INTELLIGENT DEVICE SELECTION IN FEDERATED EDGE LEARNING WITH ENERGY EFFICIENCY

by

Cheng Peng

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science

Department of Computer and Information Science

Indianapolis, Indiana

December 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Qin Hu, Chair

Department of Computer And Information Science, IUPUI

Dr. Kyubyung Kang

School of Construction Management Technology, Purdue University

Dr. Xukai Zou

Department of Computer And Information Science, IUPUI

Approved by:

Dr. Shiaofen Fang

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Due to the increasing demand from mobile devices for the real-time response of cloud computing services, *federated edge learning* (FEL) emerges as a new computing paradigm, which utilizes edge devices to achieve efficient machine learning while protecting their data privacy. Implementing efficient FEL suffers from the challenges of devices' limited computing and communication resources, as well as unevenly distributed datasets, which inspires several existing research focusing on *device selection* to optimize time consumption and data diversity. However, these studies fail to consider the energy consumption of edge devices given their limited power supply, which can seriously affect the cost-efficiency of FEL with unexpected device dropouts.

To fill this gap, we propose a device selection model capturing both energy consumption and data diversity optimization, under the constraints of time consumption and training data amount. Then we solve the optimization problem by reformulating the original model and designing a novel algorithm, named E^2DS, to reduce the time complexity greatly. By comparing with two classical FEL schemes, we validate the superiority of our proposed device selection mechanism for FEL with extensive experimental results.

Furthermore, for each device in a real FEL environment, it is the fact that multiple tasks will occupy the CPU at the same time, so the frequency of the CPU used for training fluctuates all the time, which may lead to large errors in computing energy consumption. To solve this problem, we deploy reinforcement learning to learn the frequency so as to approach real value. And compared to increasing data diversity, we consider a more direct way to improve the convergence speed using loss values. Then we formulate the optimization problem that minimizes the energy consumption and maximizes the loss values to select the appropriate set of devices. After reformulating the problem, we design a new algorithm FCE^2DS as the solution to have better performance on convergence speed and accuracy. Finally, we compare the performance of this proposed scheme with the previous scheme and the traditional scheme to verify the improvement of the proposed scheme in multiple aspects.

1. INTRODUCTION

Nowadays, the demand for the timely and reliable response of cloud computing services becomes increasingly extensive due to the prevailing of real-time applications on mobile devices, which reveals multiple critical deficiencies of the traditional central cloud computing. On the one hand, wireless communication channels may suffer from high latency, low bandwidth, and instability. On the other hand, the security of the cloud center has long been criticized, and data privacy can be easily breached under the attacks of malicious third parties. Regarding the first disadvantage, leveraging on the large-scale deployment of 5G technology, edge computing has played an important role in providing flexible and efficient services to end users. To solve the second problem, federated learning (FL) with privacy protection characteristics can effectively assist in aggregating necessary information to improve training efficiency, so as to provide automatic and intelligent computing. Combining the advantages of the above two solutions, federated edge learning (FEL) emerges recently to holistically address the problem of cloud computing for achieving efficient machine learning with the help of edge devices while protecting their data privacy at the same time.

However, due to the uncertain communication conditions and computing capabilities of edge devices without owning independent and identically distributed (IID) data, there exist huge challenges in efficiently implementing FEL. Faced with such a situation, a lot of existing studies are devoted to the research of server-oriented optimization in the FEL process, including *resource allocation* [1]–[5], *intermediate server assistance* [6], [7] and *device participation incentive* [8]–[10]. Although these methods are optimized under the premise of a given set of edge devices contributing to FEL, the participation of some devices lacking computing and communication capabilities or training data can negatively affect the training performance of FEL.

Beyond the server-oriented optimization, many researchers have tried to improve FEL from the perspective of devices, which mainly includes three types of studies, i.e., such as *model optimization* [11]–[15], *resource balance* [16]–[18] and *device selection* [19]–[24]. Among them, model optimization may sacrifice the accuracy of the final model to a certain extent, while resource balance usually assigns extra resources to low-quality participants, which can

be unfair for other high-quality devices and thus discouraging their future participation. More importantly, involving all available devices in the learning process in the first two types of schemes can heavily degrade the overall performance of FEL systems. Thus, device selection becomes a better solution under the premise of balancing resource consumption and efficiency. In fact, there are several recent works [19]–[22] focusing on device selection for FEL. But they fail to include the impact of devices' energy cost on device selection, except for time consumption and data diversity. As the power of edge devices is usually provided by built-in batteries, excessive energy consumption of participated devices may cause unanticipated dropouts and significantly lower the cost-efficiency of FEL.

To overcome the shortcomings of the existing device selection mechanisms, we consider minimizing the energy consumption of all selected devices in each round of training, which is under the constraints of communication and computing time consumption of each device, as well as the total amount of data for training. In addition, the number of devices selected for FEL in each round is also designed to be maximized so as to increase the diversity of data and thus accelerate the speed of model convergence.

However, it is nontrivial to solve the above minimization problem and the time complexity of intuitively solving this problem increases exponentially with the number of devices, which imposes a burden on the server to determine the appropriate set of devices in each FEL round and can further reduce the training efficiency of FEL.

In this paper, we firstly solve the above challenge by defining an optimization problem considering the time and energy consumption in the communication and calculation phases, as well as the number of selected devices and the local data sizes of devices. Afterwards, the optimization problem is solved by designing an efficient algorithm named E^2DS with lower time complexity. The main contributions of this work are summarized as follows:

- By elaborating the FEL procedures, we insert a device selection step in the traditional FL process, which is determined by the server through solving an optimization problem.
- Considering the limited waiting time constraint of the server and the required size of training data in FEL, we establish a mathematical model to minimize the total energy

consumption of selected devices in the communication and computing processes, as well as maximize the number of selected devices for data diversity consideration.

- To reduce the calculation complexity of the server in the stage of device selection, we reformulate the proposed optimization problem and propose an efficient solution, namely E^{2}DS, that achieves the optimization goal but brings no negative impact to the system performance of FEL.

- We implement a series of experiments to evaluate the performance of our proposed E^{2}DS scheme via investigating time and communication cost. Besides, by comparing with two classical methods, we demonstrate that E^{2}DS can perform better to save device-average energy consumption while achieving great learning performance, i.e., high accuracy with fast convergence.

The details of the model, solution and experiments are presented in Chapter 3.

Based on this work, we find that in the actual FEL environment, the CPU of each device is consumed by multiple tasks at the same time, so the CPU-cycle frequency of each device will fluctuate all the time during the whole FEL task. Since the step of collecting information of all devices for estimating energy consumption and time consumption occurs at the beginning of each communication round, including the CPU-cycle frequency, this leads to errors compared with the actual frequency while training. Therefore, instead of selecting devices only based on the information at the beginning of each communication round, we learn to obtain the frequency that is closer to the true value, so as to calculate energy consumption more accurately. In addition, the convergence speed is also an important factor to measure the performance of an FEL system. In a real FEL scenario, selecting biasing devices with higher local losses is able to speed up the convergence [23]. Therefore, we consider the loss value as an important parameter that affects the overall performance of the FEL system. We formulate the optimization problem, which is to minimize total energy consumption and maximize the loss values of selected devices at the same time in every communication round, so as to save energy and accelerate the speed of model convergence. Finally, we design an improved algorithm called FCE^{2}DS to solve this problem after reformulation, and verify the

efficiency and performance through a series of experiments. The main contributions of this work are summarized as follows:

- To accelerate the convergence speed of the global FEL model, we propose an energy-efficient loss model that the server can collect and estimate the local loss value from each device.
- Considering the time limit of each communication round and the required data size for FEL training, we propose an optimization problem to minimize the total energy consumption of selected devices in the communication and computing processes, as well as maximize the total loss values of selected devices.
- To achieve the optimization goal as well as reduce the calculation complexity in the step of device selection, we reformulate the proposed optimization problem and propose an FCE^{2}DS solution that has a good performance with the FEL system.
- To eliminate the estimation error of time and energy consumption caused by the fluctuation of the CPU-cycle frequency, we introduce a reinforcement learning method to learn the frequency in each communication round, so as to approach the real frequency and calculate the energy consumption more accurately.
- To evaluate the performance of our proposed FCE^{2}DS scheme, a series of experiments verify the high performance and high energy efficiency of the FCE^{2}DS scheme, compared with E^{2}DS scheme and Traditional scheme.

The detail of model, solution and experiments are presented in Chapter 4, and Chapter 6 concludes the whole paper.

2. RELATED WORK

As the applications of FL at the edge become popular, numerous studies have been conducted to improve FEL performance. The existing research on FEL optimization can be roughly divided into two categories, *server-oriented optimization* and *device-oriented optimization*.

Many researchers worked on server-oriented optimization through *resource allocation* [1]–[5], *intermediate server assistance* [6], [7] and *device participation incentive* [8]–[10]. Yang *et al.* [1] optimized communication bandwidth allocation to accelerate global model aggregation via superposing wireless multiple-access channels. Abad *et al.* [2] targeted at reducing communication delay by introducing mobile base stations for resource allocation. Further, Ren *et al.* [3] studied the strategy design using deep reinforcement learning for allocating communication and computing resources to reduce transmission cost during FEL. In addition, Chen *et al.* [4] designed a FL solution to increase model convergence speed by adjusting spectrum allocation and computing resources. Zhang *et al.* [5] proposed a resource management solution based on multi-agent learning to dynamically allocate spectrum and computing resources. By introducing intermediate edge servers, an algorithm employing multiple edge servers to aggregate models were proposed in [6] to speed up the convergence of FL, while Ye *et al.* [7] proposed setting up intermediate edge servers for small-scale aggregation to reduce both total communication and computing cost of mobile devices. Besides, through incentive mechanism designs, researchers in [8] and [9] studied the criteria for measuring the contribution of devices to global model updates and motivating devices to participate in alliance learning. Hu *et al.* [10] proposed a device participation decision strategy based on the correlated equilibrium to maximize both the individual and global profits.

As the server-oriented solutions, like resource allocation, will cause extra server load, and the assistance of the intermediate server requires additional stations to be built, many researchers have studied device-oriented optimization schemes in FEL, such as *ML model optimization* [11]–[15], *resource balance* [16]–[18] and *device selection* [19]–[24]. Xu *et al.* [11] studied training data preprocessing by introducing the CNN model into FEL to improve learning performance. Yu *et al.* [12] considered reducing training rounds using the proactive content caching algorithm to improve training efficiency. In addition, a model pruning

algorithm was proposed in [13] to adjust the model size and reduce training time for devices. In [14], Li *et al.* proposed a communication compression solution to reduce the energy consumption, while Sattler *et al.* also focused on communication compression, as well as the robustness of the training model in [15]. Zeng *et al.* [16] investigated resource so as to guarantee that all devices can submit local model updates using similar time cost to device condition, where the weaker channel status and computing capability would be provided with more bandwidth so as to guarantee that all devices can submit local model updates using similar time cost. Not only considering bandwidth allocation, Taïk *et al.* proposed scheduling algorithm based on data quality also considering the diversity of data size in different devices to increase the convergence speed in [17]. Ye *et al.* [18] studied the effect of bandwidth allocation algorithm with device selection on the improvement of convergence speed and proved the optimality of the proposed algorithm. Based on device selection, Wu *et al.* [19] adjusted the selection set of devices by comparing the efficiency of previous local model training of devices, while Amiri *et al.* [20] considered the channel conditions of devices and the importance of local model updates from each device. The time consumption in local computing and communication was the criteria for device selection in [21], and Zhang *et al.* [22] studied to reduce the impact of non-IID data on FEL performance through device selection. Besides, Cho *et al.* [23], [24] investigated the relationship between the loss value and convergence speed. By selecting the devices with higher local training loss values, the convergence speed can be rapidly accelerated and the communication efficiency can be improved.

However, due to the complex environment of edge devices and servers, the aforementioned studies only consider single factors affecting FEL performance, i.e., time consumption or data adversity. In addition, as an indispensable resource, energy consumption is rarely considered in the existing studies, which will lower the cost-efficiency of FEL. To fill the gaps, we first propose a comprehensive device selection scheme that takes into account the energy and time consumption in both local calculation and communication process, as well as the diversity of training data. Furthermore, since the convergence speed is also significant to the performance of the FEL system, based on the first algorithm we come up with a more comprehensive scheme that optimizes loss values of devices so as to increase the convergence

speed in a more direct way. Overall, through the device selection process before each round of learning, the total energy consumed during the FEL process will be significantly reduced with fast convergence.

3. ENERGY-EFFICIENT DEVICE SELECTION IN FEDERATED EDGE LEARNING

3.1 SYSTEM MODEL

In this paper, we consider an FEL system, consisting of one edge server and multiple devices that have been activated on the edge server. We denote the set of devices as $\mathcal{K} = \{1, 2, \cdots, k, \cdots, K\}$. Considering that numerous devices could exist, a subset of devices will be selected to participate in each round of FEL for both time and communication efficiency. For simplicity, we define a binary variable $x_k \in \{0, 1\}$ to indicate whether device k is selected or not, where $x_k = 1$ denotes that device k is chosen for local training in this FEL round while $x_k = 0$ means not being picked.

In this case, how to optimize the selection of devices based on the cost-efficiency criteria becomes an essential step to improve the system performance. To that aim, we consider selecting devices with a trade-off between resource consumption and learning performance in this paper, where the selected devices can provide the required number of data samples for model training and satisfy the time limitation of local training and updating while consuming the minimum amount of total energy.

In this section, we model the optimal device selection problem via elaborating the overall FEL working process with device selection considering resource consumption formulating an optimization problem with constraints.

3.1.1 FEL PROCESS

We first define a waiting time limitation, denoted as T^{wait}, for the server to collect local model updates from selected devices in each round, which is usually a pre-defined constant parameter for a certain FEL task. Then we present the FEL interaction process between the edge server and candidate devices during each round in Figure 3.1, involving six main steps as described below.

① **Device Information Collection.** To determine the best subset of devices satisfying the basic time limitation T^{wait} and other additional requirements, the server needs to

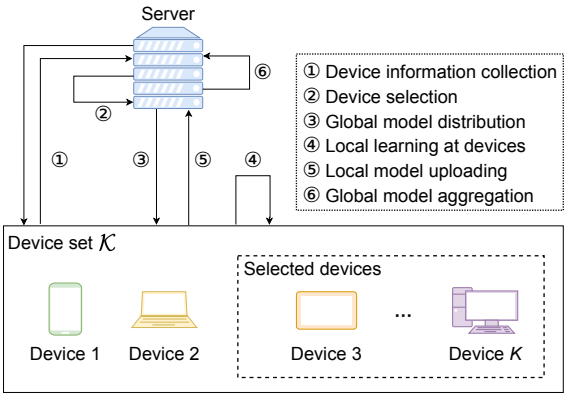


Figure 3.1. The working process of an FEL round.

collect necessary information from devices, including both computation and communication related parameters, such as local data size, CPU computing power, CPU-cycle frequency, bandwidth and transmission power.

② **Device Selection.** After the server receives requested information from all devices or reaches the maximum waiting time, the server will start to calculate the best selection plan of appropriate devices for optimizing system performance, which turns out to be the research focus of this work and will be solved in the next section.

③ **Global Model Distribution.** After the device selection step, the server sends the parameters of the global model to all selected devices.

④ **Local Learning at Devices.** Once devices receive parameters from the server, they will use their local data to train the local model.

⑤ **Local Model Uploading.** When the local model converges, each device sends the parameter updates back to the server.

⑥ **Global Model Aggregation.** Even some of the devices may fail to submit local updates within T^{wait} due to networking condition, the server aggregates the received results to derive the updated global model.

The above steps will be repeated for multiple rounds until the maximum number of FL rounds or the required model performance is reached.

3.1.2 TIME CONSUMPTION MODEL

As we mentioned above, in order to achieve efficient FEL, the server needs to select a subset of devices that can upload their locally learned model updates within T^{wait} instead of endless waiting. As we can see from Figure 4.1, the main time consumption for devices in each round of FEL is spent for steps ③ to ⑤, which will be specifically calculated in this subsection. It is worthy noting that since other steps, i.e., steps ①, ②, and ⑥, are mainly operated by the server with relatively powerful computing and communication capabilities, which is clearly independent of devices, so we skip discussing the time consumption during these steps.

Steps ③ - ⑤ describe the parameters updating process of devices. We denote the time spent of device k in the step of global model distribution by T_k^D and that in the step of local model uploading by T_k^U, which are generally determined by the value of model parameter size divided by the transmission speed and can be respectively calculated as:

$$T_k^D = \frac{D^p}{V_k^D},$$

$$T_k^U = \frac{D^p}{V_k^U}.$$

In the above equations, D^p is the size of model parameters; V_k^D and V_k^U are respectively the transmission speeds of downloading and uploading that are closely related to the condition of communication channels and can be defined as:

$$V_k^D = B_k^D \log(1 + \frac{P_k h_k^2}{N_0}), \tag{3.1}$$

$$V_k^U = B_k^U \log(1 + \frac{P_k h_k^2}{N_0}). \tag{3.2}$$

In (3.1) and (3.2), B_k^D and B_k^U are respectively the download and upload bandwidth of device k, P_k and h_k are respectively the transmission power and the channel gain of device k [21], and N_0 is the background noise. Specifically, B_k^D, B_k^U and N_0 are constant parameters, while

P_k and h_k are device information provided by each candidate device in the step of device information collection.

We denote c_k as the number of CPU cycles for each device k to complete training one sample of data, which can be measured locally and reported to the server. And the total number of CPU cycles needed for device k in each FEL round is $c_k D_k$ with D_k denoting the size of its local dataset. Given the CPU-cycle frequency of device k denoted by f_k, the time spent in the local learning step of device k, denoted by T_k^{LC}, can be calculated as[25]:

$$T_k^{LC} = \frac{c_k D_k}{f_k}.$$

Combining the time spent in each step, the total time consumption of device k in each FEL round, denoted as T_k, can be calculated as:

$$T_k = T_k^D + T_k^{LC} + T_k^U.$$

3.1.3 ENERGY CONSUMPTION MODEL

Similar to the time consumption model defined above, steps ③ to ⑤ consumes the energy of any device mainly in each round of FEL, which will be calculated in this subsection. In the rest of steps, the energy is mainly consumed for the server. Since it is not closely related to devices, and the energy supply of the server is sufficient, we skip to discuss the energy consumption during these steps.

Since the energy consumption during transmission is the product of transmission power and transmission time, we can calculate the energy consumption of device k in the step of global model distribution and local model uploading, denoted by E_k^D and E_k^U, respectively, as follows:

$$E_k^D = T_k^D P_k,$$

$$E_k^U = T_k^U P_k.$$

In the step of local learning at devices, the energy consumed of device k, denoted as E_k^{LC}, is closely related to the data amount and CPU frequency, which is expressed as[26]:

$$E_k^{LC} = \frac{\alpha_k}{2} c_k D_k f_k^2.$$

In the above equation, α_k is the effective capacitance coefficient of the computing chip-set in device k.

Thus, the overall energy consumption for device k, denoted as E_k, in each FEL round can be expressed as:

$$E_k = E_k^D + E_k^{LC} + E_k^U. \tag{3.3}$$

3.1.4 PROBLEM FORMULATION

Considering that edge devices are usually battery-powered with limited energy supply, we aim to minimize the total energy consumption of all participated devices so as to avoid unexpected device dropouts for improving the cost-efficiency of FEL. According to (4.9), the total energy consumption of selected devices can be expressed as $\sum_{k=1}^{K} E_k x_k$.

Besides, the number of participated devices is also an important parameter affecting the overall performance of FEL. In a real federated learning scenario, the quality and quantity of data from different devices can be highly diverse, and the data samples usually cannot meet the assumptions of IID. If the data distribution is severely skewed (i.e, non-IID), the accuracy of the training results will be severely reduced [27]. Thus, in the device selection stage, it is significant to increase the number of devices for improving the diversity of training data, so as to increase the accuracy of the finally trained model. Given x_k denoting the state indicator of each selected device, the number of devices can be calculated by $\sum_{k=1}^{K} x_k$.

As we mentioned at the beginning of this section, there is a time limitation T^{wait} for the server to collect local model updates, which should be a common constraint for all devices. In addition, the total amount of data for training affects the performance of FEL where excessive or extremely few data will affect the convergence speed of the training process and the model accuracy. Here we employ a parameter a to depict the ratio of the required amount of data in FEL to the total amount of data $D_{all} = \sum_{k=1}^{K} D_k$.

To achieve the above goals, we formulate an optimization problem as follows:

$$\min: \ \eta \sum_{k=1}^{K} E_k x_k - \theta \sum_{k=1}^{K} x_k, \tag{3.4}$$

$$\text{s.t.}: \ x_k \in \{0, 1\}, \tag{3.4a}$$

$$T_k \leq T^{wait}, \tag{3.4b}$$

$$\sum_{k=1}^{K} D_k x_k \geq a \cdot D_{all}. \tag{3.4c}$$

In the above formulation, the objective (3.4) is to minimize the difference between the total energy consumption and the total number of selected devices, with η and θ for value balancing. Constraint (3.4a) ensures that the state indicator should only be 1 for being selected, or 0 for not being selected. Constraint (3.4b) states that the overall time consumption of each selected device should be less than the maximum waiting time T^{wait} set by the server. And the last constraint (3.4c) guarantees that the overall data amount of selected devices has to be enough for training the model in each round.

3.2 ALGORITHM DESIGN FOR ENERGY-EFFICIENT DEVICE SELECTION

In this section, we design a specific algorithm to solve the optimization problem defined in (3.4), which consists of the problem reformulation in Section 3.2.1 and detailed algorithm design in Section 3.2.2.

3.2.1 PROBLEM REFORMULATION

From (3.4), we see that the optimization problem is to find the best subset of devices that minimizes energy consumption and maximizes the number of selected devices, under the constraints of time consumption and training data size. Solving the above optimization problem is nontrivial as it requires a complex combinatorial optimization where the difference between the total energy consumption and the number of selected devices needs to be

minimized. An intuitive solution of this problem is to traverse all combinations of devices and then compare them to obtain the optimal one. Since each device has two states of being selected and not being selected, it costs the time complexity of $O(2^K)$ to traverse K devices in total, which will undoubtedly cause a huge time consumption in each FEL round and seriously affect the training efficiency. Therefore, we transform the problem defined in (3.4) to an efficient maximization problem and propose a solution based on dynamic programming with a lower time complexity.

For simplicity, we denote $\mathcal{K}_1$ as the set of selected devices and $\mathcal{K}_0$ as the set of unselected devices, where $\mathcal{K}_1 \cap \mathcal{K}_0 =$ and $\mathcal{K}_1 \cup \mathcal{K}_0 = \mathcal{K}$. Obviously, the size of $\mathcal{K}_1$, denoted as K_1, can be calculated by:

$$K_1 = \sum_{k=1}^{K} x_k,$$

and the size of $\mathcal{K}_0$, denoted as K_0, can be calculated by:

$$K_0 = \sum_{k=1}^{K} (1 - x_k). \tag{3.5}$$

Clearly, the state indicator of device k in $\mathcal{K}_0$ is $x_k = 0$. Then, we can use y_k to indicate the state of device k which is not selected for joining FEL in this round, so we have $y_k = 1 - x_k$. In this case, (3.5) can be rewritten as:

$$K_0 = \sum_{k=1}^{K} y_k.$$

Next, we respectively denote the sum of energy consumption of devices in set $\mathcal{K}_1$ and that in set $\mathcal{K}_0$ as E_1 and E_0. According to the definitions of two sets mentioned above, E_1 and E_0 are calculated as:

$$E_1 = \sum_{k=1}^{K} E_k x_k,$$

$$E_0 = \sum_{k=1}^{K} E_k (1 - x_k) = \sum_{k=1}^{K} E_k y_k.$$

Then the total energy consumption of all devices, denoted as E_{all}, is the sum of E_1 and E_0, which is expressed as:

$$E_{all} = E_1 + E_0 = \sum_{k=1}^{K} E_k x_k + \sum_{k=1}^{K} E_k y_k. \tag{3.6}$$

In the above equation, E_{all} is clearly a fixed value in each FEL round.

In addition, the total data amount D_{all}, which appears in one of the constraints in optimization problem (3.4), can also be rewritten according to the above-defined two device sets $\mathcal{K}_1$ and $\mathcal{K}_0$. Specifically, the data amounts of devices in $\mathcal{K}_1$ and $\mathcal{K}_0$ is respectively denoted by D_1 and D_0, which are calculated as follows:

$$D_1 = \sum_{k=1}^{K} D_k x_k,$$

$$D_0 = \sum_{k=1}^{K} D_k y_k,$$

and the total amount of data D_{all} is the sum of D_1 and D_0, which is expressed as:

$$D_{all} = \sum_{k=1}^{K} D_k x_k + \sum_{k=1}^{K} D_k y_k. \tag{3.7}$$

Since D_{all} is a fixed value that only related to the set of total devices $\mathcal{K}$ in each round, the constraint (3.4c) can be transferred to:

$$\sum_{k=1}^{K} D_k y_k \leq (1 - a) \cdot D_{all}. \tag{3.8}$$

According to (3.6) and the transformed constraint (3.8), the optimization problem defined in (3.4) can be rewritten into:

$$\min: \ (\eta E_{all} - \theta K) - (\eta \sum_{k=1}^{K} E_k y_k - \theta \sum_{k=1}^{K} y_k), \tag{3.9}$$

$$\text{s.t.}: \ y_k \in \{0, 1\}, \tag{3.9a}$$

$$T_k \leq T^{wait}, \tag{3.9b}$$

$$\sum_{k=1}^{K} D_k y_k \leq (1 - a) \cdot D_{all}. \tag{3.9c}$$

As mentioned above, ηE_{all} and θK are fixed values for each FEL round. Since we aim to minimize the difference between this fixed value $\eta E_{all} - \theta K$ and $\eta \sum_{k=1}^{K} E_k y_k - \theta \sum_{k=1}^{K} y_k$ in the optimization problem (3.9), it is equivalent to maximize $(\eta \sum_{k=1}^{K} E_k y_k - \theta \sum_{k=1}^{K} y_k)$. Thus, problem (3.9) can be reformulated as:

$$\max: \ \sum_{k=1}^{K} y_k(\eta E_k - \theta), \tag{3.10}$$

$$\text{s.t.}: \ y_k \in \{0, 1\}, \tag{3.10a}$$

$$T_k \leq T^{wait}, \tag{3.10b}$$

$$\sum_{k=1}^{K} D_k y_k \leq (1 - a) \cdot D_{all}. \tag{3.10c}$$

3.2.2 ALGORITHM DESIGN

In the above, we introduce two fixed values and use the complementary relationship to transform the original minimization problem defined in (3.4) into a maximization problem defined in (3.9). To solve it, we propose an Energy-Efficient Device Selection (E^2DS) algorithm which is specified in Algorithm 1.

Generally, we can see that the above maximization problem can be transferred to a 0-1 Knapsack problem. In the 0-1 Knapsack problem, there are a number of different items and a knapsack with limited capacity. Each item has its own value and can be selected to be put

into the Knapsack or not, and the purpose is to find out which set of items being packed into the knapsack can maximize the total value under the capacity limitation.

Similarly, in our problem defined in (3.9), there are K different devices that can be selected or not for joining FEL, the capacity is the maximum remaining data size $(1-a)\cdot D_{all}$ with an extra constraint of time consumption. For the sake of convenience, we denote $D^{cap}=(1-a)\cdot D_{all}$ as the data size capacity of unselected devices, and $v_k=\eta E_k-\theta$ as the value of device k. Then we denote a sequence $\mathcal{V}=\{v_1,v_2,\cdots,v_K\}$ to contain the values of all devices. The sequence of data size is denoted by $\mathcal{D}=\{D_1,D_2,\cdots,D_K\}$.

Algorithm 1 E^{2}DS

Input: the value set of devices $\mathcal{V}=\{v_1,v_2,\cdots,v_K\}$, the size set of devices $\mathcal{D}=\{D_1,D_2,\cdots,D_K\}$, the number of devices K, the data size capacity of selected devices D^{cap}

Output: the state indicators of all devices for $\mathcal{K}_1$ $\{x_1,x_2,\cdots,x_K\}$

1: $\{y_1,y_2,\cdots,y_K\}\leftarrow UDD(\mathcal{V},\mathcal{D},K,D^{cap})$
2: **for** i $\leftarrow$ 1 to K **do**
3: $\quad x_{\mathrm{i}}\leftarrow 1-y_{\mathrm{i}}$
4: **end for**
5: **return** $\{x_1,x_2,\cdots,x_K\}$

As shown in Algorithm 1, there are four input parameters: the value set $\mathcal{V}$ presents the value of each device; the set of weight $\mathcal{D}$ presents the dataset size of each device; the parameter K presents the number of devices in the edge; the parameter D^{cap} presents the maximum data size. The set of state indicators of all devices $\{x_1,x_2,\cdots,x_K\}$ describes the result of device selection, where the state of device k with $x_k=1$ will be selected to participate in FEL of this round. Specifically, we design a dynamic programming based algorithm named Unselected Device Decision (UDD), which will be detailed in Algorithm 2, to calculate the set of unselected devices (Line 1). Then, by traversing devices from 1 to K, all the indicators of devices will be converted (Lines 2 - 4) to meet our objective in (3.4).

In Algorithm 2, a two-dimensional array $DSA(K,D^{cap})$ is defined to store the intermediate results of dynamic programming. After initializing the array $DSA(K,D^{cap})$ (Lines 1 - 6), we complete the calculation of $DSA(K,D^{cap})$ (Lines 7 - 19). By traversing devices from 1 to K, and data size from 1 to D^{cap}, $DSA(\mathrm{i},D)$ will be filled in with the calculated optimal

Algorithm 2 UDD

Input: the value set of devices $\mathcal{V} = \{v_1, v_2, \cdots, v_K\}$, the size set of devices $\mathcal{D} = \{D_1, D_2, \cdots, D_K\}$, the number of devices K, the data size capacity of selected devices D^{cap}

Output: the state indicators of all devices for $\mathcal{K}_0$ $\{y_1, y_2, \cdots, y_K\}$

1: **for** $D \leftarrow 0$ to D^{cap} **do**
2: $DSA(0, D) \leftarrow 0$
3: **end for**
4: **for** i $\leftarrow 1$ to K **do**
5: $DSA(\mathrm{i}, 0) \leftarrow 0$
6: **end for**
7: **for** i $\leftarrow 1$ to K **do**
8: **for** $D \leftarrow 1$ to D^{cap} **do**
9: **if** $D_\mathrm{i} \leq D$ **then**
10: **if** $v_\mathrm{i} + DSA(\mathrm{i}-1, D-D_\mathrm{i}) > DSA(\mathrm{i}-1, D)$ **then**
11: $DSA(\mathrm{i}, D) \leftarrow v_\mathrm{i} + DSA(\mathrm{i}-1, D-D_\mathrm{i})$
12: **else**
13: $DSA(\mathrm{i}, D) \leftarrow DSA(\mathrm{i}-1, D)$
14: **end if**
15: **else**
16: $DSA(\mathrm{i}, D) \leftarrow DSA(\mathrm{i}-1, D)$
17: **end if**
18: **end for**
19: **end for**
20: **for** i $\leftarrow K$ to 1 **do**
21: **if** $DSA(\mathrm{i}, D^{cap}) > DSA(\mathrm{i}-1, D^{cap})$ **then**
22: $y[\mathrm{i}] \leftarrow 1$
23: $D^{cap} \leftarrow D^{cap} - D[\mathrm{i}]$
24: **else**
25: $y[\mathrm{i}] \leftarrow 0$
26: **end if**
27: **end for**
28: **return** $\{y_1, y_2, \cdots, y_K\}$

solution. After the traversal is completed, the value of $DSA(K, D^{cap})$ is the solution of the entire optimization problem. Then, we update the value of state indicator of all devices (Lines 20 - 27). Since the final result is already calculated, whether each device is selected or not can be found by looking up the value of the two-dimensional array DSA in reverse. For example, if $DSA(\mathrm{i}, D) = DSA(\mathrm{i}-1, D)$, it means choosing or not choosing device k leads to the same optimization result, and then we can know that device k should not be selected

to form set $\mathcal{K}_0$, i.e., $y_k = 0$ (Line 22). Otherwise, device k should be selected to form set $\mathcal{K}_0$, i.e., $y_k = 1$ (Line 25). Finally, after traversing all devices, set $\{y_1, y_2, \cdots, y_K\}$ will be returned to Algorithm 1 (Line 28).

Overall, after obtaining the optimal result in Algorithm 2 and converting it into indicators in Algorithm 1, we can assign all the values in $\{x_1, x_2, \cdots, x_K\}$ as the states of each device, where any device k with $x_k = 1$ is the one the server would like to choose to participate in this round of FEL.

Since there are many methods to prove the correctness of applying dynamic programming to solve the 0-1 Knapsack problem [28], such as the proof by contradiction, optimization problem (3.10) is also able to be well solved by Algorithm 1. The time complexity of the algorithm is $O(KD^{cap})$, which is obviously less than that of solving it by a brute-force search with the time cost of $O(2^K)$.

3.3 EXPERIMENTAL EVALUATION

To evaluate the effectiveness and efficiency of our proposed optimization scheme E^{2}DS, we simulate the environment of FEL and perform experimental verification. We also simulate the traditional FL (TFL) process without device selection and the device selection scheme named FedCS in [21]. All three schemes are implemented on a desktop with Intel(R) Core(TM) i7-9750 CPU @2.60GHz and 16GB RAM running Windows 10 OS.

3.3.1 FEL ENVIRONMENT SIMULATION

We establish a simulated edge computing environment to employ the FEL training process, with $K = 100$ to complete the local learning and a server for device selection as well as model aggregation.

For the wireless communication simulation, a circular area with a radius of 50 meters is used as the covered area of the edge server for the experiment, with the edge server located at the center of the area. Devices are uniformly distributed with a range of 2 meters to 50 meters from the center of the circle. The channel gain h_k of device k follows the exponential distribution with the equation $g_0(d_0/d)^4$, where the reference distance $d_0 = 1$

meter, and $g_0 = -40$ dB [26]. We assume the download bandwidths of devices B_k^D follow the normal distribution with the mean and standard deviation being 5 MHz and 4 MHz, and as a practical bandwidth limitation, the upload bandwidth of devices B_k^U would be lower than the download bandwidth, which follows the normal distribution with the mean and standard deviation of 1 MHz and 0.1 MHz. In addition, we set the transmission power P_k as a normal distribution where the mean is 0.6 W and the standard deviation is 0.2 W, and the background noise $N_0 = 10^{-8}$ W. Furthermore, the data size of model parameters is set as $D^p = 25,000$ nats, which is approximately equal to 4.5 KB.

For the local learning step, the training size D_k of each device is set as a normal distribution with the mean and standard deviation being 5 MB and 4 MB, effective capacitance coefficient $\alpha_k = 2 \times 10^{-28}$, the number of CPU cycles c_k is normally distributed with the mean of 15 cycles/bits and the standard deviation of 10 cycles/bits, and the CPU-cycle frequency f_k follow a normal distribution with the mean of 0.5 GHz and the standard deviation of 0.1 GHz.

3.3.2 MODEL TRAINING SETTINGS

To verify the accuracy and efficiency of model training in the FEL environment, we use the MNIST dataset to complete the classification task in this subsection. By comparing the convergence speeds and accuracy of training in different FEL schemes, we can analyze the influence of device selection on FEL performance.

MNIST contains 70,000 handwritten digital images with 10 classes. In our experiments, we set 60,000 of the images for training and 10,000 of those for testing. The dataset is already preprocessed that every image in MNIST is gray-scale with 28×28 pixel, with the handwritten numbers displayed in the center. It is closer to the actual situation as the preprocessing process of the FEL can be completed in advance to speed up FEL.

We compare our proposed E^{2}DS algorithm with the other two schemes, i.e., TFL and FedCS [21]. Specifically, the first model randomly selects devices for training. FedCS uses a greedy algorithm to select as many devices as possible at the beginning of each round of training.

For all these schemes, all candidate devices are assigned with the number of D_k images for local training. To simulate the FEL scenario with non-IID data distribution, we distribute 68.2% of data from the same class to each device, and randomly select 31.8% of data from the remaining classes.

We build a convolutional neural network as the global model, consisting of three linear convolution layers (the first with 32 channels, the second with 64 channels, the third with 128 channels, and each followed with 2×2 max pooling), each of which is activated by the ReLU function, and a final Softmax output layer afterwards.

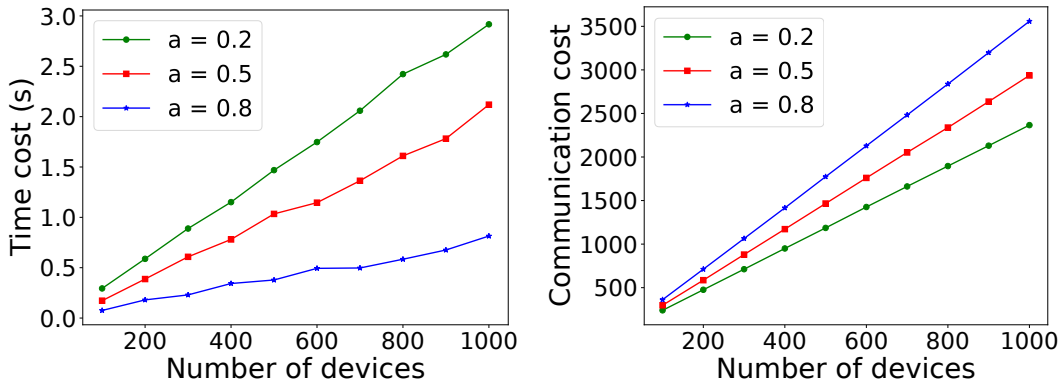
Then, we set the ratio of the necessary amount of data to the total amount of data for each round of FEL as $a = 0.75$ unless otherwise specified, which means there are three-quarters of the data used in each FEL round. In addition, the maximum waiting time in each round T^{wait} is set as 3 min, 5 min and 10 min. For the TFL scheme, as there is no time limitation, we only set the same amount of data for training. For the FedCS scheme, as there is no data amount limitation, we only set the same time limitation. While for our proposed E^{2}DS scheme, we set the weight of energy consumption as $\eta = 3$ and that of the device number as $\theta = 1$. Other sets of weight parameters are also examined, which return similar changing trends, so we omit them to avoid redundancy.

3.3.3 EVALUATION RESULTS

In this subsection, we evaluate the time cost and communication cost of the E^{2}DS algorithm in Section 3.3.3.1, which verifies the usability and effectiveness of our proposed scheme. Then, Section 3.3.3.2 verifies that the E^{2}DS algorithm performs better by evaluating the data diversity and the energy consumption of the three schemes. Finally, in Section 3.3.3.3 we evaluate the accuracy and the speed of convergence comparing in three schemes.

3.3.3.1 TIME AND COMMUNICATION COST

To test the efficiency of the device selection algorithm, Figure 3.2 shows the time consumption and communication cost when selecting different numbers of devices. In Figure 3.2(a), the time consumption increases approximately linearly with the increased number of



(a) Time cost (b) Communication cost

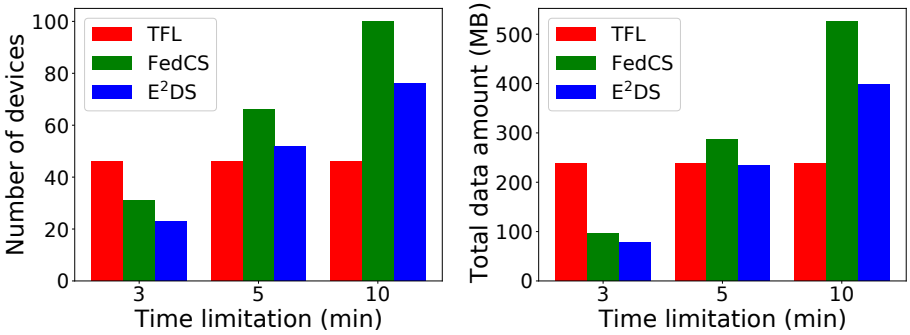
Figure 3.2. The time and communication cost of E^{2}DS.

devices. The growth trend is in line with the time complexity $O(KD^{cap})$ of Algorithm 1 as we mentioned earlier. Specifically, the time consumption is larger when $a = 0.2$ and smaller when $a = 0.8$, due to the inversely-proportional relationship between the data capacity D^{cap} and a as shown in $D^{cap} = (1 - a) \cdot D_{all}$. In contrast, as Figure 3.2(b) implies, there is a positive correlation between the number of devices and the communication cost, because more devices indicate more necessary information to be collected. At the same time, a and the communication cost are also positively correlated since a higher a indicates more data for training and hopefully more devices being selected, leading to more communication cost.

3.3.3.2 COMPARISON OF DEVICE SELECTION SCHEMES

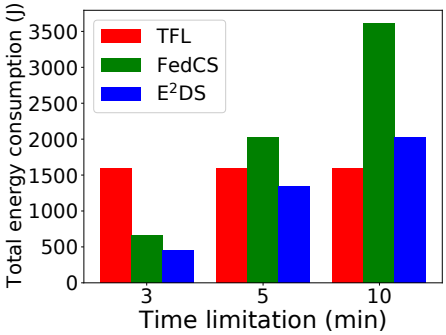
To explore the efficiency of E^{2}DS scheme, we study the performances including the number of devices, total data amount, total energy consumption and energy consumption per device of E^{2}DS scheme, TFL scheme and FedCS scheme with different time limitations T^{wait}.

In a practical FEL environment, we assume that more devices mean that the model has more diverse data. Therefore, if more devices participating in training, the model will reach better performance. As shown in Figure 3.3(a), the number of selected devices does not change with T^{wait} in TFL scheme. Since TFL scheme does not estimate the time consumption of devices in each round of learning, but the total amount of data is limited to $a \cdot D_{all}$. In contrast, E^{2}DS scheme is restricted by T^{wait} when selecting devices. When $T^{wait} = 3$ min,

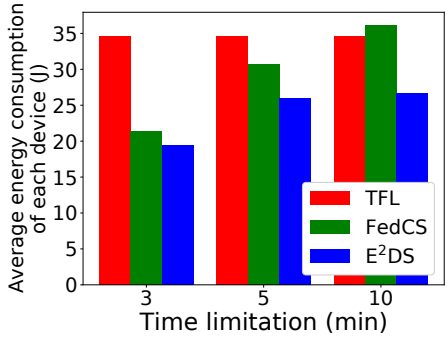


(a) Number of devices

(b) Total data amount



(c) Total energy consumption



(d) Average energy consumption of each device

Figure 3.3. The comparison results of the number of devices, total data amount, total energy consumption and energy consumption per device in different schemes.

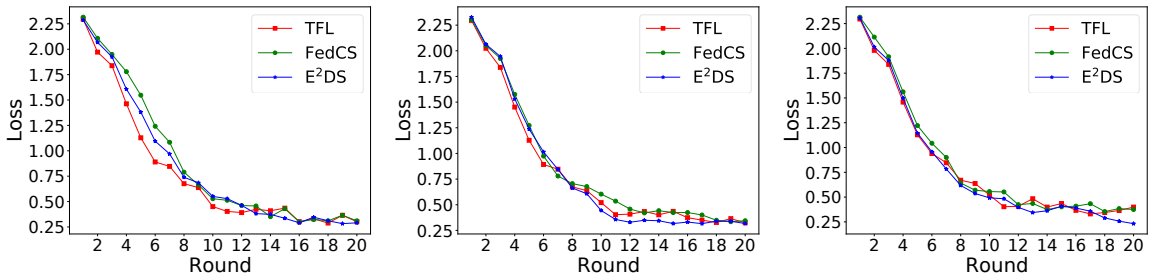
fewer devices can be selected; when $T^{wait} = 5$ min and $T^{wait} = 10$ min, the number of devices selected by E^{2}DS scheme exceeds that of TFL scheme. Besides, it is meaningless to compare the number of devices selected with FedCS scheme, because it selects as many devices as possible due to the greedy algorithm.

Regarding the total data amount shown in Figure 3.3(b), according to the experimental settings, the upper limit of the total data amount of the devices participating in E^{2}DS scheme is $a \cdot D_{all}$. When $T^{wait} = 3$ min or $T^{wait} = 5$ min, the choice of devices is mainly limited by T^{wait}. Compared with TFL scheme, in $T^{wait} = 3$ min, though the total data amount in E^{2}DS scheme is less than that of TFL scheme, E^{2}DS scheme selects more devices. Besides, since T^{wait} is the only restriction, the total data amount increases as T^{wait} becomes larger in the FedCS scheme.

As for the total energy consumption shown in Figure 3.3(c), when $T^{wait} = 3$ min and $T^{wait} = 5$ min, E^{2}DS scheme consumes the least energy. In comparison, the energy consumption of TFL scheme is 1.3-3.6 times that of E^{2}DS scheme, while the energy consumption of the FedCS scheme is 1.5-1.8 times that of E^{2}DS scheme. In particular, when $T^{wait} = 10$ min, the energy consumptions of TFL scheme and E^{2}DS scheme are less than 2000 J, while the FedCS scheme consumes nearly 3500 J. This is because FedCS scheme lacks restriction on device selection in the same task, and a load of useless energy consumption is generated.

For the average energy consumption of each device, which is the most intuitive aspect to reflect the effectiveness of the scheme on energy consumption, Figure 3.3(c) displays that the average energy consumption of E^{2}DS scheme is the lowest in all cases. Compared with TFL scheme, the energy consumption reduced by 30%-50%, while there is also 20%-30% reduction compared with FedCS scheme.

3.3.3.3 COMPARISON OF LEARNING RESULTS



(a) $T^{wait} = 3$ min (b) $T^{wait} = 5$ min (c) $T^{wait} = 10$ min

Figure 3.4. The convergence trends of the loss function over training data in different schemes.

In this subsection, we compare the learning results of E^{2}DS scheme, TFL scheme and FedCS scheme by changing the value of the time limitation as well as the number of FEL rounds. Table 3.1 implies the results of the accuracy of the three schemes after training. All schemes can achieve an accuracy of over 85% after 20 rounds of training. Specifically, FedCS and E^{2}DS schemes can increase the accuracy to 94%. Especially for E^{2}DS scheme, the

accuracy is higher than 85% after the fifth round of training, achieving a higher classification accuracy than FedCS and TFL schemes.

Table 3.1. Comparison of learning results.

Method	T^{wait}	Accuracy		
		Round=5	Round=10	Round=20
TFL	-	0.69	0.75	0.85
FedCS	3 min	0.62	0.81	0.92
	5 min	0.72	0.81	0.93
	10 min	0.72	0.85	0.94
E^2DS	3 min	0.68	0.88	0.93
	5 min	0.77	0.88	0.94
	10 min	0.73	0.89	0.94

Furthermore, in order to show the convergence difference of three schemes in the training process, the convergence trends of loss function over training data are shown in Figure 3.4. It indicates that when $T^{wait} = 3$ min, the convergence speed of TFL scheme is faster in the first ten rounds, but the convergence trend becomes roughly the same as the number of rounds grows. When $T^{wait} = 5$ min and $T^{wait} = 10$ min, the convergence trends of the three schemes are almost the same. Referencing to Figure 3.3(a), there are more devices participating in TFL scheme when $T^{wait} = 3$ min, which accelerates the convergence. When $T^{wait} = 5$ min and $T^{wait} = 10$ min, the number of devices is no longer a bottleneck that limits the convergence speed, then three schemes show the same convergence trends.

4. ENERGY-EFFICIENT DEVICE SELECTION WITH FAST CONVERGENCE IN FEDERATED EDGE LEARNING USING ONLINE BANDIT LEARNING

4.1 SYSTEM MODEL

In this paper, we consider a federated edge learning (FEL) system, where the edge is the server to aggregate model and devices are activated on the server to train their own private data. The set of devices on the edge is denoted by $\mathcal{K} = \{1, 2, \cdots, k, \cdots, K\}$. In fact, since there are a large number of devices on the edge, and the communication and computing capabilities of each device are different, devices selection in each communication round becomes a feasible method in order to improve the efficiency of training and communication.

Under this premise, how to select the optimal set of devices based on cost-effectiveness standards and training efficiency has become a necessary step to improve system performance. For this reason, in this chapter we consider the trade-off between resource consumption and convergence speed to select devices, where the required number of data samples can be provided by selected devices for model training, and meet the local training and update time constraint while consuming the minimum value of total energy and maximal value the total loss.

In this chapter, Section 4.1.1 introduces the FEL system and process, Section 4.1.2 shows the time and energy consumption models. Besides, we set the loss model in Section 4.1.3. Finally, the whole problem formulation is generated in Section 4.1.4.

4.1.1 FEL PROCESS

Usually in an FEL task, the time limit that the edge server waits for devices to upload the model update is predefined as a constant parameter. In this article, we define the time limit parameter as T^{wait}. The FEL system for device selection includes six main steps, which is shown in Figure 4.1. Since in each communication round, the server needs to choose optimal devices set for training, the server needs **Device Information Collection** before devices selection. In this step, the edge server collects the necessary information of

all the devices logged in the edge, including the parameters that are related to the specific task such as local data size and estimated loss value. In addition, parameters which are related to communication and computing power will also be collected, such as CPU computing power, transmission power and bandwidth. When the server receives the information returned by all devices or reaches the maximum waiting time, the server calculates the best choice of suitable devices to optimize system performance as **Device Selection**, which will be discussed in detail in the next chapter. After that, the server sends the parameters of the global model to all selected devices in the step of **Global Model Distribution**. Once devices receive parameters, they will use their local data to train the local model in **Local Learning at Devices** step. In **Local Model Uploading** step, when the local model is trained in certain rounds, each device sends the parameter updates back to the server, as well as the real time consumption in local learning. After that, the server aggregates the received parameter results to derive the updated global model in step **Global Model Aggregation**. At the same time, the server calculates the CPU-cycle frequency for each selected device in this communication round, records it and then learns the probability distribution using combinatorial multi-armed bandit (CMAB) algorithm. After updating the CPU-cycle frequency of each device and the global model, the server finishes this communication round of training and starts the next round with the first step, i.e., **Device Information Collection**.

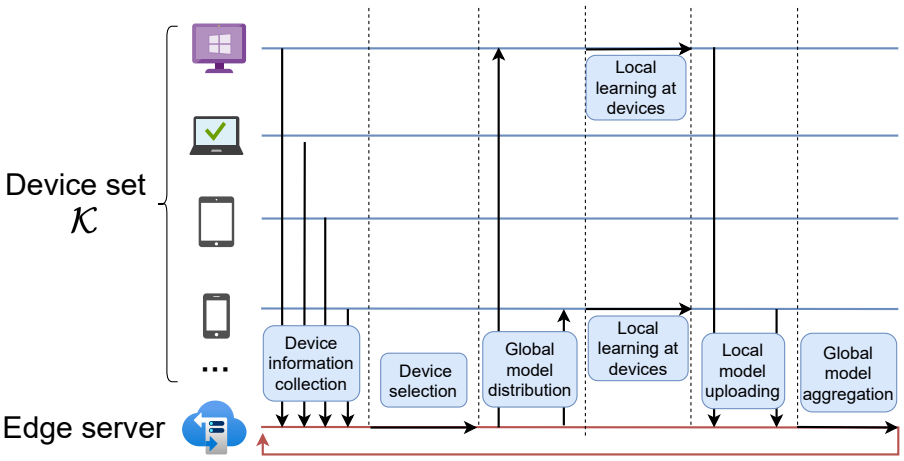


Figure 4.1. The working process of an FEL communication round.

4.1.2 TIME AND ENERGY CONSUMPTION MODELS

In order to ensure the efficient training of the FEL system, the server selects the set of devices for training by limiting the maximum upload time of the devices T^{wait}, which is mentioned above. Moreover, due to the powerful computing and communication capabilities of the server, the time and energy consumption in multiple steps can be skipped, such as **Device Selection** and **Global Model Aggregation**. Several key steps are more closely related to the devices themselves, which are worthy of analysis and optimization, such as steps of **Global Model Distribution**, **Local Learning at Devices**, and **Local Model Uploading**.

In the **Global Model Distribution** step, $T_k^D(t)$ is defined to represent the time spent of device k in round t, while $T_k^U(t)$ represents the time spent in the **Local Model Uploading** step. $T_k^D(t)$ and $T_k^U(t)$ are generally determined by the size of the model parameters divided by the transmission speed, which can be calculated as:

$$T_k^D(t) = \frac{D^p}{V_k^D(t)}, \tag{4.1}$$

$$T_k^U(t) = \frac{D^p}{V_k^U(t)}. \tag{4.2}$$

In the above equation, D^p is the size of the model parameter, while $V^D(t)$ and $V^u(t)$ are the downloading and uploading transmission speeds in round t that are closely related to the condition of the communication channel. Specifically, $V^D(t)$ and $V^u(t)$ can be calculated as:

$$V_k^D(t) = B_k^D(t)\log(1 + \frac{P_k(t)h_k^2(t)}{N_0}), \tag{4.3}$$

$$V_k^U(t) = B_k^U(t)\log(1 + \frac{P_k(t)h_k(t)^2}{N_0}). \tag{4.4}$$

In the above equations, $B_k^D(t)$ and $B_k^U(t)$ are respectively the download and upload bandwidth of device k in round t, while $P_k(t)$ and $h_k(t)$ are respectively the transmission power and the channel gain [21], and N_0 is the background noise.

Global Model Distribution, **Local Learning at Devices**, and **Local Model Uploading**

Then, we define the energy consumption of the device k in steps of **Global Model Distribution** and **Local Model Uploading** in round t, denoted by $E_k^D(t)$ and $E_k^U(t)$, respectively. As the energy consumption during transmission is the product of the transmission power and the transmission time, $E_k^D(t)$ and $E_k^U(t)$ can be calculated as:

$$E_k^D(t) = T_k^D(t)P_k(t), \tag{4.5}$$

$$E_k^U(t) = T_k^U(t)P_k(t). \tag{4.6}$$

In order to calculate the time and energy consummation of devices during the local computing process, we denote $c_k(t)$ as the number of CPU cycles for each device k to complete training one sample of data in round t, which can be measured locally and reported to the server. And the total number of CPU cycles needed for device k in round t is $c_k(t)D_k$, while D_k is defined as the size of its local dataset. Then, the CPU-cycle frequency of device k in round t is denoted by $f_k(t)$. Based on these parameters, the time spent in the local learning step of device k in round t, denoted by $T_k^{LC}(t)$, can be represented as [25]:

$$T_k^{LC}(t) = \frac{c_k(t)D_k}{F_k(t)}.$$

Since each device has different available CPU resources at different times, even during local calculations the CPU-cycle frequency may vary greatly. Therefore, instead of selecting devices based on the real-time frequency, we use the expected value $f_k(t)$ of the previous $t-1$ round as the CPU-cycle frequency of the device k in round t, which is shown as:

$$f_k(t) = \mathbb{E}[F_k(t)]. \tag{4.7}$$

After considering the time consumption in the step of **local learning at devices**, the energy consumption of device k in round t, denoted as $E_k^{LC}(t)$, is also closely related to the data amount and CPU-cycle frequency. Here $E_k^{LC}(t)$ can be expressed as [26]:

$$E_k^{LC}(t) = \frac{\alpha_k}{2} c_k D_k F_k(t)^2,$$

where α_k is the effective capacitance coefficient of the computing chip-set in device k.

Overall, by combining the time spent in each step, the total time and energy consumption of device k in round t, separately denoted as $T_k(t)$ and $E_k(t)$, can be calculated as:

$$T_k(t) = T_k^D(t) + T_k^{LC}(t) + T_k^U(t), \tag{4.8}$$

$$E_k(t) = E_k^D(t) + E_k^{LC}(t) + E_k^U(t). \tag{4.9}$$

4.1.3 LOSS MODEL

To facilitate the selection of the marking devices, we define a binary variable $x_k(t) \in \{0, 1\}$ to indicate whether device k is selected or not in round t, where $x_k(t) = 1$ denotes that device k is selected for local training in this FEL round while $x_k(t) = 0$ means not being chosen.

In the environment of federated learning, the loss value of each device after local calculation can be used as a standard to reflect the training efficiency. If device k is selected in round t, the loss value can be generated after the complete local computing process, which is denoted by $G_k^c(t)$. For devices that are not selected in the t round, the accurate loss value cannot be obtained without complete training. In order to get an approximate loss value and consume less energy, we define the the loss value $G_k^e(t)$ of training a mini-batch to estimate the loss value of the entire round of training, which is calculated as [23]:

$$G_k^e(t) = \sum_{\xi \in \widehat{\xi_k(t)}} \frac{f(k, w, \xi, t)}{|\widehat{\xi_k(t)}|}. \tag{4.10}$$

In the above equation, $\widehat{\xi_k(t)}$ is the mini-batch which is sampled uniformly at random from D_k in round t, and $f(k, w, \xi, t)$ is the loss function of device k with parameter w and sample ξ in round t. Base on these two way to efficiently and accurately collect loss values from all devices, we denote $G_k(t)$ as the loss value of device k in communication round t, which is represented as:

$$G_k(t) = \begin{cases} G_k^{\mathrm{e}}(t), & x_k(t) = 0, \\ G_k^{c}(t), & x_k(t) = 1. \end{cases} \tag{4.11}$$

4.1.4 PROBLEM FORMULATION

Given that edge devices are usually battery-powered and have limited energy supply, our goal is to minimize the total energy consumption of all participating devices to avoid accidental device disconnection, thereby improving the cost-effectiveness of FEL.

In addition, the loss value that the selected devices will get during the local computing step is also an important parameter that affects the overall performance of the FEL system. In a real federated learning scenario, biasing devices with higher local losses selection can speed up the convergence. Therefore, in the selection phase, it is important to select the devices with higher loss values for training, so as to improve the convergence speed of the final training model.

To include the factors that mentioned above, we denote a reward function $R_k(t)$ as the difference between the energy consumption and the loss of device k in round t. The reward function $R_k(t)$ can be represented as:

$$R_k(t) = \eta E_k(t) - G_k(t), \tag{4.12}$$

where η is used for value balance.

Furthermore, the amount of training data will affect the performance of FEL. Too much or too little data will affect the convergence speed and model precision. Here we use the parameter a to describe the ratio of the amount of data required in FEL to the total amount of data $D_{all} = \sum_{k=1}^{K} D_k$. Except for the data, there is a time limit T^{Wait} for the server to

collect local model updates, as defined above, which should be a common constraint for all devices.

To achieve goals and meet the constraints that mentioned above, we formulate the following optimization question:

$$\min : \ \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\sum_{k=1}^{K}R_k(t)x_k(t)], \tag{4.13}$$

$$\text{s.t.} : \ x_k(t) \in \{0,1\}, \tag{4.13a}$$

$$T_k \leq T^{wait}, \tag{4.13b}$$

$$\sum_{k=1}^{K} D_k x_k(t) \geq a \cdot D_{all}. \tag{4.13c}$$

In the above formulation, the goal (4.13) is to minimize the expected difference between the total energy consumption and the total loss of the selected devices. Constraints (4.13a) ensure that the status indicator $x_k(t)$ should only be selected as 1, not being selected as 0. The constraint (4.13b) guarantees that the total time consumption of each selected device should be less than the maximum waiting time T^{wait} set by the server. The last constraint (4.13c) stipulates that the total data volume of the selected device must be sufficient to train the model in each communication round.

4.2 ALGORITHM DESIGN FOR ENERGY-EFFICIENT DEVICE SELECTION WITH FAST CONVERGENCE

In this section, we mainly introduce a reinforcement learning method to learn the CPU-cycle frequency, and design an algorithm to solve the optimization problem (4.13) after reformulation. Firstly we introduce the general idea of CMAB in Section 4.2.1, then a specific calculation is represented in Section 4.2.2. After that, we reformulate the problem in Section 4.2.3, and design an algorithm in Section 4.2.4.

4.2.1 GENERAL IDEA OF CMAB

Since the CPU of each device is consumed by multiple tasks at the same time, the CPU-cycle frequency that can be used for the FL task is more or less lower than the rated frequency in a practical situation. To obtain a stable and real frequency of the CPU-cycle that can be used for training of each device, we introduce the CMAB method to learn the CPU-cycle frequency for approaching the real situation.

Specifically, the edge server is regarded as a player, each device is treated as an arm, and the action of device selection at the server at the beginning of every communication round is treated as pulling arms. The reward $R_k(t)$ of k can be considered as a reward in the CMAB problem.

The goal of our problem is to minimize the difference in energy consumption and loss values while satisfying the edge resource constraints. By obtaining the communication and computing power of all devices at the **Device Information Collection** step, the server can calculate and estimate the loss value, energy and time consumption as a reference for devices selection in a communication round.

In practice, however, the CPU-cycle frequency that affects time and energy consumption is unknown that only the rated value is obtained, not the frequency value dedicated to the FL task. Therefore, after local training for each communication round, the server calculates the true CPU-cycle frequency based on the information of time consumed by devices, and then relearns and updates the frequency information for devices.

In this case, in order to learn the CPU-cycle frequency of each round more accurately, we need to solve the trade-off problem of exploitation and exploration in the reinforcement learning problem. In this model, "exploitation" refers to choosing the arm (device) with a larger reward to participate in training; "exploration" refers to choosing a new arm (device) to participate in training. After obtaining the trade-off reward value, we dynamically adjust the device selection strategy under the constraints of time and data size to maximize the sum of the trade-off reward value. Therefore, through the CMAB problem, we propose an algorithm to solve the problem of device selection with unknown CPU-cycle frequency.

4.2.2 SPECIFIC CALCULATION

We can get the communication and calculation time by recording the time difference between the server sending data to device k in the **Global Model Distribution** step and receiving the result of device k in the **Local Model Uploading** step in round t, which is denoted as $T_k^{Final}(t)$. Then, same as $T_k(t)$, based on (4.1), (4.2), (4.3), (4.4), (4.7), (4.8), the real CPU-cycle frequency $F_k(t)$ of device k in round t is calculated as:

$$F_k(t) = \frac{c_k(t)D_k}{T_k^{Final}(t) - \frac{D^p}{B_k^D(t)\log(1+\frac{P_k(t)h_k^2(t)}{N_0})} - \frac{D^p}{B_k^U(t)\log(1+\frac{P_k(t)h_k(t)^2}{N_0})}}. \tag{4.14}$$

According to (4.1) - (4.12), the problem (4.13) can be rewritten as:

$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\sum_{k=1}^{K}R_k(t)x_k(t)] = \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\eta\sum_{k=1}^{K}(&\frac{D^p}{B_k^D(t)\log(1+\frac{P_k(t)h_k^2(t)}{N_0})}P_k(t) + \frac{\alpha_k}{2}c_kD_kF_k(t)^2 \\ &+ \frac{D^p}{B_k^U(t)\log(1+\frac{P_k(t)h_k(t)^2}{N_0})}P_k(t))x_k(t) - \sum_{k=1}^{K}G_kx_k(t)].\end{aligned} \tag{4.15}$$

In the above optimization goal, the other parameters included are fixed and accurate except for the CPU-cycle frequency. According to (4.7), the equation (4.15) can be updated as:

$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\sum_{k=1}^{K}R_k(t)x_k(t)] = \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\eta\sum_{k=1}^{K}(&\frac{D^p}{B_k^D(t)\log(1+\frac{P_k(t)h_k^2(t)}{N_0})}P_k(t) + \frac{\alpha_k}{2}c_kD_kf_k(t)^2 \\ &+ \frac{D^p}{B_k^U(t)\log(1+\frac{P_k(t)h_k(t)^2}{N_0})}P_k(t))x_k(t) - \sum_{k=1}^{K}G_kx_k(t)].\end{aligned} \tag{4.16}$$

To minimize (4.16), it is required to learn the unknown CPU-cycle frequency of each device, i.e., $f_k(t)$. To aim this, we denote $n_k(t)$ as the number of communication rounds that

the device k is selected to participate in training with this FL task in the first t communication round. The calculation is defined as:

$$n_k(t) = \sum_{\tau=0}^{t-1} x_k(\tau). \tag{4.17}$$

Then, in order to record a stable CPU-cycle frequency for each device, we maintain an average value of the previous t of device k, which is defined as $\bar{f}_k(t)$. According to the value $F_k(t)$ of the CPU-cycle frequency of the previous t rounds and the number of times that the device k is selected in (4.17), we can calculate the average value $\bar{f}_k(t)$ as:

$$\bar{f}_k(t) = \frac{\sum_{\tau=0}^{T-1} F_k(\tau) x_k(t)}{n_k(t)}. \tag{4.18}$$

Therefore, after receiving the real CPU-cycle frequency $F_k(t)$ in each communication round, the server will update $\bar{f}_k(t)$ and $n_k(t)$ for devices, regardless of whether it is selected to participate in this round of training. Respectively, $n_k(t)$ and $\bar{f}_k(t)$ can be updated by:

$$n_k(t) = \begin{cases} n_k(t-1), & x_k(t) = 0, \\ n_k(t-1) + 1, & x_k(t) = 1, \end{cases} \tag{4.19}$$

$$\bar{f}_k(t) = \begin{cases} \bar{f}_k(t-1), & x_k(t) = 0, \\ \dfrac{n_k(t-1)\bar{f}_k(t-1) + F_k(t)}{n_k(t)}, & x_k(t) = 1. \end{cases} \tag{4.20}$$

After updating the average CPU-cycle frequency of each device each time, we complete the "exploitation" part of reinforcement learning. But in order to allow the devices which are not selected to have more opportunities to update the data to approximate their true value, which is the "exploration" part, we introduce the variable $\widetilde{f}_k(t)$ as the estimated value of the CPU-cycle frequency of device k in the t round. In this article, we use the Upper Confidence Bound (UCB) [29] [30] algorithm to update $\widetilde{f}_k(t)$, which is shown below:

$$\widetilde{f}_k(t) = \bar{f}_k(t) + \sqrt{\frac{2 \ln K}{n_k(t)}}. \tag{4.21}$$

In the above equation, $\widetilde{f}_k(t)$ is composed of two parts, respectively considering the "exploitation" part and the "exploration" part. Specifically, $\bar{f}_k(t)$ represents that the more times we choose a certain device, the narrower the confidence interval of the device return estimate and the lower the uncertainty of the estimate, and those devices with a larger mean tend to be selected multiple times. Meanwhile, $\sqrt{\frac{2\ln K}{n_k(t)}}$ refers to that the fewer the number of attempts for a device, the wider the confidence interval and the higher the uncertainty, which means the device with a wider confidence interval tends to be selected multiple times.

Therefore, the variable $\bar{f}_k(t)$ represents the CPU-cycle frequency closest to the true value of the device, and $\widetilde{f}_k(t)$ gives suggestions for the next round of device selection, based on the perspectives of giving chances to all devices to update their real information. By doing so, the situation of selecting only the devices with better initial performance is avoided, and the server also gives the opportunity for devices with poor initial performance to be selected and update their information.

Base on this, we define $\widetilde{S}(T)$ as an average expected variable of our optimization problem that considers the whole FEL task from the beginning to the T-th round, which can be rewritten as:

$$\begin{aligned}\widetilde{S}(T) = \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\eta\sum_{k=1}^{K}(&\frac{D^p}{B_k^D(t)\log(1+\frac{P_k(t)h_k^2(t)}{N_0})}P_k(t) + \frac{\alpha_k}{2}c_k D_k \widetilde{f}_k(t)^2 \\ &+ \frac{D^p}{B_k^U(t)\log(1+\frac{P_k(t)h_k(t)^2}{N_0})}P_k(t))x_k(t) - \sum_{k=1}^{K}G_k x_k(t)].\end{aligned} \tag{4.22}$$

4.2.3 PROBLEM REFORMULATION

Now, we can rewrite the optimization problem as follows:

$$\begin{aligned}\min:\ &\widetilde{S}(T), \\ \text{s.t.}:\ &(4.13a)(4.13b)(4.13c).\end{aligned} \tag{4.23}$$

It is clear that the optimization problem is the optimal subset of devices that minimizes the total energy consumption and maximizes the total loss under the constraints of time consumption and training data size. Since the above optimization problem requires complex combinatorial optimization, the main task is to find a solution, to minimize the difference between the energy consumption and loss value of the selected devices.

The intuitive solution to this problem is to traverse all device combinations and then compare them to obtain the best device set. Since each device has two states, selected and unselected, it will take $O(2^K)$ time complexity to traverse all K devices, which will not only consume a huge amount of time in each communication round but also severely affect training efficiency. Because of that, we transform the problem defined in (4.13) into an effective maximization problem, and propose a solution with lower time complexity based on dynamic programming.

Since in each communication round, the devices registered on the edge are known and unchanged, the calculated energy consumption and loss values are fixed after the information of all devices is collected. We found that for energy, the total energy consumption of all devices in each round is fixed, and there are only two choices for devices in each communication round (to be selected or not to be selected to participate in training). Based on this, selecting a set of devices with lower energy consumption to participate in training is equivalent to selecting a set of devices with higher energy consumption, and allowing the rest of them to participate in training. As well as energy, after devices with lower loss values are selected, the rest of devices set will have the highest loss values. Above on, we can see that if a set of devices with higher sum of reward in round t can be selected and removed, the rest of devices are the optimal device set that also solves our optimization problem.

We know the state indicator of device k in $\mathcal{K}$ is $x_k(t)$, which is equal to 1 when device k is selected in round t. To better distinguish selected set and unselected set, we denote $y_k(t)$ to indicate the state of device k which is not selected for participating in training in round t, so we have $y_k = 1 - x_k$. Then, to solve the reformulated optimization, we define $\widetilde{S}_y(T)$ as

the average expected sum of reward that considers from the beginning to the T round of the whole FEL task, which can be calculated as:

$$\begin{aligned}\widetilde{S}_y(T) = \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\eta\sum_{k=1}^{K}(\frac{D^p}{B_k^D(t)\log(1+\frac{P_k(t)h_k^2(t)}{N_0})}P_k(t) + \frac{\alpha_k}{2}c_k D_k \widetilde{f}_k(t)^2 \\ + \frac{D^p}{B_k^U(t)\log(1+\frac{P_k(t)h_k(t)^2}{N_0})}P_k(t))y_k(t) - \sum_{k=1}^{K}G_k y_k(t)].\end{aligned} \tag{4.24}$$

In this case, according to (4.24), the objective (4.23) can be rewritten as:

$$\max: \ \widetilde{S}_y(T), \tag{4.25}$$

$$\text{s.t.}: \ y_k(t) \in \{0,1\}, \tag{4.25a}$$

$$T_k \leq T^{wait}, \tag{4.25b}$$

$$\sum_{k=1}^{K} D_k y_k(t) \leq (1-a)\cdot D_{all}, \tag{4.25c}$$

where the data size constraint (4.25c) means the total data size of selected devices that will not participate in training should be less than $(1-a)\cdot D_{all}$, instead, it guarantees enough data size of the devices that will participate in training.

After transforming the optimization problem, we have a maximum problem with an upper limit rather than a lower limit on the data size constraint. By doing so, we successfully reformulate the optimization that can be solved using a dynamic programming method, which is explained in detail in Section 4.2.4.

4.2.4 ALGORITHM DESIGN

In the above, we find the fact that the sum of energy consumption of all devices that are logged in the edge server in each communication round is fixed, as well as loss values. Based on the complementary relationship of selected and unselected devices, we transform the original minimization problem defined in (4.23) into a maximization problem defined in (4.25).

To solve it, we propose a Fast-Convergent Energy-Efficient Device Selection (FCE^{2}DS) algorithm and specify it in Algorithm 3.

Based on the device selection solution of E^{2}DS algorithm, the above maximization problem with an upper limit constraint can be transferred to a 0-1 Knapsack problem. Similarly, we propose a DeviceSelection algorithm to solve this problem defined in (4.25). Specifically, for every communication round t, there are K different devices that can be selected or not for joining FEL, the capacity is the maximum remaining data size $(1-a)\cdot D_{all}$ with an extra constraint of time consumption. In addition, we denote $D^{cap} = (1-a)\cdot D_{all}$ as the data size capacity of unselected devices, and $v_k(t) = \eta E_k(t) - G_k(t)$ as the value of device k in round t. Then a sequence $\mathcal{V}(t) = \{v_1(t), v_2(t), \cdots, v_K(t)\}$ is defined to contain the reward values of all devices, while the sequence of data size is denoted by $\mathcal{D} = \{D_1, D_2, \cdots, D_K\}$.

After device selection, the selected devices will participate in one communication round of local computing to update the global model. Meanwhile, the real CPU-cycle frequency data of each selected device can be calculated, so as to learn the expected CPU-cycle frequency using the CMAB learning algorithm. The expected CPU-cycle frequency will continue to improve the accuracy of energy consumption calculation, and then impact the device selection in the next communication round.

As shown in Algorithm 3, there are four input parameters: the number of current communication round t; the size set of devices $\mathcal{D} = \{D_1, D_2, \cdots, D_K\}$; the number of devices K; the data size capacity of selected devices D^{cap}. To initiate other parameters (Line 1), for every devices that logged in the edge server, the average CPU-cycle frequency of first communication round $\bar{f}_k(0)$ is set to 0 (Line 2). Then, since $n_k(t)$ will be used as a divisor in the following equation, we set its initial value to 1 (Line 3). Then, in each communication round, at the beginning we calculate the reward value of each device according to the reward function (4.12) (Line 6), and next calculate the optimal device selection solution using DeviceSelection algorithm (Line 7), which will be specifically demonstrated in Algorithm 4. After selection, the selected device will participate in the local training process and update training results to server (Lines 8-9). Then, with the latest updated parameters, the number of being selected $n_k(t)$, average frequency value $\bar{f}_k(t)$, and estimated frequency value $\widetilde{f}_k(t)$ are learnt and updated according to equations (4.19) and (4.20) (Lines 10-14).

Algorithm 3 FCE^{2}DS

Input: the number of current communication round t, the size set of devices $\mathcal{D} = \{D_1, D_2, \cdots, D_K\}$, the number of devices K, the data size capacity of selected devices D^{cap}

1: **for** $k \in$ {1,2,...,K} **do**
2: $\bar{f}_k(0) \leftarrow 0$
3: $n_k(0) \leftarrow 1$
4: **end for**
5: **for** communication round t = 1,2,... **do**
6: Calculate reward value set of devices in round t: $\mathcal{V}(t) = \{v_1(t), v_2(t), \cdots, v_K(t)\}$
7: $\{x_1, x_2, \cdots, x_K\} \leftarrow$DeviceSelection$(\mathcal{V}, \mathcal{D}, K, D^{cap})$
8: **for** $k \in$ {1,2,...,K} and $x_k = 1$ **do**
Local computing at Device according to x_k and update F_k and G_k
9: **end for**
10: **for** $k \in$ {1,2,...,K} **do**
11: $$n_k(t) = \begin{cases} n_k(t-1), & x_k(t) = 0 \\ n_k(t-1) + 1, & x_k(t) = 1 \end{cases}$$
12: $$\bar{f}_k(t) = \begin{cases} \bar{f}_k(t-1), & x_k(t) = 0 \\ \dfrac{n_k(t-1)\bar{f}_k(t-1) + F_k(t)}{n_k(t)}, & x_k(t) = 1 \end{cases}$$
13: $\tilde{f}_k(t) = \bar{f}_k(t) + \sqrt{\frac{2 \ln K}{n_k(t)}}$
14: **end for**
15: **end for**

In Algorithm 4, the reward value set of all devices $\mathcal{V}$ is input to select devices with $\mathcal{D}$, K and D^{cap}. Firstly, similar to Algorithm 2, a two-dimensional array $DSA(K, D^{cap})$ is used to store and update intermediate results of dynamic programming (Lines 1-19). Then, the output result of unselected devices are calculated according to $DSA(K, D^{cap})$ table (Lines 20-27). Finally, by converting the indicator $y_k(t)$ of unselected devices to the indicator $x_k(t)$ that device k will be selected in next round when $x_k(t) = 1$ (Lines 28-31).

Overall, the computational overhead of our solution is mainly determined by the Device-Selection part, so the time complexity of our FCE^{2}DS algorithm is $O(KD^{cap})$, where K is the number of devices that logged in the edge server in the FEL task, and D^{cap} is the data size capacity which is equal to $(1-a) \cdot D_{all}$.

Algorithm 4 DeviceSelection

Input: the value set of devices $\mathcal{V} = \{v_1, v_2, \cdots, v_K\}$, the size set of devices $\mathcal{D} = \{D_1, D_2, \cdots, D_K\}$, the number of devices K, the data size capacity of selected devices D^{cap}

Output: the state indicators of all devices for $\mathcal{K}_1$ $\{x_1, x_2, \cdots, x_K\}$

for $D \leftarrow 0$ to D^{cap} **do**
  $DSA(0, D) \leftarrow 0$
end for
for i $\leftarrow 1$ to K **do**
  $DSA(\mathrm{i}, 0) \leftarrow 0$
end for
for i $\leftarrow 1$ to K **do**
  for $D \leftarrow 1$ to D^{cap} **do**
    if $D_\mathrm{i} \leq D$ **then**
      if $v_\mathrm{i} + DSA(\mathrm{i}-1, D - D_\mathrm{i}) > DSA(\mathrm{i}-1, D)$ **then**
        $DSA(\mathrm{i}, D) \leftarrow v_\mathrm{i} + DSA(\mathrm{i}-1, D - D_\mathrm{i})$
      else
        $DSA(\mathrm{i}, D) \leftarrow DSA(\mathrm{i}-1, D)$
      end if
    else
      $DSA(\mathrm{i}, D) \leftarrow DSA(\mathrm{i}-1, D)$
    end if
  end for
end for
for i $\leftarrow K$ to 1 **do**
  if $DSA(\mathrm{i}, D^{cap}) > DSA(\mathrm{i}-1, D^{cap})$ **then**
    $y[\mathrm{i}] \leftarrow 1$
    $D^{cap} \leftarrow D^{cap} - D[\mathrm{i}]$
  else
    $y[\mathrm{i}] \leftarrow 0$
  end if
end for
for i $\leftarrow 1$ to K **do**
  $x_\mathrm{i} \leftarrow 1 - y_\mathrm{i}$
end for
return $\{x_1, x_2, \cdots, x_K\}$

4.3 EXPERIMENTAL EVALUATION

4.3.1 FEL ENVIRONMENT SIMULATION

To evaluate the effectiveness and performance of the optimization scheme FCE^{2}DS proposed in this section, we establish a simulation environment of FEL and perform experimental verification. We also simulate the traditional FL (TFL) process without device selection and the energy-efficient device selection scheme E^{2}DS proposed in [31]. All three schemes are implemented on a desktop with Intel(R) Core(TM) i7-9750 CPU @2.60GHz and 16GB RAM running Windows 10 OS.

In order to be able to make a reasonable and sufficient comparison, most of the parameter settings used in the E^{2}DS experiments are still applicable here. Specifically, for the wireless communication simulation, a circular area with a radius of 50 meters is used as the covered area of the edge server for the experiment, with the edge server located at the center of the area. In the FEL task, there are 50 devices logged in the edge server, which are uniformly distributed with a range of 2 meters to 50 meters from the center of the circle. The channel gain h_k of device k follows the exponential distribution with the equation $g_0(d_0/d)^4$, where the reference distance $d_0 = 1$ meter, and $g_0 = -40$ dB [26]. The download bandwidths of devices B_k^D is set to follow the normal distribution with the mean and standard deviation being 5 MHz and 4 MHz, and as a practical bandwidth limitation, the upload bandwidth of devices B_k^U would be lower than the download bandwidth, which follows the normal distribution with the mean and standard deviation of 1 MHz and 0.1 MHz. The transmission power P_k is set as a normal distribution where the mean is 0.6 W and the standard deviation is 0.2 W. For other fixed values, we assume the background noise N_0 as 10^{-8} W, and the data size of model parameters are set as $D^p = 25,000$ nats, which is approximately equal to 4.5 KB.

For the local learning step, the training size D_k of each device is set as a normal distribution with the mean and standard deviation being 5 MB and 4 MB. To simulate the FEL scenario with non-IID data distribution, we distribute 30% of data from the same class to each device, and randomly select 70% of data from the remaining classes. We set the effective capacitance coefficient $\alpha_k = 2 \times 10^{-28}$, the number of CPU cycles c_k is normally distributed with the mean of 15 cycles/bits and the standard deviation of 10 cycles/bits, and

both the real and estimated CPU-cycle frequency f_k follow the rule of a normal distribution with the mean of 0.5 GHz and the standard deviation of 0.1 GHz.

4.3.2 MODEL TRAINING SETTINGS

For the experiment of this FEL task, the dataset we use to train and test is MNIST which includes 60,000 handwritten digital images for training and 10,000 for testing with 10 classes. We compare our proposed FCE^{2}DS algorithm with the other two schemes, i.e., TFL and E^{2}DS [31]. In order to make the comparative experiment more convincing, we control the constraints unchanged, which means that all three schemes satisfy the constraints (4.25a), (4.25b) and (4.25c). Specifically, TFL randomly selects devices until the sum of data size reach $(1-a)\cdot D_{all}$ for training. E^{2}DS is based on the algorithm in [31], which refers to the optimization of energy consumption and the number of devices under time constraints when selecting devices. The third scheme shows the performance of FCE^{2}DS algorithm proposed in this paper.

We use the same convolutional neural network as the global model for all three schemes, including three linear convolution layers. Specifically, the first layer contains 32 channels, while the second one has 64 channels and the third one has 128 channels. All layers are followed with 2×2 max pooling, which are activated by the ReLU function, and a final Softmax output layer afterward.

Then, we set the ratio of the necessary amount of data to the total amount of data for each round of FEL as $a = 0.75$ unless otherwise specified, which means there are three-quarters of the data used in each FEL round. In addition, the maximum waiting time in each round T^{wait} is set as 10 min. For the E^{2}DS scheme, we set the default parameter value shown in [31], where the weight of energy consumption is set as 3 and that of the device number as 1. In this paper, we set the weight of energy consumption as 0.1.

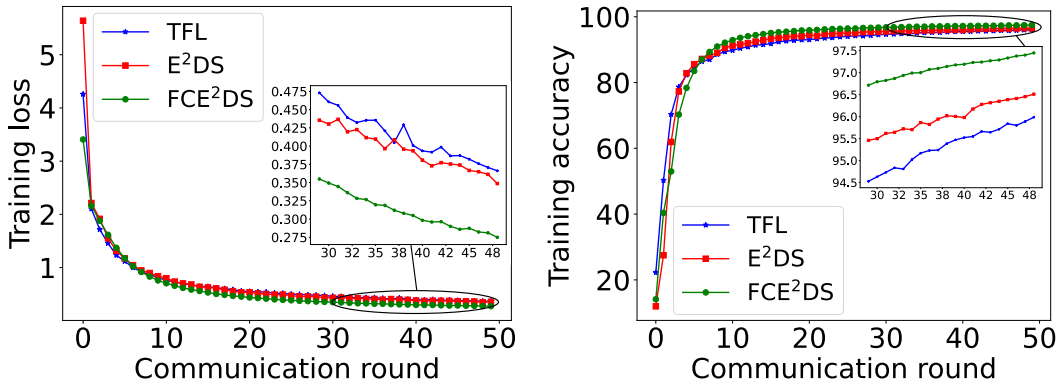
4.3.3 EVALUATION RESULTS

In this part, Section 4.3.3.1 verifies that the FCE^{2}DS algorithm performs better by evaluating the loss trend, accuracy trend, number of devices and the energy consumption of

the three schemes. Then, we evaluate the learning performance and cost of the FCE^{2}DS algorithm in Section 4.3.3.2, which verifies the usability and effectiveness of our proposed scheme.

4.3.3.1 COMPARISON EXPERIMENTS

To explore the efficiency of FCE^{2}DS scheme, we study the performances, including the loss trend, the accuracy trend, the number of devices, and energy consumption per device, of TFL scheme, E^{2}DS scheme, and FCE^{2}DS scheme.

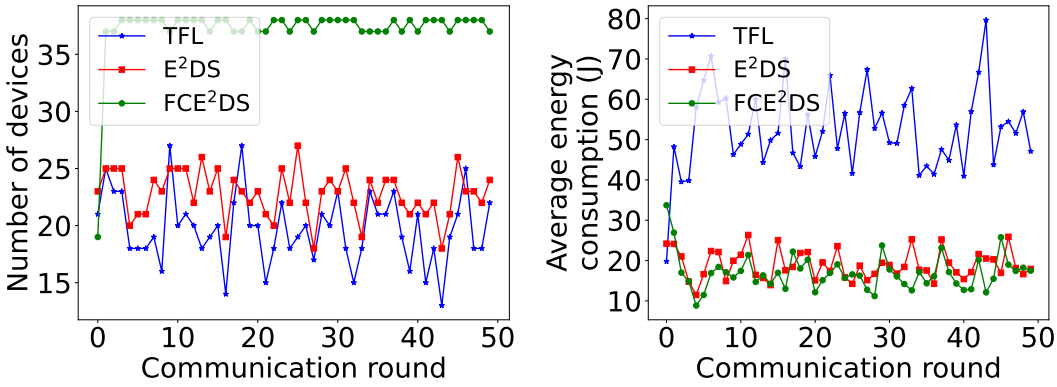


(a) Loss trend (b) Accuracy trend

Figure 4.2. The comparison results of loss and accuracy trend in different schemes.

Firstly, we compare the loss and accuracy trends in different schemes, where the results are shown in Figure 4.2. Specifically, as shown in Figure 4.2a, all three schemes converged after 50 communication round's training, but the loss of FCE^{2}DS is the smallest among them. In the final round 50, the loss of FCE^{2}DS reaches 0.274, while the losses of TFL and E^{2}DS are respectively 0.369 and 0.357, both larger than that of FCE^{2}DS. Since the loss value is the most obvious result to show the convergence speed, it is clear to see that FCE^{2}DS has a better performance in convergence. Then, in Figure 4.2b, we can see that after the training of 50 communication rounds, all the three schemes reach a high training accuracy. Separately, the accuracy of E^{2}DS scheme is 96.47%, which is a little larger than the accuracy of TFL (96.06%), while FCE^{2}DS scheme gets the highest accuracy of 97.48%. Above all,

the FCE^{2}DS scheme shows the best performance and wins the comparison of not only the convergence speed but the accuracy.



(a) Number of devices

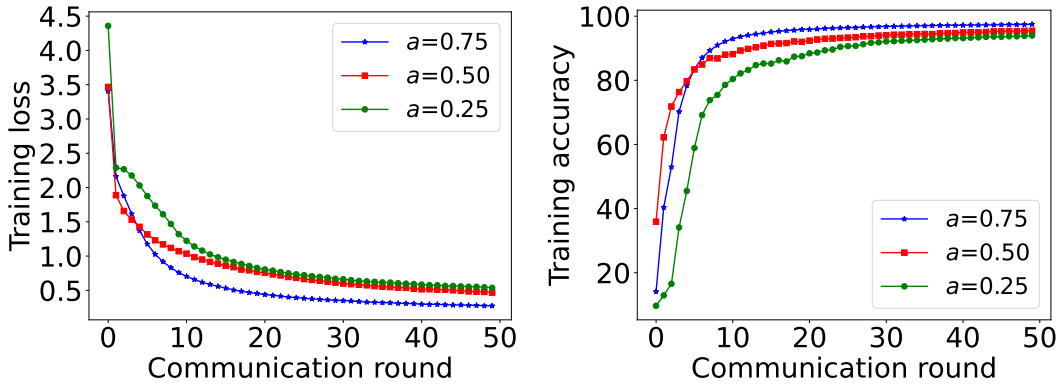
(b) Average energy consumption of each device

Figure 4.3. The comparison results of energy and devices in different schemes.

Then, Figure 4.3 shows the comparison results of three schemes regarding number of devices and average energy consumption of devices. As shown in Figure 4.3a, FCE^{2}DS scheme selects more devices (about 38 devices) in each communication round, compared with E^{2}DS scheme and TFL scheme (about 10-25 devices in each round). We know that in a practical FEL environment, more devices mean that the model has more diverse data, which is better for the model to have a fast convergence performance. The results that FCE^{2}DS scheme selects more devices confirm that it can get the lowest loss value (shown in Figure 4.2a). To explain an unusual number of selected devices in the FCE^{2}DS scheme, only 19 devices are selected to participate in the first communication round, that is because the FCE^{2}DS scheme will randomly choose devices in initialization to take part in the beginning training. For the energy consumption, Figure 4.3b shows the average energy consumption of each device in different schemes. Since the number of selected devices is different with all the three schemes, it is more appropriate to compare the average energy consumption per device instead of the total energy consumption. We can see that the average energy consumption of both FCE^{2}DS scheme and E^{2}DS scheme are around 15-25 J per device, which is much less than the average energy consumption of TFL (about 40-70 J). By comparing FCE^{2}DS scheme and E^{2}DS scheme, in most of the communication round, the energy in FCE^{2}DS

scheme consumes less and is more stable. In conclusion, FCE^{2}DS scheme has a better performance on energy consumption optimization.

4.3.3.2 LEARNING PERFORMANCE AND COST

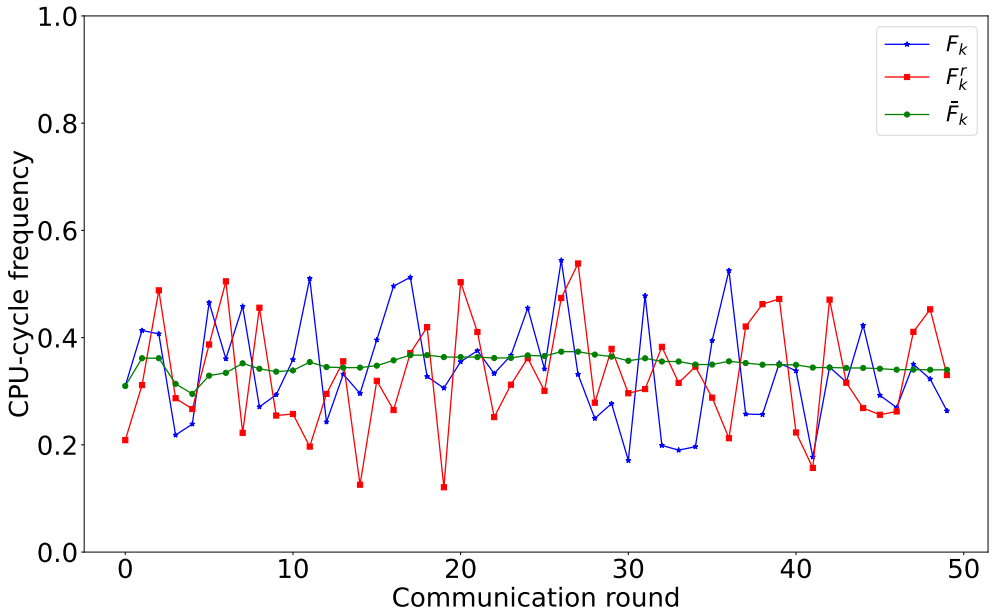


(a) Training Loss (b) Training Accuracy

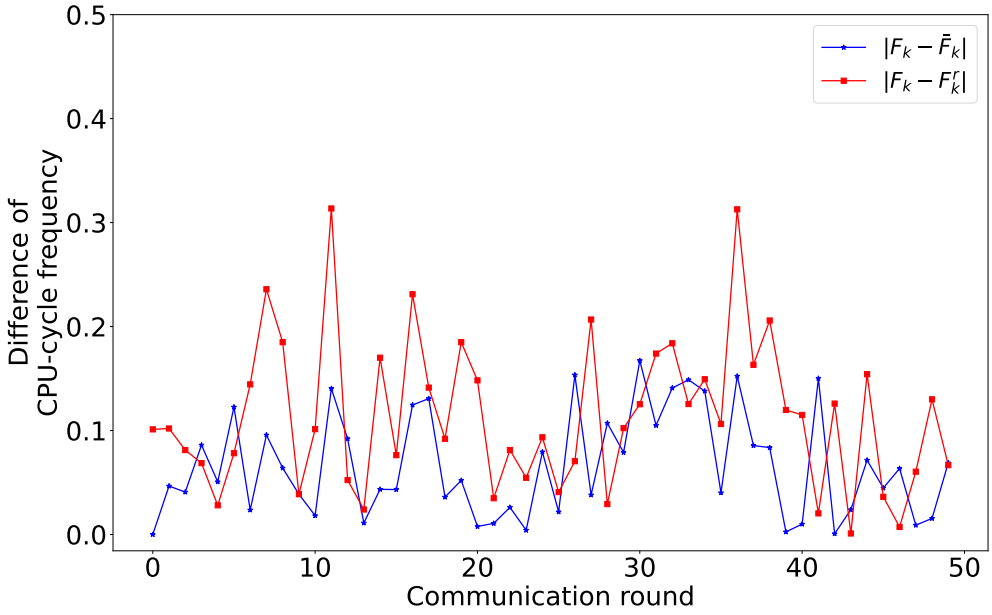
Figure 4.4. Training loss and accuracy trend with different a.

To explore the optimal parameters for training, we change the data size rate a to 0.25/0.5/0.75, and see how the proposed FCE^{2}DS algorithm performs, which is shown in Figure 4.4. Specifically, Figure 4.4a shows the difference of training loss value in each communication round. From the figure we can see that the FEL system has the fastest convergence speed when $a = 0.75$. Similarly, the FEL system gets the highest training accuracy when $a = 0.75$. Therefore, 75% of the data size rate to select devices is optimal for this FEL task to reach a high accuracy and fast convergence speed.

The performance of learning CPU-cycle frequency is shown in Figure 4.5. For each communication round, each device can measure the CPU-cycle frequency and send the value to the server, while other current FEL systems (e.g., the proposed system in [31]) will use the received frequency to calculate and select devices. In order to facilitate the comparison of the accuracy of different types of CPU-cycle frequency, we denote the received frequency here as F_k^r. In this paper we learn the frequency $\bar{f}_k$ to approach the real CPU-cycle frequency F_k. As shown in Figure 4.5a, we randomly demonstrate a device's CPU-cycle frequency trend. It is obvious that $\bar{f}_k$ fluctuates greatly in the first few communication rounds, and becomes



(a) CPU-cycle frequency trend



(b) Difference of CPU-cycle frequency

Figure 4.5. CPU-cycle frequency trend.

more and more stable and close to F_k as the number of rounds increases, compared with F_k^r. To show if the proposed system is closer to F_k, we calculate the absolute difference with F_k of both $\bar{f}_k$ and F_k^r, which is demonstrated in Figure 4.5b. We can see the blue line (difference between F_k and $\bar{f}_k$) has smaller fluctuation and is closer to 0 compared with the

red line (difference between F_k and F_k^r), which means that our system can approach the real CPU-cycle frequency by online bandit learning.

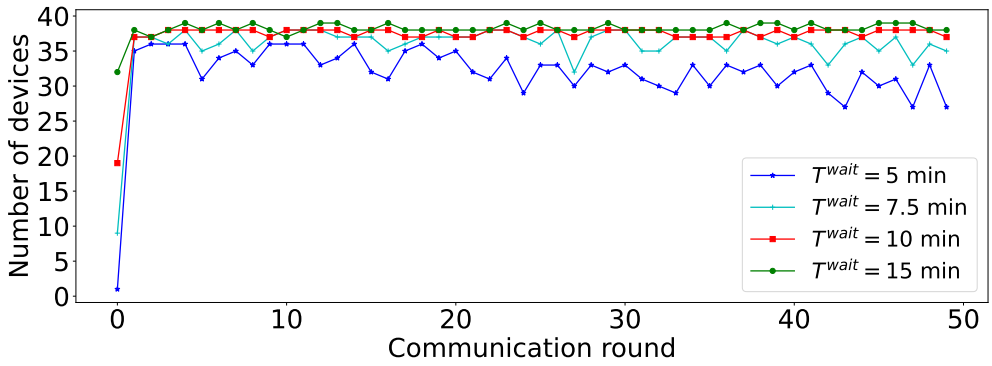
Table 4.1. Comparison of learning results with different time limits.

	T^{wait}	Communication Round t		
		$t = 10$	$t = 30$	$t = 50$
Accuracy	5 min	89.49%	96.14%	96.81%
	7.5 min	94.61%	96.99%	97.60%
	10 min	92.12%	96.71%	97.48%
	15 min	90.85%	94.37%	96.00%
Loss	5 min	0.88	0.41	0.35
	7.5 min	0.67	0.36	0.28
	10 min	0.76	0.35	0.27
	15 min	1.25	0.58	0.41

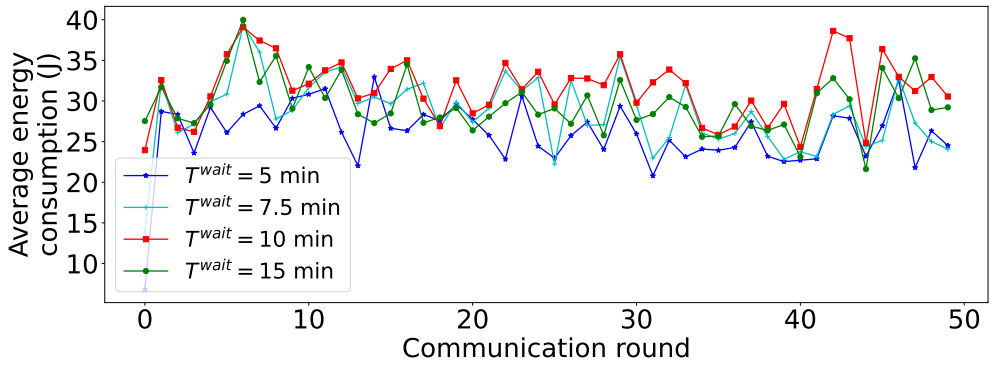
Then, by changing T^{wait} to 5 minutes, 7.5 minutes, 10 minutes and 15 minutes, we study how the time limit influences the performance and energy consumption of the FEL system. The comparison results of learning results with different time limits are shown in Table 4.1, including accuracy and loss at different communication rounds. Besides, the comparison results of energy and devices in different time limits are shown in Figure 4.6.

With a detailed analysis, in Table 4.1, the accuracy and loss values of different time limits are listed separately at different communication rounds. For the accuracy results, we can see that the FEL system could reach the highest accuracy and the lowest convergence speed when $T^{wait} = 7.5$ min through the whole task. In addition, in Figure 4.6a, it is clear that when $T^{wait} = 7.5$ min, there are more devices selected to take part in local training than the number of selected devices when $T^{wait} = 5$ min, which means the data amount used to train the model is greater as well. By combining Table 4.1 and Figure 4.6a to analyze together, the limit of training data is the reason of the FEL system not reaching the best performance when $T^{wait} = 5$ min.

Then, for the experiments with longer T^{wait} (i.e., 10 min and 15 min), as the results shown in Table 4.1, the FEL system does not perform as well as that of $T^{wait} = 7.5$ min. Until the 50-th round, the accuracy and convergence speed are still very slow, which means it needs more communication rounds to train the model and it is more difficult to have a good training



(a) Number of devices



(b) Average energy consumption of each device

Figure 4.6. The comparison results of energy consumption and devices with varying time limit.

result. This is because in an edge environment, the more required communication rounds, the more difficult to maintain the stability and durability of the edge device. Moreover, in Figure 4.6b, the FEL system is more energy-efficient when $T^{wait} = 5$ min and 7.5 min, as the average energy consumption of selected devices is lower, compared with that of $T^{wait} = 10$ min and 15 min. Above all, $T^{wait} = 7.5$ min is the most suitable time limit for this FEL task.

5. DISCUSSION

In this paper, we design algorithms E^2DS and FCE^2DS to solve the optimization problem of energy and convergence speed, and then conduct a series of experiments to find the best time limit to attain the optimal performance of the proposed algorithms. In the experiments, we find that if the time limit is too small, only a small number of devices can be selected to participate in the training, which affects the convergence speed. If an excessively large time limit is set, the time for each communication round will be extended, so the total time required to complete the task will increase a lot. Therefore, setting a suitable time limit is extremely important, too large or too small will seriously affect the performance of the FEL system.

For the FEL task in experiments, by changing the time limit of the communication round and experimenting multiple times, we manage to find the most suitable value for this task. However, different FEL tasks and environments contain a variety of devices with different datasets, communication and computing capabilities. The same parameter settings cannot make all tasks get the best performance, especially the time limit of each communication round. It will definitely consume a lot of energy and time if we experiment multiple times to find the most suitable time limit for every FEL task. Therefore, in a real situation, if the value of the time limit can be dynamically adjusted according to the performance of the devices participating in the task, the efficiency of the FEL task can be improved, which is one of the current limitations of the experiment design.

Besides, in real situations, some devices participating in the FEL task may be charged by a power source while training. Since such devices will not use built-in battery power to complete the training process, the energy consumption should be ignored, which will affect devices being selected or not by the edge server. This paper designs and solves the problem on the system mechanism, but does not take the power supply status of the device as one of the factors to select devices.

6. CONCLUSION

In this paper, we study the process of FEL in detail and consider the device selection problem under the constraints of time and training data size of participated devices in each round.

Firstly, in order to optimize both the energy consumption and the data diversity, we formulate the problem to minimize the energy consumption and maximize the number of selected devices. Then, due to the exponential time complexity of directly solving the optimization problem, we reformulate it to devise an efficient solution for achieving greatly reduced time complexity, called E^2DS. Extensive experiments based on simulation and a real-world dataset demonstrate the effectiveness and better performance of our proposed scheme than the other two classical FEL schemes.

Next, because of the volatility of the CPU-cycle frequency, we used CMAB to learn the frequency of each device to calculate the energy consumption of the device more accurately. Moreover, since the convergence speed is also important to an FEL system, we formulate the problem considering minimizing the energy consumption as well as increasing the convergence speed. To solve this more comprehensive problem, after reformulation we design an algorithm, named FCE^2DS, to solve it while still keeping a low complexity. Finally, we validate the superiority of our proposed device selection mechanism for FEL.

REFERENCES

[1] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 2022–2035, 2020.

[2] M. S. H. Abad, E. Ozfatura, D. Gunduz, and O. Ercetin, "Hierarchical federated learning across heterogeneous cellular networks," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2020, pp. 8866–8870.

[3] J. Ren, H. Wang, T. Hou, S. Zheng, and C. Tang, "Federated learning-based computation offloading optimization in edge computing-supported internet of things," *IEEE Access*, vol. 7, pp. 69 194–69 201, 2019.

[4] M. Chen, N. Shlezinger, H. V. Poor, Y. C. Eldar, and S. Cui, "Joint resource management and model compression for wireless federated learning," in *ICC 2021-IEEE International Conference on Communications*, IEEE, 2021, pp. 1–6.

[5] W. Zhang, D. Yang, W. Wu, H. Peng, N. Zhang, H. Zhang, and X. Shen, "Optimizing federated learning in distributed industrial iot: A multi-agent approach," *IEEE Journal on Selected Areas in Communications*, 2021.

[6] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, IEEE, 2020, pp. 1–6.

[7] Y. Ye, S. Li, F. Liu, Y. Tang, and W. Hu, "Edgefed: Optimized federated learning based on edge computing," *IEEE Access*, vol. 8, pp. 209 191–209 198, 2020.

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

[9] 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 of Things Journal*, vol. 6, no. 6, pp. 10 700–10 714, 2019.

[10] Q. Hu, F. Li, X. Zou, and Y. Xiao, "Correlated participation decision making for federated edge learning," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*, IEEE, 2020, pp. 1–6.

[11] Z. Xu, Z. Yang, J. Xiong, J. Yang, and X. Chen, "Elfish: Resource-aware federated learning on heterogeneous edge devices," *arXiv preprint arXiv:1912.01684*, 2019.

[12] Z. Yu, J. Hu, G. Min, H. Lu, Z. Zhao, H. Wang, and N. Georgalas, "Federated learning based proactive content caching in edge computing," in *2018 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2018, pp. 1–6.

[13] Y. Jiang, S. Wang, B. J. Ko, W.-H. Lee, and L. Tassiulas, "Model pruning enables efficient federated learning on edge devices," *arXiv preprint arXiv:1909.12326*, 2019.

[14] L. Li, D. Shi, R. Hou, H. Li, M. Pan, and Z. Han, "To talk or to work: Flexible communication compression for energy efficient federated learning over heterogeneous mobile edge devices," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, IEEE, 2021, pp. 1–10.

[15] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 9, pp. 3400–3413, 2019.

[16] Q. Zeng, Y. Du, K. Huang, and K. K. Leung, "Energy-efficient radio resource allocation for federated edge learning," in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, IEEE, 2020, pp. 1–6.

[17] A. Taik, Z. Mlika, and S. Cherkaoui, "Data-aware device scheduling for federated edge learning," *arXiv preprint arXiv:2102.09491*, 2021.

[18] D. Ye, S. Chen, and C. Wang, "Fast convergence for federated learning in ofdma systems," in *2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, IEEE, 2021, pp. 1–6.

[19] W. Wu, L. He, W. Lin, and R. Mao, "Accelerating federated learning over reliability-agnostic clients in mobile edge computing systems," *IEEE Transactions on Parallel and Distributed Systems*, 2020.

[20] M. M. Amiri, D. Gündüz, S. R. Kulkarni, and H. V. Poor, "Update aware device scheduling for federated learning at the wireless edge," in *2020 IEEE International Symposium on Information Theory (ISIT)*, IEEE, 2020, pp. 2598–2603.

[21] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, IEEE, 2019, pp. 1–7.

[22] W. Zhang, X. Wang, P. Zhou, W. Wu, and X. Zhang, "Client selection for federated learning with non-iid data in mobile edge computing," *IEEE Access*, vol. 9, pp. 24 462–24 474, 2021.

[23] Y. J. Cho, J. Wang, and G. Joshi, "Client selection in federated learning: Convergence analysis and power-of-choice selection strategies," *arXiv preprint arXiv:2010.01243*, 2020.

[24] Y. J. Cho, S. Gupta, G. Joshi, and O. Yağan, "Bandit-based communication-efficient client selection strategies for federated learning," in *2020 54th Asilomar Conference on Signals, Systems, and Computers*, IEEE, 2020, pp. 1066–1069.

[25] T. D. Burd and R. W. Brodersen, "Processor design for portable systems," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 13, no. 2, pp. 203–221, 1996.

[26] N. H. Tran, W. Bao, A. Zomaya, N. M. NH, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, IEEE, 2019, pp. 1387–1395.

[27] N. Mhaisen, A. Awad, A. Mohamed, A. Erbad, and M. Guizani, "Analysis and optimal edge assignment for hierarchical federated learning on non-iid data," *arXiv preprint arXiv:2012.05622*, 2020.

[28] P. Gilmore and R. E. Gomory, "The theory and computation of knapsack functions," *Operations Research*, vol. 14, no. 6, pp. 1045–1074, 1966.

[29] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2, pp. 235–256, 2002.

[30] A. Sengupta, S. Amuru, R. Tandon, R. M. Buehrer, and T. C. Clancy, "Learning distributed caching strategies in small cell networks," in *2014 11th International Symposium on Wireless Communications Systems (ISWCS)*, IEEE, 2014, pp. 917–921.

[31] C. Peng, Q. Hu, J. Chen, K. Kang, F. Li, and X. Zou, "Energy-efficient device selection in federated edge learning," in *2021 International Conference on Computer Communications and Networks (ICCCN)*, IEEE, 2021, pp. 1–9.