Green Federated Learning via Energy-Aware Client Selection

Rana Albelaihi*, Liangkun Yu*, Warren D. Craft*, Xiang Sun*, Chonggang Wang†, and Robert Gazda†
*University of New Mexico, Albuquerque, NM 87131, USA.
†InterDigital Communications, Conshohocken, PA 19428, USA.

GLOBECOM 2022 - 2022 IEEE Global Communications Conference | 978-1-6654-3540-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/GLOBECOM48099.2022.10001569

***Abstract*—Federated learning (FL) is a collaborative machine learning framework to enable different clients such as Internet of Things (IoT) devices to participate in a machine learning model training process, while preserving data privacy. Client selection is critical to determine the performance of FL. Most of the existing client selection methods aim to maximize the number of selected clients, who can upload their local models before the deadline, in each global iteration, thus potentially accelerating the model convergence rate. However, these methods ignore the fact that most of the IoT devices are powered by on-board batteries and harvested green energy from the environment to prolong battery life. Hence, clients selected by these methods may not have sufficient energy to upload their local models in a global iteration or are unable to participate in the training process in the near future due to battery drainage. In this paper, we propose a novel client selection, entitled "EnerGy-AwaRe CliEnt SElection for Green FeDerated Learning (GREED)", to optimize the trade-off between maximizing the number of selected clients and minimizing the energy drawn from batteries for the selected clients, while ensuring that all the selected clients have sufficient energy to upload their local models before the deadline. The performance of GREED is validated via extensive simulations.**

***Index Terms*—Federated learning, client selection, green energy.**

I. Introduction

Various Internet of Things (IoT) devices are widely deployed to facilitate our daily lives [1]. For example, smart vehicles, which are equipped with various sensors and intelligent controllers, can assist drivers to avoid accidents, alleviate road congestion, and finally achieve autonomous drive/control. Here, the intelligent controller of an autonomous vehicle must precisely and promptly execute different navigation decisions according to the sensed road conditions [2]. Taking stop-and-go traffic as an example, the controller of an autonomous vehicle must make frequent slow-down and speed-up decisions based on traffic lights, the distance to the vehicle ahead, and other potential obstacles [3]. The controller has to utilize machine learning (ML) models to promptly make accurate decisions. For instance, a Convolutional Neural Network (CNN) model was trained to detect the lanes and cars on highways in real-time based on various on-board sensors, and then determine the speed and orientation of the vehicle to avoid a collision [4]. Traditionally, these ML models must be trained over a large volume of data in a centralized cloud data center. However, since these data may contain private information (such as users' voices and locations captured by the cameras), users are not willing to upload these data to the cloud, thus hindering the model training process.

To ensure data privacy during the training process, federated learning (FL) has been proposed to enable each IoT device to train an ML model over its local data samples. Thus, instead of uploading local data to a data center, each IoT device only uploads its local model to a centralized FL server, which aggregates all the received local models to generate a new global model that is sent back to the IoT devices so they can benefit from the FL results. FL has been widely used in various applications to train different ML models, while preserving data privacy. For example, Google applied FL to train a recurrent neural network model which predicts subsequent words when a user is typing a message in a smartphone [5].

Client selection in FL seeks to determine which clients are suitable to be selected to improve the performance of FL in terms of convergence rate and model accuracy. Many client selection methods have been investigated. For example, in order to solve the straggler problem in wireless FL (where different clients have different computing and communications capabilities, and thus some clients take much longer to upload their local models to the FL server, leading to a long delay for each global iteration), one of the existing methods is to set up a deadline for each global iteration, and requires FL client selection to maximize *qualified clients*, who can upload their local models before the deadline [6]–[10]. Note that having more clients participate in each global iteration can substantially reduce the number of global iterations required for convergence [11]. However, these client selection solutions ignore the fact that most of the IoT devices are powered by portable batteries or harvest green energy from the environment (such as solar and wind) to prolong battery life [12]. Hence, some clients selected by the existing methods: 1) cannot finish their model training and uploading before the deadline, due to a shortage of energy, thus end up wasting energy, or 2) by may quickly drain their batteries by actively participating in the FL process, and are thus unable to participate in the future.

In this paper, we propose a novel client selection solution tailored for green FL, where each client uses harvested green energy as a main power source and portable batteries as a backup. The goal of the proposed client selection is to optimize the trade-off between maximizing the number of selected clients and minimizing the overall energy pulled from the

This work was supported by the National Science Foundation under Award CNS-2148178.

portable battery for the selected clients in each global iteration, while ensuring that all the selected clients have sufficient energy to upload their local models before the deadline. The major contributions of the paper are summarized as follows.

1) We propose a green FL architecture that requires a new client selection to ensure the selected clients to have sufficient energy.
2) We formulate the client selection in green FL as an optimization problem.
3) We design the Ener**G**y-Awa**R**e Cli**E**nt S**E**lection for Green Fe**D**erated Learning (GREED) algorithm to efficiently solve this problem and analyze the performance of GREED via extensive simulations.

The rest of the paper is organized as follows. Section II briefly reviews related work. Section III introduces the related system models and formulates the client selection problem. Section IV designs the GREED algorithm to solve the problem. Section V presents the simulation results, and Section VI concludes the paper.

II. Related Work

Client selection is a critical component of the FL process, potentially affecting convergence rate, model accuracy, and clients' total energy consumption [13]. Many approaches aim to select the maximum number of clients who can finish their model training and uploading before the deadline in each global iteration. For example, Nishio and Yonetani [6] sought to increase the number of selected clients in each global iteration, proposing FedCS to jointly optimize the uploading schedule and client selection. Albelaihi *et al.* [7] developed the LEARN client selection algorithm to maximize the number of selected clients who then go on to share a wireless channel based on time-division duplexing (TDD), with the algorithm taking into account a client's forced upload delay when waiting for the communication channel to become available. Yu *et al.* [8] designed client selection to optimize the trade-off between maximizing the number of selected clients and minimizing the total energy consumption of the selected clients. By estimating the number of global iterations, which is a function of the number of selected clients in a global iteration, Shi *et al.* [14] optimized client selection to minimize overall training latency (i.e., the sum of the latency for all the global iterations). Xu *et al.* [15] argued for selecting fewer clients in early global iterations where learning performance is less sensitive to the number of selected clients and more clients in later global iterations, where learning performance is more sensitive to the number of selected clients. The suggested dynamic client sampling method has been shown to reduce overall energy consumption while improving training loss and model accuracy.

Some work focused on resource management and client selection in green FL. Silva *et al.* [16] proposed to simultaneously transmit a global model and power from the BS to clients. Accordingly, they selected all the clients and adjusted the computing latency of the clients to optimize the trade-off between the number of global iterations and the average delay of a global iteration. However, they did not consider the battery capacity constraints of the clients. Hamdi *et al.* [17] assumed that clients can harvest energy from nearby energy sources, and they designed a joint client selection and transmission power allocation method to minimize the training loss of the model, while making use of the harvested energy and ensuring a minimum SINR to each selected client. Liu *et al.* [18] designed a joint client selection and bandwidth allocation solution in green FL to optimize the trade-off between minimizing the ages of local models among all the clients and minimizing the latency of a global iteration. However, neither [17] nor [18] can guarantee the selected clients can successfully upload their local models before the deadline in a global iteration, thus leading to the straggler problem.

III. System Models and Problem Formulation

Let $\boldsymbol{\mathcal{I}}$ be the set of clients in terms of IoT devices in the BS's coverage area and each client powered by both green energy and on-board battery. Let x_i be the binary variable to indicate whether client i is selected to participate in the FL process (i.e., $x_i = 1$) or not (i.e., $x_i = 0$). Denote $\boldsymbol{\mathcal{D}}_i$ and $|\boldsymbol{\mathcal{D}}_i|$ as the set and number of data samples of client i, respectively, and so $\boldsymbol{\mathcal{D}} = \bigcup_{i \in \boldsymbol{\mathcal{I}}} \boldsymbol{\mathcal{D}}_i$ is the set of all the training data samples.

A. Traditional Federated Learning Process

FL is to determine the parameters $\boldsymbol{\omega}$ in a global model such that the global loss function $\boldsymbol{\mathcal{F}}(\boldsymbol{\omega})$ can be minimized, i.e.,

$$\arg\min_{\boldsymbol{\omega}} \boldsymbol{\mathcal{F}}(\boldsymbol{\omega}) = \arg\min_{\boldsymbol{\omega}} \sum_{i \in \boldsymbol{\mathcal{I}}} \frac{|\boldsymbol{\mathcal{D}}_i|}{|\boldsymbol{\mathcal{D}}|} f_i(\boldsymbol{\omega})\, x_i, \tag{1}$$

where $f_i(\boldsymbol{\omega})$ is the local loss function of client i, i.e.,

$$f_i(\boldsymbol{\omega}) = \frac{1}{|\boldsymbol{\mathcal{D}}_i|} \sum_{n \in \boldsymbol{\mathcal{D}}_i} f(\boldsymbol{\omega}, \boldsymbol{a}_{i,n}, b_{i,n}). \tag{2}$$

Here, $\boldsymbol{a}_{i,n}$ and $b_{i,n}$ are the inputs and output of n^{th} data sample, respectively, in $\boldsymbol{\mathcal{D}}_i$, and $f(\boldsymbol{\omega}, \boldsymbol{a}_{i,n}, b_{i,n})$ denotes the error of the local model $\boldsymbol{\omega}$. To solve Problem (1), FL comprises many global iterations, each of which comprises four steps.

1) The FL server broadcasts the current global model, denoted as $\boldsymbol{\omega}^{(k)}$, to the selected clients (where $x_i = 1$).
2) Each selected client i calculates its local model by training the received global model over its local data set $\boldsymbol{\mathcal{D}}_i$ based on, for example, the gradient descent method, i.e., $\boldsymbol{\omega}_i^{(k+1)} = \boldsymbol{\omega}_i^{(k)} - \delta \nabla f_i\left(\boldsymbol{\omega}_i^{(k)}\right)$, where δ indicates the step size or learning rate.
3) Client i uploads its derived local model $\boldsymbol{\omega}_i^{(k+1)}$ to the FL server via its wireless network.
4) The FL server aggregates the received local models and updates the global model based on FedAvg [5].

The FL continues to update the global model in each global iteration until it converges. Note that, without loss of generality, we do not consider the latency and energy consumption of the clients in downloading the global model in Step 1) and the latency of the FL server in updating the global model in Step 4) for the rest of the paper.

B. Latency Models

1) Computing latency: The computing latency of client i in training its local model over $\boldsymbol{\mathcal{D}}_i$ in a global iteration is [19]

$$t_i^{comp} = \frac{C_i|\boldsymbol{\mathcal{D}}_i|v\log_2(1/\eta)}{f_i}, \tag{3}$$

where f_i is the CPU frequency of client i in Hz, C_i is the average number of CPU cycles required for training one data sample for client i, and $v\log_2(1/\eta)$ gives the number of required local iterations to achieve the desired accuracy η. Here, $v = \frac{2}{(2-L\delta)\delta\gamma}$, where δ is the step size, and L and γ are calculated based on the eigenvalues of the Hessian matrix for the loss function (i.e., Eq. (2)).

2) Upload latency: Assume that the clients are covered by a base station (BS), and they would upload their local models to the FL server via the BS, which uses TDD to share the wireless channel among the clients. Hence, the achievable data rate of client i to upload its local model is

$$r_i = B\log_2\left(1 + \frac{p_i\,|h_i|^2}{N_0}\right), \tag{4}$$

where B is the amount of bandwidth for the BS, p_i is the maximum transmission power of client i, h_i is the channel coefficient from client i to the BS, and N_0 is the noise power. Hence, the latency of client i in uploading its local model is

$$t_i^{upload} = \frac{s}{r_i}, \tag{5}$$

where s is the size of the local model.

3) Waiting time: TDD is applied at the BS, and so if the wireless channel is not available (i.e., another client is uploading its local model), a client must wait until the wireless channel is available before uploading its local model. Specifically, let $\boldsymbol{\mathcal{J}}$ be the set of selected clients, i.e., $\boldsymbol{\mathcal{J}} = \{i \in \boldsymbol{\mathcal{I}} \mid x_i = 1\}$. The index of a selected client in $\boldsymbol{\mathcal{J}}$ is derived according to its computing latency. That is, if a selected client has a lower computing latency, then the client has a lower index, i.e., if $j_1 < j_2$, then $t_{j_1}^{comp} \leq t_{j_2}^{comp}$, where $j_1,\ j_2 \in \boldsymbol{\mathcal{J}}$. According to [8], the waiting time for client $j+1$ is

$$t_{j+1}^{wait} = \max\left\{0, t_j^{comp} + t_j^{wait} + t_j^{upload} - t_{j+1}^{comp}\right\}. \tag{6}$$

C. Energy Models

In each global iteration, the harvested green energy (as the main energy source) and the portable battery (as the backup energy source) are used to power IoT devices in terms of the selected clients for their local model computing and uploading. It is necessary to estimate the energy consumption of computing and uploading a local model for a client to see if the harvested green energy is sufficient.

1) Energy consumption of computing a local model: Assume that the CPU frequency applied at client i is f_i, and so the energy consumption of running one CPU cycle for client i is σf_i^2, where σ is a coefficient determined by the switched capacitance [20]. Also, the number of CPU cycles required by training the local model over $|\boldsymbol{\mathcal{D}}_i|$ data samples is $t_i^{comp} f_i$. Thus, the energy consumption of client i in computing its local model in a global iteration is

$$E_i^{comp} = t_i^{comp} f_i \times \sigma f_i^2 = C_i|\boldsymbol{\mathcal{D}}_i|v\log_2(1/\eta)\sigma f_i^2. \tag{7}$$

2) Energy consumption of uploading a local model: The energy consumption of client i in uploading a local model is

$$E_i^{upload} = p_i \times t_i^{upload} = \frac{p_i s}{B\log_2\left(1 + \frac{p_i h_i^2}{N_0}\right)}, \tag{8}$$

where t_i^{upload} is the uploading latency of client i in Eq. (5).

D. Green Energy Generation and Remaining Energy

Let G_i be the amount of green energy harvested by client i in a global iteration. If client i is selected, then the generated green energy will first be utilized by client i to power its model training and uploading. If the green energy generation is sufficient, then the residual green energy will be used to charge the battery of client i; otherwise, extra energy would be drawn from client i's battery to meet the energy demand of model training and uploading. Let

$$\mu_i = \left(E_i^{comp} + E_i^{upload}\right) x_i - G_i. \tag{9}$$

Here, $\mu_i \geq 0$ indicates the energy that need to be drawn from client i's battery, and $\mu_i < 0$ gives the green energy used to charge client i's battery. Hence, at the end of a global iteration, the energy in client i's battery is updated based on

$$E_i^{battery} := \begin{cases} 0, & \text{if } E_i^{battery} - \mu_i \leq 0; \\ E_i^{max}, & \text{if } E_i^{battery} - \mu_i \geq E_i^{max}; \\ E_i^{battery} - \mu_i, & \text{otherwise;} \end{cases} \tag{10}$$

where E_i^{max} is the battery capacity for client i.

E. Problem Formulation

Selecting more clients to participate in the training process can speed up the training process [11]. However, selecting more clients in each global iteration would lead to more energy drawn from the batteries for those clients, some of whom do not harvest sufficient green energy to power the model training and uploading. Hence, it is necessary to optimize the trade-off between maximizing the number of selected clients and minimizing the energy pulled from the batteries for the selected clients. We then formulate the client selection in Green FL as follows.

$$\boldsymbol{P0}: \arg\min_{\boldsymbol{x}} \alpha\sum_{i\in\boldsymbol{\mathcal{I}}}\mu_i - (1-\alpha)\sum_{i\in\boldsymbol{\mathcal{I}}} x_i, \tag{11}$$

$$\text{s.t.} \quad \forall i \in \boldsymbol{\mathcal{I}}, G_i + E_i^{battery} \geq (E_i^{comp} + E_i^{upload})x_i, \tag{12}$$

$$\boldsymbol{\mathcal{J}} = \{i \in \boldsymbol{\mathcal{I}} \mid x_i = 1\}, \tag{13}$$

$$\forall j \in \boldsymbol{\mathcal{J}}, t_j^{comp} + t_j^{wait} + t_j^{upload} \leq \tau, \tag{14}$$

$$\forall j \in \boldsymbol{\mathcal{J}}, t_{j+1}^{wait} = \max\left\{0, t_j^{comp} + t_j^{wait} + t_j^{upload} - t_{j+1}^{comp}\right\}, \tag{15}$$

$$\forall i \in \boldsymbol{\mathcal{I}}, x_i \in \{0, 1\}, \tag{16}$$

The objective is to optimize the trade-off between minimizing the amount of energy pulled from the batteries for the clients (i.e., $\sum_{i\in\boldsymbol{\mathcal{I}}} \mu_i$) and maximizing the number of selected clients (i.e., $\sum_{i\in\boldsymbol{\mathcal{I}}} x_i$), where α is a parameter to adjust the trade-off. Constraint (12) ensures that the energy available to client i, which equals the harvested green energy plus remaining energy in client i's battery, is more than the energy consumption of model training and uploading for client i if it is selected. Constraint (13) defines the set of selected clients. Constraint (14) implies that a selected client should finish its model training and uploading before the deadline τ. Constraint (15) calculates the waiting time of a selected client, and Constraint (16) indicates x_i to be a binary variable.

IV. Energy Aware Client Selection for Green Federated Learning

It is non-trivial to solve $\boldsymbol{P0}$ since the waiting time t_j^{wait} in Constraint (15) is a nonconvex function of x_i. By plugging Eq. (9) into the objective of $\boldsymbol{P0}$, i.e., Eq. (11), we have

$$\begin{aligned}&\alpha\sum_{i\in\boldsymbol{\mathcal{I}}}\left(\left(E_i^{comp}+E_i^{upload}\right)x_i-G_i\right)-(1-\alpha)\sum_{i\in\boldsymbol{\mathcal{I}}}x_i\\=&\sum_{i\in\boldsymbol{\mathcal{I}}}\left(\alpha\left(E_i^{comp}+E_i^{upload}\right)-1+\alpha\right)x_i-\sum_{i\in\boldsymbol{\mathcal{I}}}\alpha G_i. \quad (17)\end{aligned}$$

Two intuitions can guide us in minimizing such an objective function. 1) Selecting only clients with negative values of $\alpha\left(E_i^{comp}+E_i^{upload}\right)-1+\alpha$ can minimize the objective function, and 2) prioritizing clients with smaller values of $\alpha\left(E_i^{comp}+E_i^{upload}\right)-1+\alpha$ will help minimize the objective function. Based on those intuitions, we designed the Ener**G**y-Awa**R**e Cli**E**nt S**E**lection for Green Fe**D**erated Learning (GREED) algorithm to solve $\boldsymbol{P0}$. Specifically,

1) Select the clients whose values of $\alpha\left(E_i^{comp}+E_i^{upload}\right)-1+\alpha$ are negative and the energy requirements in Constraint (12) are met. Denote $\boldsymbol{\mathcal{I}}'$ as the set of these clients, i.e.,

$$\begin{aligned}\boldsymbol{\mathcal{I}}'=\Big\{i\in\boldsymbol{\mathcal{I}}\;\big|\;&\alpha\left(E_i^{comp}+E_i^{upload}\right)-1+\alpha<0\\&\&\&\;G_i+E_i^{battery}\geq E_i^{comp}+E_i^{upload}\Big\}. \quad (18)\end{aligned}$$

   Sort the clients in $\boldsymbol{\mathcal{I}}'$ based on the increasing order of $\alpha\left(E_i^{comp}+E_i^{upload}\right)-1+\alpha$, and let i' be the index of these clients.
2) Iteratively evaluate the clients in $\boldsymbol{\mathcal{I}}'$ to see if they are suitable to be selected. Here, a client being suitable to be selected means that all the constraints in $\boldsymbol{P0}$ are satisfied. That is, for example, if client i' is assumed to be selected $x_{i'}=1$, then the waiting time of all the selected clients $\boldsymbol{\mathcal{J}}$ will be updated based on Eq. (15), where $\boldsymbol{\mathcal{J}}=\left\{i\in\boldsymbol{\mathcal{I}}'\,|x_{i'}=1\right\}$. If all the selected clients can still meet the deadline requirement, i.e., Constraint (14), then client i' is suitable to be selected, i.e., $x_{i'}=1$; otherwise, $x_{i'}=0$.
3) The iteration continues until all the clients in $\boldsymbol{\mathcal{I}}'$ have been evaluated.

Algorithm 1: GREED algorithm

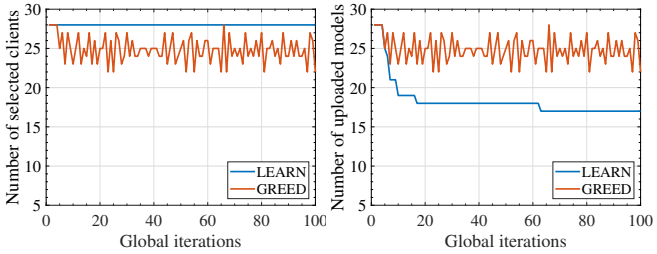
```
Calculate t_i^comp, t_i^upload, E_i^comp, E_i^upload, and E_i^battery
  for all the clients in I.
Initialize I' based on Eq. (18).
Sort the clients in I' based on the increasing order of
  α(E_i^comp + E_i^upload) − 1 + α, and let i' be the
  index of these clients.
Initialize x_{i'} = 0, ∀i' ∈ I'.
for i' = 1 to |I'| do
    x_{i'} = 1;
    Update set of the selected clients J based on its
     definition.
    for j = 1 to |J| do
        Calculate t_j^wait based on Eq. (6).
        if t_j^comp + t_j^wait + t_j^up > τ then
            x_{i'} = 0; break;
        end
    end
end
```

Algorithm 1 summarizes the proposed GREED algorithm.

V. Simulations

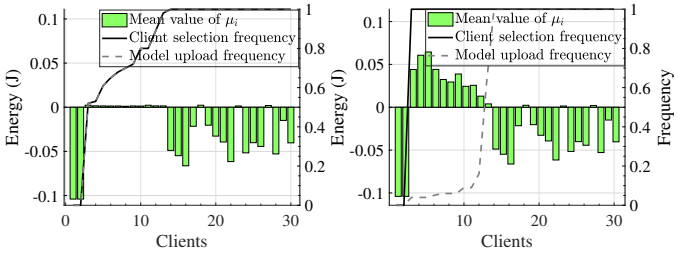
In this section, we conduct extensive simulations to evaluate the performance of GREED. As a comparison, we apply the LEARN algorithm [7] as a baseline method to select clients. The objective of LEARN is to maximize the number of selected clients without considering the green energy generation and energy supply constraint, i.e., Constraint (12). That is, the client selected by LEARN may not have sufficient energy for model training and uploading in a global iteration. We consider a scenario with 30 clients deployed in a BS's coverage area. d_i denotes the distance between client i and the BS in kilometers, and so $128.1+37.6\log_{10} d_i$ is applied to calculate the path loss between client i and the BS [21]. TDD is applied at the BS and $B=5$ MHz. In addition, we assume that the average number of CPU cycles required for training one data sample is randomly selected for different clients, i.e., $C_i\sim U(5,15)\times10^4$ cycles/sample, and the CPU frequency of a client is also randomly selected, i.e., $f_i\sim U(1.5,2)$ GHz. Moreover, we set $\gamma=2$, $L=4$, $\delta=0.1$, and $\eta=0.1$ to calculate the computing latency of a client in Eq. (3). With respect to green energy generation, we assume that each client is equipped with a solar panel, and the size of the solar panel of a client ν_i is randomly selected, i.e., $\nu_i\sim U(5,20)$ cm^2. Also, the solar radiation in the BS's coverage area is assumed to be $\zeta=0.01$ mW/cm^2, and so the amount of green energy generation of a client in a global iteration is $G_i=\zeta\nu_i\tau$. Initially, all client batteries are assumed to be fully charged. Other parameters are listed in Table I.

Let $\alpha=0.02$. Figs. 1(a) and 1(b) show the average number of selected clients and the average number of uploaded models



(a) (b)

Fig. 1: Performance of GREED and LEARN over 100 global iterations, where a) average number of the selected clients, and b) average number of uploaded local models.



(a) (b)

Fig. 2: Average values of μ_i, frequency of selection, frequency of local model uploading for all the clients among 100 global iterations, where a) GREED and b) LEARN.

TABLE I: Simulation Parameters

Parameter	Value
Size of data samples ($\|\boldsymbol{\mathcal{D}}_i\|$)	50 samples
Transmission power (p_i)	10 dBm/MHz
Noise (N_0)	-104 dBm/10MHz
Size of local model (s)	100 Kbits
Deadline of a global iteration (τ)	1.5 second
Switch capacitance coeff. in Eq. (7) (σ)	10^{-28}
Capacity of the battery E^{max}	1 J

in a global iteration, respectively, for LEARN and GREED among 100 iterations. From Fig. 1(a), we can see that although the number clients selected by LEARN (in blue) is more than GREED (in orange), among these clients selected by LEARN, only a portion of them have sufficient energy to eventually upload their local models to the FL server before the deadline in each global iteration. As shown in Fig. 1(b), since all the clients' batteries are fully charged, the clients selected by LEARN have sufficient energy supplies (in terms of green energy generation plus remaining energy in the battery) to finally upload their local models to the FL server in the first couple of global iterations. However, as the remaining energy in the selected clients' batteries keeps reducing over the global iterations, more clients selected by LEARN do not have sufficient energy to finally upload their local models. On the other hand, GREED always guarantees the selected clients have enough energy to eventually upload their local models (i.e., the orange curve in Fig. 1(a) is the same as the one in Fig. 1(b)). As a result, LEARN initially selects more clients but has fewer uploaded local models than GREED.

Figs. 2(a) and 2(b) show the average values of μ_i, frequency of selection, frequency of local model uploading incurred for all the clients in the GREED and LEARN algorithms, respectively, during 100 global iterations (clients are ordered by upload frequency). Here, μ_i in each global iteration is calculated based on Eq. (9), which indicates if client i used the amount of harvested green energy to charge its battery (i.e., $\mu_i < 0$) or pulled the amount of energy from its battery (i.e., $\mu_i > 0$). A client's "selection frequency" is the proportion of the global iterations in which that client is selected to participate (regardless of energy constraints); "model upload frequency" is the proportion of global iterations in which that client was able to upload its local model to the FL server.

In this example run, Clients 1 and 2 are never selected by LEARN and GREED because of their high computing and uploading latency, and so Constraint (14) in $\boldsymbol{P0}$ cannot be satisfied by these clients. As a result, these clients do not expend energy for FL. They continue to harvest green energy and their batteries are always at full capacity with harvested green energy being wasted. For Clients 3-13, the average μ_i value incurred by LEARN is much higher than that incurred by GREED, which is close to 0. This is because these clients are always selected by LEARN in every global iteration regardless of whether they have sufficient energy, thus leading to larger values of μ_i. GREED, on the other hand, does not always select these clients, depending on if they have sufficient energy. Note that, in Fig. 2(a), the curves for frequency of selection and model uploading in GREED overlap, demonstrating that GREED guarantees the selected clients have enough energy to eventually upload their local models. Clients 14-30 all have sufficient energy to eventually upload their local models, and thus are always selected by both GREED and LEARN (producing the same values of μ_i for GREED and LEARN).

Fig. 3 shows the trade-off in GREED between maximizing the number of selected clients and minimizing the energy pulled from the batteries (i.e., μ_i) as α increases from 0 to 1. Here, reducing α means that the system prefers to select more clients in the current global iteration rather than saving clients' energy. Changing α over global iterations can improve model accuracy since selecting fewer clients (in terms of larger α) in earlier iterations and more clients (in terms of smaller α) in later iterations can have higher model accuracy [15].

Letting $\alpha = 0.02$, we also evaluated the model testing accuracy over global iterations for LEARN and GREED. We use CIFAR-10 as a benchmark dataset. Here, CIFAR-10 is an image dataset with 10 different classes/image labels. [22]. LeNet [23] is trimmed as the ML model to be trained, where there are 1) an input layer; 2) two convolution layers (6 and 16 channels, respectively) with the kernel size of 5×5. Each convolution layer is followed by a 2 × 2 max pooling; 3) two fully connected layers (120 and 84 units, respectively) with ReLU activation; 4) a final softmax output

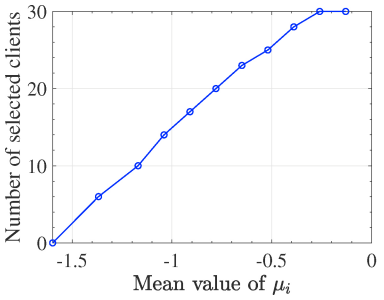


Fig. 3: The trade-off over α.

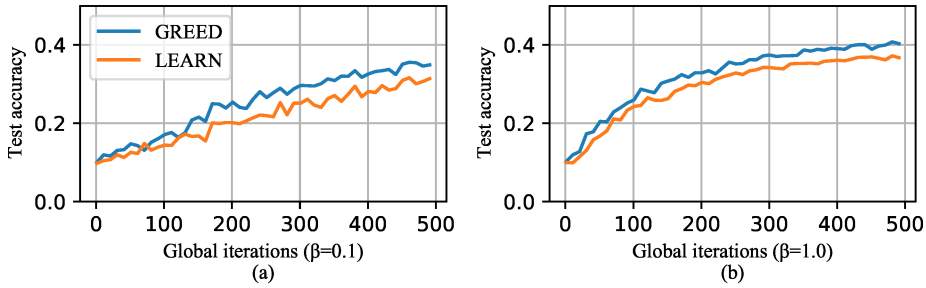


Fig. 4: Test accuracy over the global iterations, where (a) $\beta = 0.1$ and (b) $\beta = 1$.

layer. In addition, we use Dirichlet distribution to distribute the dataset (i.e., images in CIFAR-10) among the clients [24], i.e., $f(\eta_1, \eta_2, \ldots, \eta_M; \beta) = \frac{\Gamma(\beta M)}{\Gamma(\beta)^M} \prod_{m=1}^{M} \eta_m^{\beta-1}$, where $M = 10$ is the number of classes in CIFAR-10, $\Gamma(z) = \int_0^\infty x^{z-1} e^{-x} dx$ is the gamma function, and β is used to decide the level of label balance among the clients, i.e., a larger β implies less skewed label distribution among clients. That is, data are closer to independent and identical distribution (IID) among clients, and vice versa. The model is trained by the selected clients based on batch gradient descent at a learning rate of 0.05 with the batch size and number of local iterations equal 50 samples and 1 iteration, respectively. Fig. 4 shows model testing accuracy over global iterations for LEARN and GREED for two values of β. GREED always has a faster convergence rate and higher test accuracy than LEARN because GREED produces more uploaded local models from the clients.

VI. Conclusion

In this paper, we investigated client selection in green FL, where clients are powered by green energy and have limited on-board batteries. We have formulated an optimization problem to optimize the trade-off between maximizing the number of selected clients and minimizing the energy pulled from the batteries of the clients in each global iteration, while ensuring that all the selected clients have sufficient energy to upload their local models before the deadline. We designed GREED to efficiently solve the problem. The performance of GREED is demonstrated via extensive simulations.

References

[1] X. Sun and N. Ansari, "Edgeiot: Mobile edge computing for the internet of things," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 22–29, 2016.

[2] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[3] A. Ibrahim, M. Čičić, D. Goswami, T. Basten, and K. H. Johansson, "Control of platooned vehicles in presence of traffic shock waves," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 1727–1734.

[4] B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue, F. Mujica, A. Coates, and A. Y. Ng, "An empirical evaluation of deep learning on highway driving," 2015.

[5] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proc. 20th Intl. Conf. Artif. Intell. Stat.* PMLR, 20–22 Apr 2017, pp. 1273–1282.

[6] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *2019 IEEE Intl. Conf.Commun. (ICC)*, 2019, pp. 1–7.

[7] R. Albelaihi, X. Sun, W. D. Craft, L. Yu, and C. Wang, "Adaptive participant selection in heterogeneous federated learning," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.

[8] L. Yu, R. Albelaihi, X. Sun, N. Ansari, and M. Devetsikiotis, "Jointly optimizing client selection and resource management in wireless federated learning for internet of things," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[9] S. Zhai, X. Jin, L. Wei, H. Luo, and M. Cao, "Dynamic federated learning for GMEC with time-varying wireless link," *IEEE Access*, vol. 9, pp. 10 400–10 412, 2021.

[10] L. Li, H. Xiong, Z. Guo, J. Wang, and C.-Z. Xu, "SmartPC: Hierarchical pace control in real-time federated learning system," in *2019 IEEE Real-Time Systems Symposium (RTSS)*, 2019, pp. 406–418.

[11] S. U. Stich, "Local SGD Converges Fast and Communicates Little," *arXiv e-prints*, p. arXiv:1805.09767, May 2018.

[12] X. Sun and N. Ansari, "Dynamic resource caching in the iot application layer for smart cities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 606–613, 2018.

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

[14] W. Shi, S. Zhou, and Z. Niu, "Device scheduling with fast convergence for wireless federated learning," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[15] J. Xu and H. Wang, "Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective," *IEEE Trans. Wirel. Commun.*, vol. 20, no. 2, pp. 1188–1200, 2021.

[16] J. M. B. da Silva, K. Ntougias, I. Krikidis, G. Fodor, and C. Fischione, "Simultaneous wireless information and power transfer for federated learning," in *2021 IEEE 22nd International Workshop on Signal Processing Advances in Wireless Communications*, 2021, pp. 296–300.

[17] R. Hamdi, M. Chen, A. B. Said, M. Qaraqe, and H. V. Poor, "Federated learning over energy harvesting wireless networks," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 92–103, 2022.

[18] X. Liu, X. Qin, H. Chen, Y. Liu, B. Liu, and P. Zhang, "Age-aware communication strategy in federated learning with energy harvesting devices," in *2021 IEEE/CIC International Conference on Communications in China (ICCC)*, 2021, pp. 358–363.

[19] Z. Yang, M. Chen, W. Saad, C. S. Hong, M. Shikh-Bahaei, H. V. Poor, and S. Cui, "Delay Minimization for Federated Learning Over Wireless Communication Networks," *arXiv e-prints*, Jul. 2020.

[20] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power cmos digital design," *IEICE Trans. Electron.*, vol. 75, no. 4, pp. 371–382, 1992.

[21] "Radio frequency (rf) system scenarios, document tr 25.942, v.14.0.0, 3gpp, 2017."

[22] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[23] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[24] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-iid data silos: An experimental study," *arXiv preprint arXiv:2102.02079*, 2021.