

Received December 23, 2020, accepted January 4, 2021, date of publication January 8, 2021, date of current version January 20, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3050172

Dynamic Federated Learning for GMEC With Time-Varying Wireless Link

SHAOLEI ZHAI¹, XIN JIN², LING WEI¹, HONGXUAN LUO², AND MIN CAO¹

¹Electric Power Research Institute, Yunnan Power Grid Company Ltd., Kunming 650041, China

²China Southern Power Grid Research Institute Company Ltd., Guangzhou 510080, China

Corresponding author: Shaolei Zhai (slzhai12@163.com)

ABSTRACT Smart grid applications, such as predicting energy consumption, grid user behavior analysis and predicting energy theft, etc., are data-driven applications that require machine learning with a wealth of data generated from Internet of Things (IoT) based metering devices. However, traditional methods of uploading this huge data to the remote cloud for data analytics may be low efficient due to the non-negligible network transmission delay. By deploying a number of computing-enabled devices at the network edge, edge computing supports the implementation of machine learning close to the power grid environment. Considering the limited computing resources of edge devices and non-independent and identical (non-IID) data source, federated learning is a feasible edge computing based machine learning model. In federated learning, distributed mobile clients and a federated server collaborate to perform machine learning. Generally, the more clients to join the federated learning, the faster to obtain learning convergence and the higher resource utility. However, the communications between clients and the server in training rounds of federated learning may fail due to time-varying link reliability properties in a wireless network of smart grid, which not only slows down the model convergence rate but also wastes resources, such as energy consumption for invalid local training. This paper studies a dynamic federated learning problem in a power grid mobile edge computing (GMEC) environment, considering the high dynamic of link reliability. We design a delay deadline constrained federated learning framework to avoid extremely long training delay, and then formulate a dynamic client selection problem for computing utility maximization in such learning framework. Two online client selection algorithms, including *cli-max greedy* and *uti-positive guarantee*, are proposed to address the problem. The theoretical analysis and simulation results are conducted to illustrate the efficiency of the proposal.

INDEX TERMS Mobile edge computing, machine learning, federated learning, smart grid, link reliability.

I. INTRODUCTION

With the development of Internet of things (IoT), artificial intelligence (AI) as well as big data, smart grid has been a promising paradigm of the power grid systems. In such smart grid environment, multitude of data is gathered from massive IoT based electricity meters, distribution transformers, as well as other metering devices [1]. Most of data is from computation intensive applications, such as forecasting energy consumption, prediction of power quality, analytics of energy consumption trends and prediction of energy theft [2], which require big data analytics with low latency. Take predicting energy theft as an example, the faster and the higher accurate in detecting the energy theft, the smaller loss.

The associate editor coordinating the review of this manuscript and approving it for publication was Giacomo Verticale .

Traditionally, data analytics is performed in a remote cloud with AI-based algorithms, such as machine learning. The massive amount of data from smart grid is thus required to upload to the remote cloud via communication networks. However, such computing paradigm is low efficient for the ever-increasing smart grid applications that require quick and high-accurate analytics for value maximization. The reasons are as the following. Firstly, the data uploading delay from the data source to the remote cloud is inevitable and becomes the bottleneck for low latency [3]. Secondly, the data gathered from various metering devices is typically based on the usage of that device, the datasets from different metering devices in the smart grid system will be non-independent and identical distribution (non-IID).

Edge computing [4], [5], which extends cloud-like service at the network edge, is a promising computing paradigm for

smart grid. In edge computing, computing devices, such as computing-enabled mobile nodes and servers, are deployed at the network edge close to metering devices [4]. Therefore, data analytics can be performed at network edge by running computing algorithms at edge devices close to metering devices, such that data uploading delay could be explicitly reduced. However, the computation and storage resources of an edge device are quite smaller in comparison with a cloud, data analytics at only a single edge device may be low efficient [6]. Indeed, distributed learning with the collaboration of multiple edge computing devices would be more feasible.

Federated learning [7], [8] is an efficient distributed learning paradigm that supports machine learning at network edge with non-IID data. In federated learning, a number of clients (e.g., computing-enabled mobile nodes) and a federated server (located at the center of network edge) collaborate to perform multiple training rounds to obtain a learning model of special data-driven applications, e.g., a learning model of predicting the energy consumption. In each round, the federated server selects a number of clients to join this round of training. Then, the global algorithm state (e.g., model parameters) is diffused to the selected clients via a communication network. The clients then execute local training with local dataset to update the model parameters. Then, the model parameters are uploaded to the federated server for parameter aggregation. The training round repeats until the model is converged or interrupted.

Generally, the more number of clients to join the federated learning, the smaller number of rounds requiring to obtain learning convergence [7], [8]. However, since the mobile clients in smart grid are generally deployed in an outdoor environment with high dynamic of link reliability [9]–[11], not all selected clients could finally upload their trained model parameters to the federated server for parameter aggregation. For example, some selected client may not receive the global algorithm state due to a sudden loss of the link between the mobile client and the server. The failure of selected clients in uploading the trained model parameters to the server would not only slow down the learning convergence rate but also waste consumed energy for local training as well as prolong the delay of a training round. Therefore, how to select mobile clients to join training rounds of federated learning for resource utility maximization in a power grid mobile edge computing (GMEC) environment considering time-varying link reliability deserves further study.

This paper studies a dynamic distributed learning (e.g., federated learning) problem for GMEC with time-varying wireless link. In such GMEC, cloud-like computing service within a power grid wireless access network is provided by deploying a server at the center of the wireless access network and computing-enabled mobile clients (e.g., intelligent grid terminals) randomly distributed within the network. Every mobile client has a non-IID dataset, in which data is generated from nearby or equipped metering devices. The links between the mobile clients and the server are highly dynamic, that is, it may be unreliable at some time slots.

We design a delay deadline constrained federated learning scheme, where the clients of each training round are dynamically chosen considering the link state. In special, we analyze the delay as well as cost of the proposed federated learning framework. Then, a dynamic client selection problem for such framework in GMEC with time-varying link reliability is formulated. Two online client selection algorithms, e.g., *cli-max greedy* and *uti-positive guarantee*, are proposed to address the problem. The theoretical analysis and simulation studies have been conducted to illustrate the efficiency of the proposal.

The remainder of this paper is organized as follows. Section II overviews the related work. Section III describes the proposed federated learning framework in GMEC and then formulates the problem. Section IV proposes two online algorithms to address the problem. Simulation studies are conducted to demonstrate the efficiency of the proposal in Section V. We conclude this paper in Section VI.

II. RELATED WORK

Edge computing based machine learning has been widely explored in existing works for modeling, design and prediction in power grid systems.

An intelligent edge analytics approach for load identification in smart meters has been studied in [12]. Sirojan *et al.* have designed an embedded edge computing paradigm for real-time smart meter data analytics in [13]. However, since the computation resource of an embedded edge computing device is quite limit, it only analyzes the high frequency component, while high performance computing for historical data analytics is needed to perform in the remote cloud. An edge computing framework for real-time monitoring in smart grid has been proposed in [14], which used a heuristic algorithm to schedule computation tasks among edge devices to maximize the benefits. A cost-efficient tasks scheduling for smart grid communication network with edge computing system has been studied in [15]. In the proposal, a green greedy algorithm is proposed to reduce the cost of the edge computing system while satisfying the task completion needs. In order to save energy consumption and reduce the task computational latency, a deep reinforcement learning based mobile offloading scheme was proposed in [16]. To balance the computation loads of edge computing nodes, an edge-based load-balancing algorithm based on popularity and centrality was proposed in [17]. In the above proposals, the computation tasks, such as real-time power data analytics, are executed in one of the edge computing devices. Thus, these proposals are low efficient in running complex algorithms with massive amount of data. Besides, offloading data from one edge device to another may lead to a long transmission delay when the bandwidth of the link between edge devices is small or the link is unreliable.

Recently, a number of proposals have focused on edge computing based distributed learning. A fog computing model for anomaly detection in smart grids has been proposed in reference [18]. In the proposed model, distributed

computing devices at the edge of smart grid network collaborate to detect the anomalous patterns in the electricity consumption data. Kumar *et al.* have used vehicular delay-tolerant networks (VDTNs) for data dissemination to various devices in the smart grid environment using mobile edge computing [19]. In the proposal, the store-and-carry forward mechanism for message dissemination has been designed to reduce transmission delays. A big data management system, including smart grid and smart local grid, based on fog infrastructures has been designed in [20] to support real time computing services. Due to limited computing resource in fog devices, a cloud-fog-based smart grid model for efficient resource utilization was explored in [21]. Zhao *et al.* presented a smart and practical privacy-preserving data aggregation (PDA) scheme with smart pricing and packing method in fog based smart grids. An IoT-based energy management system with edge computing was proposed to improve the energy efficiency in smart cities [22].

Among various distributed learning models [23], federated learning [7], [8] is a new paradigm particularly feasible for edge computing. Forecasting electrical load using edge computing and federated learning has been explored in [24]. Saputra *et al.* also used federated learning to predict energy demand for electric vehicle networks [25]. Nishio *et al.* studied the client selection problem with resource constraints, e.g., limited computation resources, poor wireless channel conditions (longer upload time) [26]. In the proposal, the authors designed a new federated learning protocol, referred to FedCS, to select clients to join federated learning, aiming at completing the training process in a shorter time. Although the proposals in [24], [25] have illustrated the efficiency of edge computing based federated learning for smart grid, the deployment of federated learning combined with edge computing, such as which clients should be selected to join federated learning, is presently in its initial stage [26].

III. A FEDERATED LEARNING FRAMEWORK FOR GMEC

We consider a GMEC environment with computing-enabled mobile clients (e.g., intelligent grid terminals) and an edge server, which is endowed with cloud-like computing service. The mobile clients are randomly distributed in a time-varying wireless network, while the edge server is located at the center of that network. Each mobile client has a dataset comprising a number of samples from the nearest IoT sensors (e.g., ammeter, water meter, camera). Generally, the dataset of a mobile client is small and contains a partial information of the GMEC system. For example, the dataset of a mobile client only includes several grid users' ammeter information. However, smart grid applications, such as forecasting energy consumption, grid user behavior analysis and predicting energy theft, require machine learning based big data analytics with more datasets.

Assuming that computation intensive power grid applications, e.g., forecasting energy consumption, are deployed in both mobile clients and the server. We design a delay

deadline constrained federated learning framework as shown in *Framework 1* to support machine learning based power grid applications at the network edge with the collaboration of mobile clients and the edge server.

As shown in *Framework 1*, when the edge server, which acts as a federated server, has a computation task, for example, a task of forecasting energy consumption, it initially tells the mobile clients what computation to run and a special data structure that will be used in the training process as well as other information required for the task of forecasting energy consumption, via broadcasting the above information in GMEC. Then, a number of learning rounds, where a round includes the stages from *client selection* to *algorithm state aggregation*, are iterated until the objective is achieved.

Framework 1 Delay Deadline Constrained Federated Learning

- 1: *Initialization*: The edge server tells the mobile clients in GMEC that there is a learning to start.
 - 2: *Client selection*: A *timer* is set for receiving the request from clients. Once the timer reaches the delay deadline, the edge server selects a number of clients from the requesting clients to join this round of training.
 - 3: *Training*: At the beginning of this step, a *timer* is set in the edge server.
 - a) *Algorithm state diffusion*: The edge server sends the most updated global model parameters to the selected clients.
 - b) *Local training*: The selected clients perform local computing based on the global model parameters and its local dataset.
 - c) *Algorithm state uploading*: The clients send the trained model parameters to the edge server.
 - 4: *Algorithm state aggregation*: After receiving all the algorithm state uploading messages, or, when the timer of the training step reaches its delay deadline, the server averages the trained model parameters from the selected clients to global model parameters, and stores them as a checkpoint in the server.
-

Due to the time-varying link reliability properties, the communications between some clients and the server may fail at some time slots. In order to avoid that the edge server may waste a long time to wait for the response from the unreachable clients, as shown in *Framework 1*, we have added two timers at *client selection* and *training* stages of a round of training, respectively. When a timer reaches its delay deadline, the learning process will go to next step immediately.

In the following, we will mathematically analyze the delay as well as cost models of the proposed delay deadline constrained federated learning framework. Then, a dynamic client selection problem for cost minimization under the time-varying wireless link is formulated.

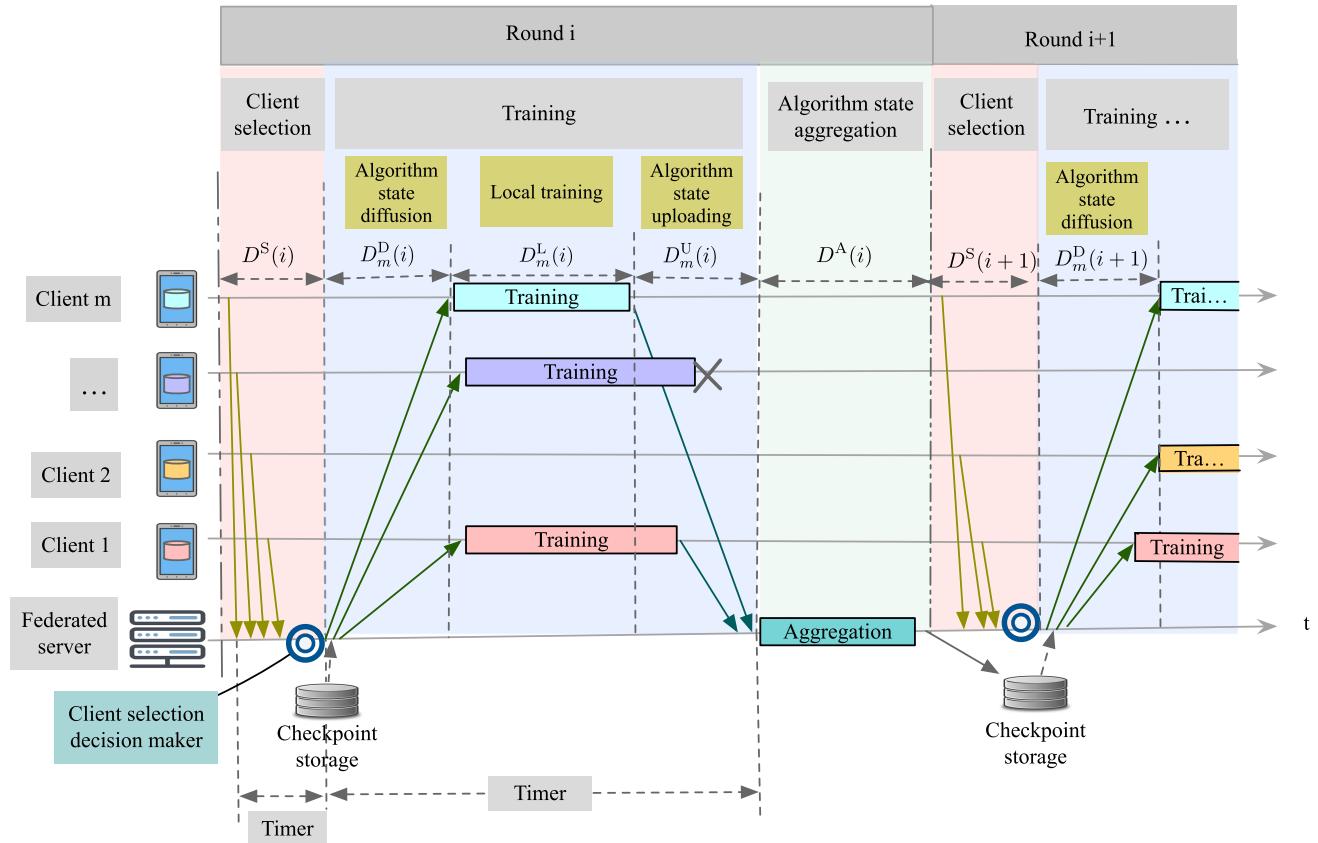


FIGURE 1. Per-round evolution of the delay deadline constrained federated learning.

A. SYSTEM MODEL

Let $\mathbb{M} = \{1, 2, \dots, M\}$ be the mobile client set in the GMEC system. Let $\mathcal{I}(i) = (I_1(i), I_2(i), \dots, I_M(i))$ be a binary client selection decision vector at i th round of training for mobile clients in the GMEC system. $I_m(i) = 1$ for $m \in \mathbb{M}$ indicates that mobile client m is selected to join the i th round of federated learning; otherwise, $I_m(i) = 0$, which indicates the client is not selected. The summary of notations used in this paper are listed in Table 1.

B. TIME-VARYING LINK RELIABILITY

According to *Framework 1*, we depict the detail of a round of training as in Fig. 1. As shown in Fig. 1, at each round of training, there are three types of communications between mobile clients and the federated server.

1) TIME-VARYING UPLINK FOR REQUESTING FEDERATED LEARNING

The first is in the *client selection* stage, where the mobile client sends a request to the edge server for joining the federated learning, which requires the uplink communication from a mobile client to the server. Assume that at the beginning of each round, every mobile client is willing to join the training. It tries to send the request to the server via a time-varying uplink. Let $\Psi_m^J(i)$ (which is assumed to be i.i.d. over rounds)

be a binary uplink reliability variable of client m at the client selection stage of the i th round, where $\Psi_m^J(i) = 1$ indicates that the link is reliable and thus that the request can be successfully transmitted to the server via the link; and $\Psi_m^J(i) = 0$, we say that the link is fail and thus the request is unable to reach the server.

2) TIME-VARYING DOWNLINK FOR DIFFUSING ALGORITHM STATE

As shown in Fig. 1, once a client is selected to join federated learning, the federated server will send the most updated global algorithm state (e.g., model parameters) to the client, which requires the downlink communication. Let $\Psi_m^D(i)$ be a binary downlink reliability variable of client m at *algorithm state diffusion* stage of the i th round. Then, $\Psi_m^D(i) = 1$ indicates that the link is reliable; and $\Psi_m^D(i) = 0$ indicates that the link is fail and the global algorithm state cannot reach the client.

3) TIME-VARYING UPLINK FOR ALGORITHM STATE UPLOADING

After local training, the client will send the up-to-date algorithm state (e.g., trained model parameters) to the edge server for federated aggregation, which requires the uplink communication, as shown in Fig. 1. Let $\Psi_m^U(i)$ be a binary uplink

TABLE 1. Summary of notations.

Symbols	Definition
M	The number of mobile clients in a GMEC system
N	The number of rounds in a federated learning
i	The round index of a federated learning process
$M(i)$	The number of mobile clients selected to join the i th round of training
$M_{\text{succ}}(i)$	The number of selected clients that have successfully uploaded the algorithm state update
$I_m(i)$	The binary client selection variable of client m at the i th round
$\Psi_m^J(i)$	The binary uplink reliability variable of client m in the i th round at the time requesting FL
$\Psi_m^D(i)$	The binary downlink reliability variable of client m in the i th round at the stage of algorithm state diffusion
$\Psi_m^U(i)$	The binary uplink reliability variable of client m in the i th round at the stage of algorithm state uploading
f_m	The computation capability (samples per second) of the m th mobile client
ρ_m	The expected link reliability of the link between client m and the server.
$N_m^L(i)$	The number of local update iterations of client m at the i th round
$S_m(i)$	The number of samples in the local dataset at the i th round
$B_m(i)$	The local batch size of client m at the i th round
$D_m^J(i)$	The uplink transmission delay of sending request
$D^S(i)$	The client selection delay of the i th round of training
$D_m^D(i)$	The algorithm state diffusion delay of mobile client m at the i th round of training
$D_m^L(i)$	The local training delay of mobile client m at the i th round
$D_m^U(i)$	The algorithm state uploading delay from mobile client m to the federated server at the i th round
$D_m^T(i)$	The training delay of mobile client m at the i th round
$D^T(i)$	The training delay of the i th round
$D^A(i)$	The algorithm state aggregation delay of the i th round
D_{\max}^S	A timer of federated server for waiting the request from clients
D_{\max}^T	A timer of federated server for waiting the training feedback from clients
$E_m(i)$	The local-training energy consumption of client m at the i th round
γ_m	The local computation power factor of client m
α, β, ω	The weighted parameters
\mathbb{M}	The mobile client vector
$\mathbb{M}(i)$	The selected mobile client vector at the i th round
$\mathbb{M}_{\text{succ}}(i)$	The mobile client vector that have successfully uploaded the algorithm state update
$\mathcal{I}(i)$	The binary client selection vector at the i th round

reliability variable of client m in the i th round at the stage of *algorithm state uploading*, where $\Psi_m^U(i) = 1$ indicates that the link is reliable; and $\Psi_m^U(i) = 0$ means the link fails, thus the algorithm state cannot upload to the server.

C. DELAY MODEL

Let $\mathbb{M}(i) \subseteq \mathbb{M}$ be a set of mobile clients selected for the i th round of federated learning at the *client selection* stage. Then, for any $m \in \mathbb{M}(i)$, we have $I_m(i) = 1$.

1) CLIENT SELECTION DELAY

As shown in Fig. 1, the client selection delay refers to the time interval from the beginning of a round to the time that

decision is made on which clients will be selected to join federated learning of this round, representing by $D^S(i)$, where i means at the i th round. Since the time interval for making client selection is far shorter than that of sending the request from clients to the server, the client selection delay can be expressed by

$$D^S(i) = \min[\max_{m \in \mathbb{M}} \Psi_m^J(i) D_m^J(i), D_{\max}^S], \quad (1)$$

where D_{\max}^S is a timer of federated server for waiting the request from clients, as illustrated in Framework 1. (1) indicates that, once the timer reaches its deadline, it will make client selection decision immediately based on the clients that the server has successfully received their requests.

2) TRAINING DELAY

As shown in Fig. 1, the training delay consists of the following three types of delays.

- (a) *Algorithm state diffusion delay*. After the *client selection* stage, the federated server initiates *algorithm state diffusion* stage to send the most updated global model parameters to each of the selected clients. The time interval from the federated server sends out the messages to a client receives the information is called this client's algorithm state diffusion delay, e.g., $D_m^D(i)$ represents the algorithm state diffusion delay of mobile client m at the i th round of training. The algorithm state diffusion delay is affected by the downlink state and the packet size of the diffusing information.
- (b) *Local training delay*. When a client receives the global model parameter, it starts local training based on the global model parameters and its local dataset. The local training delay is affected by the size of its local dataset, size of a batch, number of local update iterations and local computing capability. Let $D_m^L(i)$ be the local training delay of client m at the i th round, which can be expressed by

$$D_m^L(i) = \frac{N_m^L(i)S_m(i)}{B_m(i)} D_m^{\text{CPU},B}, \quad (2)$$

where $D_m^{\text{CPU},B} = B_m(i)/f_m$ is the delay of processing a local training with a batch.

- (c) *Algorithm state uploading delay*. When local training finishes, the client will upload the local algorithm state (e.g., local trained model parameters) to the federated server via the uplink. The time interval from a client uploads the algorithm state to the federated server receives it is called algorithm state uploading delay of this client, e.g., $D_m^U(i)$ is the algorithm state uploading delay of client m at the i th round of training. The algorithm state uploading delay is affected by the uplink state and the packet size of the uploading information.

Since different clients may experience different training delay due to downlink and uplink states,¹ local computing

¹We will extend the special diffusion/uploading delay model related to special access technology in the future work.

capabilities and size of local datasets, in synchronous update scheme, the training delay of a round is the maximum training delay of the selected clients at this round. That is, the training delay of the i th round can be derived by

$$D^T(i) = \max_{m \in \mathbb{M}(i)} D_m^T(i), \quad (3)$$

where $D_m^T(i)$ is the training delay of client m at the i th round, which is derived by

$$D_m^T(i) = \Psi_m^D(i) \Psi_m^U(i) \left(D_m^D(i) + D_m^L(i) + D_m^U(i) \right). \quad (4)$$

Notice that, once the uplink is fail (e.g., $\Psi_m^D(i) = 0$) at *algorithm state diffusion* stage, the subsequent substages for client m , including *local training* and *algorithm state uploading*, would not come. Similarly, if the uplink is fail at the beginning of the *algorithm state uploading* substage, the *algorithm state uploading* substage would not come. Therefore, we set $D_m^T(i) = \infty$ when $\Psi_m^D(i) = 0$ or $\Psi_m^U(i) = 0$.

3) ALGORITHM STATE AGGREGATION DELAY

When the federated server receives all the algorithm state uploading messages from the selected clients, or, when the training timer reaches the delay deadline, it starts to average the received local trained model parameters to global model parameter, and storing them as a checkpoint. We use $D^A(i)$ to represent the algorithm state aggregation delay at the i th round of training.

4) DELAY OF A ROUND

As shown in Fig. 1, when the federated server selects a number of clients to join a round of training, it goes into the state of waiting the selected clients to feedback the algorithm state. However, the events of downlink failure at the algorithm state diffusion substage, or uplink failure in the algorithm state uploading substage would make the feedback from that client fails. Accordingly, we have set a training timer, defined as D_{\max}^T ,² in the training stage to avoid the federated server waits infinitely for unreachable clients, as illustrated in *Framework 1*.

Therefore, the delay of the i th round could be expressed by

$$D(i) = D^S(i) + \min[D^T(i), D_{\max}^T] + D^A(i). \quad (5)$$

D. COST MODEL

This paper mainly considers the energy consumption of mobile clients that have been selected to join the federated learning but fail in the training stage, including failure in algorithm state diffusion and algorithm state uploading substages due to downlink/uplink unreliability.

As shown in Fig. 1, if the downlink at the algorithm state diffusion substage fails, there will be no further action at the unreachable client, such that there is no energy consumption in that client. Differently, if the uplink at the algorithm

²Note that, for simplification, assume that the value of a timer at every round is identical.

state uploading substage fails, the unreachable client has consumed energy for local training, which will be a waste since this round of local training has no contribution to the global algorithm state update. Accordingly, the wasted energy of round i can be expressed by

$$E(i) = \sum_{m \in \mathbb{M}(i)} (1 - \Psi_m^U(i)) E_m(i), \quad (6)$$

where $E_m(i) = \gamma_m (S_m(i))^3 / (D_m^L(i))^2$.

Therefore, we define the cost of a round i as the following

$$\mathcal{C}(i) = \alpha E(i) + \beta D(i). \quad (7)$$

E. PROBLEM FORMULATION

Reference [7], [8] have pointed out that, the more number of clients joining the federated learning, the smaller number of rounds requiring to obtain learning convergence. However, due to the time-varying link reliability property, not all presenting available clients could finally finish a round of training. The client that fails in uploading its algorithm state may waste its energy for the invalid local training. Therefore, we define the *computing utility* of the GMEC system at the i th round as the following

$$\mathcal{G}(i) = M_{\text{succ}}(i) - \omega \mathcal{C}(i), \quad (8)$$

where $M_{\text{succ}}(i)$ is the number of selected clients that have successfully uploaded the algorithm state update.

The long-term computing utility is then defined as

$$\mathcal{G} = \mathbb{E}[\mathcal{G}(i)] = \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{i=1}^N \mathcal{G}(i). \quad (9)$$

Then, the dynamic client selection for federated learning problem (CSFL) in a time-varying wireless network environment can be formulated as

$$\text{Maximize: } \mathcal{G} \quad (10)$$

$$\text{Subject to: } \Psi_m^J(i), \Psi_m^D(i), \Psi_m^U(i) \in \{0, 1\}, \quad \forall m \in \mathbb{M} \quad (11)$$

$$\mathcal{I} = (\mathcal{I}(1), \mathcal{I}(2), \dots, \mathcal{I}(i), \dots, \mathcal{I}(N)) \quad (12)$$

$$\mathcal{I}(i) = (I_1(i), I_2(i), \dots, I_m(i), \dots, I_M(i)) \quad (13)$$

$$\mathbb{M}_{\text{succ}}(i) \subseteq \mathbb{M}(i) \quad (14)$$

$$\Psi_m^J(i) = \Psi_m^D(i) = \Psi_m^U(i) = 1, \quad \forall m \in \mathbb{M}_{\text{succ}}(i) \quad (15)$$

$$I_m(i) = 1, \quad \forall m \in \mathbb{M}(i) \quad (16)$$

$$\alpha, \beta, \omega > 0. \quad (17)$$

IV. THE ONLINE ALGORITHMS FOR CSFL

A. THE CLI-MAX GREEDY ALGORITHM

Because of the high dynamic of the wireless network, it would be difficult and very costly to accurately estimate the link reliability over time slots. Therefore, offline methods (e.g., the linear/nonlinear programming solver) are unsuitable for solving the dynamic client selection problem for federated learning at a time-varying link reliability environment described in (10)-(17). Accordingly, in this subsection, we first design

an online client maximum greedy (*Cli-max greedy*) algorithm for solving the problem. Then, we theoretically analyze the performance in terms of *computing utility* of the proposed cli-max greedy algorithm.

1) THE ALGORITHM

As illustrated in *Algorithm 2*, at each round of training, when the federated server receives the computation requests from all the mobile clients, or, the timer of the client selection stage reaches the deadline, the *cli-max greedy* algorithm is initiated to greedily select all the clients that have successfully sent their requests to the server, equivalently, the clients whose links are reliable during the client selection stage would be chosen to join that round of training.

Algorithm 2 Cli-Max Greedy Algorithm

At each round i for $i = 1, 2, \dots, N$ of federated learning, **do**

- 1) At the beginning of each round i , while $\Psi_m^J(i) = 1, \forall m \in \mathbb{M}$, **or**, the timer of the client selection stage reaches the deadline (e.g., client selection delay has reached D_{\max}^S), **do**
 - a) **Initialization:** $\mathbb{M}(i) = \{\}$.
 - b) **Greedy decision:** for all $m \in \mathbb{M}$, **do**
If $\Psi_m^J(i) = 1$ then, set

$$\begin{cases} I_m(i) = 1, \\ \mathbb{M}(i) = \mathbb{M}(i) \cup \{i\}. \end{cases} \quad (18)$$

else set $I_m(i) = 0$.

- 2) Performing the remaining stages as shown in Fig. 1 with determined $\mathbb{M}(i)$ and $\mathcal{I}(i)$.
-

2) PERFORMANCE ANALYSIS

Assuming that the reliability of every time-varying link follows i.i.d over time slots. Let $P(\Psi_m^J(i) = 1) = P(\Psi_m^D(i) = 1) = P(\Psi_m^U(i) = 1) = \rho_m, \forall m \in \mathbb{M}$ and $i = 1, 2, \dots, N$, where $P(\cdot)$ is a probability function.

We have the following theorem.

Theorem 1: Consider a federated learning system as illustrated in Framework 1 with time-varying links whose link reliability follows i.i.d over time slots, then, under the cli-max greedy algorithm, the long-term computing utility satisfies

$$\mathcal{G} \geq \sum_{m \in \mathbb{M}} \rho_m^3 - A \sum_{m \in \mathbb{M}} \rho_m (1 - \rho_m) E_m^{\max} - B, \quad (19)$$

where $A = \omega \alpha$ and $B = \omega \beta (D_{\max}^S + D_{\max}^T + D^A)$ are constants.

Proof: According to (5), we have

$$D(i) \leq D^S(i) + D_{\max}^T + D^A(i). \quad (20)$$

Similarly, according to (1), $D^S(i) \leq D_{\max}^S$ holds. Substituting into (20), we obtain that

$$D(i) \leq D_{\max}^S + D_{\max}^T + D^A,$$

where $D^A = \max[D^A(i) : i = 1, 2, \dots, N]$. Subsituting into (7), we have

$$C(i) \leq \alpha E(i) + \beta (D_{\max}^S + D_{\max}^T + D^A),$$

further substituting into (8), we obtain

$$\mathcal{G}(i) \geq M_{\text{succ}}(i) - \omega \alpha E(i) - \omega \beta (D_{\max}^S + D_{\max}^T + D^A). \quad (21)$$

Substituting into (9) and with some calculus, we have

$$\mathcal{G} \geq \mathbb{E}[M_{\text{succ}}(i)] - \omega \alpha \mathbb{E}[E(i)] - \omega \beta (D_{\max}^S + D_{\max}^T + D^A). \quad (22)$$

On the other hand, for any client $m \in \mathbb{M}$ that would successfully join the i th round of local training, its link reliability must satisfies $\Psi_m^J(i) = \Psi_m^D(i) = \Psi_m^U(i) = 1$. Under the *cli-max greedy* client selection policy, any client $m \in \mathbb{M}$ that satisfies $\Psi_m^J(i) = 1$ would be selected to join the i th round of training, as illustrated in *Algorithm 2*. That is, $I_m(i) = 1$ holds when $\Psi_m^J(i) = 1$. Therefore,

$$\begin{aligned} \mathbb{E}[M_{\text{succ}}(i)] &= \sum_{m \in \mathbb{M}} \mathbb{E}[\mathbb{1}(\Psi_m^J(i) = \Psi_m^D(i) = \Psi_m^U(i) = 1) I_m(i)] \\ &\stackrel{\text{i.i.d.}}{=} \sum_{m \in \mathbb{M}} \mathbb{E}[\mathbb{1}(\Psi_m^J(i) = 1) I_m(i)] \mathbb{E}[\mathbb{1}(\Psi_m^D(i) = 1)] \\ &\quad \times \mathbb{E}[\mathbb{1}(\Psi_m^U(i) = 1)] \\ &\stackrel{\text{Alg.2}}{=} \sum_{m \in \mathbb{M}} 1 \cdot P(\Psi_m^J(i) = 1) P(\Psi_m^D(i) = 1) P(\Psi_m^U(i) = 1) \\ &= \sum_{m \in \mathbb{M}} \rho_m^3, \end{aligned}$$

where $\mathbb{1}(X) = \{0, 1\}$, if X is true then $\mathbb{1}(X) = 1$; and $\mathbb{1}(X) = 0$, otherwise.

Note that, under some other client selection policies, such as the policy that the client m satisfying $\Psi_m^J(i) = \Psi_m^D(i) = \Psi_m^U(i) = 1$ may not be chosen (e.g., $I_m(i) = 0$), $\mathbb{E}[M_{\text{succ}}(i)] \leq \sum_{m \in \mathbb{M}} \rho_m^3$ holds.

Again, for any client $m \in \mathbb{M}$ that would be selected to join the i th round of training, its link reliability must satisfy $\Psi_m^J(i) = 1$. Under the *cli-max greedy* algorithm, $I_m(i) = 1$ holds when $\Psi_m^J(i) = 1$. Accordingly,

$$\begin{aligned} \mathbb{E}[E(i)] &= \sum_{m \in \mathbb{M}} \mathbb{E}[\mathbb{1}(\Psi_m^J(i) = 1) I_m(i) (1 - \Psi_m^U(i)) E_m(i)] \\ &= \sum_{m \in \mathbb{M}} 1 \cdot P(\Psi_m^J(i) = 1) (1 - P(\Psi_m^U(i) = 1)) \mathbb{E}[E_m(i)] \\ &= \sum_{m \in \mathbb{M}} \rho_m (1 - \rho_m) \mathbb{E}[E_m(i)] \\ &\leq \sum_{m \in \mathbb{M}} \rho_m (1 - \rho_m) E_m^{\max}, \end{aligned}$$

where $E_m^{\max} = \max[E_m(i) : i = 1, 2, \dots, N]$.

Define $A = \omega\alpha$ and $B = \omega\beta(D_{\max}^S + D_{\max}^T + D^A)$, and substituting the above results into (22), we obtain

$$\mathcal{G} \geq \sum_{m \in \mathbb{M}} \rho_m^3 - A \sum_{m \in \mathbb{M}} \rho_m (1 - \rho_m) E_m^{\max} - B.$$

Then the statement follows. \square

B. PERFORMANCE IMPROVEMENT

Although the *cli-max greedy* algorithm is low-complexity, the cost (e.g., energy consumption for invalid local training) may be high due to the high dynamic of wireless link state. It is a sub-optimal solver for the CSFL problem described in (10)-(17). To further reduce the cost, in this subsection, we analyze the computing utility of per selected client in the long-term. Then, the client selection policy is improved based on the analytical results.

1) PER-CLIENT COMPUTING UTILITY

Assuming that the server has received the request from a client $i \in \mathbb{M}$ at the i th round of training. We define the computing utility of the client at that round as the following

$$\mathcal{G}_m(i)|_{\Psi_m^J=1} = I_m(i)(\mathbb{1}_{\text{suss}}(i) - \omega\mathcal{C}_m(i)), \quad (23)$$

where $\mathbb{1}_{\text{suss}}(i)$ is a binary variable indicating the success of the local training in that client. If the local training succeeds, then $\mathbb{1}_{\text{suss}}(i) = 1$; $\mathbb{1}_{\text{suss}}(i) = 0$, otherwise. $\mathcal{C}_m(i)$ is the cost of client m , which is defined as

$$\mathcal{C}_m(i) = \alpha(1 - \Psi_m^U(i))E_m(i) + \beta D_m^T(i)/M. \quad (24)$$

For every client m with $\Psi_m^J = 1$, we have the following theorem.

Theorem 2: Consider a federated learning system as illustrated in Framework 1 with time-varying links whose link reliability follows i.i.d over time slots, then, under any client selection algorithm, the expected computing utility of any client m with $\Psi_m^J(i) = 1$ for $i = 1, 2, \dots, N$ follows

$$\mathcal{G}_m|_{\Psi_m^J=1} = \mathbb{E}[I_m(i)](\rho_m^2 - A(1 - \rho_m)\bar{E}_m - C), \quad (25)$$

where $\bar{E}_m = \mathbb{E}[E_m(i)]$, $\bar{D}_m^T = \mathbb{E}[D_m^T(i)]$, $A = \omega\alpha$ and $C = \omega\beta\bar{D}_m^T/M$.

Proof: According to the assumptions in Section III, if the link between a selected client m and the server at both the *algorithm state diffusion* and *algorithm state uploading* substages are reliable (e.g., $\Psi_m^D(i) = \Psi_m^U(i) = 1$), then the local training succeeds; otherwise, the local training fails. Therefore, the successful indication function of local training can be expressed as $\mathbb{1}_{\text{suss}}(i) = \mathbb{1}(\Psi_m^D(i) = \Psi_m^U(i) = 1)$. Substituting it and (24) into (23), we have

$$\begin{aligned} \mathcal{G}_m(i)|_{\Psi_m^J(i)=1} &= I_m(i)(\mathbb{1}(\Psi_m^D(i) = \Psi_m^U(i) = 1) \\ &\quad - \omega I_m(i)(\alpha(1 - \Psi_m^U(i))E_m(i) + \beta D_m^T(i)/M)). \end{aligned} \quad (26)$$

Take expectation over rounds in both sides, we obtain

$$\begin{aligned} \mathcal{G}_m|_{\Psi_m^J=1} &= \mathbb{E}\left[I_m(i)\left(\mathbb{1}(\Psi_m^D(i) = \Psi_m^U(i) = 1)\right)\right] \\ &\quad - \mathbb{E}\left[\omega I_m(i)\left(\alpha(1 - \Psi_m^U(i))E_m(i) + \beta D_m^T(i)/M\right)\right] \\ &= \mathbb{E}[I_m(i)]P(\Psi_m^D(i) = 1)P(\Psi_m^U(i) = 1) \\ &\quad - \omega\alpha\mathbb{E}[I_m(i)](1 - P(\Psi_m^U(i) = 1))\mathbb{E}[E_m(i)] \\ &\quad - \omega\beta\mathbb{E}[I_m(i)]\mathbb{E}[D_m^T(i)/M] \\ &= \rho_m^2\mathbb{E}[I_m(i)] - \omega\alpha(1 - \rho_m)\mathbb{E}[I_m(i)]\mathbb{E}[E_m(i)] \\ &\quad - \omega\beta\mathbb{E}[I_m(i)]\mathbb{E}[D_m^T(i)/M]. \end{aligned} \quad (27)$$

Let $\mathbb{E}[E_m(i)] = \bar{E}_m$ and $\mathbb{E}[D_m^T(i)] = \bar{D}_m^T$, substituting into (27), we have

$$\mathcal{G}_m|_{\Psi_m^J=1} = \mathbb{E}[I_m(i)]\left(\rho_m^2 - \omega\alpha(1 - \rho_m)\bar{E}_m - \omega\beta\bar{D}_m^T/M\right).$$

Define $A = \omega\alpha$ and $C = \omega\beta\bar{D}_m^T/M$, accordingly, we have

$$\mathcal{G}_m|_{\Psi_m^J=1} = \mathbb{E}[I_m(i)]\left(\rho_m^2 - A(1 - \rho_m)\bar{E}_m - C\right),$$

which ends the proof. \square

2) THE UTILITY-POSITIVE GUARANTEE ALGORITHM

The result of *Theorem 2* shows that, as to a client who has successfully sent its request to the server (e.g., $\Psi_m^J = 1$), if the server determines to reject the client for joining this round of training (e.g., $I_m = 0$), then it gets $\mathcal{G}_m = 0$ from this client. However, if it selects the client to join the training, then, $\mathcal{G}_m = \rho_m^2 - A(1 - \rho_m)\bar{E}_m - C$, which could be positive or negative due to the dynamic link reliability property. Generally, if at a decision epoch, the server can assure that the behavior of selecting the client (e.g., set $I_m = 1$) would achieve a positive computing utility, then, the better decision is to select the client; otherwise, it is better not to select it.

Based on the above analysis and the result of *Theorem 2*, we design the utility-positive guarantee (*cli-positive guarantee*) algorithm for improving the performance in terms of computing utility of the CSFL problem. Specifically, as shown in *Algorithm 3*, at each round of training, when the server receives the requests from all the mobile clients, or, the timer of the client selection stage reaches the deadline, the *uti-positive guarantee* algorithm is initiated to select the clients to join that round of training. In the algorithm, we use prediction methods, such as recurrent neural network (RNN) [27], [28], to predict the link reliability (e.g., ρ_m for $m \in \mathbb{M}_{\text{req}}(i)$) at both *algorithm state diffusion* and *algorithm state aggregation* substages. Then, the per-client computing utility at that round can be estimated. Finally, the clients who have the positive per-client computing utility are selected to join that round of training.

V. PERFORMANCE EVALUATION

This section investigates the computing utility performance of the proposed *cli-max greedy* and *uti-positive guarantee* algorithms.

Algorithm 3 Uti-Positive Guarantee Algorithm

At each round i for $i = 1, 2, \dots, N$ of federated learning, **do**

- 1) At the beginning of each round i , while $\Psi_m^J(i) = 1$, $\forall m \in \mathbb{M}$, **or**, the timer of the client selection stage reaches the deadline (e.g., client selection delay has reached D_{\max}^S), **do**
 - a) **Initialization:** $\mathbb{M}(i) = \{\}$.
 - b) **Estimating the link reliability:** obtain ρ_m for $m \in \mathbb{M}$ with predicting methods, such as RNN.
 - c) **Uti-positive guarantee decision:** for $m \in \mathbb{M}_{req}(i)$, where $\mathbb{M}_{req}(i) = \{m : m \in \mathbb{M} \text{ and } \Psi_m^J(i) = 1\}$, **do**
 - i) Calculate $\mathcal{G}'_m(i)$ with (28).

$$\mathcal{G}'_m(i) = \rho_m^2 - A(1 - \rho_m)\bar{E}_m - C, \quad (28)$$

where A and C are constants that from *Theorem 2*.

- ii) If $\mathcal{G}'_m(i) > 0$ then, set

$$\begin{cases} I_m(i) = 1, \\ \mathbb{M}(i) = \mathbb{M}(i) \cup \{i\}. \end{cases} \quad (29)$$

else set $I_m(i) = 0$.

- 2) Performing the remaining stages as shown in Fig. 1 with determined $\mathbb{M}(i)$ and $\mathcal{I}(i)$.

TABLE 2. The basic parameter settings.

Parameters	Value
Number of clients M	100
Computing capability f_m (samples/s)	312.5
Local update N_m^L (iterations/round)	1
Dataset size S_m (samples/round/client)	[2000, 4000]
Batch size (samples/batch)	100
Link reliability probability	[0.1, 1.0]
$P(\Psi_m^J = 1) = P(\Psi_m^D = 1) = P(\Psi_m^U = 1)$	
Mean D_m^J (s)	0.01
Mean D_m^D, D_m^U (s)	0.019
Power factor γ_m	10^{-14}

In each of the simulation run, the GMEC system has a computation task of object classification, which will be run in the federated learning model with independent image datasets in distributed mobile clients. Each dataset is a subset of MNIST [29], which comprise 60,000 training images and 10,000 testing images with 10 object classes (i.e., 10 digits). An image is equivalent to a sample. At each round of training, the federated server selects a number of clients with the investigated client selection algorithms. For model accuracy, the number of rounds at each simulation run satisfies $N \geq 1,000$ rounds. The basic parameter settings are listed in Table 2.

We compare the proposed two algorithms, e.g., *cli-max greedy* and *uti-positive guarantee* with the theoretical lower-bound of the *cli-max greedy* algorithm and a benchmarked algorithm, *client selection for federated learning*

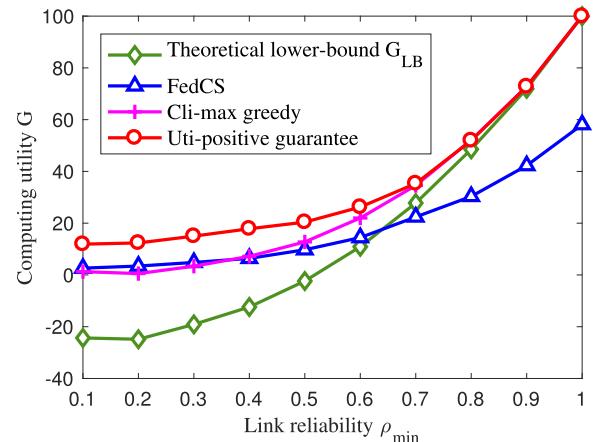


FIGURE 2. Computing utility with respect to link reliability probability.

(FedCS) [26]. Based on the result of *Theorem 1*, assuming the expected link reliability (e.g., ρ_m for $m \in \mathbb{M}$) is given, then the theoretical lower-bound of the *cli-max greedy* algorithm is obtained by $\mathcal{G}_{LB} = \sum_{m \in \mathbb{M}} \rho_m^3 - A \sum_{m \in \mathbb{M}} \rho_m (1 - \rho_m) E_m^{\max} - B$. In FedCS [26], the mobile clients that consume the least local model learning delay are iteratively added to the federated learning set, until the maximum local training delay exceeds the delay deadline $D_m^{T,\max}$. We evaluate the performance of FedCS with the deadline of $D_m^{T,\max} = 10s$.

A. VARIOUS LINK RELIABILITY

This scenario observes the performance by varying the link reliability in the GMEC system. In special, we define ρ_{\min} and ρ_{\max} as the minimum and maximum link reliability. Thus, the range of link reliability is expressed by $[\rho_{\min}, \rho_{\max}]$. The link reliabilities of clients are uniformly picked from the range. We set $\rho_{\max} = 1.0$, and observe the performance by varying ρ_{\min} from 0.1 to 1.0.

As illustrated in Fig. 2, under all the investigated algorithms, including the theoretical analysis, the computing utility increases with the increasing link reliability. This is because, the link reliabilities at the *algorithm state diffusion* and *algorithm state uploading* substages increase with increasing ρ_{\min} . Thus, the probabilities of distributed selected clients successfully receiving the global algorithm state as well as uploading the local training results (e.g., local trained algorithm state) to the server for aggregation also increase, leading to the increasing computing utility.

The computing utility given by the *cli-max greedy* algorithm is always higher than that of theoretical lower bound under various link reliability, as shown in Fig. 2, which demonstrates the accuracy of *Theorem 1*.

When ρ_{\min} is small (e.g., $\rho_{\min} \leq 0.4$), FedCS and the *cli-max greedy* algorithm provide similar performance in terms of the computing utility. However, with the increasing ρ_{\min} , the *cli-max greedy* algorithm outperforms FedCS by given higher computing utility. This is because, under FedCS, the clients that with the local training delays exceed the delay deadline of 10s would be rejected to join the training. Thus,

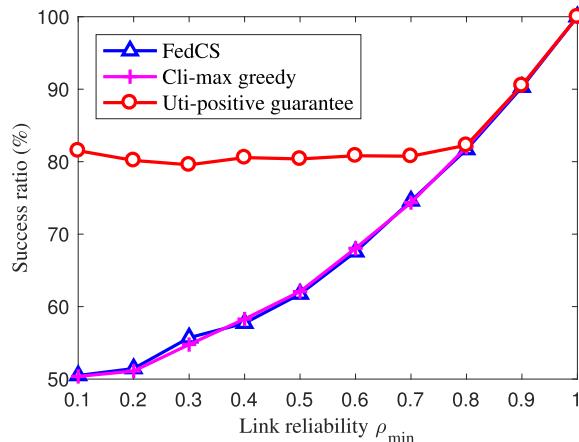


FIGURE 3. Local training success ratio with respect to link reliability probability.

although some clients have high probability to finish the training stage with high ρ_{\min} (e.g., $\rho_{\min} > 0.4$), they might be rejected by FedCS due to their long local training delays.

Comparing to both the cli-max greedy and FedCS algorithms, the performance improvement by the uti-positive guarantee algorithm is obvious by providing the highest computing utility under various ρ_{\min} , as illustrated in Fig. 2. In particular, when $\rho_{\min} = 1.0$, the computing utility of uti-positive guarantee is 1.72 times of that of FedCS, which improves 0.72. This is because, under the uti-positive guarantee algorithm, only the clients that would provide at least a positive computing utility would be selected to join the federated learning. Thus, the *success ratio*, which is defined as the ratio of the number of clients have successfully finished the training stage to the number of clients selected to join that round of training, given by the uti-positive guarantee algorithm is the highest comparing to the cli-max greedy and FedCS algorithms, as shown in Fig. 3. Note that, the uti-positive guarantee and cli-max greedy algorithms provide similar performance when ρ_{\min} is high (e.g., $\rho_{\min} > 0.8$). This is because, with the high link reliability, the probability of providing a positive per-client computing utility is high. Thus, the selected clients are similar under both the cli-max greedy and FedCS algorithms, which is illustrated in Fig. 3 that, the success ratio given by both are similar.

B. VARIOUS NUMBER OF MOBILE CLIENTS

In this scenario, we observe the performance by varying the number of mobile clients from 1 to 300. Generally, the theoretical number of clients that have selected but finally fail in the training stage increases with the increasing number of clients in the GMEC system. Since the failure client would introduce high cost (e.g., wasting consumed energy for invalid local training), it is unsurprising that the theoretical lower bound of the computing utility decreases quickly with the increasing number of clients, as shown in Fig. 4.

The simulated computing utility given by the cli-max greedy algorithm is greater than that of the theoretical lower

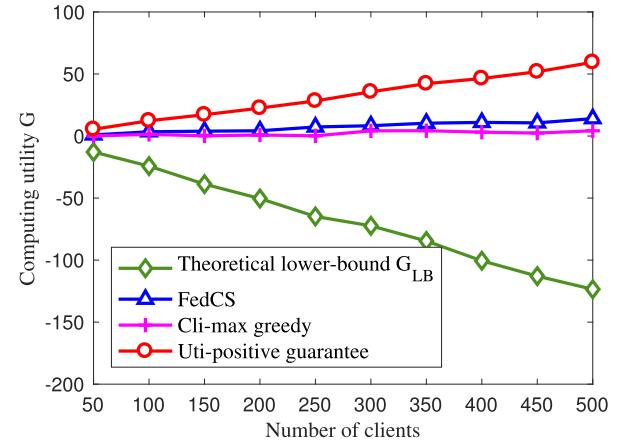


FIGURE 4. Utility with respect to client numbers.

bound, as shown in Fig. 4, which again illustrates the accuracy of *Theorem 1*. Note that, although the number of clients that have selected but finally fail in the training stage increases with the increasing number of clients, leading to a negative increasing in the computing utility, the number of clients that have selected and will finally finish the training stage also increases with the increasing number of clients, thus, the probability that the positive increment dominates the computing utility increases with the increasing number of clients. Thus, as shown in Fig. 4, under the simulated cli-max greedy algorithm, the computing utility increases slowly with the increasing number of clients.

The performance improvement of the uti-positive guarantee algorithm is again illustrated by providing the highest computing utility compared to the other investigated algorithms, as shown in Fig. 4. Particularly, when the number of clients reaches 300, the computing utility of uti-positive guarantee is 4 times of that of FedCS. Note that, due to the fact that the link reliability distribution is set to a constant (e.g., [0.1, 1.0]), the success ratios given by all the investigated schemes, including the FedCS, cli-max greedy and uti-positive guarantee, are stable under various number of clients, as illustrated in Fig. 5.

C. VARIOUS DATASET SIZES

We further evaluate the performance of the proposed algorithms under various dataset sizes. We set the number of mobile clients to $M = 100$. Other parameters are listed in Table 2. The sizes of datasets of various clients follow the uniform distribution among S_{\min} and S_{\max} , where $S_{\max} = 5,000$ samples. We observe the delay performance at various S_{\min} . All samples of all datasets are from MNIST [29].

As shown in Fig. 6, the computing utility given by all the investigated algorithms, including the FedCS, cli-max greedy and uti-positive guarantee algorithms, are greater than that of the theoretical lower bound. Interesting that, the computing utility given by FedCS becomes zero when S_{\min} is greater than 3,000 samples. This is because, when the size of local dataset becomes large, the local training delay would exceed

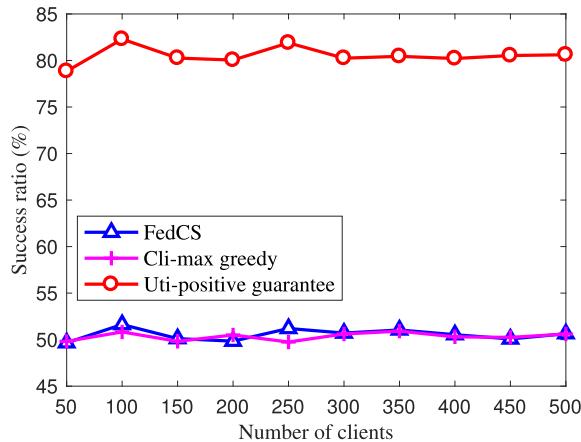


FIGURE 5. Local training success ratio with respect to client numbers.

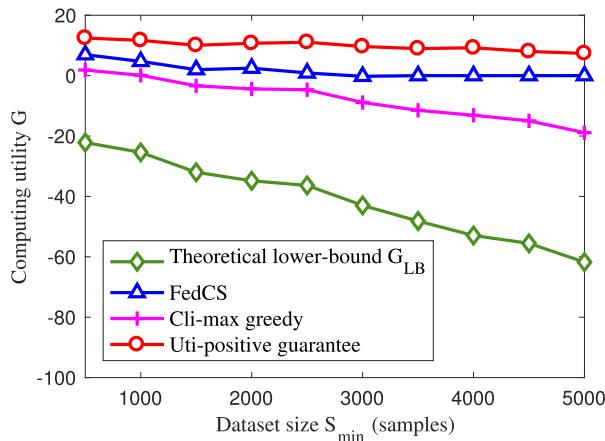


FIGURE 6. Computing utility with respect to dataset size.

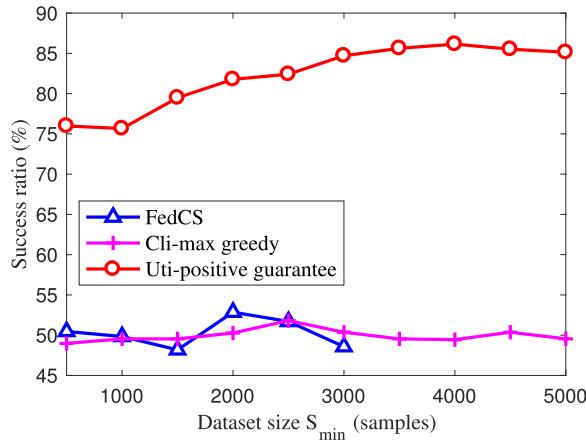


FIGURE 7. Local training success ratio with respect to dataset size.

the delay deadline of 10s, leading to all clients are rejected to join the training by FedCS, which is illustrated in Fig. 7 that, the success ratio given by FedCS becomes zero when $S_{\min} \geq 3,000$.

The computing utility given by the cli-max greedy algorithm decreases to a negative value when the dataset size increases. This is because, due to the high dynamic of the

link reliability (range of [0.1, 1.0]), there is a high probability of a selected client to fail in the training stage, including both the algorithm state diffusion and algorithm state uploading substages. The failure of both stages would lead to a negative increment in the computing utility. Particularly, when a client fails in the algorithm state uploading substage, it has wasted a lot of energy consumed for invalid local training, which would heavy the negative of the computing utility. Therefore, under a high dynamic environment, the larger size of local datasets, the higher cost to select a client to join federated learning.

Since the client that may provide negative computing utility would not be selected to join the training by the uti-positive guarantee algorithm, it is unsurprising that, the computing utility given by the uti-positive guarantee algorithm can be guaranteed to a positive value, as illustrated in Fig. 6. Note that, as shown in Fig. 7, the success ratio given by the uti-positive guarantee algorithm increases with the increasing S_{\min} . This is because, under the uti-positive guarantee algorithm, the client that has the large size of dataset but has low reliability would be rejected, thus the selected clients would have high probability to finish that round of training.

The simulation results in Figs. 2-7 demonstrate that, the computing utility given by the simulated cli-max greedy algorithm is greater than that of the theoretical lower bound under various link reliability, various number of clients and various sizes of local datasets, which illustrates the accuracy of *Theorem 1*. The computing utility given by the uti-positive guarantee algorithm outperforms that by the cli-max greedy algorithm under various link reliability, various number of clients and various sizes of local datasets, which demonstrates the performance improvement of the uti-positive guarantee algorithm over the cli-max greedy algorithm. The efficiency of the proposed algorithms, including the cli-max greedy and the uti-positive guarantee algorithms have been illustrated by comparing with the existing FedCS algorithm.

D. ENERGY CONSUMPTION AND LEARNING DELAY

Finally, we investigate the efficiency of the proposed *cli-max greedy* and *uti-positive guarantee* algorithms for energy saving and learning delay reduction by investigating the performance metrics in terms of wasted energy as well as the learning delay in comparison with the FedCS algorithm [26] under the scenario of various link reliability in Section V-A.

The wasted energy of a learning process is defined by

$$E = E(i)N_{\text{opt}},$$

where $E(i)$ follows (6), N_{opt} is the number of rounds required to repeat until achieving the model accuracy, which could be approximated by $N_{\text{opt}} = \alpha/\mathbb{E}[M_{\text{succ}}(i)]$ according to [30], where α is a factor parameter related to training accuracy.

Similarly, the learning delay of a learning process is defined by

$$D = D(i)N_{\text{opt}}.$$

TABLE 3. Comparison of wasted energy and learning delay.

Link reliability (ρ_{\min})	Wasted energy (J)			Learning delay ($\times 10^3$ s)		
	FedCS	Cli-max greedy	Uti-positive guarantee	FedCS	Cli-max greedy	Uti-positive guarantee
0.1	8.07	9.69	3.23	12.57	7.51	12.16
0.2	7.93	9.44	2.94	12.01	6.79	10.92
0.3	8.15	9.05	3.12	10.60	5.99	9.62
0.4	7.25	8.50	3.06	9.675	5.36	8.53
0.5	5.98	7.22	3.22	7.81	4.51	6.98
0.6	5.08	6.08	3.17	6.84	3.93	5.69
0.7	3.97	4.54	3.1	6.12	3.38	4.21
0.8	2.48	2.95	2.81	4.98	2.86	2.93
0.9	1.29	1.48	1.48	4.39	2.46	2.46
1.0	0	0	0	1.91	1.38	1.38

The simulation results are summarized in Table 3. As shown in Table 3, the cli-max greedy algorithm wastes the largest energy in comparison with FedCS and uti-positive guarantee under various link reliability (excepted $\rho_{\min} = 1$). This is because, under cli-max greedy, all the clients whose links are reliable during the client selection stage would be chosen to join the training, thus the failure ratio during the subsequent training and algorithm state aggregation stages would be the highest in comparison with both of FedCS and uti-positive guarantee. Therefore, it is unsurprising that, under cli-max greedy, the wasted energy of the local training that has no contribution to the global algorithm state update would be the highest.

The uti-positive guarantee algorithm outperforms the other two algorithms by providing the lowest wasted energy under various link reliabilities, as shown in Table 3. Particularly, when the link reliability is low, e.g., $\rho_{\min} < 0.7$, the wasted energy of uti-positive guarantee is one-third of that of cli-max greedy and a quarter of that of FedCS.

The learning delay reduction of the proposals are illustrated in Table 3, where the learning delays given by both cli-max greedy and uti-positive guarantee are lower than that of FedCS under various link reliabilities. In particular, when the link reliability is high, e.g., $\rho_{\min} \geq 0.8$, the learning delays of both cli-max greedy and uti-positive guarantee are one sixth of that of FedCS. The cli-max greedy provides the lowest learning delay. In particular, the learning delay of cli-max greedy is less than one sixth of that of FedCS when $\rho_{\min} \leq 0.9$. This is because, under cli-max greedy, the number of clients successfully finish a round of training is the highest. Then, according to [26], [30], the more number of clients joining a training, the fast to converge. Therefore, the number of rounds given by cli-max greedy is the lowest. Accordingly, the corresponding learning delay is the lowest in comparison with FedCS and uti-positive guarantee.

VI. CONCLUSION

This paper has proposed a dynamic federated learning scheme in a high dynamic power grid mobile edge computing environment. In special, a delay deadline constrained federated learning framework has been proposed to support machine learning based power grid applications. Then,

a dynamic client selection problem for computing utility maximization in the proposed federated learning framework, considering the link reliability in various communication stages of training rounds, has been formulated. An online greedy algorithm, e.g., *cli-max greedy*, was proposed to address the problem. Theoretical lower bound of the algorithm was also given. After theoretically analyzing the per-client computing utility, we improved the performance by the uti-positive guarantee algorithm. The simulation results have shown that, the uti-positive guarantee algorithm outperforms FedCS and cli-max greedy by providing high computing utility under various link reliability, various number of clients and various sizes of local datasets. The cli-max greedy algorithm can significantly reduce the learning delay to one sixth of that of FedCS in tradeoff wasting more energy for invalid local training. The uti-positive guarantee algorithm outperforms FedCS by providing shorter learning delay and lower wasted energy.

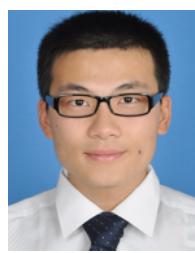
REFERENCES

- [1] A. Gomez-Exposito, A. J. Conejo, and C. Canizares, *Electric Energy Systems: Analysis and Operation*. Boca Raton, FL, USA: CRC Press, 2018.
- [2] J. C. Olivares-Rojas, E. Reyes-Archundia, J. A. Gutiérrez-Gneccchi, J. W. González-Murueta, and J. Cerdá-Jacobo, “A multi-tier architecture for data analytics in smart metering systems,” *Simul. Model. Pract. Theory*, vol. 102, Jul. 2020, Art. no. 102024.
- [3] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, “All one needs to know about fog computing and related edge computing paradigms: A complete survey,” *J. Syst. Archit.*, vol. 98, pp. 289–330, Sep. 2019.
- [4] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: Vision and challenges,” *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.
- [5] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the Internet of Things,” in *Proc. 1st Ed. MCC Workshop Mobile Cloud Comput. (MCC)*, New York, NY, USA, 2012, pp. 13–16.
- [6] M. Guo, L. Li, and Q. Guan, “Energy-efficient and delay-guaranteed workload allocation in IoT-edge-cloud computing systems,” *IEEE Access*, vol. 7, pp. 78685–78697, 2019.
- [7] H. B. McMahan, E. Moore, D. Ramage, and B. A. Y. Arcas, “Federated learning of deep networks using model averaging,” 2016, *arXiv:1602.05629*. [Online]. Available: <https://arxiv.org/abs/1602.05629>
- [8] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning: Concept and applications,” Feb. 2019, pp. 1–19, *arXiv:1902.04885*. [Online]. Available: <https://arxiv.org/abs/1902.04885>
- [9] M. Aazam, S. Zeadally, and K. A. Harras, “Deploying fog computing in industrial Internet of Things and industry 4.0,” *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4674–4682, Oct. 2018.

- [10] A. S. González and C. C. Pastor, "Edge computing node placement in 5G networks: A latency and reliability constrained framework," in *Proc. 6th IEEE Int. Conf. Cyber Secur. Cloud Comput. (CSCloud), 5th IEEE Int. Conf. Edge Comput. Scalable Cloud (EdgeCom)*, Jun. 2019, pp. 183–189.
- [11] A. F. R. Trajano, A. A. M. de Sousa, E. B. Rodrigues, J. N. de Souza, A. de Castro Callado, and E. F. Coutinho, "Leveraging mobile edge computing on smart grids using LTE cellular networks," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jun. 2019, pp. 1–7.
- [12] T. Sirojan, T. Phung, and E. Ambikairajah, "Intelligent edge analytics for load identification in smart meters," in *Proc. IEEE Innov. Smart Grid Technol.—Asia (ISGT-Asia)*, Dec. 2017, pp. 1–5.
- [13] T. Sirojan, S. Lu, B. T. Phung, and E. Ambikairajah, "Embedded edge computing for real-time smart meter data analytics," in *Proc. Int. Conf. Smart Energy Syst. Technol. (SEST)*, Sep. 2019, pp. 1–5.
- [14] Y. Huang, Y. Lu, F. Wang, X. Fan, J. Liu, and V. C. M. Leung, "An edge computing framework for real-time monitoring in smart grid," in *Proc. IEEE Int. Conf. Ind. Internet (ICII)*, Oct. 2018, pp. 99–108.
- [15] J. Yao, Z. Li, Y. Li, J. Bai, J. Wang, and P. Lin, "Cost-efficient tasks scheduling for smart grid communication network with edge computing system," in *Proc. 15th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2019, pp. 272–277.
- [16] L. Xiao, X. Lu, T. Xu, X. Wan, W. Ji, and Y. Zhang, "Reinforcement learning-based mobile offloading for edge computing against jamming and interference," *IEEE Trans. Commun.*, vol. 68, no. 10, pp. 6114–6126, Oct. 2020.
- [17] M. Li, L. Rui, X. Qiu, S. Guo, and X. Yu, "Design of a service caching and task offloading mechanism in smart grid edge network," in *Proc. 15th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2019, pp. 249–254.
- [18] R. El-Awadi, A. Fernández-Vilas, and R. P. D. Redondo, "Fog computing solution for distributed anomaly detection in smart grids," in *Proc. Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, Oct. 2019, pp. 348–353.
- [19] N. Kumar, S. Zeadally, and J. J. P. C. Rodrigues, "Vehicular delay-tolerant networks for smart grid data management using mobile edge computing," *IEEE Commun. Mag.*, vol. 54, no. 10, pp. 60–66, Oct. 2016.
- [20] T. Islam and M. M. A. Hashem, "A big data management system for providing real time services using fog infrastructure," in *Proc. IEEE Symp. Comput. Appl. Ind. Electron. (ISCAIE)*, Apr. 2018, pp. 85–89.
- [21] S. Zahoor, N. Javaid, A. Khan, B. Ruqia, F. J. Muhammad, and M. Zahid, "A cloud-fog-based smart grid model for efficient resource utilization," in *Proc. 14th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2018, pp. 1154–1160.
- [22] Y. Liu, C. Yang, L. Jiang, S. Xie, and Y. Zhang, "Intelligent edge computing for IoT-based energy management in smart cities," *IEEE Netw.*, vol. 33, no. 2, pp. 111–117, Mar. 2019.
- [23] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A survey on distributed machine learning," *ACM Comput. Surv.*, vol. 53, no. 2, pp. 1–33, Mar. 2020.
- [24] A. Taik and S. Cherkaoui, "Electrical load forecasting using edge computing and federated learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–6.
- [25] Y. M. Saputra, D. T. Hoang, D. N. Nguyen, E. Dutkiewicz, M. D. Mueck, and S. Srikantheswara, "Energy demand prediction with federated learning for electric vehicle networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.
- [26] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7.
- [27] T. Mikolov, S. Kombrink, L. Burget, J. Černocká, and S. Khudanpur, "Extensions of recurrent neural network language model," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2011, pp. 5528–5531.
- [28] K. Gregor, I. Danihelka, A. Graves, and D. Wierstra, "DRAW: A recurrent neural network for image generation," Feb. 2015, pp. 1–10, *arXiv:1502.04623*. [Online]. Available: <https://arxiv.org/abs/1502.04623>
- [29] TensorFlow. *MNIST*. Accessed: Jun. 1, 2020. [Online]. Available: <https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz>
- [30] M. Guo, M. Mukherjee, G. Liang, and J. Zhang, "Computation offloading for machine learning in industrial environments," in *Proc. IECON 46th Annu. Conf. IEEE Ind. Electron. Soc.*, Oct. 2020, pp. 4465–4470.



SHAOLEI ZHAI received the master's degree in control engineering and control theory from North China Electric Power University, China, in 2009. He currently works with the Electric Power Research Institute, Yunnan Power Grid Company Ltd., China. His research interests include electrical engineering, power system automation, high-voltage measurement technology, and so on. As a Senior Engineer, he has won dozens of science and technology progress awards.



XIN JIN received the master's degree from North China Electric Power University (NCEPU), Baoding, China, in 2011. He currently works with China Southern Power Grid Electric Power Research Institute Company Ltd., China. He is also a Senior Engineer with the China Southern Power Grid Research Institute. He also works as the Director of the Intelligent Measurement and New Technology Laboratory. His main research interests include power line carrier communication and wireless communication technology for smart grid.



LING WEI received the master's degree in control engineering and control theory from the Kunming University of Science and Technology, in 2014. As a Visiting Scholar of China and USA, she went to USA for exchange, from 2012 to 2013. She currently works with the Electric Power Research Institute, Yunnan Power Grid Company Ltd., China. Her research interests include power system automation and energy metering.



HONGXUAN LUO received the master's degree in electric engineering from the South China University of Technology, China, in 2017. He currently works with China Southern Power Grid Electric Power Research Institute Company Ltd., China. His research interests include smart meter terminal and power demand side management.



MIN CAO received the Bachelor of Engineering degree in industrial automation instrument from the Kunming University of Science and Technology, in 1982. He currently works with the Electric Power Research Institute, Yunnan Power Grid Company Ltd., China. He is also a Senior Technical Expert of China Southern Power Grid. He is also a Professor of engineering. His research interests include electrical engineering, power system automation, high-voltage measurement technology, and so on. He has received 30 provincial and ministerial awards.