Joint Job Assignment and Resource Allocation for Multi-Job Wireless Federated Learning

Tan Li*, Zeheng Wei*, Hai Liu*, Zhiyong Lin†, Tse-Tin Chan‡
* Department of Computer Science, The Hang Seng University of Hong Kong, Hong Kong SAR, China
† School of Computer Science, Guangdong Polytechnic Normal University, China
‡ Department of Mathematics and Information Technology, The Education University of Hong Kong, Hong Kong SAR, China
E-mails: tanli@hsu.edu.hk, laowei_szu@foxmail.com, hliu@hsu.edu.hk, zylin@gpnu.edu.cn, tsetinchan@eduhk.hk

***Abstract*—Wireless Federated Learning (WFL) marks a significant evolution in edge artificial intelligence (AI), allowing for collaborative learning while preserving the privacy of edge devices. Recently, the enhanced data collection and storage capability of edge devices have precipitated a paradigm shift from training an individual model (single-job) to multiple AI models (multi-job). This shift poses a new challenge in maintaining high system performance while managing resource management among multiple jobs. In this work, we address the challenge of a multi-job WFL framework by optimizing its dual efficiency. We formulate a multi-objective optimization problem with the goal of minimizing energy consumption and execution time to enhance system efficiency, while ensuring that all AI jobs meet their predefined performance criteria, thereby guaranteeing learning efficiency. To solve the problem, we propose an algorithm that jointly optimizes job assignment and resource allocation, by considering the triple-heterogeneity of data, devices, and jobs in the multi-job WFL framework. Specifically, a matching game-based method is utilized to assign jobs to capable devices, considering their respective contributions and costs, while convex optimization techniques are employed to refine the resource allocation toward computing frequency and transmission power. The performance of the algorithm is evaluated through numerical simulations in terms of system and learning performance over multiple AI model training jobs, showing that our proposed algorithm ensures time and energy efficiency under the same learning performance constraints.**

***Index Terms*—Learning efficiency, multi-job optimization, resource allocation, system efficiency, wireless federated learning.**

I. Introduction

The advent of 5G and future 6G networks, along with cutting-edge advancements in artificial intelligence (AI), are driving us towards a future filled with edge intelligence, where mobile devices boast substantial capabilities for data collection, processing, and storage [1]. A crucial challenge in edge intelligence is collaborative learning from the vast data produced by edge devices while preserving privacy. Wireless Federated Learning (WFL) [2], [3] addresses this by enabling devices to share learned knowledge through model parameters, thus eliminating the need to transmit sensitive user data to centralized servers [4].

This work was partially supported by RMGS "Towards Adaptive Federated Learning Against Gradient Leakage Attacks" and "Intelligent Wireless Edge Networks: Theories and Applications" of HSUHK. (*Corresponding author: Zeheng Wei.*)

Measuring the utility of a WFL framework relies on key performance indicators from both wireless communication and learning domains. Firstly, system efficiency [5], [6] is paramount, encompassing the time and energy devices required to partake in federated training. Some research focuses on minimizing training delays [7] or reducing the total energy cost [8], [9] to improve WFL system efficiency. Such improvements are typically realized through the strategic adjustment of resource allocation, including bandwidth, computational frequency, and transmission power. Another line of research emphasizes performance optimization, such as the minimization of the learning loss value [10], [11] by the design of client scheduling and model aggregation mechanisms.

Despite some progress in the aforementioned research, these methods still face limitations in practical applications because of their anchoring in single-job paradigm. Central servers (or platforms) often face the necessity of training multiple jobs concurrently [12]. In response to these, research has delved into the development of multi-job FL frameworks [13]–[16]. Studies such as [13] and [14] have proposed client selection strategies for multi-job (or multi-model) scenarios, yet these strategies have not fully addressed the complexities of wireless communication. Furthermore, research like [15], [16] have considered the multi-job FL in a vehicular network but have focused more on system efficiency, neglecting important learning performance. More importantly, various heterogeneities in multi-job scenario have not been comprehensively considered, such as *job heterogeneity* (the distinct requirements of different jobs), *device heterogeneity* (the variation in hardware configurations), and *data heterogeneity* (the difference in data quantity, distribution, and quality).

To address these issues, we introduce a multi-job WFL framework that offers a more robust and realistic approach to contemporary settings, accounting for the triple-heterogeneity. Within the proposed multi-job WFL framework, the objective is determined to be balancing the trade-offs between learning and system efficiencies, and a two-step optimization algorithm is proposed to tackle the problem. The specific contributions of our work are outlined as follows:

- We develop a multi-job WFL framework that accounts for balancing dual efficiency. We formulate a performance-constrained multi-objective optimization problem, aiming

at minimizing the time and energy required for the training process until all jobs reach their predetermined performance.

- To tackle the formulated problem, we develop a joint optimization algorithm involving both job assignment and resource allocation. Specifically, we employ a many-to-one matching game to efficiently match jobs with devices while processing triple-heterogeneity. A resource allocation algorithm is then implemented by fine-tuning computing and power resources to approach objectives.
- Our experiments feature various datasets and deep learning models. Our algorithm stands out in comparison to baseline strategies on both training duration and energy usage while meeting the same performance.

II. Related Work

A. Single-job WFL

Optimizing the WFL system demands an integrated consideration of both system and learning domains. In the first line of research, time delay and energy consumption during the training process are two key indicators directly linked to system efficiency. Some studies aim to minimize energy cost by implementing energy-aware client selection [17] and adaptive resource allocation policies [9]. On the other hand, research like [6], [7], [18], [19] has focused on minimizing training delay, with contributions that include optimization of resource allocation [7], and methods to client scheduling [6].

Conversely, training loss or accuracy serve as vital metrics for assessing learning efficiency and are commonly considered alongside system efficiency. A short-sighted strategy that only optimizes energy and time consumption is insufficient since total time and energy costs are influenced by the entire learning process. To achieve a balance between dual efficiency, previous works often formulate multi-objective optimization problems [20] or optimization problems with constraints [21]. Despite the progress, a notable limitation is that most of these algorithms focus on a single-job setting. When operating multiple jobs training in parallel, the schedule steps for different jobs may overlap, leading to resource competition.

B. Multi-Job WFL

Some existing studies have explored multi-job FL, but the insights gained are limited. Authors in [13] and [14] have both introduced client scheduling schemes for multiple jobs or models. However, both works employ a first-come-first-serve job scheduling mode, where a job immediately selects devices from the unoccupied ones, leading to a suboptimal candidate pool for job selection. Additionally, they are not conducted in a wireless communication environment.

The works most closely related to ours are [15], [16], which considered multi-job FL in the vehicular network with job assignment and bandwidth allocation. However, our work differs from theirs in several significant ways. First, we consider the dual-efficiency while [15], [16] only focused on system efficiency. Secondly, their works do not fully consider the triple-heterogeneity proposed in this work. In particular, they demonstrate that jobs prefer devices with lower per-round energy/time cost, which could result in jobs preferring devices with smaller datasets (data heterogeneity), consequently hindering model aggregation and extend learning rounds. Lastly, [15], [16] do not utilize real-world multiple deep learning models and datasets for experimentation. In summary, our proposed method is the first to consider optimizing dual-efficiency in multi-job WFL within the context of triple-heterogeneity.

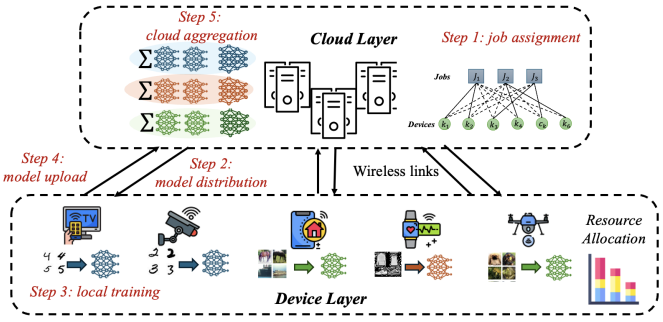


Fig. 1. Multi-job wireless federated learning framework.

III. System Model and Problem Formulation

As shown in Fig. 1, our proposed paradigm, named MJ-WFL, involves multiple devices working collaboratively to train a variety of AI jobs disseminated by the platform. Given that each device is equipped with an array of sensors (such as cameras, temperature and audio sensors), it is assumed that they each possess distinct local datasets that can be leveraged across several jobs. We use $\mathcal{N} = \{1, ..., n, ..., N\}$ and $\mathcal{K} = \{1, ..., k, ..., K\}$ to denote the sets of jobs and devices, respectively. Each device k is assumed to have N local datasets corresponding to the N jobs and the dataset of the n-th job on device k is expressed as $\mathcal{D}_{n,k}$ with $D_{n,k}$ samples. Typically, the global learning objective of MJ-WFL is to find a set of parameters $\mathbf{W} = \{\mathbf{w}_1, ..., \mathbf{w}_n, ..., \mathbf{w}_N\}$ to minimize the overall performance loss:

$$\min_{\mathbf{W}} \sum_{n=1}^{N} \sum_{k=1}^{K} \frac{D_{n,k}}{D_n} L_n^k(\mathbf{w}_n) \tag{1}$$

where D_n is the size of $\mathcal{D}^n = \cup_k \mathcal{D}_{n,k}$ and $L_n^k(\mathbf{w}_n)$ is the loss value of job n at device k. There are some scheduling steps throughout the learning process. We formally index the scheduling steps of job n as t_n, at which job n asks for the cloud server to assign a subset of qualified devices to collaboratively participate in its FL training (Step 1). We use a binary variable $a_{n,k}(t_n)$ to indicate whether job n is scheduled to device k. After job assignment, the subset of devices $K_n(t_n) := \{k \in \mathcal{K} | a_{n,k}(t_n) = 1\}$ receives the corresponding global model $\mathbf{w}_n(t_n - 1)$ of job n, distributed by the cloud server (Step 2). Each device $k \in K_n(t_n)$ initializes $\mathbf{w}_n(t_n - 1)$ as its local model $\mathbf{w}_{n,k}^0(t_n)$ and starts κ_n times local training (Step 3). For each local step $\tau \in [1, \kappa_n]$, device

k updates its local model using stochastic gradients over the local dataset:

$$\mathbf{w}_{n,k}^{\tau}(t_n) = \mathbf{w}_{n,k}^{\tau-1}(t_n) - \eta_n \nabla L_n^k(\mathbf{w}_{n,k}^{\tau-1}(t_n), \mathcal{D}_{n,k}) \quad (2)$$

where η_n is the learning rate of job n. After κ_n times local training, the device $k \in K_n(t_n)$ sets $\mathbf{w}_{n,k}(t) = \mathbf{w}_{n,k}^{\kappa_n}(t)$ and uploads its updated local model to the cloud server through wireless links (Step 4). The cloud server then performs aggregation over all collected models and updates the global model of job n (Step 5):

$$\mathbf{w}_n(t_n) = \frac{D_{n,k}}{\sum_{k \in K_n(t_n)} D_{n,k}} \mathbf{w}_{n,k}(t) \quad (3)$$

The steps outlined above will be iterated until the global model achieves the required performance. From this procedure, it is apparent that setting the scheduling point t_n and choosing the device subset $K_n(t_n)$ are pivotal to the success of our learning process. This distinction also highlights the primary difference between the MJ-WFL framework and traditional single-job WFL approaches. Detailed strategies regarding these aspects will be elaborated in the algorithm described in Section IV.

A. Problem Formulation

Building on prior discussions, our work wants to explore the dual efficiency within the MJ-WFL system, focusing on both system and learning efficacy. In this section, we first characterize the system cost within a single learning round t_n, arising from two principal factors: time delays and energy costs incurred by computational and communication processes.

1) Computing Model: Assuming that $\nu_{n,k}$ denotes the number of CPU cycles performing the forward-backward propagation algorithm with one data for job n at k. Note that $\nu_{n,k}$ will be influenced by the job profile, such as the model type, e.g., long-short term memory (LSTM) or convolutional neural network (CNN), the adopted training methods, e.g., SGD or Adam, and the size of data sample, e.g., the image size. By allocation $f_k(t_n)$ CPU frequency for device $k \in K_n(t_n)$ at round t_n, the time and energy cost can be expressed as:

$$T_{n,k}^{L}(t_n) = \kappa_n \cdot \nu_{n,k} D_{n,k} / f_k(t_n) \quad (4)$$

$$E_{n,k}^{L}(t_n) = \kappa_n \cdot \gamma_k \nu_{n,k} f_k^2(t_n) D_{n,k} \quad (5)$$

where γ_k is the effective switched capacitance that depends on the chip architecture [22].

2) NOMA-aided Transmission Model: The majority of work in WFL utilizes Orthogonal Multiple Access (OMA) techniques [16], [9], which require that devices only work in different time slots or different frequencies, resulting in a waste of time and energy. Consequently, we employ Non-Orthogonal Multiple Access (NOMA), enabling model uploading through the shared channel, so that higher spectral efficiency and better resource allocation can be achieved based on the specific channel conditions of each device [1]. To solve the inter-device interference incured by simultaneous transmission, we apply the successive interference cancellation (SIC) technique, which decodes the received information according to their channel gains. With the loss of generality, we assume that at the end of t_n, only $|K_n(t_n)|$ devices within the set $K_n(t_n)$ need to upload their models. We sort their channel gains $h_k(t_n), k \in K_n(t_n)$ in descending order. Therefore, the achievable data rate of device k is given by:

$$R_{n,k}(t_n) = B(t_n) \log_2(1 + \frac{p_k(t_n) h_k(t_n)}{\sum_{j=k+1}^{|K_n(t_n)|} p_j(t_n) h_j(t_n) + N_0}) \quad (6)$$

where $B(t_n)$ is the bandwidth, $p_k(t_n)$ is the allocated transmission power during round t_n for device k, and N_0 is the additive white Gaussian noise. Assuming that the channel gains are unchanged during the learning round but vary between different scheduled steps due to the mobility of the device. The transmission time and energy for device k transmitting local model to cloud server at t_n can be characterized as,

$$T_{n,k}^{U}(t_n) = V_n / R_{n,k}(t_n) \quad (7)$$

$$E_{n,k}^{U}(t_n) = p_k(t_n) \cdot T_{n,k}^{U}(t_n) \quad (8)$$

where V_n is the model parameter size (bits) of job n.

Overall, the total amount of time and energy consumed for k within round t_n are:

$$T_{n,k}(t_n) = T_{n,k}^{L}(t_n) + T_{n,k}^{U}(t_n) \quad (9)$$

$$E_{n,k}(t_n) = E_{n,k}^{L}(t_n) + E_{n,k}^{U}(t_n). \quad (10)$$

After analyzing the per-round system cost, we turn our attention to learning efficiency, which pertains to the final performance of the global models. Indeed, a trade-off exists between these two types of efficiency. Solely focusing on minimizing the per-round system cost may force the cloud server to favor clients with fewer data samples to reduce computational costs. However, this can lead to insufficient training of the local models, negatively impacting the global aggregation and, consequently, the final model performance. To balance this trade-off, our paper introduces the concept of performance-constrained system cost. That is, we evaluate the total system cost until all jobs meet predefined performance metrics, for instance, exceeding a set accuracy threshold. In particular, for job n we set t_n^* as the required round after which the n-th job's performance $\sigma(t_n^*) \geq \sigma_n^*$.

Then the total system time delay can be expressed as:

$$\mathcal{T} = \max_{n \in \mathcal{N}} \sum_{t_n=1}^{t_n^*} (\max_{k \in K_n(t_n)} T_{n,k}(t_n)) \quad (11)$$

which implies that the total time delay hinges on the time at which the last job meets the target performance. Furthermore, the completion time for each job is influenced by the total number of rounds and the slowest client in each round to complete both training and transmission.

Then the total system energy cost can be expressed as:

$$\mathcal{E} = \sum_{n=1}^{N} \sum_{t_n=1}^{t_n^*} \sum_{k \in K_n(t_n)} E_{n,k}(t_n) \quad (12)$$

In this paper, we consider a joint Job Assignment and Resource Allocation (JARA) problem in MJ-WFL by minimizing the learning performance-constrained system cost. Specifically, we

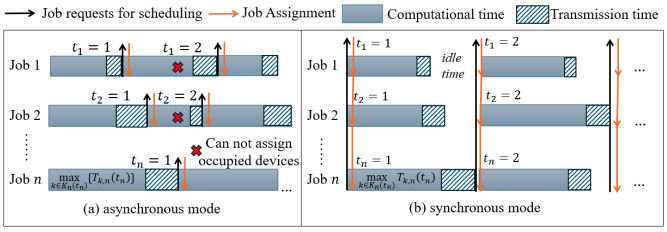


Fig. 2. Two examples of job scheduling mode.

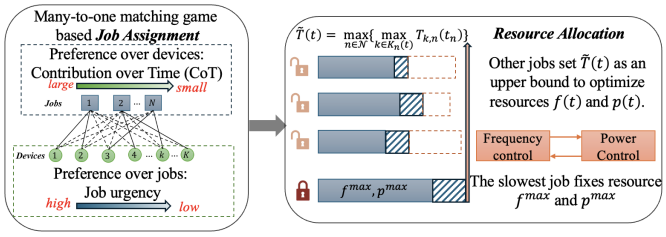


Fig. 3. Joint optimization algorithm for MJ-WFL.

aim at minimizing both the system time delay $\mathcal{T}$ and energy cost $\mathcal{E}$. The overall objective of JARA is presented as:

$$\min_{\{\mathbf{K}(t_n),\mathbf{f}(t_n),\mathbf{p}(t_n)\}_{t_n=1}^{t_n^*}} \quad \mathcal{T} \ \& \ \mathcal{E} \tag{13}$$

$$\text{s.t. } \sigma(t_n^*) \geq \sigma_n^*, \ \forall n \in \mathcal{N} \tag{14}$$

$$K_n(t_n) \cap K_{n'}(t_n) = \emptyset, \ \forall n, n' \in \mathcal{N} \tag{15}$$

$$|K_n(t_n)| \leq C_n, \ n \in \mathcal{N} \tag{16}$$

$$p_k^{min} \leq p_k(t_n) \leq p_k^{max}, \ \forall k \in \mathcal{K} \tag{17}$$

$$f_k^{min} \leq f_k(t_n) \leq f_k^{max}, \ \forall k \in \mathcal{K} \tag{18}$$

The set of variables $\mathbf{K}(t_n) = \{K_1(t_n), ..., K_N(t_n)\}$ delineates the decisions regarding job assignments, while $\mathbf{f}(t_n) = \{f_1(t_n), ..., f_K(t_n)\}$ and $\mathbf{p}(t_n) = \{p_1(t_n), ..., p_K(t_n)\}$ correspond to the allocation of computational and power resources, respectively. Constraint (14) shows our predetermined performance constraint, balancing the trade-off between system and learning efficiency. Constraints (15) and (16) ensure that a device can be assigned to only one job and that the total number of devices assigned to job n does not exceed its budget C_n. The hardware constraints are put as (17) and (18), indicating that the allocated CPU frequency and transmission power are within the constraints of hardware capabilities.

IV. Proposed Algorithm

In this section, we first examine two different modes of job scheduling. Following the selection of a preferred mode, we develop a joint optimization algorithm for job assignment and resource allocation aimed at optimizing the objectives we introduced in the previous section.

A. Job Scheduling Mode

Regarding the update strategy for t_n, which defines the job scheduling mode, there are two prevalent methods, shown in Fig. 2. The first is the asynchronous mode [13], [14], often referred to as the first-come-first-served mode, where the cloud server employs an event-driven mechanism that allows jobs to request device scheduling upon completing their respective learning rounds. In this setup, each job independently updates its t_n using a local counter, leading to asynchronous progress among the jobs due to varying round duration. In contrast, the synchronous mode [15], [16] operates under a collective scheduling policy, where all jobs proceed to the next round simultaneously after the slowest job completes its local training, ensuring synchronous advancement across jobs.

The reason we opt for the synchronous mode for our optimization algorithms is twofold: 1) Under the asynchronous mode, jobs are limited to selecting from the unoccupied devices, which leads to fluctuations in the aggregated global model because of the incomplete device information. This, in turn, could extend the total time and energy costs before achieving the target performance (Constraint (14)) since any extra round of training brings additional computation and transmission expenses. 2) The synchronous mode, despite potentially introducing idle waiting times for simpler jobs, does not conflict with our optimization objective in Eq. (13). Job idleness does not affect the overall time cost because it is determined by the slowest job. Furthermore, during waiting periods, devices operate at low idle power, which are considered negligible in this work.

The synchronous mode determines that the learning rounds for each job $t_1 = t_2 =, ..., = t_N = t$ are the same, and the strategic decisions regarding $\{\mathbf{K}(t), \mathbf{f}(t), \mathbf{p}(t)\}$ being made at the beginning of round t should be utilizing the full information of jobs and devices. In this work, we propose a two-step method: we initially fix the results of resource allocation at the max level to facilitate job assignment $\mathbf{K}(t)$ and subsequently fine-tune the CPU frequency $\mathbf{f}(t)$ and power $\mathbf{p}(t)$ for the selected devices of each job. The whole algorithm is illustrated in Fig. 3.

B. Matching Game-Based Job Assignment

In this part, our aim is to minimize the total learning time $\mathcal{T}$ in our objective through the strategic assignment of jobs. Such a problem can be formulated as a many-to-one matching game since each job can be assigned to multiple devices while each device can only undertake one job. Formally, we define the two-sided many-to-one matching game as a tuple $(\mathcal{N}, \mathcal{K}, \succ_{\mathcal{N}}, \succ_{\mathcal{K}})$. Here, the job set $\mathcal{N}$ and device set $\mathcal{K}$ act as two sets of players, $\succ_{\mathcal{N}} = \{\succ_n\}_{n\in\mathcal{N}}$ and $\succ_{\mathcal{K}} = \{\succ_k\}_{k\in\mathcal{K}}$ denote the preference relations of ENs and devices over the other side. Accordingly, the matching μ is defined as follows:

Definition 1 (Two-side matching game)**.** A many-to-one matching μ is a function from the set $\mathcal{N} \cup \mathcal{K}$ in to the set of all subsets of $\mathcal{N} \cup \mathcal{K}$ such that i) $|\mu(k)| = 1, \forall k \in \mathcal{K}$; ii) $|\mu(n)| \leq C_n, \forall n \in \mathcal{N}$; iii) $\mu(k) = n$ if and only of $k \in \mu(n)$.

Here i) implies that each device is only matched with one job; ii) states that job n can be matched with at most C_n devices; and iii) indicates that if device k is matched with job n, then job n is also matched with device k. Before the matching

process, we need to design the utility functions of both sides to build their preference relations.

1) Preference List of Jobs: This is evaluated by the utility function $r_n^t(k)$, denoted as the *Contribution over Time* (CoT) function, where the denominator indicates the contribution of device k to job n, and the numerator reflects the time introduced by device k when participating in training job n.

$$r_n^t(k) = \frac{\sum_{t'=1}^{t} \beta^{t-t'} \theta_{n,k}(t') + \sqrt{\frac{2 \ln t}{S_{n,k}(t)}}}{T_{n,k}(t)} \tag{19}$$

The design of contribution leverages the idea of exploration-exploitation from bandit algorithms [23]. The first component is an exponential forgetting function based on the instantaneous contribution of device k from past rounds. We use the *loss function reduction* at device k after each round t', defined as $F_{n,k}^{\Delta}(t')$, to measure the data quality. Together with the data quantity, we define the instantaneous contribution of device k for job n as $\theta_{n,k}(t') = 1 - e^{-\phi_1 (F_{n,k}^{\Delta}(t') D_{n,k})^{\phi_2}}$, where ϕ_1 and ϕ_2 are weight factors. An exponential forgetting function is adopted to assign weights to each instantaneous contribution, which favors more recent records by assigning them larger weights and older ones with smaller weights ($\beta < 1$). The second component is a confidence upper bound, where $S_{n,k}(t)$ indicates the number of times device k has been assigned job n. Furthermore, to ensure achieving the theoretically shortest per-round training time in the denominator, we set the maximum frequency f_k^{max} and power p_k^{max} for all devices.

The idea behind the CoT function is that the total learning time of job n hinges on the duration of per round $T_{n,k}(t)$ as well as the required rounds t_n^* before convergence. The CoT function tries to balance the trade-off between minimizing the required round t_n^* and the per-round time $T_{n,k}(t)$. Jobs should prefer devices with high-quality datasets (data heterogeneity) to minimize the required round while ensuring the per-round time does not become a bottleneck (device heterogeneity). Therefore, for any two devices $k, k' \in \mathcal{K}$, the preference relation of job n can be expressed as:

$$k \succ_n k' \Leftrightarrow r_n(k) > r_n(k') \tag{20}$$

which indicates that the job n prefers device k to k' if k can achieve a higher CoT value.

2) Preference List of Devices: As identified in our analysis, the per-round time is decided by the slowest job, while the total training time is determined by the last job that achieves its desired performance. Jobs with large individual data samples or model parameter sizes tend to require more learning time (job heterogeneity). This complexity leads to not only the longer per-round training time but also the increased number of rounds needed for convergence. Consequently, such complex jobs are more likely to become bottlenecks and should be given priority when selecting devices. In this paper, we define *job urgency* as the utility function of job n:

$$r_k^t(n) = \phi_3(\sigma_n^* - \sigma_n(t-1)) + \phi_4 * V_n \tag{21}$$

where ϕ_3 and ϕ_4 are weight factors. The two components represent the gap between the current performance and the target performance at previous round $t-1$ and the size of model parameters, respectively. A larger gap indicates a more urgent job, and a larger size suggests a more complex job, necessitating high-quality devices to participate in training to reduce the total number of learning rounds. In this work, we make all devices demonstrate a preference for jobs with high-level urgency. Specifically, for any two jobs $n, n' \in \mathcal{N}, n \neq n'$, the preference relation can be expressed as:

$$n \succ_k n' \Leftrightarrow r_k^t(n) < r_k^t(n') \tag{22}$$

By adopting this rule, jobs with urgent demands are able to obtain a higher matching priority. Overall, while constructing the preference lists mentioned above, we comprehensively considered the triple heterogeneity of MJ-WFL and balanced the dual efficiency in the objective function.

3) Matching-game Based Algorithm: We then employ the Gale-Shapley algorithm [24] to obtain the job assignment result, which is summarized in Algorithm 1. The devices and jobs first initialize their preference lists $P_k(t)$ and $P_n(t)$ according to Eq. (22) and (20). In the matching process, every device would propose its most preferred job according to its preference list. If any job n receives less than the C_n proposals, the requested job n would hold the devices as its candidates. Otherwise, if more than the C_n devices propose the same job n, the job would retain the most C_n preferred users and reject the others based on its preference list. The matching algorithm would end when every user has already been matched with one job or has been rejected by all jobs.

Algorithm 1: Matching Game-based Job Assignment (Matching-JA)

Data: Device set $\mathcal{K}$ and Job set $\mathcal{N}$
Result: Job assignment solution $\mathbf{K}(t)$
Establish the preference lists $P_k(t)$ and $P_n(t)$ for all $k \in \mathcal{K}, n \in \mathcal{N}$;
Initialize the unmatched device set $\mathcal{K}' = \mathcal{K}$;
while $\mathcal{K}' \neq \emptyset$ **do**
  for $k \in \mathcal{K}'$ **do**
    Apply for its most preferred job among who have not rejected it in its preference list ;
  end
  for $n \in \mathcal{N}$ **do**
    if $\sum_{k \in \mathcal{K}} a_{n,k}(t) < C_n$ **then**
      Job n keeps all the proposed devices.;
      Remove the matched device k from $\mathcal{K}'$;
      else
        Job n retains the most preferred C_n device and rejects others;
      end
    end
  end
end
return $\mathbf{K}(t)$

C. Resource Allocation

In this part, we further adjust the resource on the assigned devices to minimize the energy costs $\mathcal{E}$ without compromising the settled $\mathcal{T}$. From the results of job assignments, we can calculate the pre-round training delay for all jobs and sort them in descending order. Denote $n_{slw}(t)$ and $T_{slw}(t)$ as the slowest job and the corresponding time cost at round t, respectively. Since the per-round time cost is determined by the slowest job, the $\mathcal{T}$ achieved in Alg. 1 will not extend when $T_{slw}(t)$ remains unchanged. That is, devices $k \in \{\mathcal{K} \backslash K_{n_{slw}}(t)\}$ can reduce their frequency and power to optimize energy cost without compromising $\mathcal{T}$ if devices $k \in K_{n_{slw}}(t)$ maintain f_k^{max} and p_k^{max} during round t. This leaves room to process resource allocation to achieve the objective of minimizing $\mathcal{E}$. In particular, we decompose the resource allocation problem for $K_n(t)$ into two subproblems: the power control $\mathbf{p}_n$ and CPU frequency control $\mathbf{f}_n$ problems.

1) Power Control: By fixing the CPU frequency $\mathbf{f}^*(t) = \{f_k^*(t)\}_{k \in K_n(t)}$, we can eliminate the effect of computation energy cost $E_{n,k}^L(t)$, and only consider the transmission energy cost $E_{n,k}^U(t)$ minimization problem. Such a problem is equal to the power minimization problem since the transmission time is upper bounded by the fixed computation time $T_k^{L*}(t)$ and $T_{slw}(t)$. That is, power minimization gives the solution to the transmission energy minimization problem under the upper bounded transmission rate. Formally, we formulate the power minimization problem for devices in $K_n(t)$ as follows:

$$\min_{\mathbf{p}_n(t)} \sum\nolimits_{k \in \mathcal{K}_n(t)} p_k(t) \tag{23}$$

$$\text{s.t. } R_k(t) \geq \frac{V_n}{T_{slw}(t) - T_k^{L*}(t)} \tag{24}$$

$$p_k^{min} < p_k(t) < p_k^{max} \tag{25}$$

where $\mathbf{p}_n(t) = \{p_k(t)\}\}_{k \in \mathcal{K}_n(t)}$. Once $\mathbf{f}^*(t)$ and $T_k^{L*}(t)$ are fixed, the maximum tolerable transmission time for round t is restricted to $T_{slw}(t) - T_k^{L*}(t)$, where $T_k^{L*}(t) = \frac{\nu_{n,k} D_{n,k}}{f_k^*(t)}$. This further results in the lowest transmission rate $R_k(t)$ in Eq. (24). We can see that Eq. (24) and Eq. (25) are non-convex constraints, leading to the non-convexity of problem Eq. (23). To tackle this, we first introduce the intermediate variable:

$$z_k(t) = \frac{p_k(t) h_k(t)}{\sum_{j=k+1}^{|\hat{K}_n(t)|} a_{j,m}(t) p_k(t) h_k(t) + N_0} \tag{26}$$

where $\hat{K}_n(t)$ includes devices from $K_n(t)$ as well as devices that are concurrently uploading their models. Due to job heterogeneity, various jobs may complete local computing at different times, which can lead to overlaps in model uploading phases. Consequently, in light of NOMA characteristics, it is essential to incorporate all devices that are simultaneously utilizing the same channel when calculating their power levels. Then we can rewrite $R_k(t) = B(t) \log_2(1 + z_k(t))$. We next show the lower bound of $R_k(t)$ by the following inequality,

$$\log_2(1 + x) \geq \alpha \log_2(x) + \beta \tag{27}$$

where $\alpha = \frac{x_0}{1+x_0}$ and $\beta = \log_1(1 + x_0) - \frac{x_0}{1+x_0} \log_2 x_0$. The bound holds when $x = x_0$. Therefore, we denote the lower bound $\hat{R}_k(t)$ of $R_{k,m}(t)$ as follow:

$$\hat{R}_k(t) = B(t)(\alpha_k(t) \log_2(z_k(t)) + \beta_k(t)) \tag{28}$$

By defining $p_k(t) = e^{q_k(t)}$, we reformulate problem (23) as:

$$\min_{\mathbf{Q}_n(t)} \sum\nolimits_{k \in \mathcal{K}_n(t)} e^{q_k(t)} \tag{29}$$

$$\text{s.t. } \hat{R}_k(t) \geq \frac{V_n}{T_{slw}(t) - T_k^{L*}(t)} \tag{30}$$

$$p_k^{min} < e^{q_k(t)} < p_k^{max} \tag{31}$$

We consider the convexity of constraint (30). For simplicity, t is omitted. Since $\hat{R}_k$ can be rearranged as:

$$\begin{aligned}
\hat{R}_k &= B(\alpha_k \log_2(z_k) + \beta_k) \\
&= B\alpha_k \log_2\left(\frac{e^{q_k} h_k}{\sum_{j=k+1}^{\hat{K}_n} a_j e^{q_k} h_k + N_0}\right) + B\beta_k \\
&= B\alpha_k (q_k + \log_2(h_k) \\
&\quad - \log_2\left(\sum\nolimits_{j=k+1}^{\hat{K}_n} a_j e^{q_k} h_k + N_0\right)) + B\beta_k
\end{aligned}$$

The constraint Eq. (30) is concave because $\hat{R}_{k,m}$ is a log-sum-exponential function, which is convex. The constraint Eq. (31) is an exponential function, a convex function of q_k. Besides, the objective of Eq. (29) is a sum of the exponential functions and also a convex function of q_k. Then we can use convex optimization methods, like the interior point method [25], to solve it and obtain $\mathbf{Q}_n(t) = \{q_k(t)\}_{k \in \mathcal{K}_n(t)}$.

2) CPU Frequency Control: The CPU frequency control problem optimizes the $\mathbf{f}_n(t) = \{f_k^*(t)\}_{k \in K_n(t)}$ with fixed transmission power $\mathbf{p}^*(t) = \{p_k^*(t)\}_{k \in K_n}$ by only minimizing the computation energy cost in $T_{n,k}^L(t)$, which can be formulated as:

$$\min_{\mathbf{f}_n(t)} \sum\nolimits_{k \in \mathcal{K}_n(t)} v_{n,k} D_{n,k} \gamma f_k(t)^2 \tag{32}$$

$$\text{s.t. } \frac{\nu_k D_k}{f_k(t)} \leq T_{slw}(t) - \frac{V_n}{R_k^*(t)} \tag{33}$$

$$f_k^{min} < f_k(t) < f_k^{max} \tag{34}$$

where $R_k^*(t)$ depends on the transmission power $\mathbf{p}_n^*(t)$. When $f_k(t) > 0$, problem (32) is monotonically increasing with respect to $f_k(t)$. Therefore, problem (32) has a closed form solution $f_k(t) = \min\{f_k^{min}, \hat{f}_k(t)\}$,where $\hat{f}_k(t) = \frac{\nu_{n,k} D_{n,k}}{T_{slw}(t) - \frac{V_n}{R_k^*(t)}}$, obtaining by the time constraint (17). The closed form solution indicates that the frequency $f_k(t)$ should take the minimum value in the feasible set $\{f_k^{min}, \hat{f}_k(t)\}$.

The overall resource allocation algorithm is summarized in Algorithm 2. Given the job assignment solution $\mathbf{K}(t)$ as input, the frequency and power control are executed interactively until satisfying their corresponding thresholds ϵ_f, ϵ_p, or meeting the max number of iterations N_f, N_p. Specifically, given the feasible set of frequency values, we can obtain $p_k(t)$ by solving problem (29) until the end condition in Line 5 is satisfied. Each time when $p_k(t)$ is fixed, the CPU frequency

Algorithm 2: Resource allocation Algorithm

Data: Job assignment solution $\mathbf{K}(t)$
Result: Resource allocation solution $\mathbf{p}(t), \mathbf{f}(t)$

Find job $n_{slw} = \arg\max_{n\in\mathcal{N}}\{\max_{k\in K_n(t)} T_{n,k}(t)\}$;
Calculate $T_{slw}(t) = \max_{k\in K_{n_{slw}}(t)} T_{n,k}(t)$;
Set $f_k(t) = f_k^{max}$ and $p_k(t) = p_k^{max}$ for $k \in K_{n_{slw}}(t)$;
Set $\epsilon_p, \epsilon_f \in [0,1]$, N_f, N_p, $\alpha_k^{(0)}(t) = 1, f_k^{(0)}(t) = 0, f_k^{(1)}(t) = f_k^{max}$, $\forall k \in \{\mathcal{K}\backslash K_{n_{slw}}(t)\}$;
for $n \in \mathcal{N}\backslash n_{slw}$ **do**
  for $k \in \mathcal{K}_n(t)$ **do**
    while $|f_k^{(l_f)}(t) - f_k^{(l_f-1)}(t)| \geq \epsilon_f$ & $l_f < N_f$ **do**
      Randomly select set $\mathbf{Q}_n^{(0)}(t)$ and $\mathbf{Q}_n^{(1)}(t)$;
      while $|\sum_{k\in\mathcal{K}_n(t)} e^{q_k^{(l_p)}(t)}| - |\sum_{k\in\mathcal{K}_n(t)} e^{q_k^{(l_p-1)}(t)}| \geq \epsilon_p$& $l_p < N_p$ **do**
        Solve problem (29) to obtain $\mathbf{Q}_n^{(l_p)}(t)$;
        Update $z_k^{(l_p)}(t)$, $\alpha_k^{(l_p)}(t)$ and $\beta_k^{(l_p)}(t)$;
        $l_p = l_p + 1$;
      end
      Set $p_k^{(l_f)}(t) = e^{q_k^{(l_p)}(t)}$;
      Calculate $f_k^{(l)}(t) = \min\{f_k^{min}, \hat{f}_k^{(l_f)}(t)\}$;
      $l_f = l_f + 1$.
    end
  end
end

can be obtained by solving problem (32). The CPU frequency adjustment ends when the terminal conditions are met.

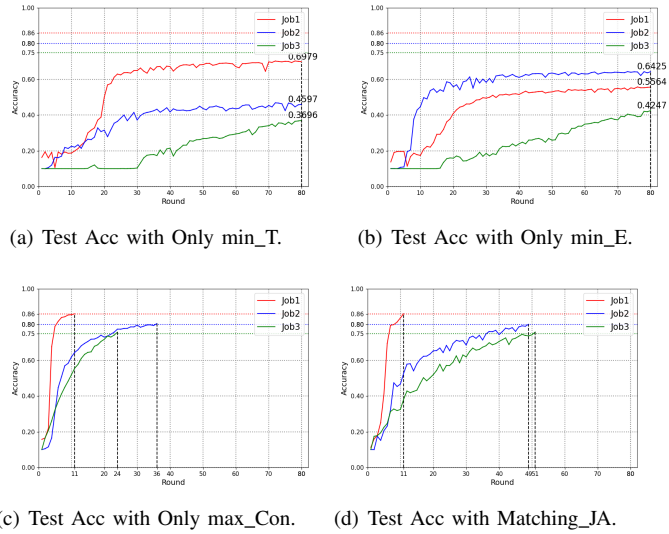
3) Complexity: The computational complexity of Alg. 1 mainly depends on the number of the device proposing. In the worst case, the proposing number is NK because it is possible for each device to propose to each job in its preference list. For the Alg. 2, we suppose that the CPU frequency and power control need at most N_f and N_p iterations to converge. Considering $C_n \in [\underline{C}, \overline{C}]$, the complexity of solving the power control problem Eq. (29) by the interior point method [25] is $O(\overline{C}^2)$. Then the complexity of the joint algorithm is $O((K-\underline{C})N_f N_p \overline{C}^2)$. Therefore, the complexity of per-round joint optimization is $O(NK+(K-\underline{C})N_f N_p \overline{C}^2)$. In practice, we can manually set the numbers of N_f and N_p to control the complexity of the whole algorithm.

V. Experiment

In this section, we first introduce the settings to simulate the triple heterogeneity in the MJ-WFL framework, followed by the analysis of the effect on job assignment (Alg. 1) and resource allocation (Alg. 2). Lastly, we also show the improvement in system efficiency brought by the NOMA.

A. Experiment Setting

To validate that our proposed method overcomes the context of triple-heterogeneity in a practical training environment of



(a) Test Acc with Only min_T. (b) Test Acc with Only min_E.

(c) Test Acc with Only max_Con. (d) Test Acc with Matching_JA.

Fig. 4. Test performance with different job assignment schemes.

MJ-WFL, we adopt the following experiment and simulation settings. 1) Job Settings: We arrange two datasets, SVHN and CIFAR-10, and two deep learning models, ResNet-50 and VGG-9, to achieve three different FL jobs, i.e., SVHN with ResNet-50, CIFAR-10 with ResNet-50, and CIFAR-10 with VGG-9. We will refer to these three jobs as Job1, Job2, and Job3, with 0.86, 0.80, and 0.75 as their target accuracy, respectively. Through different deep learning models, reaching different target accuracies for different datasets ensures job heterogeneity. 2) Dataset Settings: We use the data partitioning strategy proposed by [26] to construct local datasets for each job on 30 devices. The strategy ensures that the data samples on various clients for each job maintain different data quality, implementing data heterogeneity. 3) Device Settings: We randomly set computing frequency and transmission power for each device to guarantee device heterogeneity. The maximum frequency ranges from 1 to 8 GHz, while the maximum transmission power ranges from 0.5 to 1 watt.

B. Effect of Job Assignment

We first implement Alg. 1 to perform job assignment, where the weight factors ϕ_1, ϕ_2, ϕ_3, and ϕ_4 in the utility functions are 0.1, 0.44, 0.8, 0.2 and the forgetting weight of the instantaneous contribution $\beta = 0.8$. To verify the efficiency of our proposed job assignment algorithm, we adopt a serial method, which finishes each job one by one; an asynchronous method [13], by which jobs only select from the unoccupied devices; and three different synchronous assigning strategies as baselines. Three synchronous strategies include minimizing per-round time (Only min_T), minimizing per-round energy (Only min_E) [16], and maximizing per-round contribution (Only max_Con). Our proposed matching_JA algorithm will be compared with the baselines from both learning and system efficiency perspectives.

1) Learning Performance: Firstly, we evaluate the learning efficiency from the perspective of test accuracy. Fig. 4 shows the accuracy curves of all synchronous methods. The Only

TABLE I
SYSTEM PERFORMANCE

		Job 1	Job 2	Job 3	Total
serial	Energy cost(J)	1530.31	4266.12	2120.30	7916.73
	Round/time(s)	12	44	29	21763.14
asynchronous	Energy cost(J)	1414.89	4923.68	1800.23	8138.80
	Round/time(s)	-	-	-	13700.87
Only min_T	Energy Cost(J)	4528.58	3713.27	1669.92	9911.77
	Round/Time(s)	80	80	80	4360.24
Only min_E	Energy cost(J)	3320.48	3616.47	1273.46	8210.41
	Round/time(s)	80	80	80	13332.94
Only max_Con	Energy cost(J)	1536.54	4710.26	1948.82	8195.62
	Round/time(s)	11	36	24	13438.62
matching_JA	Energy cost(J)	1110.05	4143.35	2348.51	7601.91
	Round/time(s)	11	49	51	11886.08

min_T and the Only min_E algorithms fail to reach the preset target performance, where the Only min_T reaches 0.70, 0.46, and 0.37 on each job and the Only min_E reaches 0.56, 0.64, and 0.42 on each job, respectively. These two strategies are keen on selecting devices with small quantities of data samples to reduce the per-round energy or time consumption, leading to insufficient data quality for each training round. Due to the data heterogeneity, however, Job1 and Job2 show different increase trends on these two baselines.

The remaining baselines and our proposed algorithm reach the target accuracy. Specifically, our proposed algorithm spends 11, 49, and 51 rounds for 3 jobs, while the Only max_Con algorithm also used 11 rounds for Job1 but only used 36 and 24 rounds for Job2 and Job3, respectively. This is because the Only max_Con algorithm only considers maximizing the device contribution when allocating devices, disregarding the energy consumption and time required for training. Besides, the serial and the asynchronous methods also reach the target accuracies with fewer training rounds but do not reduce system costs. This will be explained in the following part.

2) System Performance: The serial method adopts similar assigning strategies as our algorithm but processes each job one by one, thereby achieving a similar energy cost of 7961.73 J but the longest training time of 21763.14 seconds. The total time and energy of the asynchronous method are 13700.87 seconds and 8138.80 J, neither of which surpasses the matching_JA because it only allows jobs to select devices from the unoccupied ones. With all 3 jobs undergoing 80 training rounds, neither the Only min_T nor Only min_E reached the target performance. But the costs of the system they introduced were completely different. This discrepancy is due to a trade-off between the per-round time and energy costs, leading to entirely different device preferences for each algorithm. The Only min_T algorithm minimizes time, clocking in at just 4360.24 seconds, but it does so at the expense of the highest energy cost of 9911.77 J, because it consistently favors high-performance devices with large frequencies and power. In contrast, Only min_E adopts the opposite selection criteria, prioritizing a reduction in energy per round, which results in lower energy usage but a longer running time. These two sets of results further illustrate that short-sighted algorithms not only fail to achieve learning convergence but also cannot attain satisfactory system performance.

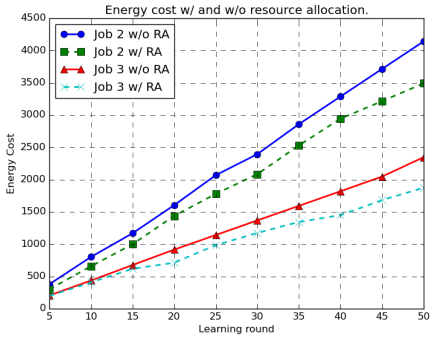


Fig. 5. Energy cost of Job 2 and Job 3 with/without resource allocation.

When comparing two synchronous algorithms that achieve the target performance, our proposed algorithm outperforms the max_Con algorithm in both energy and time consumption, with a reduction in time by 11.55% and energy cost by 7.24%. The Only max_Con favors devices with larger datasets, aiming to reduce the total number of learning rounds. As shown in Fig. 4(c), it succeeds in this goal. However, this comes at the cost of introducing longer delays and higher energy costs per round. In conclusion, our matching_JA algorithm strikes a well-balanced trade-off between contribution and per-round cost. Details are shown in Table. I.

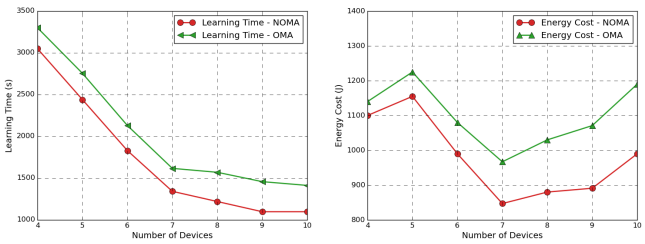
C. Effect of Resource Allocation

In this subsection, we implement Alg. 2 to perform resource adjustment based on the output of Alg. 1. As observed from the outcomes depicted in Fig. 5, the energy cost of Job2 and Job3 without resource adjustment exhibits predominantly linear growth. This trend is attributable to the strategy of operating all devices at their maximum computational frequencies and transmission powers in each round. However, following the adjustment through Alg. 2, the energy cost demonstrates a non-linear progression. The disparity in energy cost compared to the scenario without resource allocation progressively widens as the training advances. For Job2 and Job3, adopting resource adjustment led to a reduction in energy consumption by 8.5% and 20.2%, respectively, compared to no adjustment (as shown in Table. I). Additionally, it can be observed from the curve that the reduction of energy per round is not fixed. This is because we only perform resource adjustments for jobs other than n_{slw} in each round. When that job is the n_{slw} of the round, its energy consumption does not decrease. By doing this, Alg. 2 does not affect the total training time.

D. Effect of NOMA vs. OMA

In the final part, we compare the differences between using NOMA and OMA for model updates. We explore how changes in the job budget, defined as the number of devices permitted to engage in training per round, affect the time and energy cost for job 1 under NOMA and OMA.

Firstly, for time (Fig. 6(a)), both NOMA-based and OMA-based methods exhibit a decrease as the number of devices



(a) Learning Time (b) Energy Cost

Fig. 6. Performance of Job 1 with NOMA and OMA.

increases. This is because accepting more devices does not increase the per-round time (always determined by the slowest device) but contributes to faster convergence due to the increased participating data sample volume. Additionally, when the job budget increases from 7 to 10, the reduction in time becomes less pronounced, which means that adding three extra devices does not shorten the total rounds needed for convergence. The disparity between NOMA-based and OMA-based methods is primarily in the transmission time, and this gap widens as the job budget increases. The reason for this is that NOMA is better adapted to scenarios with multiple device connections. In evaluating the energy cost of NOMA-based and OMA-based methods, Fig. 6(b) shows a trend where energy initially increases with the number of devices from 3 to 4, decreases between 4 to 7 then rises again beyond 7 devices. The initial decrease is due to the reduction in learning rounds, which leads to a simultaneous reduction in both transmission and computation costs. The subsequent increase after more than 7 devices participate in training is because, while the learning rounds are no longer significantly reduced, the computing cost introduced in each round continues to rise.

VI. Conclusion

In this paper, we have addressed the complexities of a Multi-job WFL system characterized by tri-heterogeneity arising from data, devices, and jobs. Our goal is to balance the dual efficiency of the system by formulating a performance-constrained system cost minimization problem. To tackle this challenge, we introduce a joint optimization algorithm. Given the constraints on resources, we put forward a matching-game based job assignment algorithm designed to minimize the total training time. This matching process takes into consideration the preferences over both devices and job sides. After the assignment of jobs to devices, we employ a two-stage approach to fine-tune resource allocation to reduce energy costs. Our simulation experiments demonstrate superiority in terms of dual efficiency over other baselines.

References

[1] P. S. Bouzinis, P. D. Diamantoulakis, and G. K. Karagiannidis, "Wireless federated learning (WFL) for 6G networks part I: Research challenges and future trends," *IEEE Commun. Lett.*, vol. 26, no. 1, pp. 3–7, Jan. 2022.

[2] N. H. Tran, W. Bao, A. Zomaya, M. N. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *Proc. IEEE INFOCOM*, Apr. 2019, pp. 1387–1395.

[3] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 269–283, Jan. 2021.

[4] T. Li, L. Song, and C. Fragouli, "Federated recommendation system via differential privacy," in *Proc. IEEE ISIT*, Jun. 2020, pp. 2592–2597.

[5] 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.

[6] R. Chen, D. Shi, X. Qin, D. Liu, M. Pan, and S. Cui, "Service delay minimization for federated learning over mobile devices," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 990–1006, Apr. 2023.

[7] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Convergence time optimization for federated learning over wireless networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 4, pp. 2457–2471, Apr. 2021.

[8] M. Kim, W. Saad, M. Mozaffari, and M. Debbah, "Green, quantized federated learning over wireless networks: An energy-efficient design," *IEEE Trans. Wireless Commun.*, vol. 23, no. 2, pp. 1386–1402, Feb. 2024.

[9] L. Yu, R. Albelaihi, X. Sun, N. Ansari, and M. Devetsikiotis, "Jointly optimizing client selection and resource management in wireless federated learning for Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 6, pp. 4385–4395, Mar. 2022.

[10] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "Performance optimization of federated learning over wireless networks," in *Proc. IEEE GLOBECOM*, Dec. 2019, pp. 1–6.

[11] S. Wang, M. Chen, C. G. Brinton, C. Yin, W. Saad, and S. Cui, "Performance optimization for variable bitwidth federated learning in wireless networks," *IEEE Trans. Wireless Commun.*, vol. 23, no. 3, pp. 2340–2356, Mar. 2024.

[12] Q. Wang, X. Mei, H. Liu, Y.-W. Leung, Z. Li, and X. Chu, "Energy-aware non-preemptive task scheduling with deadline constraint in DVFS-enabled heterogeneous clusters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 4083–4099, Dec. 2022.

[13] C. Zhou et al., "Efficient device scheduling with multi-job federated learning," in *Proc. AAAI*, vol. 36, no. 9, Feb. 2022, pp. 9971–9979.

[14] N. Bhuyan and S. Moharir, "Multi-model federated learning," in *Proc. COMSNETS*, Jan. 2022, pp. 779–783.

[15] Z. Li, H. Wu, and Y. Lu, "Coalition based utility and efficiency optimization for multi-task federated learning in Internet of Vehicles," *Future Gener. Comput. Syst.*, vol. 140, pp. 196–208, Mar. 2023.

[16] Z. Li, H. Wu, Y. Lu, B. Ai, Z. Zhong, and Y. Zhang, "Matching game for multi-task federated learning in Internet of Vehicles," *IEEE Trans. Veh. Technol.*, vol. 73, no. 2, pp. 1623–1636, Feb. 2024.

[17] J. Zhao, Y. Feng, X. Chang, and C. H. Liu, "Energy-efficient client selection in federated learning with heterogeneous data on edge," *Peer-to-Peer Networking Appl.*, vol. 15, no. 2, pp. 1139–1151, 2022.

[18] W. Xia, W. Wen, K.-K. Wong, T. Q. Quek, J. Zhang, and H. Zhu, "Federated-learning-based client scheduling for low-latency wireless communications," *IEEE Wireless Commun.*, vol. 28, no. 2, pp. 32–38, Apr. 2021.

[19] K. Wei et al., "Low-latency federated learning over wireless channels with differential privacy," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 1, pp. 290–307, Jan. 2022.

[20] Z.-y. Chai, C.-d. Yang, and Y.-l. Li, "Communication efficiency optimization in federated learning based on multi-objective evolutionary algorithm," *Evol. Intell.*, vol. 16, no. 3, pp. 1033–1044, 2023.

[21] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in IoT," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5986–5994, Jul. 2020.

[22] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. USENIX HotCloud*, Jun. 2010, pp. 1–7.

[23] T. Lattimore and C. Szepesvári, *Bandit algorithms*. Cambridge University Press, 2020.

[24] J. Sethuraman, C.-P. Teo, and L. Qian, "Many-to-one stable matching: Geometry and fairness," *Math. Oper. Res.*, vol. 31, no. 3, pp. 581–596, Aug. 2006.

[25] J. Papandriopoulos and J. S. Evans, "Low-complexity distributed algorithms for spectrum balancing in multi-user DSL networks," in *Proc. IEEE ICC*, Jun. 2006, pp. 3270–3275.

[26] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-IID data silos: An experimental study," in *Proc. IEEE ICDE*, May 2022, pp. 965–978.