computational capabilities and network conditions. For IoT client devices, Dinh et al. [25] argued that data heterogeneity affects energy consumption and convergence time. Yang et al. [26] proposed an iterative algorithm to minimize energy consumption. These findings underscore the importance of energy-efficient resource allocation strategies in FL, which contribute to developing sustainable and effective FL systems.

2.2 Incentive Mechanism for Federated Learning

Contract theory [5], [6], [7], [8] creates binding agreements between the server and clients that specify rewards and penalties for different participation and performance levels. Contracts can be customized for each client, considering their unique constraints and preferences. This approach allows for clear expectations and rewards for clients and thus ensures their active participation in the FL process. In the context of contract theory, Kang et al. [5] introduced

reputation-based participant selection and incentive mechanisms in FL to improve learning accuracy. Lim et al. [6] proposed a hierarchical framework with contract-based rewards to facilitate FL in mobile crowdsensing environments. Ding et al. [7] explored the impact of incomplete information on server costs in contract-based incentive mechanisms. Lim et al. [8] designed a multidimensional contract framework for drone owners to truthfully report their types in a drone-enabled FL. More recently, Wang et al. [27] proposed a mechanism that applies contract theory to maximize a single CVFL server's utility while ensuring task convergence and accuracy across multiple vehicle clusters. While this work is effective at modeling bilateral agreements with information asymmetry, it assumes a single server and thus does not address multi-server competition or resource sharing between different servers.

Another incentive mechanism design is based on the Stackelberg game model [9], [11], [10], [12]. In this model, the central server acting as a leader sets the rules of the game, which typically determine how the rewards are allocated to participating clients. On the other hand, clients acting as followers strategically respond with their resource contributions to the training task to maximize their payoffs. Despite the self-interested nature of the leader and followers, this approach still aligns with the goals of the FL process. The Stackelberg game model can be used to analyze whether the interplay between participants reaches a Stackelberg equilibrium, where no participant has an incentive to deviate from the chosen strategy unilaterally.

Auction-based approaches [14], [13] prioritize social welfare over individual payoff by selecting the most qualified and motivated clients to participate. The selection requires private information from all participants, but a well-designed pricing rule can motivate participants to report such information truthfully through the submitted bids. Jiao et al. [14] presented an auction framework for FL to maximize social welfare through truthful bidding, with winners determined by reinforcement learning. Thi Li et al. [13] used a random auction strategy to minimize social cost in FL auctions, guaranteeing approximate truthfulness while participants' expected utilities are non-negative. Kim [28] incorporated privacy levels into the cost calculation for IoT devices in FL auctions, using the Vickrey-Clarke-Groves mechanism to determine winners and payments while preserving incentive compatibility. More recently, Zhang et al. [29] proposed an auction-based incentive mechanism that integrates communication path finding, jointly optimizing both resource allocation and communication quality.

A comprehensive survey by Tu et al. [30] compared all three paradigms and concluded that auction-based approaches are uniquely suited to multi-party settings where participants have private information and social welfare maximization is the design goal. Unlike contract theory and Stackelberg game approaches, auction mechanisms naturally handle asymmetric information from multiple competing parties simultaneously and come with provable economic guarantees on individual rationality, budget balance, and incentive compatibility.

2.3 Many-to-Many Federated Learning

All studies mentioned in the previous subsection assumed a single server with a single global model. When multiple servers are present (each with a distinct learning model), many existing studies [6], [18], [19] partitioned all clients into disjoint groups, one for each training task. This treatment renders the problem into multiple one-to-many FL tasks, which does not fully utilize the client's training resources but avoids potential competition among servers for the client's training resources. Only a few studies [20], [15], [16] have explored the potential of many-to-many FL, which allows servers to share client resources in a non-exclusive manner. These works are the most closely related to ours, and we compare them in detail below and in Table 1.

Cho et al. [20] modeled the interaction between multiple FL servers and clients as a multi-leader multi-follower Stackelberg game and proposed an exact method for finding the Stackelberg equilibrium. However, their work accounted for clients' training data but did not consider computing power (CPU-cycle frequency) and energy costs. Furthermore, their work assumes complete information, applying backward induction to obtain solutions. By contrast, IDA operates under asymmetric information, thereby better preserving participants' privacy.

Mai et al. [15] proposed a double auction mechanism for many-to-many FL that maximizes social welfare, with FL tasks as buyers and data owners as sellers. However, there are four important differences. First, the decision variable in [15] is the *model accuracy* level that each data owner contributes to each FL task, whereas our work allocates *training data size* and *CPU-cycle frequency* — concrete, measurable resources that directly determine both training quality and energy consumption. Second, [15] does not incorporate training time into the optimization objective at all, which means their allocation may assign excessive work to slow clients, exacerbating the straggler effect and causing severe training delays in practice. In contrast, we explicitly model training time as a component of server utility so that the allocation actively penalizes assignments that cause long training delays. Third, we provide a convergence rate analysis for the proposed method (Sec. 5.1) while [15] does not. Fourth, [15] does not jointly optimize training data, computational frequency, and training time as we do, limiting its applicability to settings where resource heterogeneity is a key concern.

Chen and Yen [16] also studied many-to-many FL with a regular (single-sided) auction mechanism, allocating training data from clients to servers to maximize social welfare subject to a training time constraint. Our work differs from [16] in the following points. First, we allocate both training data *and* CPU-cycle frequency, while [16] treats CPU-cycle frequency as fixed. Jointly optimizing both resources allows IDA to achieve a better trade-off between model quality, training time, and energy cost. Second, training time in [16] is only a hard constraint (i.e., it must not exceed a threshold), whereas in our formulation training time is embedded in the server utility function as an objective, allowing the allocation to continuously balance model quality against training delay rather than simply capping it. Third, [16] uses a regular auction where only clients submit bids, while we

use a *double* auction where both servers and clients submit bids. Double auction is more appropriate for the many-to-many setting because it allows servers to express their demand-side preferences, leading to more efficient market outcomes. Fourth, [16] uses ADMM as the solver, which requires all participants to share their private parameters with the auctioneer. IDA operates under asymmetric information as each participant solves only its own local subproblem without revealing private utility or cost functions. Fifth, we consider energy cost as an explicit objective component, whereas [16] does not model client energy consumption.

Table 1 summarizes the comparison.

TABLE 1: Comparison of Many-to-Many FL Incentive Mechanism Studies

Feature	[20]	[15]	[16]	Ours
Method	Stackelberg game	Double auction	Regular auction	Double auction
Decision variable	Data size	Acc. level	Data size	Data + CPU freq.
Training time as objective	✓	×	Constr. only	✓
Energy cost as objective	×	✓	×	✓
Asymmetric information	×	✓	×	✓
Proven convergence rate	N/A	×	×	✓

3 System Model and Problem Formulation

Multiple servers and clients participate in our multi-server FL framework (Fig. 2). We use $S = \{1, 2, \cdots, s\}$ and $C = \{1, 2, \cdots, c\}$ to denote the sets of servers and clients in the system, respectively. Servers and clients are self-interested and make participation and contribution decisions independently. Each server has its global model, which needs the client's training resource (training data and computing power) for model training, and is willing to offer monetary rewards to clients for the client's training contributions. The client's contribution is gauged by model quality and training time. On the other hand, clients incur energy costs by allocating training resources to the server's model training. Each client's payoff is the reward it receives minus the energy cost incurred. Since the client's training resources are limited, servers may need to compete with each other for the client's training contributions. We subsequently formulate these factors as server utility functions, client cost functions, and a social welfare maximization problem.

It is worth noting that our resource allocation framework is designed as a *pre-training phase* that determines the allocation plan (i.e., dataset size and CPU-cycle frequency per client per server) before FL training begins. The resulting allocation plan is then applied across all rounds of the FL training process. One might consider reapplying IDA before each FL round to adapt to changing conditions. However, two fundamental challenges arise. First, the accuracy model in our formulation captures the *final* trained model quality as a function of total allocated data, rather than the incremental improvement from a single round, making it unsuitable as a per-round objective without substantial modification. Second, reallocating data sizes and CPU-cycle frequencies across rounds may disrupt the convergence behavior of FedAvg, whose guarantees assume consistent client participation throughout training [1]. Addressing these challenges and extending the framework to fully dynamic, per-round resource allocation is therefore a non-trivial and interesting direction for future work, which we discuss in Sec. 7.

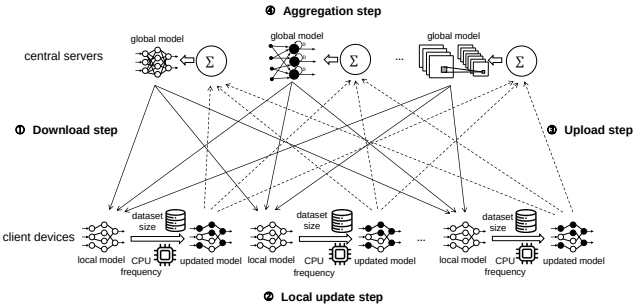


Fig. 2: Multi-server Federated Learning Framework

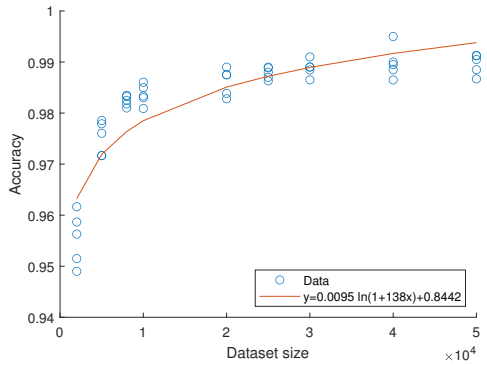
3.1 Server Utility

We consider two factors in defining server utility: model accuracy and training time. Model accuracy is crucial for many applications such as accident warning and prediction. It is well known that model accuracy depends on the quantity and quality of the training data [3]. Nevertheless, we assume that the client's datasets are independent and identically distributed, so the amount of training data positively affects the model accuracy but exhibits diminishing marginal contribution [9]. Let d_{ij}^s be the data size demanded by server $j \in S$ from client device i for j's model training. Our experimental results (shown in Fig. 3) suggested that the expected model accuracy A_j of server j can be defined as

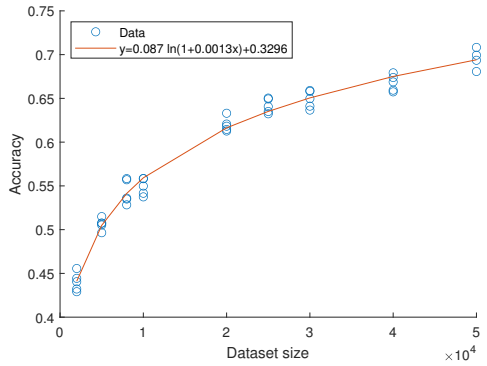
$$A_j = \beta_j \ln\left(1 + \sigma_j \sum_{i \in C} d_{ij}^s\right) + \gamma_j, \tag{1}$$

where $\beta_j > 0$, $\sigma_j > 0$, and $0 \leq \gamma_j < 1$ are all model-specific parameters. Specifically, γ_j represents the baseline accuracy achievable with minimal data (i.e., the intercept of the curve); σ_j controls the rate at which accuracy improves with additional data, capturing the diminishing marginal contribution effect; and β_j scales the overall magnitude of the accuracy gain. These parameters are model-specific and can be estimated empirically by fitting (1) to accuracy measurements collected on a small held-out dataset, as illustrated in Fig. 3 for the MNIST and CIFAR-10 datasets. When $\gamma_j = 0$, (1) is consistent with the equation reported by Zhan et al. [9], where $A_j = \beta_j \log\left(1 + \sigma_j \sum_{i \in C} d_{ij}^s\right)$.

The training time for a server in the FL framework includes the time required for downloading, uploading, and locally updating the model. The time spent by each client i in downloading or uploading the model of server j is the size of the data transmitted divided by the corresponding transmission rate. We assume some model compression approach such as gradient compression [33], which compresses the model into a sparse matrix and transmits only gradients with values larger than a threshold. We assume that the volume transmitted is a fraction of the original model size w_j and that the fraction is $d_{ij}^s r_j \leq 1$, where $r_j \in (0, 1]$ is a model-dependent compression ratio representing the proportion of model parameters retained after



(a) MNIST



(b) CIFAR-10

Fig. 3: Accuracy measurements with fitted curves using two open image datasets: MNIST [31] and CIFAR-10 [32]. We configured 10 clients with datasets of both equal and unequal sizes. We employed FedAvg [1], conducting training over 10 epochs with 300 iterations each, using a batch size of 64.

compression (i.e., a smaller r_j indicates higher compression). The value of r_j depends on the sparsity threshold applied in the compression scheme and can be tuned per model. Thus, the communication time is:

$$t_{ij}^{\text{com}} = \frac{r_j d_{ij}^s w_j}{R_{ji}} + \frac{r_j d_{ij}^s w_j}{R_{ij}}, \tag{2}$$

where R_{ji} and R_{ij} are the downloading and uploading transmission rates, respectively.

The time taken by each client for the local update step depends on its computational resource and dataset size. Let f_i^s be client i's CPU-cycle frequency for local update. The time taken for the local update step can be expressed as follows:

$$t_{ij}^{\text{cmp}} = \frac{d_{ij}^s}{f_i^s}\tau_i, \tag{3}$$

where τ_i is a client-specific coefficient that captures the computing efficiency of client i. The slowest client determines the total training time T_j for server j, which could be the bottleneck of the training process.

$$\begin{aligned} T_j &= \max_{i\in C}\left\{t_{ij}^{\text{com}} + t_{ij}^{\text{cmp}}\right\} \\ &= \max_{i\in C}\left\{\frac{r_j d_{ij}^s w_j}{R_{ji}} + \frac{r_j d_{ij}^s w_j}{R_{ij}} + \frac{d_{ij}^s}{f_i^s}\tau_i\right\}. \end{aligned} \tag{4}$$

Server j's utility for a particular setting of d_{ij}^s and f_i^s for all client i is the value brought in by the model minus the cost associated with the training time.

$$\begin{aligned} &U_j(\{d_{ij}^s\}_{i\in C}, \{f_i^s\}_{i\in C}) \\ &= \rho_j^{\text{acc}} A_j - \rho_j^t T_j \\ &= \rho_j^{\text{acc}}\left(\beta_j \ln\left(1+\sigma_j\sum_{i\in C} d_{ij}^s\right)+\gamma_j\right) \\ &\quad - \rho_j^{\text{time}}\left(\max_{i\in C}\left\{\frac{r_j d_{ij}^s w_j}{R_{ji}} + \frac{r_j d_{ij}^s w_j}{R_{ij}} + \frac{d_{ij}^s}{f_i^s}\tau_i\right\}\right), \end{aligned} \tag{5}$$

where ρ_j^{acc} and ρ_j^{time} represent the weights of the model accuracy and training time, respectively, in the monetary value and cost calculations. An appropriate setup of these two terms allows server j to strike a trade-off between model accuracy and training time.

3.2 Client Cost

Client cost comes from energy consumption for model training, which consists of computation and communication parts. The computation part depends on the amount of training data and CPU-cycle frequency. We assume that client $i \in C$ decides to contribute an amount of training data with size d_{ij}^c and CPU-cycle frequency f_i^c to the model training of each server j. Since servers and clients make decisions independently, d_{ij}^c and f_i^c can differ from d_{ij}^s and f_i^s, respectively.

It is known that energy consumption in CMOS circuits is proportional to the quadratic supply voltage to the chip [34]. Furthermore, it has been observed that the CPU-cycle frequency is approximately linearly proportional to the voltage supply when operating at low voltage limits [34]. As a result, the energy consumption of client i in computation can be expressed as

$$E_i^{\text{cmp}} = \kappa_i \sum_{j\in S} d_{ij}^c f_i^{c2}, \tag{6}$$

where κ_i is a coefficient for the effective capacitance in client i. In reality, E_i^{cmp} contains a static part (i.e., idle power consumption) that is independent of f_i^c and the client's allocation decisions. Including this static term would add a constant offset to the client cost function $C_i(\cdot)$, which in turn uniformly decreases the social welfare objective by a fixed amount. However, since this constant does not affect the gradient of the objective with respect to the decision variables d_{ij}^c and f_i^c, the optimal allocation decisions remain unchanged. Similarly, the pricing and reward rules and all economic properties (individual rationality, budget balance, and incentive compatibility) are unaffected. Therefore, we omit the static part for simplicity without loss of generality.

On the other hand, the energy consumption by client i in communications is dominated by model uploading as the energy consumption in reception is relatively small. It depends on the client's transmission power p_{ij} and the uploading time w_j/R_{ij}:

$$E_i^{\text{com}} = \sum_{j\in S} p_{ij}\left(\frac{r_j d_{ij}^c w_j}{R_{ij}}\right). \tag{7}$$

TABLE 2: Notation

	For Each Server j
C	Set of all clients
$\{d_{ij}^s\}_{i\in C}$	Request of dataset size to each client i
$\{f_i^s\}_{i\in C}$	Request of CPU-cycle frequency to each client i
A_j	Model accuracy
σ_j	Model accuracy parameter in (1)
r_j	Compression ratio: fraction of model parameters transmitted after gradient compression
R_{ij}	Transmission rate from client i
R_{ji}	Transmission rate to client i
w_j	Size of the model parameter (in bits)
t_{ij}^{com}	Communication time with client i
t_{ij}^{cmp}	Computing time by client i
T_j	Total training time
δ_j	Proportion of the slowest training time to the total training time
ρ_j^{acc}	Coefficient from model accuracy to monetary payoff
ρ_j^{time}	Coefficient from total training time to monetary cost
$U_j(\cdot)$	Utility function
	For Each Client i
S	Set of all servers
$\{d_{ij}^c\}_{j\in S}$	Dataset size provided to each server j
f_i^c	Setup of CPU-cycle frequency
τ_i	Coefficient for converting (d_{ij}^s/f_i^s) to t_{ij}^{cmp}
κ_i	Coefficient for the effective capacitance
p_{ij}	Transmission power to server j
D_i	Maximal dataset size
$f_i^{\max}$	Maximal CPU-cycle frequency
E_i^{cmp}	Energy consumption due to computation
E_i^{com}	Energy consumption due to communication
ρ_i^{eng}	Monetary cost per unit of energy consumption
$C_i(\cdot)$	Cost function

The client cost function counts the energy consumption in computation and communication for the FL process:

$$\begin{aligned} &C_i\left(\{d_{ij}^c\}_{j\in S}, f_i^c\right) \\ =&\rho_i^{\text{eng}}\left(E_i^{\text{cmp}} + E_i^{\text{com}}\right) \\ =&\rho_i^{\text{eng}}\left(\kappa_i \sum_{j\in S} d_{ij}^c f_i^{c2} + \sum_{j\in S} p_{ij}\left(\frac{r_j d_{ij}^c w_j}{R_{ij}}\right)\right), \end{aligned} \tag{8}$$

where ρ_i^{eng} is the monetary cost per unit of energy in client i.

Table 2 lists the notation used in this paper.

3.3 Objective: Social Welfare Maximization

As discussed earlier, servers improve global model accuracy and training efficiency by acquiring more training data and a higher CPU-cycle frequency from clients. By contrast, clients reduce the number of training data and CPU-cycle frequencies to minimize energy costs. These objectives create a conflict of interest between the central servers and the client devices. From a social perspective, the optimal solution lies in maximizing social welfare through efficient resource trading. To achieve this, a centralized market broker plays a crucial role in determining the optimal sizes of training data d_{ij}^s and d_{ij}^c, as well as the CPU-cycle frequencies f_i^s and f_i^c. We assume the centralized market broker maintains a neutral stance in mediating these resource allocations. This optimization process involves solving a social welfare maximization (SWM) problem formulated as follows:

Definition 1. *The social welfare maximization (SWM) problem is*

$$\max_{\{d_{ij}^s, f_i^s, d_{ij}^c, f_i^c\}} \quad \sum_{j\in S} U_j(\{d_{ij}^s\}_{i\in C}, \{f_i^s\}_{i\in C}) - \sum_{i\in C} C_i(\{d_{ij}^c\}_{j\in S}, f_i^c), \tag{9}$$

$$\text{s.t.} \quad d_{ij}^c \le D_i, \forall i \in C, j \in S, \tag{10}$$

$$f_i^c \le f_i^{\max}, \forall i \in C, \tag{11}$$

$$d_{ij}^s \le d_{ij}^c, \forall i \in C, j \in S, \tag{12}$$

$$f_i^s \le f_i^c, \forall i \in C, \tag{13}$$

$$d_{ij}^s, f_i^s, d_{ij}^c, f_i^c \ge 0, \forall i \in C, j \in S, \tag{14}$$

where D_i and f_i^{max} are the maximum values of d_{ij}^c and f_i^c, respectively.

Eqs. (10) and (11) ensure that clients operate within their resource limits. Eq. (12) ensures that the dataset size supplied by each client device satisfies the corresponding dataset size requested by the central server, while constraint (13) guarantees that the CPU-cycle frequency supplied by each client device meets the corresponding CPU-cycle frequency requested by the central server. Eq. (14) ensures that all the supplies and provisions are non-negative.

As the maximum operation in (5) poses challenges, we introduce the use of δ_j, the ratio of the slowest training time to the total training time, to replace the maximum operation. Specifically,

$$\begin{aligned} T_j &= \max_{i\in C}\left\{t_{ij}^{\text{com}} + t_{ij}^{\text{cmp}}\right\} \\ &\approx \delta_j \sum_{i\in C}\left\{t_{ij}^{\text{com}} + t_{ij}^{\text{cmp}}\right\}. \end{aligned} \tag{15}$$

Thus, server j's utility function (5) can be expressed as

$$\begin{aligned} &U_j(\{d_{ij}^s\}_{i\in C}, \{f_i^s\}_{i\in C}) \\ \approx&\rho_j^{\text{acc}}\left(\ln\left(1+\sigma_j\sum_{i\in C} d_{ij}^s\right)\right) - \rho_j^{\text{time}}\left(\delta_j\sum_{i\in C}\left(\frac{r_j d_{ij}^s w_j}{R_{ji}} + \frac{r_j d_{ij}^s w_j}{R_{ij}} + \frac{d_{ij}^s}{f_i^s}\tau_i\right)\right). \end{aligned} \tag{16}$$

It is worth noting that, with this definition, the objective function of the SWM problem is concave, and the constraint set is both compact and convex. Refer to the proofs in the Appendix.

The quality of this approximation depends on δ_j, which represents how dominant the slowest client is relative to the total training time. When clients are homogeneous, $\delta_j \approx 1/|C|$ and the approximation is exact by symmetry. When one client is significantly slower than the others (severe straggler effect), $\delta_j \to 1$ and the approximation also becomes tight, since the maximum is well-approximated by the sum scaled by a factor close to 1. The approximation is least accurate when δ_j takes intermediate values and client times are moderately heterogeneous. In our simulations we set $\delta_j = 0.3$, which corresponds to a moderately heterogeneous scenario. We assess sensitivity to this parameter in Sec. 6.5.

4 Proposed Approach

4.1 Social Welfare Optimum

The SWM problem possesses an optimal solution that can be precisely characterized using the necessary and sufficient

Karush-Kuhn-Tucker (KKT) conditions. To proceed, we relax the constraints and define the Lagrangian function of the SWM problem as follows:

$$\begin{aligned}
&\mathcal{L}_1(\{d_{ij}^s\}_{i\in C,j\in S},\{f_i^s\}_{i\in C},\{d_{ij}^c\}_{i\in C,j\in S},\{f_i^c\}_{i\in C},\\
&\quad\{\mu_{ij}\}_{i\in C,j\in S},\{\theta_i\}_{i\in C},\{\nu_{ij}\}_{i\in C,j\in S},\{\omega_i\}_{i\in C})\\
&=\sum_{j\in S}U_j(\{d_{ij}^s\}_{i\in C},\{f_i^s\}_{i\in C})-\sum_{i\in C}C_i(\{d_{ij}^c\}_{j\in S},f_i^c)\\
&\quad-\sum_{i\in C}\sum_{j\in S}\mu_{ij}(d_{ij}^c-D_i)-\sum_{i\in C}\theta_i(f_i^c-f_i^{\max})\\
&\quad-\sum_{i\in C}\sum_{j\in S}\nu_{ij}(d_{ij}^s-d_{ij}^c)-\sum_{i\in C}\omega_i(f_i^s-f_i^c),
\end{aligned}\tag{17}$$

where μ_{ij}, θ_i, ν_{ij}, and ω_i are Lagrange multipliers associated with the constraints and must be non-negative. The KKT conditions of the SWM problem that yield the optimal primal variables $\hat{d}_{ij}^s$, $\hat{f}_i^s$, $\hat{d}_{ij}^c$ and $\hat{f}_i^c$ and the optimal dual variables $\hat{\mu}_{ij}$, $\hat{\theta}_i$, $\hat{\nu}_{ij}$ and $\hat{\boldsymbol{\omega}}_i$ are given by the following set of equations:

$$\frac{\partial U_j(\{\hat{d}_{ij}^s\}_{i\in C},\{\hat{f}_i^s\}_{i\in C})}{\partial \hat{d}_{ij}^s}-\hat{\nu}_{ij}=0,\tag{18}$$

$$\frac{\partial U_j(\{\hat{d}_{ij}^s\}_{i\in C},\{\hat{f}_i^s\}_{i\in C})}{\partial \hat{f}_i^s}-\hat{\omega}_i=0,\tag{19}$$

$$\frac{\partial C_i(\{\hat{d}_{ij}^c\}_{j\in S},\hat{f}_i^c)}{\partial \hat{d}_{ij}^c}-\hat{\mu}_{ij}+\hat{\nu}_{ij}=0,\tag{20}$$

$$\frac{\partial C_i(\{\hat{d}_{ij}^c\}_{j\in S},\hat{f}_i^c)}{\partial \hat{f}_i^c}-\hat{\theta}_i+\hat{\omega}_i=0,\tag{21}$$

$$\sum_{i\in C}\sum_{j\in S}\hat{\mu}_{ij}(\hat{d}_{ij}^c-D_i)=0,\tag{22}$$

$$\sum_{i\in C}\hat{\theta}_i(\hat{f}_i^c-\hat{f}_i^{\max})=0,\tag{23}$$

$$\sum_{i\in C}\sum_{j\in S}\hat{\nu}_{ij}(\hat{d}_{ij}^c-\hat{d}_{ij}^s)=0,\tag{24}$$

$$\sum_{i\in C}\hat{\omega}_i(\hat{f}_i^c-\hat{f}_i^s)=0,\tag{25}$$

$$\hat{d}_{ij}^s,\hat{f}_i^s,\hat{d}_{ij}^c,\hat{f}_i^c\geq 0,\forall i\in C,j\in S,\tag{26}$$

$$\hat{\mu}_{ij},\hat{\theta}_i,\hat{\nu}_{ij},\hat{\omega}_i\geq 0,\forall i\in C,j\in S.\tag{27}$$

According to the equations from (18) to (21), the optimal solutions of d_{ij}^s, f_i^s, d_{ij}^c and f_i^c satisfy

$$\begin{cases}
\frac{\partial U_j\left(\{\hat{d}_{ij}^s\}_{i\in C},\{\hat{f}_i^s\}_{i\in C}\right)}{\partial \hat{d}_{ij}^s}=\hat{\nu}_{ij},\\
\frac{\partial U_j\left(\{\hat{d}_{ij}^s\}_{i\in C},\{\hat{f}_i^s\}_{i\in C}\right)}{\partial \hat{f}_i^s}=\hat{\omega}_i,\\
\frac{\partial C_i\left(\{\hat{d}_{ij}^c\}_{j\in S},\hat{f}_i^c\right)}{\partial \hat{d}_{ij}^c}=\hat{\mu}_{ij}-\hat{\nu}_{ij},\\
\frac{\partial C_i\left(\{\hat{d}_{ij}^c\}_{j\in S},\hat{f}_i^c\right)}{\partial \hat{f}_i^c}=\hat{\theta}_i-\hat{\omega}_i.
\end{cases}\tag{28}$$

However, the broker cannot derive the optimal solution to the SWM problem in practice. This is because evaluating the server utility function set $\{U_j(\{d_{ij}^s\}_{i\in C},\{f_i^s\}_{i\in C})\mid j\in S\}$ and the client cost function set $\{C_i(\{d_{ij}^c\}_{i\in S},f_i^c)\mid i\in C\}$ need local information known only to the servers and clients, respectively. The broker lacks access to this information for various reasons, such as privacy and security concerns [35]. Therefore, it is necessary to design an incentive mechanism that addresses the information asymmetry by encouraging servers and clients to truthfully report their demands and supplies (in terms of dataset sizes and CPU-cycle frequencies), respectively.

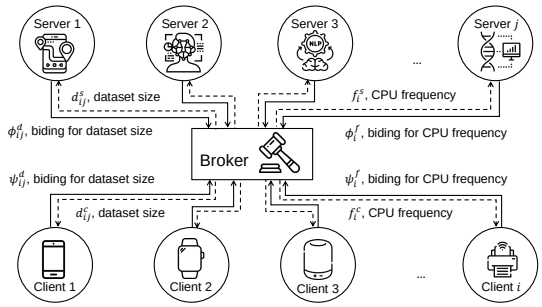


Fig. 4: An Overview of the Iterative Double Auction Framework

4.2 Iterative Double Auction (IDA)

To address the issue of asymmetric information, we propose using an iterative double auction (IDA) algorithm [36] that allows the broker to extract hidden information from sellers and buyers. As depicted in Fig. 4, the broker assists multiple servers and clients to adjust their bids iteratively to maximize individual payoffs and general social welfare. Specifically, the broker solves an alternative optimization problem to determine the optimal values of d_{ij}^s, f_i^s, d_{ij}^c, and f_i^c. The values are then combined with appropriate pricing rules, including payments for servers and rewards for clients, to achieve the optimal solution for the original SWM problem. We assume that all servers and clients comply with their constraints and refrain from submitting bids that exceed their capabilities. We also assume that all servers and clients are *price-takers*, meaning that they submit bids as their best responses to the broker announcements; bidders do not anticipate the impact of their bids on the results in the next iteration (in fact, they lack information for making such an anticipation). In the following subsections, we will elaborate on the allocation rule, the optimal pricing rules, and the implementation of the IDA algorithm.

4.3 Resource Allocation

The IDA algorithm consists of two stages. In the first stage, each server $j\in S$ submits a bid $B_j^s=(\{\phi_{ij}^d\}_{i\in C},\{\phi_i^f\}_{i\in C})$ to the broker as its demands of training data and computing power, where ϕ_{ij}^d and ϕ_i^f are the dataset size and the CPU-cycle frequency it demands from client i. Meanwhile, each client $i\in C$ submits a bid $B_i^c=(\{\psi_{ij}^d\}_{j\in S},\psi_i^f)$ to the broker as its supply of data and computational resource, where ψ_{ij}^d and ψ_i^f are the dataset size and CPU-cycle frequency, respectively, it supplies to server j. These bids are collectively the input to the broker for resource allocation and pricing rules. However, these bids may not reflect actual demands and supplies.

With bids from all servers and clients, the broker performs resource allocation, i.e., determining $\{d_{ij}^s\}_{i\in C,j\in S}$, $\{f_i^s\}_{i\in C}$, $\{d_{ij}^c\}_{i\in C,j\in S}$, and $\{f_i^c\}_{i\in C}$. Inspired by previous studies [37], we use the logarithmic function along with the

quadratic function to formulate the allocation rule, which can obtain the same concave and convex properties of the server utility function and the client cost function, respectively.

Definition 2. *The broker allocation (BA) problem is to*

$$\max_{\{d_{ij}^s, f_i^s, d_{ij}^c, f_i^c\}} \quad \alpha \sum_{j\in S}\sum_{i\in C}\left(\phi_{ij}^d \ln d_{ij}^s - \frac{1}{2}\psi_{ij}^d {d_{ij}^c}^2\right) + (1-\alpha)\sum_{i\in C}\left(\phi_i^f \ln f_i^s - \frac{1}{2}\psi_i^f {f_i^c}^2\right), \tag{29}$$

where $\alpha \in (0,1)$ is a weighting factor for broker, subject to (10) *to* (14).

The BA problem shares the constraints with the SWM problem but features a distinct, strictly concave objective function, ensuring a unique optimal solution. Since the BA problem is also a convex optimization problem, we can use the Lagrange function and the KKT conditions for optimization. Accordingly, we define the corresponding Lagrange function of the BA problem as follows:

$$\begin{aligned}
&\mathcal{L}_2\left(\{d_{ij}^s\}_{i\in C, j\in S}, \{f_i^s\}_{i\in C}, \{d_{ij}^c\}_{i\in C, j\in S}, \{f_i^c\}_{i\in C}, \right.\\
&\quad \left.\{\mu'_{ij}\}_{i\in C, j\in S}, \{\theta'_i\}_{i\in C}, \{\nu'_{ij}\}_{i\in C, j\in S}, \{\omega'_i\}_{i\in C}\right)\\
&= \alpha \sum_{j\in S}\sum_{i\in C}\left(\phi_{ij}^d \ln d_{ij}^s - \frac{1}{2}\psi_{ij}^d {d_{ij}^c}^2\right) + (1-\alpha)\sum_{i\in C}\left(\phi_i^f \ln f_i^s - \frac{1}{2}\psi_i^f {f_i^c}^2\right)\\
&\quad - \sum_{j\in S}\sum_{i\in C}\mu'_{ij}\left(d_{ij}^c - D_i\right) - \sum_{i\in C}\theta'_i\left(f_i^c - f_i^{\max}\right)\\
&\quad - \sum_{j\in S}\sum_{i\in C}\nu'_{ij}\left(d_{ij}^s - d_{ij}^c\right) - \sum_{i\in C}\omega'_i\left(f_i^s - f_i^c\right),
\end{aligned} \tag{30}$$

where μ'_{ij}, θ'_i, ν'_{ij}, and ω'_i are Lagrange multipliers associated with the constraints and must be non-negative. By applying the KKT conditions, we obtain the optimal values of the primal variables $\hat{d}_{ij}^s, \hat{f}_i^s, \hat{d}_{ij}^c, \hat{f}_i^c$ and the optimal values of the dual variables $\hat{\mu}'_{ij}$, $\hat{\theta}'_i$, $\hat{\nu}'_{ij}$, $\hat{\omega}'_i$ through the following set of equations:

$$\alpha\frac{\phi_{ij}^d}{\hat{d}_{ij}^s} - \hat{\nu}'_{ij} = 0, \tag{31}$$

$$(1-\alpha)\frac{\phi_i^f}{\hat{f}_i^s} - \hat{\omega}'_i = 0, \tag{32}$$

$$\alpha\psi_{ij}^d \hat{d}_{ij}^c - \hat{\mu}'_{ij} + \hat{\nu}'_{ij} = 0, \tag{33}$$

$$(1-\alpha)\,\psi_i^f \hat{f}_i^c - \hat{\theta}'_i + \hat{\omega}'_i = 0, \tag{34}$$

$$\sum_{j\in S}\sum_{i\in C}\hat{\mu}'_{ij}\left(\hat{d}_{ij}^c - D_i\right) = 0, \tag{35}$$

$$\sum_{i\in C}\hat{\theta}'_i\left(\hat{f}_i^c - \hat{f}_i^{\max}\right) = 0, \tag{36}$$

$$\sum_{j\in S}\sum_{i\in C}\hat{\nu}'_{ij}\left(\hat{d}_{ij}^c - \hat{d}_{ij}^s\right) = 0, \tag{37}$$

$$\sum_{i\in C}\hat{\omega}'_i\left(\hat{f}_i^c - \hat{f}_i^s\right) = 0, \tag{38}$$

$$\hat{d}_{ij}^s, \hat{f}_i^s, \hat{d}_{ij}^c, \hat{f}_i^c \geq 0, \forall i \in C, j \in S, \tag{39}$$

$$\hat{\mu}'_{ij}, \hat{\theta}'_i, \hat{\nu}'_{ij}, \hat{\omega}'_i \geq 0, \forall i \in C, j \in S. \tag{40}$$

By (31) and (32), the optimal allocation for the server side can be obtained as follows:

$$\begin{cases} \hat{d}_{ij}^s = \alpha\frac{\phi_{ij}^d}{\hat{\nu}'_{ij}}, \\ \hat{f}_i^s = (1-\alpha)\,\frac{\phi_i^f}{\hat{\omega}'_i}. \end{cases} \tag{41}$$

Similarly, by (33) and (34), the optimal dataset size and CPU-cycle frequency for the client side can be derived as follows:

$$\begin{cases} \hat{d}_{ij}^c = \frac{\hat{\mu}'_{ij} - \hat{\nu}'_{ij}}{\alpha\psi_{ij}^d}, \\ \hat{f}_i^c = \frac{\hat{\theta}'_i - \hat{\omega}'_i}{(1-\alpha)\psi_i^f}. \end{cases} \tag{42}$$

4.4 Optimal Pricing Rules

When solving the SWM or the BA problem, we do not consider the payments from servers and the rewards to clients. However, clients are willing to participate in the training process *only if* they receive higher rewards to cover their training costs. Accordingly, the broker needs to collect server payments to offer clients rewards. Optimal pricing determines the server's payments and client's rewards that maximize their respective payoffs while aligning with the optimal solution to the SWM problem.

First, we make the optimal solution to the BA problem align with the solution to the SWM problem. Comparing the two sets of KKT conditions of the SWM problem and the BA problem, we observe that (18) to (21) differ from (31) to (34), while the remaining conditions are identical. Thus, the optimal solution to the BA problem is also the optimal solution to the SWM problem if the server's bids meet the following conditions:

$$\begin{cases} \phi_{ij}^d = \frac{\hat{d}_{ij}^s}{\alpha}\frac{\partial U_j\left(\{\hat{d}_{ij}^s\}_{i\in C}, \{\hat{f}_i^s\}_{i\in C}\right)}{\partial \hat{d}_{ij}^s}, \\ \phi_i^f = \frac{\hat{f}_i^s}{1-\alpha}\frac{\partial U_j\left(\{\hat{d}_{ij}^s\}_{i\in C}, \{\hat{f}_i^s\}_{i\in C}\right)}{\partial \hat{f}_i^s}, \end{cases} \tag{43}$$

and the client's bids meet the following conditions:

$$\begin{cases} \psi_{ij}^d = \frac{1}{\alpha\hat{d}_{ij}^c}\frac{\partial C_i\left(\{\hat{d}_{ij}^c\}_{j\in S}, \hat{f}_i^c\right)}{\partial \hat{d}_{ij}^c}, \\ \psi_i^f = \frac{1}{(1-\alpha)\hat{f}_i^c}\frac{\partial C_i\left(\{\hat{d}_{ij}^c\}_{j\in S}, \hat{f}_i^c\right)}{\partial \hat{f}_i^c}. \end{cases} \tag{44}$$

Next, we determine the payment rule to make (43) the optimal server bids that maximize the server's payoffs defined below.

Definition 3. *The server payoff maximization (SPM) problem is defined as follows.*

$$\max_{\{\phi_{ij}^d, \phi_i^f\}} \quad U_j\left(\{d_{ij}^s\}_{i\in C}, \{f_i^s\}_{i\in C}\right) - P_j\left(\{\phi_{ij}^d\}_{i\in C}, \{\phi_i^f\}_{i\in C}\right), \tag{45}$$

where $P_j(B_j^s)$ is server j's payment with respect to B_j^s, such that $\phi_{ij}^d, \phi_i^f \geq 0, \forall i \in C, j \in S$.

The utility part $U_j(\cdot)$ is determined by the broker's resource allocation rule as in (41). Therefore, server j can only lower $P_j(\cdot)$ by setting up B_j^s. Let $\hat{B}_j^s = (\{\hat{\phi}_{ij}^d\}_{i\in C}, \{\hat{\phi}_i^f\}_{i\in C})$ be the optimal setting of B_j^s that maximizes j's payoff. To find $\hat{B}_j$, we begin by taking the first-order derivatives of the objective function (45) with respect to ϕ_{ij}^d and ϕ_i^f. Setting the

derivatives to zeros leads to the following two optimality conditions:

$$\frac{\partial P_j\left(\{\hat{\phi}_{ij}^d\}_{i\in C},\{\hat{\phi}_i^f\}_{i\in C}\right)}{\partial \hat{\phi}_{ij}^d}=\frac{\partial U_j\left(\{\hat{d}_{ij}^s\}_{i\in C},\{\hat{f}_i^s\}_{i\in C}\right)}{\partial \hat{d}_{ij}^s}\cdot\frac{\partial \hat{d}_{ij}^s}{\partial \hat{\phi}_{ij}^d}, \quad (46)$$

$$\frac{\partial P_j\left(\{\hat{\phi}_{ij}^d\}_{i\in C},\{\hat{\phi}_i^f\}_{i\in C}\right)}{\partial \hat{\phi}_i^f}=\frac{\partial U_j\left(\{\hat{d}_{ij}^s\}_{i\in C},\{\hat{f}_i^s\}_{i\in C}\right)}{\partial \hat{f}_i^s}\cdot\frac{\partial \hat{f}_i^s}{\partial \hat{\phi}_i^f}. \quad (47)$$

By applying the optimal solution (41) derived from the BA problem and substituting it with the bids in (43), we obtain the following expressions:

$$\frac{\partial P_j\left(\{\hat{\phi}_{ij}^d\}_{i\in C},\{\hat{\phi}_i^f\}_{i\in C}\right)}{\partial \hat{\phi}_{ij}^d}=\alpha\frac{\phi_{ij}^d}{\hat{d}_{ij}^s}\cdot\alpha\frac{1}{\hat{\nu}'_{ij}}=\alpha, \quad (48)$$

$$\frac{\partial P_j\left(\{\hat{\phi}_{ij}^d\}_{i\in C},\{\hat{\phi}_i^f\}_{i\in C}\right)}{\partial \hat{\phi}_i^f}=(1-\alpha)\frac{\phi_i^f}{\hat{f}_i^s}\cdot(1-\alpha)\frac{1}{\hat{\omega}'_i}=1-\alpha. \quad (49)$$

Then, the payment rule for server j can be expressed as follows:

$$P_j\left(\{\phi_{ij}^d\}_{i\in C},\{\phi_i^f\}_{i\in C}\right)=\alpha\sum_{i\in C}\phi_{ij}^d+(1-\alpha)\sum_{i\in C}\phi_i^f. \quad (50)$$

Finally, we determine the reward rule to make (44) the optimal client bids that maximize the client's payoffs. Let $Q_i(B_i^c)$ be the reward to client i when client i submits $B_i^c=(\{\psi_{ij}^d\}_{j\in S},\psi_i^f)$. Each client aims to maximize its payoff by setting up an optimal bid, which is defined as the client payoff maximization (CPM) problem.

Definition 4. *The client payoff maximization (CPM) problem is defined as follows.*

$$\max_{\{\psi_{ij}^d,\psi_i^f\}} \quad Q_i\left(\{\psi_{ij}^d\}_{j\in S},\psi_i^f\right)-C_i\left(\{d_{ij}^c\}_{j\in S},f_i^c\right), \quad (51)$$

such that $\psi_{ij}^d,\ \psi_i^f\geq 0, \forall i\in C, j\in S$.

Since $C_i(\cdot)$ is determined by the broker's resource allocation rule in (42), client i can only find the optimal bid B_i^c that maximizes $Q_i(B_i^c)$. This is done by taking the first-order derivatives of the objective function (51) with respect to ψ_{ij}^d and ψ_i^f, respectively, to obtain the following two optimality conditions:

$$\frac{\partial Q_i\left(\{\hat{\psi}_{ij}^d\}_{j\in S},\hat{\psi}_i^f\right)}{\partial \hat{\psi}_{ij}^d}=\frac{\partial C_i\left(\{d_{ij}^c\}_{j\in S},\hat{f}_i^c\right)}{\partial \hat{d}_{ij}^c}\cdot\frac{\partial \hat{d}_{ij}^c}{\partial \hat{\psi}_{ij}^d}, \quad (52)$$

$$\frac{\partial Q_i\left(\{\hat{\psi}_{ij}^d\}_{j\in S},\hat{\psi}_i^f\right)}{\partial \hat{\psi}_i^f}=\frac{\partial C_i\left(\{d_{ij}^c\}_{j\in S},\hat{f}_i^c\right)}{\partial \hat{f}_i^c}\cdot\frac{\partial \hat{f}_i^c}{\partial \hat{\psi}_i^f}. \quad (53)$$

By applying the optimal solution (42) from the BA problem and substituting it with the bids in (44), we obtain the following expressions:

$$\frac{\partial Q_i\left(\{\hat{\psi}_{ij}^d\}_{j\in S},\hat{\psi}_i^f\right)}{\partial \hat{\psi}_{ij}^d}=\alpha\hat{d}_{ij}^c\psi_{ij}^d\cdot\frac{-\left(\hat{\mu}'_{ij}-\hat{\nu}'_{ij}\right)}{\alpha{\psi_{ij}^d}^2}=\frac{-\left(\hat{\mu}'_{ij}-\hat{\nu}'_{ij}\right)^2}{\alpha{\psi_{ij}^d}^2}, \quad (54)$$

$$\frac{\partial Q_i\left(\{\hat{\psi}_{ij}^d\}_{j\in S},\hat{\psi}_i^f\right)}{\partial \hat{\psi}_i^f}=(1-\alpha)\hat{f}_i^c\psi_i^f\cdot\frac{-\left(\hat{\theta}'_i-\hat{\omega}'_i\right)}{(1-\alpha){\psi_i^f}^2}=\frac{-\left(\hat{\theta}'_i-\hat{\omega}'_i\right)^2}{(1-\alpha){\psi_i^f}^2}. \quad (55)$$

Then, the reward rule for client i is defined as follows:

$$Q_i\left(\{\psi_{ij}^d\}_{j\in S},\psi_i^f\right)=\sum_{j\in S}\frac{\left(\hat{\mu}'_{ij}-\hat{\nu}'_{ij}\right)^2}{\alpha\psi_{ij}^d}+\frac{(\hat{\theta}'_i-\hat{\omega}'_i)^2}{(1-\alpha)\psi_i^f}. \quad (56)$$

4.5 The Algorithm

After deriving the allocation rules (41) (42), along with the payment rule (50) and the reward rule (56), the servers and clients can determine their own optimal bids. This enables the broker to achieve market equilibrium, maximizing social welfare in a single round, given complete market information that includes server utility functions (5) and client cost functions (8). However, due to the information asymmetry mentioned earlier, we can only employ an iterative algorithm that gradually fine-tunes the market dynamics and allows for convergence toward the desired outcome.

In each iteration of IDA, the broker announces the updated dual variables, specifically the Lagrange multipliers $\mu'_{ij},\theta'_i,\nu'_{ij}$ and ω'_i. Subsequently, servers and clients determine their optimal bids by solving their respective SPM and CPM problems. More specifically, each server j first turns $U_j(\cdot)$ into a function of B_j^s by replacing d_{ij}^s and f_i^s in (5) with $\hat{d}_{ij}^s$ and $\hat{f}_i^s$ in (41), respectively. Server j then combines $U_j(B_j^s)$ with $P_j(B_j^s)$ given by (50) to solve the SPM problem. The client part is similar. The optimal bids are then submitted back to the broker to determine the optimal resource allocation by (41) and (42) and uses the results to update the Lagrange multipliers $\mu'_{ij},\theta'_i,\nu'_{ij}$ and ω'_i following the primal-dual Lagrangian decomposition approach [37]. Specifically, the broker performs the updates using a gradient descent technique as follows:

$$\begin{aligned}{\mu'_{ij}}^{(t)}&=\max\left\{{\mu'_{ij}}^{(t-1)}-\eta\frac{\partial\mathcal{L}_2}{\partial\mu'_{ij}},0\right\}\\&=\max\left\{{\mu'_{ij}}^{(t-1)}+\eta\left(d_{ij}^c-D_i\right),0\right\},\forall i\in C,j\in S,\end{aligned} \quad (57)$$

$$\begin{aligned}\theta_i'^{(t)}&=\max\left\{\theta_i'^{(t-1)}-\eta\frac{\partial\mathcal{L}_2}{\partial\theta'_i},0\right\}\\&=\max\left\{\theta_i'^{(t-1)}+\eta\left(f_i^c-f_i^{\max}\right),0\right\},\forall i\in C,\end{aligned} \quad (58)$$

$$\begin{aligned}{\nu'_{ij}}^{(t)}&=\max\left\{{\nu'_{ij}}^{(t-1)}-\eta\frac{\partial\mathcal{L}_2}{\partial\nu'_{ij}},0\right\}\\&=\max\left\{{\nu'_{ij}}^{(t-1)}+\eta\left(d_{ij}^s-d_{ij}^c\right),0\right\},\forall i\in C,j\in S,\end{aligned} \quad (59)$$

$$\begin{aligned}\omega_i'^{(t)}&=\max\left\{\omega_i'^{(t-1)}-\eta\frac{\partial\mathcal{L}_2}{\partial\omega'_i},0\right\}\\&=\max\left\{\omega_i'^{(t-1)}+\eta\left(f_i^s-f_i^c\right),0\right\},\forall i\in C,\end{aligned} \quad (60)$$

where η represents the step size and t is the iteration count. The broker stops the iteration process when all the submitted bids do not significantly differ from the previously submitted ones (i.e., the difference is within some small ratio ϵ). Otherwise, the broker commences another iteration by announcing the updated Lagrange multipliers. The above process is shown in Algorithm 1.

Algorithm 1 Iterative Double Auction (IDA)

Initialize $d_{ij}^s, f_i^s, d_{ij}^c, f_i^c$.
Initialize Lagrangian multiplier ${\mu'_{ij}}^{(0)}, {\theta'_i}^{(0)}, {\nu'_{ij}}^{(0)}, {\omega'_i}^{(0)}$.
Initialize iteration count $t = 0$.
repeat
  $t \leftarrow t + 1$
  Broker announces ${\mu'_{ij}}^{(t-1)}, {\theta'_i}^{(t-1)}, {\nu'_{ij}}^{(t-1)}, {\omega'_i}^{(t-1)}$.
  Each server $j \in S$ solves the SPM problem and sends $B_j^s = (\{\hat{\phi}_{ij}^d\}_{i\in C}, \{\hat{\phi}_i^f\}_{i\in C})$ to the broker
  Each client $i \in C$ solves the CPM problem and sends $B_i^c = (\{\hat{\psi}_{ij}^d\}_{j\in S}, \hat{\psi}_i^f)$ to the broker
  Broker computes $\hat{d}_{ij}^s, \hat{f}_i^s, \hat{d}_{ij}^c, \hat{f}_i^c$ by (41) and (42).
  Broker updates ${\mu'_{ij}}^{(t)}, {\theta'_i}^{(t)}, {\nu'_{ij}}^{(t)}, {\omega'_i}^{(t)}$ by (57) to (60).
until $|(x^{(t)} - x^{(t-1)})/x^{(t-1)}| \leq \epsilon$ for $x = \hat{\phi}_{ij}^d, \hat{\phi}_i^f, \hat{\psi}_{ij}^d, \hat{\psi}_i^f$

4.6 Computational and Communication Overhead

Each iteration of IDA involves $s + c$ independent optimization subproblems: one SPM problem per server and one CPM problem per client, each solved locally by the respective participant. Each subproblem is a smooth, unconstrained optimization over a small number of variables (of order c or s), solvable efficiently by standard gradient-based methods. The broker's update step requires only element-wise gradient descent on the Lagrange multipliers, which takes $O(cs)$ operations.

Since all servers (resp. clients) solve their SPM (resp. CPM) problems in parallel and both groups operate simultaneously, the per-iteration parallel time for the subproblems is $O(\max(P_{\text{SPM}}, P_{\text{CPM}}))$, where P_{SPM} and P_{CPM} are the costs of solving a single SPM and CPM problem, respectively. The overall per-iteration parallel time complexity is therefore $O(\max(P_{\text{SPM}}, P_{\text{CPM}}) + cs)$. The subproblem component is independent of the number of servers s and clients c. The broker's update step scales as $O(cs)$ with the number of participants.

The communication overhead per iteration consists of: (1) the broker sending each server j the $2c$ multipliers ($\{\nu'_{ij}\}_{i\in C}$ and $\{\omega'_i\}_{i\in C}$) it needs to solve its SPM problem, and sending each client i the $2s + 2$ multipliers ($\{\mu'_{ij}\}_{j\in S}$, $\{\nu'_{ij}\}_{j\in S}$, θ'_i, and ω'_i) it needs to solve its CPM problem, amounting to $2cs + (2s + 2)c = 4cs + 2c$ values in total; (2) each server j submitting its bid B_j^s of size $2c$ scalars; and (3) each client i submitting its bid B_i^c of size $s + 1$ scalars. The total messages exchanged per iteration are therefore $O(cs)$, which is the same order as the number of decision variables in the SWM problem and cannot be reduced in general without sacrificing convergence. In our simulation with $s = 4$ servers and $c = 4$ to 12 clients, IDA converged in 3 to 8 iterations, resulting in a modest total communication cost that is well within the capacity of typical network infrastructures.

5 Properties of the Proposed Approach

5.1 Convergence Analysis

In this subsection, we derive the convergence rate of the IDA algorithm. By the gradient updating rule (57 – 60), the dynamics of the dual variables are defined as follows:

$$\dot{\mu}'_{ij}(t) = {\mu'_{ij}}^{(t)} - {\mu'_{ij}}^{(t-1)} = \begin{cases} \eta\left(d_{ij}^c - D_i\right), & \text{if } {\mu'_{ij}}^{(t)} > 0, \\ 0, & \text{if } {\mu'_{ij}}^{(t)} = 0, \end{cases} \tag{61}$$

$$\dot{\theta}'_i(t) = {\theta'_i}^{(t)} - {\theta'_i}^{(t-1)} = \begin{cases} \eta\left(f_i^c - f_i^{\max}\right), & \text{if } {\theta'_i}^{(t)} > 0, \\ 0, & \text{if } {\theta'_i}^{(t)} = 0, \end{cases} \tag{62}$$

$$\dot{\nu}'_{ij}(t) = {\nu'_{ij}}^{(t)} - {\nu'_{ij}}^{(t-1)} = \begin{cases} \eta\left(d_{ij}^s - d_{ij}^c\right), & \text{if } {\nu'_{ij}}^{(t)} > 0, \\ 0, & \text{if } {\nu'_{ij}}^{(t)} = 0, \end{cases} \tag{63}$$

$$\dot{\omega}'_i(t) = {\omega'_i}^{(t)} - {\omega'_i}^{(t-1)} = \begin{cases} \eta\left(f_i^s - f_i^c\right), & \text{if } {\omega'_i}^{(t)} > 0, \\ 0, & \text{if } {\omega'_i}^{(t)} = 0. \end{cases} \tag{64}$$

We use the Lyapunov function to quantify the difference between the optimal solution and the solution at iteration t:

$$\begin{aligned} \mathcal{V}(\boldsymbol{\mu}', \boldsymbol{\theta}', \boldsymbol{\nu}', \boldsymbol{\omega}') = \frac{1}{2}\sum_{j\in S}\sum_{i\in C}\left(\mu'_{ij} - \hat{\mu}'_{ij}\right)^2 + \frac{1}{2}\sum_{i\in C}\left(\theta'_i - \hat{\theta}'_i\right)^2 \\ + \frac{1}{2}\sum_{j\in S}\sum_{i\in C}\left(\nu'_{ij} - \hat{\nu}'_{ij}\right)^2 + \frac{1}{2}\sum_{i\in C}\left(\omega'_i - \hat{\omega}'_i\right)^2. \end{aligned} \tag{65}$$

We derive the convergence rate of the IDA algorithm by analyzing the one-step improvement of the Lyapunov function (65). We first note that this function achieves Lipschitz smoothness property since it is derived from the algebraic sum of square loss from system dynamics. The Lipschitz smoothness property is crucial as it ensures bounded changes in the Lyapunov function concerning variations in the dual variables. Mathematically, we have the Lipschitz smoothness property as follows:

$$\|\nabla\mathcal{V}(\boldsymbol{\lambda}'^{(t)}) - \nabla\mathcal{V}(\boldsymbol{\lambda}'^{(t')})\| \leq L\|\boldsymbol{\lambda}'^{(t)} - \boldsymbol{\lambda}'^{(t')}\|, \tag{66}$$

where $\boldsymbol{\lambda}'^{(t)} = (\boldsymbol{\mu}'^{(t)}, \boldsymbol{\theta}'^{(t)}, \boldsymbol{\nu}'^{(t)}, \boldsymbol{\omega}'^{(t)})$ denotes the vector of dual variables at iteration t and L represents a positive Lipschitz constant. Exploiting the Lipschitz smoothness property (66) along with the gradient updating rule (57 – 60), we find that the Lyapunov function exhibits the following improvement in successive iterations

$$\mathcal{V}(\boldsymbol{\lambda}'^{(t)}) - \mathcal{V}(\boldsymbol{\lambda}'^{(t+1)}) \geq \frac{1}{2L}\|\nabla\mathcal{V}(\boldsymbol{\lambda}'^{(t)})\|^2. \tag{67}$$

By applying the telescoping sum technique, we obtain

$$\begin{aligned} \mathcal{V}(\boldsymbol{\lambda}'^{(0)}) - \mathcal{V}(\boldsymbol{\lambda}'^{(T)}) &\geq \frac{1}{2L}\sum_{k=0}^{T-1}\|\nabla\mathcal{V}(\boldsymbol{\lambda}'^{(k)})\|^2 \\ &\geq \frac{T}{2L}\min_{0\leq k<T}\|\nabla\mathcal{V}(\boldsymbol{\lambda}'^{(k)})\|^2. \end{aligned} \tag{68}$$

As a result, we can deduce that

$$\min_{0\leq k<T}\|\nabla\mathcal{V}(\boldsymbol{\lambda}'^{(k)})\| \leq \sqrt{\frac{2L(\mathcal{V}(\boldsymbol{\lambda}'^{(0)}) - \mathcal{V}(\boldsymbol{\lambda}'^{(T)}))}{T}}, \tag{69}$$

which guarantees that the IDA algorithm achieves the rate of convergence $O(\frac{1}{\sqrt{T}})$. This convergence rate demonstrates the efficiency of the IDA algorithm in approaching the optimal solution for the SWM problem.

5.2 Individual Rationality

Any participant in the proposed IDA framework is *individually rational* if the participant's payoff is always non-negative.

Recall that the server's utility functions are concave and possess the property that $U_j(\mathbf{0},\mathbf{0}) = 0$ for all server j. Therefore, we can apply the Lagrange's Mean Value Theorem to derive the following result:

$$\begin{aligned}
&U_j(\{\hat{d}^s_{ij}\}_{i\in C}, \{\hat{f}^s_i\}_{i\in C}) \\
&\geq U_j(\mathbf{0},\mathbf{0}) + \hat{d}^s_{ij}\frac{\partial U_j(\{\hat{d}^s_{ij}\}_{i\in C}, \{\hat{f}^s_i\}_{i\in C})}{\partial \hat{d}^s_{ij}} + \hat{f}^s_i\frac{\partial U_j(\{\hat{d}^s_{ij}\}_{i\in C}, \{\hat{f}^s_i\}_{i\in C})}{\partial \hat{f}^s_i} \\
&= \hat{d}^s_{ij}\frac{\partial U_j(\{\hat{d}^s_{ij}\}_{i\in C}, \{\hat{f}^s_i\}_{i\in C})}{\partial \hat{d}^s_{ij}} + \hat{f}^s_i\frac{\partial U_j(\{\hat{d}^s_{ij}\}_{i\in C}, \{\hat{f}^s_i\}_{i\in C})}{\partial \hat{f}^s_i}.
\end{aligned} \tag{70}$$

Considering the optimal bids (43) and the payment rule (50) for servers, we obtain

$$\begin{aligned}
&\hat{d}^s_{ij}\frac{\partial U_j(\{\hat{d}^s_{ij}\}_{i\in C}, \{\hat{f}^s_i\}_{i\in C})}{\partial \hat{d}^s_{ij}} + \hat{f}^s_i\frac{\partial U_j(\{\hat{d}^s_{ij}\}_{i\in C}, \{\hat{f}^s_i\}_{i\in C})}{\partial \hat{f}^s_i} \\
&= \alpha\hat{\phi}^d_{ij} + (1-\alpha)\hat{\phi}^f_i \\
&= P_j(\{\hat{\phi}^d_{ij}\}_{i\in C}, \{\hat{\phi}^f_i\}_{i\in C}).
\end{aligned} \tag{71}$$

This guarantees the consistent satisfaction of the following inequalities:

$$U_j(\{\hat{d}^s_{ij}\}_{i\in C}, \{\hat{f}^s_i\}_{i\in C}) \geq P_j(\{\hat{\phi}^d_{ij}\}_{i\in C}, \{\hat{\phi}^f_i\}_{i\in C}), \forall j \in S, \tag{72}$$

which implies non-negative payoffs of servers.

Similarly, the cost functions of clients are convex and have the property that $C_i(\mathbf{0}, 0) = 0$ for all client i. Therefore, we can also employ the Lagrange's Mean Value Theorem to obtain the following result:

$$\begin{aligned}
&C_i(\{\hat{d}^c_{ij}\}_{j\in S}, \hat{f}^c_i) \\
&\leq C_i(\mathbf{0}, 0) + \hat{d}^c_{ij}\frac{\partial C_i(\{\hat{d}^c_{ij}\}_{j\in S}, \hat{f}^c_i)}{\partial \hat{d}^c_{ij}} + \hat{f}^c_i\frac{\partial C_i(\{\hat{d}^c_{ij}\}_{j\in S}, \hat{f}^c_i)}{\partial \hat{f}^c_i} \\
&= \hat{d}^c_{ij}\frac{\partial C_i(\{\hat{d}^c_{ij}\}_{j\in S}, \hat{f}^c_i)}{\partial \hat{d}^c_{ij}} + \hat{f}^c_i\frac{\partial C_i(\{\hat{d}^c_{ij}\}_{j\in S}, \hat{f}^c_i)}{\partial \hat{f}^c_i}.
\end{aligned} \tag{73}$$

By taking account of the optimal bids (44) and the reward rule (56) of client devices, we can derive

$$\begin{aligned}
&\hat{d}^c_{ij}\frac{\partial C_i(\{\hat{d}^c_{ij}\}_{j\in S}, \hat{f}^c_i)}{\partial \hat{d}^c_{ij}} + \hat{f}^c_i\frac{\partial C_i(\{\hat{d}^c_{ij}\}_{j\in S}, \hat{f}^c_i)}{\partial \hat{f}^c_i} \\
&= \frac{(\hat{\mu}'_{ij} - \hat{\nu}'_{ij})^2}{\alpha\hat{\psi}^d_{ij}} + \frac{(\hat{\theta}'_i - \hat{\omega}'_i)^2}{(1-\alpha)\hat{\psi}^f_i} \\
&= Q_i(\{\hat{\psi}^d_{ij}\}_{j\in S}, \hat{\psi}^f_i).
\end{aligned} \tag{74}$$

This ensures the consistent fulfillment of the following inequalities:

$$C_i(\{\hat{d}^c_{ij}\}_{j\in S}, \hat{f}^c_i) \leq Q_i(\{\hat{\psi}^d_{ij}\}_{j\in S}, \hat{\psi}^f_i), \forall i \in C, \tag{75}$$

which implies non-negative payoffs of clients.

We thus conclude that all servers and clients get non-negative payoffs by participating in the IDA framework.

5.3 Budget Balance

The IDA framework ensures *budget balance* if the total payment collected from servers is not less than the total reward offered to clients. This property ensures no additional investment from the broker to maintain a non-negative budget throughout the entire procedure. To verify the broker's budget balance, we expand the payment rule (50) and the reward rule (56) as follows:

$$\begin{aligned}
&\sum_{j\in S} P_j(\{\phi^d_{ij}\}_{i\in C}, \{\phi^f_i\}_{i\in C}) - \sum_{i\in C} Q_i(\{\psi^d_{ij}\}_{j\in S}, \psi^f_i) \\
&= \sum_{j\in S}\sum_{i\in C}\left[\alpha\hat{\phi}^d_{ij} + (1-\alpha)\hat{\phi}^f_i + \frac{(\hat{\mu}'_{ij} - \hat{\nu}'_{ij})^2}{\alpha\hat{\psi}^d_{ij}} + \frac{(\hat{\theta}'_i - \hat{\omega}'_i)^2}{(1-\alpha)\hat{\psi}^f_i}\right].
\end{aligned} \tag{76}$$

Consequently, we substitute the above equation with the BA optimum (41, 42) and apply the constraints (12, 13) in the BA problem as follows:

$$\begin{aligned}
&\sum_{j\in S} P_j(\{\phi^d_{ij}\}_{i\in C}, \{\phi^f_i\}_{i\in C}) - \sum_{i\in C} Q_i(\{\psi^d_{ij}\}_{j\in S}, \psi^f_i) \\
&= \sum_{j\in S}\sum_{i\in C}\left[\hat{d}^s_{ij}\hat{\nu}'_{ij} + \hat{f}^s_i\hat{\omega}'_i + \hat{d}^c_{ij}(\hat{\mu}'_{ij} - \hat{\nu}'_{ij}) + \hat{f}^c_i(\hat{\theta}'_i - \hat{\omega}'_i)\right] \\
&\geq \sum_{j\in S}\sum_{i\in C}\left[\hat{d}^c_{ij}\hat{\mu}'_{ij} + \hat{f}^c_i\hat{\theta}'_i\right] \geq 0.
\end{aligned} \tag{77}$$

It indicates that the total payment is sufficient to cover the total reward. So, the IDA framework ensures budget balance.

5.4 Incentive Compatibility

An action mechanism is *incentive compatible* if it is the best strategy for all participants to bid truthfully, i.e., reveal their hidden utility and cost functions. With the broker's announcement, servers and clients can solve their respective payoff maximization problems. As participants are all price-takers, they simply submit their bids to maximize their respective payoffs considering the payment and reward rules. With the submitted bids, the broker proceeds to solve the BA problem, seeking a solution aligned with the SWM problem. Through iterations, the hidden costs of clients and the hidden utilities of servers in the SWM problem are gradually revealed. By revealing these hidden factors, the iterative mechanism accurately reflects each participant's true preferences and contributions. This promotes an environment where participants are motivated to truthfully reveal their preferences, as their bids directly influence resource allocation, impacting their payoffs. On the other hand, if participants choose not to bid truthfully, they risk receiving suboptimal resource allocation, resulting in lower payoffs for themselves. Deviation from truthfulness by participants may undermine the efficiency of the resource allocation process, potentially impeding their benefits within the FL framework. Thus, the IDA framework ensures that participants are incentivized to provide accurate information and engage actively in the FL process.

5.5 Economic Efficiency

In Sec. 5.1, we have shown that the proposed approach will converge to some result that meets constraints (31) to (40). In the previous subsection, we have shown that the hidden costs of clients and the hidden utilities of servers in the SWM problem are gradually revealed when clients and servers submit their bids (i.e., their respective solutions to (45) and (51)) as responses to given pricing and reward rules (50) and (56). With revealed client costs and server utilities, the broker can find a socially optimal solution that meets (9) to (14). Therefore, the IDA framework will eventually lead to a socially optimal solution, achieving economic efficiency.

6 Experimental Results

We conducted a series of experiments for a numerical analysis of the proposed framework, including the convergence and the impacts of the resource supply-demand ratio, model value, energy cost, and time cost. We also compared the performance of the proposed approach with competitive methods. It is important to note that the simulations in this section evaluate the resource allocation mechanism directly through the analytical model (i.e., the utility, cost, and social welfare functions defined in Sec. 3), without executing FL training. This is sufficient because the allocation decisions and their economic properties are fully determined by the system model. FedAvg [1] was used separately in Fig. 3 solely to generate empirical accuracy measurements for fitting the accuracy model parameters, as described in Sec. 3. We assumed 4 to 12 clients and 2 to 20 servers. The transmissions between clients and servers were assumed LoRA with transmission rates R_{ij} and R_{ji} set to 50 Kbps [38] and the client's transmission power set to 20 dBm [39]. For each client i, we set the maximum dataset size, D_i, to 256 MB and CPU-cycle frequency, $f_i^{\max}$, to 240 MHz, based on the specifications of the LoPy 4 [40]. We adopted the SqueezeNet [41] as our DNN model with parameter w_j in each round. SqueezeNet offers AlexNet-level accuracy with significantly fewer parameters, resulting in a model size of less than 0.5 MB. We considered heterogeneous clients by varying their computing efficiency (τ_i in (3)) and energy efficiency (κ_i in (6)), where κ_i was in the range between 1e-11 and 7e-11 [42]. Furthermore, we set the proportion of the slowest training time compared to the total training time, denoted as δ_j, to 0.3. Table 3 summarizes the parameter setting.

TABLE 3: Simulation Settings

Parameter	Unit	Default Value	Range
Number of clients		4	[4, 12]
Number of servers		4	[2, 20]
β_j in (1)		1.0	—
σ_j in (1)		0.002	[0.001, 0.0025]
γ_j in (1)		0	—
r_j in (2)		0.5	[0.3333, 0.8090]
w_j in (2)	MB	0.5	—
R_{ij} in (2)	Kbps	50	—
R_{ji} in (2)	Kbps	50	—
τ_i in (3)		1	[0.5, 2.0]
ρ_j^{acc} in (5)		10^4	[5000, 15000]
ρ_j^{time} in (5)		1	[0.5, 5]
κ_i in (6)		1e-11	[1e-11, 7e-11]
p_{ij} in (7)	dBm	20	—
ρ_i^{eng} in (8)		5000	[1000, 8000]
D_i in (10)		256	—
$f_i^{\max}$ in (11)	MHz	240	—
δ_j in (16)		0.3	—
α in (29)		0.5	—
ϵ in Algorithm 1		0.01	—
η: step size in (57) to (60)		0.05	[0.03, 0.07]

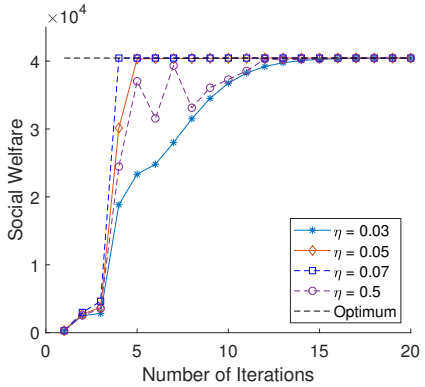


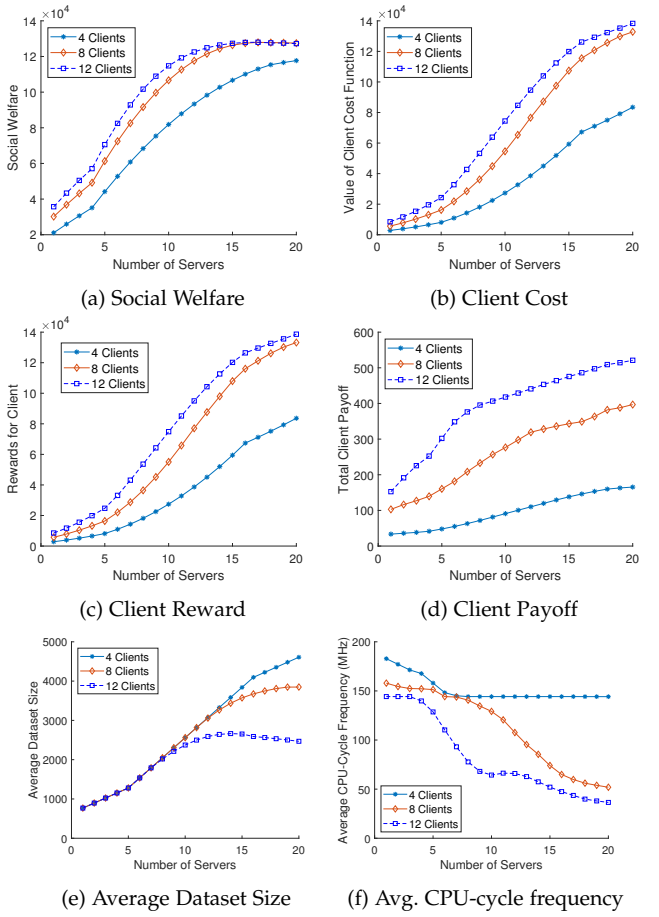
Fig. 5: Convergence of the IDA algorithm with different settings of η

6.1 Convergence of IDA

We consider different settings of the step size η in the gradient descent process. Fig. 5 illustrates the change in social welfare as the iteration count of the IDA algorithm increased. In all cases, the social welfare generally increased with the number of iterations and eventually converged to the maximum value. This trend confirms the economic viability and convergence of the proposed IDA algorithm. The step size affected the convergence rate but not the achievable maximum social welfare. Generally speaking, a larger step size leads to a quicker convergence of social welfare. However, this faster convergence should be carefully considered, as it might come at the expense of sacrificing solution stability, as observed in the case of $\eta = 0.5$.

6.2 Impacts of Resource Supply-Demand Ratio

We studied the impacts of the supply-demand ratio on performance metrics by fixing the number of clients to four and varying the number of servers from 1 to 20. Fig. 6a shows the result of social welfare, where the social welfare increased as the number of servers increased initially. However, increased participation eventually led to diminishing returns in social welfare due to limited client resources. To provide a clearer picture of the overall system performance, we further analyzed the costs, the rewards, and the payoffs of clients. The client costs increased with the number of servers (Fig. 6b), and the client rewards (Fig. 6c) exhibited the same trend but were slightly higher than the costs. Consequently, client payoffs slightly increased with increasing number of servers (Fig. 6d). Fig. 6e shows how the average amount of data offered by each client, $\bar{d} = \sum_{i \in C} \sum_{j \in S} d_{ij}^c / |C|$, changed with the number of servers. Recall that both the server's utilities and the client's costs increase with $\bar{d}$. When more servers were involved, the server's utilities were weighed more than the client's costs, so we can observe a trend of increasing $\bar{d}$. However, the server's utilities increase logarithmically. In contrast, the client's costs increase linearly with $\bar{d}$, so the increasing rate

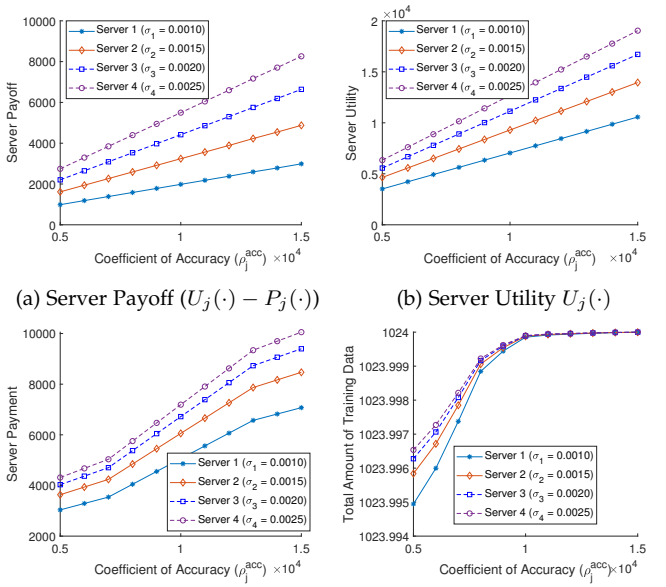


(a) Social Welfare (b) Client Cost

(c) Client Reward (d) Client Payoff

(e) Average Dataset Size (f) Avg. CPU-cycle frequency

Fig. 6: Impacts of the supply-demand ratio on (a) social welfare, (b) client cost, (c) client reward, (d) client payoff, (e) average dataset size, and (f) average CPU-cycle frequency.



(a) Server Payoff ($U_j(\cdot) - P_j(\cdot)$) (b) Server Utility $U_j(\cdot)$

(c) Server Payment $P_j(\cdot)$ (d) Total Amount of Training Data

Fig. 7: Impact of model value (ρ_j^{acc}) on (a) server payoff, (b) server utility, (c) server payment, and (d) total amount of training data. Each server j had a distinct setting of σ_j.

decreased when the total amount of client training data was sufficient (i.e., with 12 or 8 clients). On the other hand, the server's utilities and the client's costs also increased with the average client CPU-cycle frequency $\bar{f} = \sum_{i \in C} f_i^c / |C|$. Fig. 6f shows that $\bar{f}$ did not increase when more servers were involved. This can be justified as the client's costs increase quadratically while the server's utilities increase only linearly with $\bar{f}$. With a large number of clients (e.g., 12 clients), $\bar{f}$ decreased to maximize social welfare.

6.3 Impact of Model Value

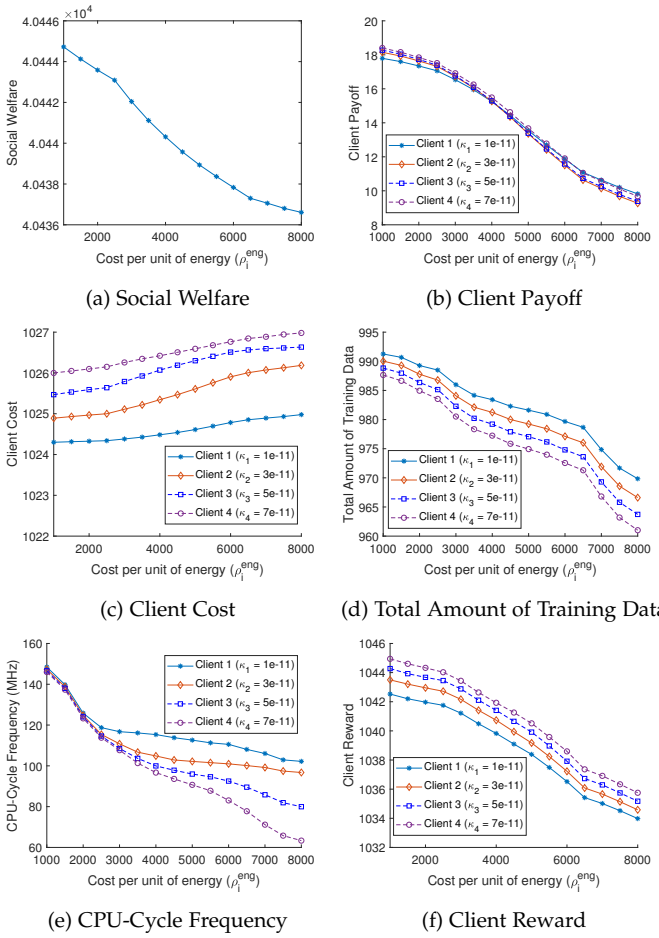
Next, we tested the impact of model value on social welfare. We used a fixed setting with four clients and four servers. These servers were set up with different σ_j values, representing a scenario where different models had different sensitivity to diminishing marginal contribution by dataset size. We then examined the impact of the parameter for model value, ρ_j^{acc} in (5), which controls the trade-off between model accuracy and training time in defining server j's utility.

As depicted in Fig. 7a, each server's payoff increased with the value of ρ_j^{acc}. This is reasonable as a higher ρ_j^{acc} value brings in a higher server utility. For a fixed ρ_j^{acc} value, a server with a higher σ_j value also had a higher payoff, which is expected as a higher σ_j value implies a higher $A_j(\cdot)$ value. We decomposed the server's payoffs into the utility part (Fig. 7b) and the payment part (Fig. 7c). The server's utilities exhibited the same trend as the server's payoffs, while the server's payments also increased with ρ_j^{acc}. Fig. 7d demonstrates that the increase in server's payoffs was due to the increase of the valuation on the same amount of training data (either through a larger ρ_j^{acc} or a larger σ_j value) because the amounts of training data were almost unchanged as ρ_j^{acc} increased (the difference was within 0.005).

6.4 Impact of Energy Cost

Client cost comes from energy consumption in our system model. We thus systematically varied the cost per unit of energy, ρ_i^{eng}, from 1000 to 8000 with a step size of 1000 to observe the impact of energy cost. We considered four servers and four clients with different effective capacitances: $(\kappa_1, \kappa_2, \kappa_3, \kappa_4) = (1\times10^{-11}, 3\times10^{-11}, 5\times10^{-11}, 7\times10^{-11})$.

As shown in Fig. 8a, the social welfare decreased as the value of ρ_i^{eng} increased, which is expected. Fig. 8b shows each client's payoff. The client payoff was negatively impacted by the increase in ρ_i^{eng} and acted as a diminishing factor to the social welfare. However, each client's cost increased slightly, as illustrated in Fig. 8c. A deeper inspection revealed that the IDA algorithm suppressed the potentially linear increase of client cost due to increasing ρ_i^{eng} by decreasing the total amount of training data (Fig. 8d) and the CPU-cycle frequency (Fig. 8e). We highlight that a client with a higher setting of κ_i (e.g., Client 4) faced a greater computational burden and thus provided less training data and lower CPU-cycle frequency. The price of the cost suppression was the reduction of client rewards, as shown in Fig. 8f.

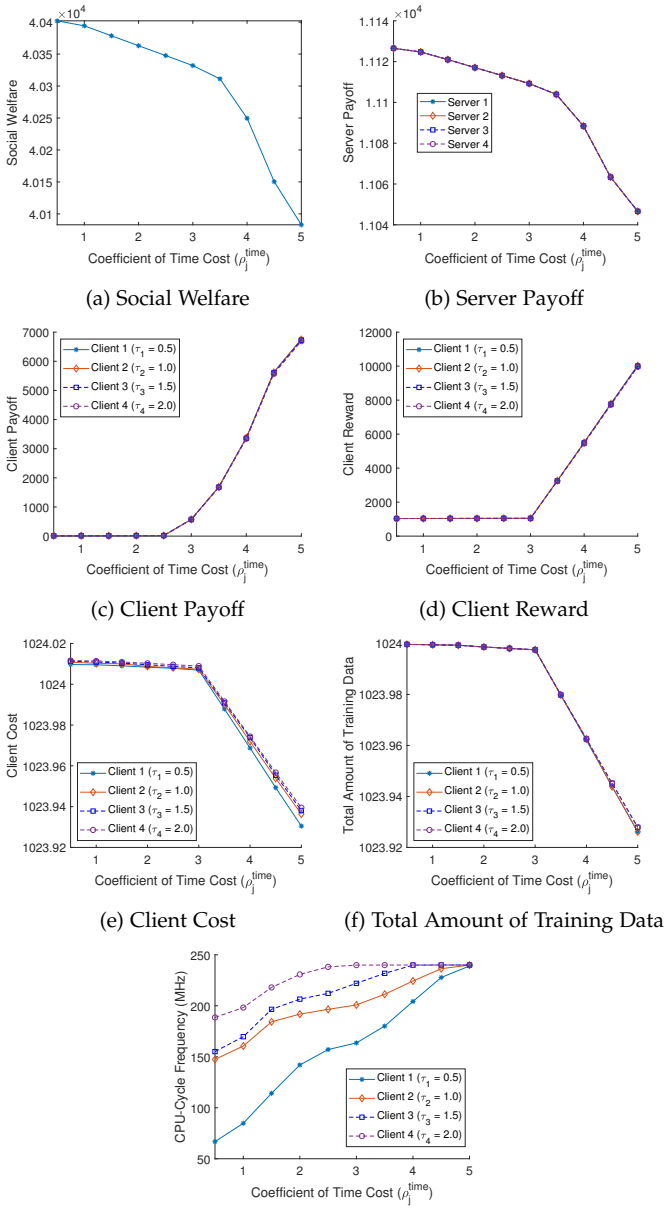


(a) Social Welfare (b) Client Payoff

(c) Client Cost (d) Total Amount of Training Data

(e) CPU-Cycle Frequency (f) Client Reward

Fig. 8: Impact of cost per unit of energy (ρ_i^{eng}) on (a) social welfare, (b) client payoff, (c) client cost, (d) total amount of training data, (e) CPU-cycle frequency, and (f) client reward. Each client i had a distinct setting of effective capacitance (κ_i).



(a) Social Welfare (b) Server Payoff

(c) Client Payoff (d) Client Reward

(e) Client Cost (f) Total Amount of Training Data

(g) CPU-Cycle Frequency

Fig. 9: Impact of time cost (ρ_j^{time}) on (a) social welfare, (b) server payoff, (c) client payoff, (d) client reward, (e) client cost, (f)total amount of training data, and (f) CPU-cycle frequency. Each client i had a different setting of τ_i.

6.5 Impact of Time Cost

As indicated in (5), server utility $U_j(\cdot)$ is adversely affected by the cost of the training time T_j. To study the impact of time cost, we varied ρ_j^{time} as a way to change the weight of time cost in utility evaluation. As expected, an increase of ρ_j^{time} resulted in a decrease in social welfare (Fig. 9a). Supporting this observation, Fig. 9b illustrated a trend of decrease in each server's payoff as ρ_j^{time} increased. By contrast, Fig. 9c shows that the client payoff exponentially increased as ρ_j^{time} increased. This is because the IDA algorithm linearly increased the rewards to clients (Fig. 9d) and slightly decreased the client costs (Fig. 9e) when ρ_j^{time} increased to 3 or larger. Regarding the client costs, Fig. 9f shows that the amount of training data did not change significantly, but the CPU-cycle frequency increased to suppress the increase in time cost as ρ_j^{time} increased. Therefore, the increase in the client's payoff in Fig. 9c was mainly due to the server's compensations for the increase of the client's energy cost due to the rise of CPU-cycle frequency.

Note that varying ρ_j^{time} from 1 to 5 varies the effective time-cost weight $\rho_j^{\text{time}} \cdot \delta_j$ from 0.3 to 1.5. Since the qualitative behavior of the system remains consistent across this range, it is confirmed that the solution is not overly sensitive to the choice of δ_j.

6.6 Competitive Methods

We compared the performance of the IDA algorithm with three competitive methods, including NSGA-II [43], PAM-N, and PAM-S [44], [45], in terms of market social welfare. NSGA-II is a population-based multi-objective GA that searches for Pareto-optimal solutions by evolving a population of candidate allocations across generations. While it can approximate good solutions relatively quickly, it requires complete knowledge of all participants' utility and cost functions. Since these are inherently private in our setting, NSGA-II is not directly applicable; we include it as a performance reference by assuming full information is available to the algorithm. PAM-N is an auction-based method that considers bids from participants but allocates

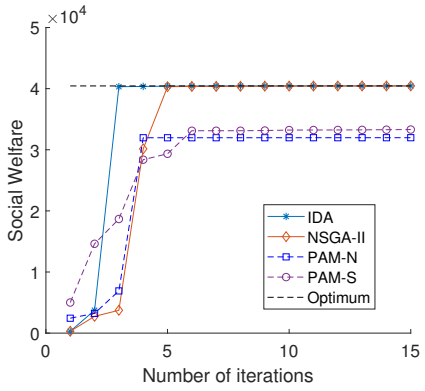


Fig. 10: Comparison of the IDA algorithm with competitive methods (NSGA-II, PAM-N, and PAM-S) in terms of market social welfare

TABLE 4: Comparison of Converged Social Welfare and Computation Time

Method	Social Welfare (% of opt.)	Time (s)
IDA	40,446.07 (100.0%)	3.09 ± 0.16
NSGA-II	40,446.06 (100.0%)	0.26 ± 0.04
PAM-N	31,985.74 (79.1%)	1.18 ± 0.07
PAM-S	33,591.40 (83.1%)	0.81 ± 0.06
Optimum	40,446.07 (100.0%)	—
Time: mean $\pm$ std over 9 trials (warm-up excluded).		

resources based on a fixed iteration count rather than the convergence of dual variables, which can lead to premature or suboptimal termination. PAM-S adopts a Stackelberg game approach, where clients act as leaders who declare their bids first, followed by servers acting as followers who decide their bids in each round. While PAM-S accounts for strategic behavior, the sequential leader-follower structure introduces an asymmetry that may disadvantage servers and result in suboptimal social welfare.

Fig. 10 shows the convergence trajectories of all mechanisms, and Table 4 summarizes their converged social welfare and computation time averaged over 10 trials. IDA and NSGA-II both converged to the optimal social welfare (40,446.07), after three and five iterations respectively, while PAM-N and PAM-S converged to only 79.1% and 83.1% of the optimum. The failure of PAM-N to reach the optimum confirms that iteration-count-based termination is less effective than the dual-variable-driven convergence criterion of IDA. The underperformance of PAM-S suggests that the Stackelberg leader-follower asymmetry limits the achievable social welfare compared to the symmetric double-auction framework of IDA. In terms of computation time, NSGA-II is the fastest (0.26 s), followed by PAM-S (0.81 s), PAM-N (1.18 s), and IDA (3.09 s). The higher computation time of IDA is attributable to the sequential execution of the $s + c$ individual optimization subproblems via `fmincon`. In practice, each server solves its SPM problem and each client solves its CPM problem independently and in parallel, so the per-iteration computation time is determined by the slowest subproblem (rather than their sum). Crucially, unlike NSGA-II, IDA achieves the same optimal social welfare without requiring complete information from participants, and its accompanying pricing rule provides provable guarantees of individual rationality and incentive compatibility that no other compared method can offer.

7 Conclusions and Future Work

This study has addressed the allocation of training resources, i.e., training data and CPU-cycle frequency, from clients to multiple servers in FL. It differs from previous studies in our joint consideration of model accuracy, training time, and energy consumption. To maximize social welfare in an asymmetric information market, we have proposed an IDA mechanism where the broker motivates servers and clients to submit bids that represent their true resource demands and provisions while servers and clients set up their bids only to maximize their respective payoffs. We have analyzed the convergence of the IDA algorithm and examined its desirable economic properties, including individual rationality, budget balance, incentive compatibility, and economic efficiency. We have conducted simulations to observe how various factors, including the resource supply-demand ratio, the model value, the energy cost, and the time cost, affect social welfare. The simulation results also demonstrate the superior performance of the IDA algorithm over other competitive methods, including NSGA-II, PAM-N, and PAM-S.

There are two possible directions for future work. One is an online allocation rule to adaptively allocate resources in scenarios where server utilities and client costs change dynamically. The other is a decentralized and autonomous mechanism to eliminate the role of the broker.

References

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int'l Conf. on Artificial Intelligence and Statistics*, Apr. 2017, pp. 1273–1282.

[2] 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.

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

[4] M. R. Sprague, A. Jalalirad, M. Scavuzzo, C. Capota, M. Neun, L. Do, and M. Kopp, "Asynchronous federated learning for geospatial applications," in *Joint European Conference Machine Learning and Knowledge Discovery in Databases*, vol. 967, 2018, pp. 21–28.

[5] 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.

[6] 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, Oct. 2020.

[7] 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.

[8] W. Y. B. Lim, J. Huang, Z. Xiong, J. Kang, D. Niyato, X.-S. Hua, C. Leung, and C. Miao, "Towards federated learning in UAV-enabled internet of vehicles: A multi-dimensional contract-matching approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 8, pp. 5140–5154, Aug. 2021.

[9] 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.

[10] 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.

[11] 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, Oct. 2020.

[12] D. Wang, X. Pang, J. Hu, S. Yue, and J. Ren, "Two-dimensional Stackelberg game-based incentive mechanism for differential private federated learning with non-IID data," *IEEE Trans. Mobile Comput.*, no. in press, Mar. 2026.

[13] T. H. Thi 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. 8, pp. 4874–4887, Aug. 2021.

[14] 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.

[15] T. Mai, H. Yao, J. Xu, N. Zhang, Q. Liu, and S. Guo, "Automatic double-auction mechanism for federated learning service market in Internet of Things," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 5, pp. 3123–3135, Sep.–Oct. 2022.

[16] F.-Y. Chen and L.-H. Yen, "Incentive-aware resource allocation for multiple model owners in federated learning," *IEEE Trans. Serv. Comput.*, vol. 12, no. 2, pp. 549–562, Mar.–Apr. 2024.

[17] 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.

[18] 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.

[19] 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.

[20] M.-C. Cho, L.-H. Yen, and J.-X. Wang, "Seeking stability for multi-leader Stackelberg game as an incentive mechanism for multi-requester federated learning," *IEEE Access*, vol. 13, pp. 5922–5936, Jan. 2025.

[21] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," in *Proc. Int'l Conf. on Learning Representations (ICLR)*, 2020, pp. 1–6.

[22] A. Li, J. Sun, P. Li, Y. Pu, H. Li, and Y. Chen, "Hermes: An efficient federated learning framework for heterogeneous mobile clients," in *Proc. 27th ACM MobiCom*, Oct. 2021, pp. 420–437.

[23] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "A novel framework for the analysis and design of heterogeneous federated learning," *IEEE Trans. Signal Process.*, vol. 69, pp. 5234–5249, Aug. 2021.

[24] Q. Ma, Y. Xu, H. Xu, Z. Jiang, L. Huang, and H. Huang, "FedSA: A semi-asynchronous federated learning mechanism in heterogeneous edge computing," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3654–3672, Dec. 2021.

[25] C. T. Dinh, N. H. Tran, M. N. 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.

[26] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 3, pp. 1935–1949, Mar. 2021.

[27] S. Wang, H. Zhao, W. Wen, W. Xia, B. Wang, and H. Zhu, "Contract theory based incentive mechanism for clustered vehicular federated learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 7, pp. 8134–8147, Jul. 2024.

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

[29] N. Zhang, Q. Ma, and X. Chen, "Auction based incentive mechanism in federated learning considering communication path finding," in *Proc. IEEE INFOCOM*, 2024, pp. 1–10.

[30] X. Tu, K. Zhu, N. C. Luong, D. Niyato, Y. Zhang, and J. Li, "Incentive mechanisms for federated learning: From economic and game theoretic perspective," *IEEE Trans. Cogn. Commun. Netw.*, vol. 8, no. 3, pp. 1566–1593, 2022.

[31] 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.

[32] A. Krizhevsky, "Learning multiple layers of features from tiny images," Dept. Comput. Sci., Univ. Toronto, Toronto, Tech. Rep., 2009.

[33] Y. Liu, S. Garg, J. Nie, Y. Zhang, Z. Xiong, J. Kang, and M. S. Hossain, "Deep anomaly detection for time-series data in industrial IoT: A communication-efficient on-device federated learning approach," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6348–6358, Apr. 2021.

[34] T. D. Burd and R. W. Brodersen, "Processor design for portable systems," *J. VLSI Sign. Process. Syst. Sign. Image. Video Technol.*, vol. 13, no. 2-3, pp. 203–221, Aug. 1996.

[35] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained IoT devices," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 1–24, Jan. 2022.

[36] B. P. Majumder, M. N. Faqiry, S. Das, and A. Pahwa, "An efficient iterative double auction for energy trading in microgrids," in *Proc. IEEE Symp. on Computational Intelligence Applications in Smart Grid (CIASG)*, Dec. 2014, pp. 1–7.

[37] D. P. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1439–1451, Aug. 2006.

[38] M. Bor and U. Roedig, "LoRa transmission parameter selection," in *Proc. 13th Int'l Conf. on Distributed Computing in Sensor Systems (DCOSS)*, Jun. 2017, pp. 27–34.

[39] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the limits of LoRaWAN," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 34–40, Sep. 2017.

[40] Pycom Ltd., "LoPy 4 Datasheet," [Online]. Available: https://docs.pycom.io/datasheets/development/lopy4/, accessed: 2023-06-22.

[41] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and¡ 0.5 MB model size," *arXiv preprint arXiv:1602.07360*, Nov. 2016.

[42] P. Tam, S. Math, and S. Kim, "Optimized multi-service tasks offloading for federated learning in edge virtualization," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 6, pp. 4363–4378, Nov.–Dec. 2022.

[43] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. Hoboken, NJ, USA: Wiley, 2001, vol. 16.

[44] K. P. Naveen and R. Sundaresan, "Double-auction mechanisms for resource trading markets," *IEEE/ACM Trans. Netw.*, vol. 29, no. 3, pp. 1210–1223, Jun. 2021.

[45] "MATLAB Simulation for IDADET," [Online]. Available: https://github.com/NII-Tohoku-Xidian/Data-Transaction-Ecosystem-in-IoV/tree/IDADET, 2022, accessed: 2024-07-17.

Jing-Xuan Wang received the B.S. and M.S. degrees from National Yang Ming Chiao Tung University, Hsinchu, Taiwan, in 2021 and 2023, respectively. Since 2023, he has been an engineer with Taiwan Semiconductor Manufacturing Company (TSMC), Taiwan. His research interests include game theory and optimization.

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. Before he joined the Department of Computer Science, National Chiao Tung University, he was with Chung Hua University, Hsinchu, Taiwan, from 1998 to 2006 and the National University of Kaohsiung, Kaohsiung, Taiwan, from 2006 to 2016. His current research interests include distributed computing, wireless networking, and game theory. He was the Editor Boards of the Springer's Wireless Networks. He was the recipient of the Best Paper Awards of the IEEE WCNC 2013, ISPA 2015, and APNOMS 2021.