Analyzing Optimization Trade-Off Between Energy and Model Convergence in Federated Learning

Nilo Marchiori Louzada(✉), Renato Elias Nunes de Moraes, Rodolfo S. Villaça, Vinícius F. S. Mota, and Maria Claudia Silva Boeres

Universidade Federal do Espírito Santo, Vitória, ES, Brazil
nilo.louzada@edu.ufes.br, {renato.moraes,rodolfo.villaca,vinicius.f.mota, maria.boeres}@ufes.br

Abstract. Federated Learning (FL) has established itself as a promising solution for training artificial intelligence models on edge devices while preserving data privacy. However, the hardware heterogeneity and energy constraints of IoT devices pose severe challenges to the sustainability of this paradigm. This work addresses energy efficiency optimization in FL networks by formulating the problem through the joint control of training time and device CPU frequency. Unlike purely theoretical approaches, the proposed methodology employs Lagrangian Relaxation and Karush-Kuhn-Tucker (KKT) conditions to determine the optimal frequency, validating results through realistic network emulation using a container-based tool. Experiments conducted with the MNIST dataset in mid-iid and non-iid data distribution scenarios demonstrate that strict time limit control, although reducing consumption per round, significantly impairs the global model's convergence. In contrast, the proposed optimization via frequency adjustment proved capable of reducing total energy consumption without degrading final accuracy. Additionally, combining frequency optimization with client selection strategies yields further reductions in accumulated energy consumption.

Keywords: Federated Learning · Lagrangian Relaxation · KKT · Convex Optimization

1 Introduction

The accelerated proliferation of smart edge devices, driven by the Internet of Things (IoT), has generated an unprecedented volume of data that challenges centralized processing architectures [8]. Although training Artificial Intelligence (AI) models on these data is highly valuable, transferring it to the cloud imposes critical latency, bandwidth, and privacy barriers [14]. Federated Learning (FL) has emerged as the solution to this scenario, enabling the collaborative training of global models while keeping the data residing on local devices [15,20].

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026, LNCS 16767, pp. 269–286, 2027.
https://doi.org/10.1007/978-3-032-30491-9_18

However, the practical implementation of FL in wireless networks faces the inherent heterogeneity of the edge environment. Devices vary in processing capacity and, crucially, in energy constraints [19,21]. Energy efficiency is therefore a primary requirement for the sustainability of FL. Although reducing CPU frequency is a common strategy to save energy, it directly impacts computing time and, consequently, the total latency of training rounds.

The optimization problem can be defined as, given a federated learning network, determining the optimal values for CPU frequency and training operation time, maximizing accuracy while minimizing energy consumption, considering the heterogeneity of the devices and their energy constraints. In this context, the objective of this work is to analyze the real impacts on the global model convergence when applying energy optimization based on CPU frequency control.

The literature has established rigorous theoretical foundations for this challenge. The authors in [19] formulated the trade-off between learning time and energy consumption through the FEDL (Federated Learning over wireless networks problem), whose solution requires the joint control of computation (via CPU frequency) and communication latencies. Extending these concepts, [11] introduced the DCM (Dataset and Computation Management) framework, which considers the dataset fraction as a control variable.

Additional complexity is given by the network topology and the nature of the data. Indeed, [14] and [20] demonstrated that edge hierarchy and adaptive participant selection are vital for minimizing costs. In network scenarios formed by low-energy devices, [2] highlighted the need for routing topology awareness to avoid the depletion of critical nodes. Moreover, the non-iid distribution of data imposes extra challenges to convergence, requiring adaptive learning rate strategies [8].

Despite theoretical advancements, the literature still presents a methodological gap in the use of CPU frequency as a decision variable in optimization models, which frequently rely on purely numerical solutions [11,14,19,21]. This approach provides limited insights into network behavior in a realistic emulation environment. To narrow the gap between theoretical optimization and practical assessment, this work provides the following contributions:

- An experimental characterization of the convergence impact of CPU frequency based energy optimization in FL, validated through realistic network emulation;
- An empirical demonstration that frequency control reduces total energy consumption by up to 30% without accuracy degradation under both mid-iid and non-iid data distributions.

The remainder of this article is organized as follows. Section 3 introduces the mathematical formalism for abstracting federated learning networks. Next, Sect. 4 discusses the optimization problem associated with processing time and energy consumption of client devices. The employed network emulation methodology, as well as the presentation and analysis of the obtained results, are

described in Sect. 5. Finally, Sect. 6 concludes this paper and provides suggestions for future work.

2 Related Work

Resource efficiency and client selection in Federated Learning (FL) have attracted significant attention from the scientific community, which can be categorized into strategies focused on resource allocation, multi-criteria selection, and system-level validations.

Resource Allocation and Energy Efficiency: Several studies address the optimization of system parameters to enable FL. Tran *et al.* [19] modeled FL over wireless networks as a convex optimization problem, proposing an algorithm that minimizes both training time and device energy consumption. Similarly, Zhou *et al.* [21] investigated multi-objective optimization in bandwidth-limited systems. Despite their strong mathematical foundations, such works often validate their approaches primarily through numerical simulations.

Client Selection in Heterogeneous Networks: Mitigating the effect of *stragglers* is commonly addressed through intelligent client selection. Nishio and Yonetani [16] introduced the FedCS protocol, which proactively selects clients based on their resource capacities to maximize the number of participants within a specific time window. More recent approaches, such as those by Tahir and Ali [18] and Ali *et al.* [1], expand this idea by applying heuristic and multi-criteria selection (*e.g.*, energy, data quality, and bandwidth).

The Gap in Realistic Emulation: Recent review papers, such as those by Badra *et al.* [10] and Daly *et al.* [6], emphasize that one of the greatest challenges in modern FL is the transition from mathematical theory to production environments. Emulation tools have begun to bridge this gap. Bastos *et al.* [3] and the developers of MininetFed [17] provided container-based platforms for FL evaluation.

Our work differs from the existing literature by closing the loop between rigorous mathematical optimization and realistic systemic emulation. Table 1 summarizes the main contributions of this paper in contrast to the current literature.

3 The Federated Learning System Formulation

This section details the modeling of the FL system. The focus of Sect. 3.1 establishes the mathematical notation and describes the distributed learning network architecture analyzed. Section 3.2 introduces the mathematical framework that approximates the physical energy consumption and estimates the devices' processing times.

Table 1. Expanded qualitative comparison between this work and the Federated Learning literature.

Work	Optimization Focus	Solution Method	Opt Model	Validation Strategy	Emulation Tool
Ali *et al.* [1]	Client selection	Multi-criteria Greedy		Network Emulation	EdgeCloudSim
Bastos *et al.* [3]	Energy in IoT (RPL)	Energy-Aware Selection		Network Emulation	MininetFed
Kim *et al.* [11]	Energy and Convergence Time	Convex Optimization	x	Numerical	FL-Pytorch
Luo *et al.* [14]	Cost and Latency	Convex Opt. + Heuristic	x	Numerical	Not mentioned
Nishio *et al.* [16]	Device selection	Heuristic	x	Numerical	Not mentioned
Tahir *et al.* [18]	Device selection	Greedy Heuristic	x	Numerical	Not mentioned
Tran *et al.* [19]	Energy and Time	Analytical Relaxation	x	Numerical	Not mentioned
Wei *et al.* [20]	Energy and Selection	Selection and Topology	x	Numerical	Not mentioned
Zhou *et al.* [21]	Energy and Selection	Multi-objective Opt.	x	Numerical	Not mentioned
This Work	**Energy and Time**	**Convex Optimization**	**x**	**Network Emulation**	**MininetFed**

3.1 Distributed Neural Network Training

Let R be a tree-structured network consisting of a set of $\mathbf{N}$ clients $CD_1, \ldots, CD_{\mathbf{N}}$, and a Centralized Aggregation Server (AS). The CDs train local neural networks, while AS performs the global aggregation of weights or gradients sent by the CDs. In this topology, each $CD_n, n = 1, \ldots, N$ is connected to the AS by single-path routes comprising one or more hops. Furthermore, communication between CDs and the AS nodes of R is performed via a wireless network. An example of this topology with N = 6 is depicted in Fig. 1.

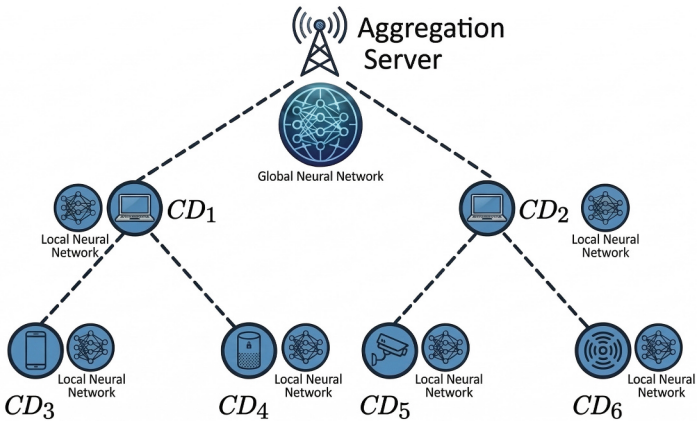


Fig. 1. Tree-structured topology of a FL.

In this topology, each client device $CD_n, n = 1, \ldots, N$ is characterized by its own set of input and output pairs $\mathbf{D_n}$ of size d_n, where each sample $\{x_n^i, y_n^i\}$, $x_n^i, y_n^i \in \mathbb{R}$, comprises a vector of s features and its expected output label. The parameter S_n specifies the size in bits of the CD_n dataset. Hardware constraints for each device are characterized by a CPU frequency $f_n \in [f_n^m in, f_n^m ax]$, a

chipset capacitance coefficient α_n, and a computational factor c_n (CPU cycles required to execute the model training task per sample). The parameters notation necessary for the problem formulation are summarized in Table 2.

The aforementioned scenario characterizes the learning problem as a typical classification problem. Therefore, each client device CD_n has the task of solving Problem 1, as defined in Eq. 1, considering the neural network weights w_n that minimize a local loss function $L(w)$ formulated as a sample loss function $l(x^i, y^i, w)$:

$$Prob.1: \quad w_n = \arg\min_{w_n} \; L(w_n), \text{ where } L(w_n) = \frac{1}{d_n} \sum_{i=1}^{d_n} l(x^i, y^i, w_n). \quad (1)$$

For instance, $l(x^i, y^i, w) = 1/2((x^i)^T w - y)^2$ [11]. Several methods can be employed to solve Problem 1; among them, SGD (stochastic gradient descent) is widely adopted [9]. AS, in turn, has the task of aggregating the distinct local weights w_n into a single object w. Furthermore, AS is responsible for determining the CPU frequency for each CD during the training phase. To achieve this, the AS leverages the constants characterizing each device, while maintaining no access to their local datasets.

Table 2. Notation Table - Federated Learning Model Variables

Symbol	Description
AS	Aggregation Server
CD	Client Device
N	Total number of client devices (CDs), $N \in \mathbb{N}$
n	Client device index, $n \in \{1, 2, \ldots, N\}$
s	Number of attributes (features), $s \in \mathbb{N}$
d_n	Number of samples of the n-th device, $d_n \in \mathbb{N}$
x_n^i	i-th vector of s input attributes of CD_n, $i = (1, 2, ..., d_n)$
y_n^i	i-th expected output label for x_n^i, $y_n^i \in \mathbb{R}$
$\mathbf{D_n}$	Set of attributes and labels of a CD_n: $\{x_n^i, y_n^i\}_{i=1}^{d_n}$
S_n	Size in bits of $\mathbf{D_n}$, $S_n \in \mathbb{R}$
f_n^{min}	Minimum CPU frequency of CD n, $f_n^{min} \in \mathbb{R}^+$
f_n^{max}	Maximum CPU frequency of CD n, $f_n^{max} \in \mathbb{R}^+$
α_n	Chipset capacitance of CD n, $\alpha_n \in \mathbb{R}^+$
c_n	CPU cycles per bit of training sample, $c_n \in \mathbb{R}^+$
w_n	Local model weights of CD_n, $w_n \in \mathbb{R}^K$, $K \in \mathbb{N}^+$
w	Global model weights (aggregated), $w \in \mathbb{R}^K$, $K \in \mathbb{N}^+$
$L(w_n)$	Local loss function of CD_n, $L(w_n) \in \mathbb{R}$
$l(x^i, y^i, w_n)$	Sample loss function, $l(\cdot) \in \mathbb{R}$
t	Training round index, $t \in \mathbb{N}$
w_n^t	Local weights of a CD_n in round t, $w_n^t \in \mathbb{R}^K$, $K \in \mathbb{N}^+$
w^t	Global model weights (aggregated), in round t. $w \in \mathbb{R}^K$, $K \in \mathbb{N}^+$
d_n^t	Number of samples of CD_n in round t, $d_n^t \in \mathbb{N}$
f_n^t	Allocated frequency for CD_n in round t, $f_n^t \in \mathbb{R}^+$
E_n^{cmp}	Computational energy consumption of CD_n, $E_n^{cmp} \in \mathbb{R}^+$
T_n^{cmp}	Computational time of CD_n, $T_n^{cmp} \in \mathbb{R}^+$

The way the AS aggregates the different weights can vary, as different methods are proposed in the literature. Perhaps the most classic algorithm, also adopted in this work, is *FedAvg* [15], where the local weights $w_n, n = 1, \ldots, \mathbf{N}$ are aggregated into w as defined in Eq. 2.

$$w = \frac{1}{d} \sum_{n=1}^{N} d_n \cdot w_n. \tag{2}$$

The FL scheme performed on the network is summarized in Algorithm 1, which introduces the index t associated with each training round (see Table 2).

Algorithm 1: Distributed Federated Learning Scheme

Input: T,$\{S_n^0, c_n, \alpha_n, f_n^{min}, f_n^{max}\}$
Output: Final global model w^T

1 The AS initializes and distributes w^0 to all $CD_n, n = 1, \ldots, N$;
2 Each CD_n sends parameters $\{c_n, \alpha_n, f_n^{min}, f_n^{max}\}$ to the AS;
3 **for** $t = 0$ **to** $T - 1$ **do**
  `// Coordination and Optimization Phase`
4 Each CD_n sends $\{S_n^t\}$ to the AS;
5 The server defines the devices that will participate in the training round;
6 The server executes the optimization model;
7 The server informs each CD_n of its operating frequency f_n^t;
  `// Local Training Phase at the CDs`
8 **for** *each client device* CD_n *in parallel* **do**
9 Each CD_n obtains w_n^{t+1} by solving Problem 1 for $L(w_n^t)$ via SGD;
10 Sends w_n^{t+1} to the AS;
11 **end for**
  `// Aggregation Phase at the Server`
12 The AS aggregates the received weights using Eq. 2:;
13 $w^{t+1} \leftarrow \frac{1}{d} \sum_{n=1}^{N} d_n \cdot w_n^{t+1}$;
14 The AS distributes the new global model w^{t+1} to the devices;
15 **end for**

- **Inputs and Outputs:** The algorithm receives essential configuration parameters, such as the total number of rounds T, and the physical parameters and hardware constraints for each CD_n ($f_{min}, f_{max}, c_n, \alpha_n, S_n$). The output is the optimized global model w^T.
- **Lines 1–2:**: The AS initializes the system by broadcasting the global model weights w^0 (an array initialized with arbitrary values without loss of generality) to all CD_n. In response, each client transmits its fixed hardware parameters to the server.
- **Line 3:** The main federated training loop begins, iterating from round $t = 0$ to $t = T - 1$.

- **Line 4:** Each device informs the server of the amount of data (S_n^t) it holds for that specific round. This step is necessary because the data volume at a client may vary as new data is acquired.
- **Line 5:** The server defines which clients will participate in the current training round. This is because certain optimization strategies select only a subset of clients to save energy.
- **Lines 6–7:** The server acts as a coordinator, executing the mathematical optimization model to determine the optimal CPU frequency for the selected clients. Subsequently, the server informs each CD_n of its specific operating frequency f_n^t.
- **Lines 8–11 (Local Training Phase):** A parallel loop begins where each selected client adjusts its CPU and computes the new model weights (w_n^{t+1}) by solving its local optimization problem (Prob. 1) using the SGD method. After the computation, the updated weights are sent back to the AS server.
- **Lines 12–14 (Aggregation Phase):** Upon receiving the local updates, the server aggregates them using Eq. 2 to form the new global model w^{t+1}. This updated model is then distributed to the devices, preparing the network for the next round.

3.2 CPU Processing Time and Energy Consumption of Client Devices

Based on the distributed training scheme presented in Sect. 3.1, this section establishes the mathematical formulation to quantify the CPU processing time and energy consumption for each CD. This formulation follows the several works in the literature, such as [11,19], and [20], the energy consumption E_n^{cmp} and the training time per round T_n^{cmp} for CD_n are presented in Eqs. 3 and 4:

$$E_n^{cmp} = \frac{\alpha_n}{2} c_n S_n f_n^2, \tag{3}$$

$$T_n^{cmp} = \frac{c_n S_n}{f_n}. \tag{4}$$

The Eqs. 3 and 4 are derived by adapting theoretical consumption and processing models. The variables used were previously defined in Sect. 3.1 and described in Table 2. Following [5,7], these equations were simplified for an ideal scenario, were the energy consumption and computational effort of the system itself are neglected, as well as energy losses through heating and leakage currents.

4 Optimization Problem Associated with Processing Time and Energy Consumption of Client Devices

Concerning the mathematical tools introduced in the previous sections, Sect. 4.1 presents the optimization problem. It incorporates an objective function designed

to balance energy consumption and processing time, which is subject to the problem constraints. Section 4.2 describes the methodology used to achieve the optimal solution, while Sect. 4.3 outlines the characteristics of the network instance constructed, which follow the structure of the example presented in Fig. 1.

4.1 The Mathematical Formulation

In this section, we present the formulation of the optimization problem of interest, described in Prob. 2, comprising Eqs. 5a to 5c:

$$Prob.2: \quad \min_{f_n, T^{cmp}} \quad \kappa T^{cmp} + \sum_{n \in \mathbf{N}} E_n^{cmp}, \tag{5a}$$

$$\text{s.t:} \quad T_n^{cmp} \leq T^{cmp} \quad \forall n \in \mathbf{N}, \tag{5b}$$

$$f_n^{min} \leq f_n \leq f_n^{max}, \quad \forall n \in \mathbf{N} \tag{5c}$$

Prob. 2 is a multi-objective optimization problem that seeks to balance two competing resources with its objective function (5a): the CPU processing time T^{cmp} and the energy consumption E_n^{cmp}. Since Prob. 2 is a bi-objective optimization problem, its objective function is compound by two conflicting criteria. The priority assigned to each cost (either energy consumption or round latency) is a subjective choice. The trade-off is managed by the parameter κ. When coupled with T^{cmp}, a is sufficiently high, κ renders the second term of the objective function negligible compared to the first. Conversely, as κ approaches 0, the first term becomes irrelevant. The parameter κ is also used to normalize these values, once they may have different orders of magnitude.

Constraint 5b imposes that the decision variable T^{cmp} is large enough so that all clients finish the task before the round ends, while constraint 5c ensures that f_n respects the computational power limits of each client device CD_n.

This formulation is a simplified version of the one proposed by [19]. In its original form, the computational resource optimization is a subproblem addressed jointly with the communication resource subproblem. We aim to investigate the feasibility and consequences of optimizing computational resources exclusively; consequently, the communication subproblem is omitted.

4.2 Obtaining the Optimal Solution

According to [19], the convex optimization problem presented in Sect. 4.1 can be solved optimally by employing a Lagrangian Relaxation [4], jointly with the Karush-Kuhn-Tucker (KKT) conditions [13]. The authors propose partitioning the set of $\mathbf{N}$ client devices into three subsets, N_1, N_2, N_3 ($N_i \cap N_j = \emptyset$ and $N_i \cup N_j = \mathbf{N}$, $i \neq j$) and prove mathematically that the optimal solution for the problem formulated in 4.1 is given by Eqs. 6 and 7a to 7d.

$$f_n^* = \begin{cases} f_n^{max}, & \forall n \in \mathcal{N}_1, \\ f_n^{min}, & \forall n \in \mathcal{N}_2, \\ \frac{c_n S_n}{T_{cmp}^*}, & \forall n \in \mathcal{N}_3, \end{cases} \tag{6}$$

$$T^*_{cmp} = \max\{T_{\mathcal{N}_1}, T_{\mathcal{N}_2}, T_{\mathcal{N}_3}\}, \tag{7a}$$

$$T_{N_1} = \max_{n \in N_1} \frac{c_n S_n}{f_n^{max}}, \tag{7b}$$

$$T_{N_2} = \max_{n \in N_2} \frac{c_n S_n}{f_n^{min}}, \tag{7c}$$

$$T_{N_3} = \left(\frac{\sum_{n \in \mathcal{N}_\ni} \alpha_n (c_n S_n)^3}{\kappa} \right)^{1/3}. \tag{7d}$$

To determine this partitioning, the procedure proposed in [19] is reproduced in Algorithm 2.

Algorithm 2: Determining partitions $\mathcal{N}_1$, $\mathcal{N}_2$ and $\mathcal{N}_3$ [19]

Input: $\mathcal{N}_1, \mathcal{N}_2 = \emptyset$, $\mathcal{N}_3 = \{CD_n, n = 1, 2, \ldots, d_n\}$
Output: $\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3$

1 Sort $\mathcal{N}_3$ by the value of $\frac{c_n S_n}{f_n^{min}}$ in ascending order;
2 $T_{\mathcal{N}_3} \leftarrow$ Value obtained from Equation (7d);
3 **for** $i \in \mathcal{N}_3$ **do**
4 **if** $\max_{n \in \mathcal{N}} \frac{c_n S_n}{f_n^{min}} \geq T_{\mathcal{N}_3}$ ***and*** $\mathcal{N}_1 == \emptyset$ **then**
5 $\mathcal{N}_1 = \mathcal{N}_1 \cup \{n\}$
6 $\mathcal{N}_3 = \mathcal{N}_3 \setminus \mathcal{N}_1$
7 Update $T_{\mathcal{N}_3}$
8 **end if**
9 **if** $\frac{c_i D_i}{f_i^{min}} \leq T_{\mathcal{N}_3}$ **then**
10 $\mathcal{N}_2 = \mathcal{N}_2 \cup \{i\}$
11 $\mathcal{N}_3 = \mathcal{N}_3 \setminus \{i\}$
12 Update $T_{\mathcal{N}_3}$
13 **end if**
14 **end for**
15 **return** $\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3$

- **Inputs and Outputs:** The algorithm begins by assuming that all clients will have their frequencies actively scaled, placing them all in set $\mathcal{N}_3$, while $\mathcal{N}_1$ and $\mathcal{N}_2$ start empty. The algorithm returns these three sets correctly populated.
- **Line 1:** The devices in set $\mathcal{N}_3$ are sorted in ascending order. The metric used for sorting is the time each device would take to process its data if it operated at its minimum possible CPU frequency ($\frac{c_n S_n}{f_n^{min}}$).
- **Line 2:** The target processing time for set $\mathcal{N}_3$ ($T_{\mathcal{N}_3}$) is initially calculated based on Eq. 7d.
- **Line 3:** A loop begins, iterating through each device i currently within set $\mathcal{N}_3$.

- **Lines 4–8 (Bottleneck Handling):** The algorithm evaluates whether a "bottleneck" device exists in the system. It checks if the maximum limiting processing time is greater than or equal to the current target time $T_{\mathcal{N}_3}$. If such a bottleneck is identified and set $\mathcal{N}_1$ is empty, this device is removed from $\mathcal{N}_3$ and included in $\mathcal{N}_1$. Since the group dynamics have changed, the target time $T_{\mathcal{N}_3}$ is immediately recalculated.
- **Lines 9–13 (Fast Client Handling):** The algorithm then seeks to identify highly efficient clients. It tests whether the iterated device i, even when operating at its minimum CPU frequency, would still complete its task in less time than the target $T_{\mathcal{N}_3}$. If this condition holds, the device is added to set $\mathcal{N}_2$ and removed from $\mathcal{N}_3$. Again, this removal requires $T_{\mathcal{N}_3}$ to be updated.
- **Line 15:** Once the loop concludes and all reallocations to handle bottlenecks and fast clients are performed, the algorithm returns the final partitions $\mathcal{N}_1$, $\mathcal{N}_2$, and $\mathcal{N}_3$ to the server.

4.3 Network Instance Description

The network instance used in the experiments follows a topology similar to the one shown in Fig. 1. It consists of $\mathbf{N} = 6$ client devices, in addition to the aggregation server (AS), organized in a binary tree topology with the AS as the root. Although the optimization methodology is scalable, the number of devices was limited due to computational resource constraints on the emulation host, without compromising the quality and reliability of the experimental results.

The configuration of each device and its parameters are presented in Table 3. The values for c_n were obtained in a preliminary experimental stage. The training process was conducted across a range of f_n values to produce an empirical characterization of T_n^{cmp}. Consequently, c_n was adjusted to align the empirical data with the theoretical predictions established in Eq. 4.

Table 3. Configuration of the client devices in the learning network

CD_n	S_n (MB)	c_n	f_n^{max} (GHz)	f_n^{min} (GHz)
CD_1	32.5	9	2.3	1.3
CD_2	54.1	5	2.8	1.3
CD_3	81.2	4	2.7	1.3
CD_4	43.3	8	2.1	1.3
CD_5	65.0	4	2.5	1.3
CD_6	21.6	11	2.1	1.3

On the other hand, since it depends on hardware characteristics, α_n was set to the same value for all devices as $\alpha_n = 2 \cdot 10^{-28}$, associated with the machine that emulated the network. This assumption does not affect the generality of the

results, as device heterogeneity is still guaranteed by the values of f_n^{min}, f_n^{max}, and c_n, given that the constant α_n is always multiplied by one of these factors. Other authors, such as [19] and [11], maintain this same assumption.

Each client device received samples from the MNIST dataset [12], a widely used benchmark in the literature. MNIST was selected deliberately: its well-understood convergence behavior allows the effects of frequency-based optimization to be isolated and attributed without confounding factors introduced by dataset complexity. This controlled setting facilitates reproducibility and direct comparison with prior emulation-based studies. Evaluating the approach on more complex datasets remains as future work.

A qualitative analysis of the optimal solution for Problem 2 can be conducted from Fig. 2, obtained by applying Algorithm 2 to the client configuration presented in Table 3. The graph shows the change in the optimal frequency value and also the processing time according to the increase in the value of κ. We will divide the solution analysis into different intervals of kappa:

- $\log(\kappa) \leq -0.35$: saving energy becomes so important that all devices execute at minimum frequency, taking the maximum possible time to finish the task; in this situation $\mathcal{N}_2 = \mathcal{N}$.
- $\log(\kappa) \cong 0.75$: T^{cmp} and f_n are not defined by boundary conditions, but rather by a value determined by Eqs. 7d and 6. In this situation, $\mathcal{N}_3 = \mathcal{N}$.
- $\log(\kappa) \geq 0.85$: In this region, a low T^{cmp} is chosen because this term has more weight in Eq. (5a). Here, the bottleneck device (CD_4) operates at maximum frequency, while the frequency of the other devices is adjusted to intermediate values ($\mathcal{N}_1 \neq \emptyset$).

The plots reinforce the role of κ in resource prioritization, as evidenced by the comparison between the two graphs in Fig. 2. When κ is adjusted to low values, the objective function prioritizes energy saving over latency, resulting in lower operating frequencies and higher processing times. Conversely, as κ increases, the priority shifts toward minimizing latency, leading to higher frequencies and increased energy consumption.

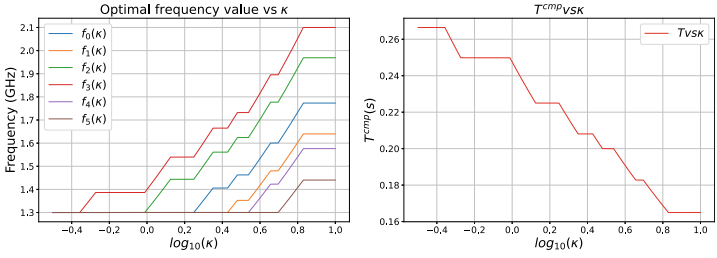


Fig. 2. Solution to Problem 2 for different values of κ.

5 Experiments and Results

Simulations were performed in virtual networks to validate the dependencies of energy and processing time discussed in the theoretical model. In particular, to evaluate whether saving resources generates impacts on the learning model's convergence and, if so, to discuss in which scenario this trade-off of efficiencies is advantageous.

To create the virtual network laboratories, MininetFed [17] was used. This tool allows the emulation of federated learning based on container networks managed by Docker. The emulation environment imposes inherent resource constraints, as each virtual device runs as an isolated Docker container sharing CPU and memory with the host machine. Scaling beyond six devices would compromise the reliability of timing and energy measurements - a known challenge in container-based FL emulation. Importantly, the optimization methodology itself is analytically scalable to arbitrary $\mathbf{N}$, as the analytical solution in Sect. 4.2 is determined by Algorithm 2 with a $\mathcal{O}(n \log n)$ complexity; the experimental scope is a constraint of the validation environment, not of the proposed approach. The differential of MininetFed is the ability to emulate networks sensitive to energy consumption, which is crucial for the objective of this work, as well as the possibility to define heterogeneous devices regarding computational power, amount of data, and other variables to which Problem 2 is sensitive.

The general network configuration (topology, configuration, and communication protocols) was inspired by [2], while changes to the server and clients were necessary to accommodate the proposed problem. Section 5.1 describes one of the modifications made to the library so that the energy consumption measurement was effective regarding CPU frequency changes. Next, Sect. 5.2 presents the training methodologies, directly linked to the specifications of Algorithm 1. Finally, the results are presented and discussed in Sect. 5.3.

5.1 Energy Consumption and Power

Although the energy consumption throughout a training round is estimated by Eq. 3 (Sect. 3.2), it is worth remembering that these are approximations. The real time of the training process can oscillate, just as the computer's clock does not remain constant [21].

Therefore, throughout the experimentation, energy consumption was measured through a function that calculates the power (J/s) consumed by the CPU, defined in Eq. 8 [7]:

$$P(f) = CV^2 f, \tag{8}$$

where C is a constant adjusted experimentally by varying f and measuring the actual CPU consumption of the experiment host. Thus, the instantaneous power $P(f, t)$ is measured by reading the frequency at predefined time intervals Δt, thus arriving at values close to reality. The energy spent in this interval is then given by $\Delta E(t) = P(f, t) \cdot \Delta t$. This is an adaptation of [7], considering only the CPU workload consumption, since this is the investigation target of this study.

Implementing the power function of Eq. (8) was necessary because the default function used by [3] only considers the CPU usage percentage and did not prove sensitive to CPU frequency changes.

5.2 Experiments

The network devices were configured following the instance from Table 3. Different training strategies were implemented, combining different forms of data distribution, client selection at each round, and resource consumption, as described below:

Regarding the Training Methodology

- **Standard:** in this configuration, no optimization is done, all devices participate in all rounds, with frequency control assigned to the operating system.
- **LeastEnergy:** Client selection proposed by [2], where only the set $\mathcal{M} \subseteq \mathcal{N}$ of devices that consumed less than the average consumption of the previous round participates in the current round (Acting on Line 5 of Algorithm 1). Frequency is also controlled by the kernel.
- **Opt and Opt&fixed T^{cmp}:** In both, the frequency control is defined by the solution to Problem 2, with namespace governor. However, "Opt&fixed T^{cmp}" is the only one where the training round stopping criterion is not a maximum number of epochs (defined as 10), but rather T^{cmp*}, the result of Equation (7a).
- **Opt&LeastEnergy:** The $\mathcal{M}$ devices selected by LeastEnergy are the input for Problem 2.

For the methodologies *Opt*, *Opt&T^{cmp} fixed*, and *Opt&LeastEnergy*, the parameter κ was set to $10^{3/4}$. This value was considered suitable for the experiments because, as shown in Fig. 2, it represents a solution regime where none of the devices operate at their maximum or minimum frequency limits.

Regarding Data Distribution

- **mid-iid:** The devices receive random samples from the MNIST dataset; however, all of them have examples from the 10 possible labels of this dataset, but in unbalanced quantities among them.
- **non-iid:** The data is unbalanced, and there may be devices that "do not know" certain image categories. In the experiments, each client received images with 4 categories individually chosen at random among the 10 possible ones. (Note: the non-iid graphs only include results for three methodology scenarios to facilitate visualization).

5.3 Results

Figure 3 illustrates that the Standard training mode is the least efficient in terms of both consumption per round and time. The *opt-fixed* T^{cmp} methodology presents the lowest absolute energy consumption at the end of the process. However, fixing the time per round directly impairs the model's quality, as evidenced in Fig. 4. This fixed time limit leads to a smaller total number of epochs, which justifies the shorter training time shown in Fig. 3c. Consequently, to achieve the same target accuracy, the total energy consumed would be higher.

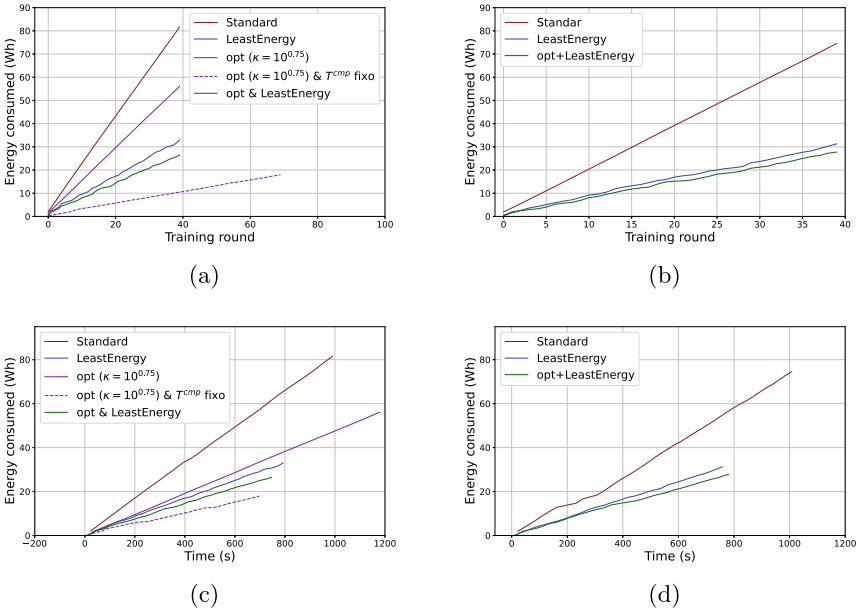
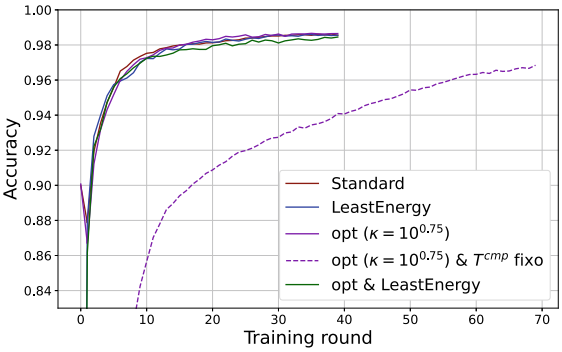


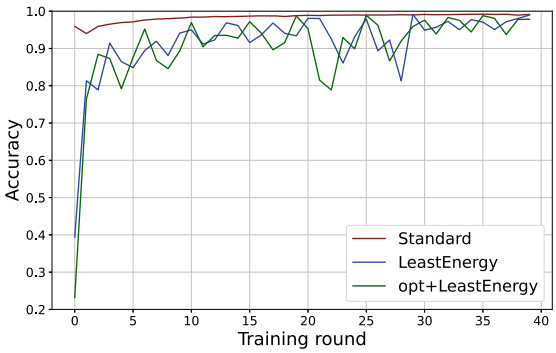
Fig. 3. Energy consumption behavior for different scenarios: (a) Energy vs Round (mid-iid), (b) Energy vs Round (non-iid), (c) Energy vs Time (mid-iid), (d) Energy vs Time (non-iid).

Conversely, comparing the pairs "Standard vs. Opt" and "LeastEnergy vs. Opt&Least Energy" demonstrates that energy optimization reduces total consumption. Furthermore, as shown in Fig. 4, it does not harm the model's performance. This result is expected, given that the number of epochs remained constant across these cases. The graphs in Figs. 3c and 3d also show that the training process time, for the same number of rounds, is not drastically altered by the optimization model under analysis, except for the fixed-time scenario.

Analyzing Fig. 5 in conjunction with the others brings an important reflection: reaching a target accuracy demands the same or less energy when employing the optimized methodology instead of the standard client selection approach. Specifically, the optimization approach achieves a reduction of 30% and 20%



(a)



(b)

Fig. 4. Accuracy evolution behavior for different scenarios: (a) Mean accuracy for mid-iid data, (b) Mean accuracy for non-iid data.

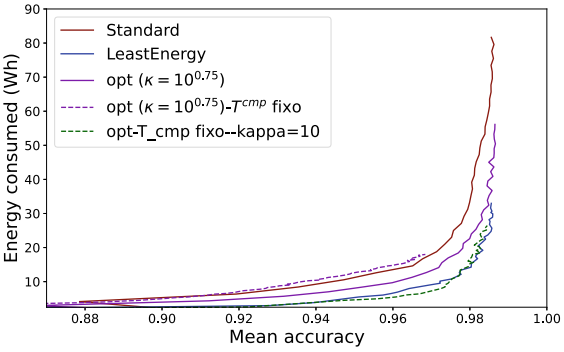


Fig. 5. Relationship between consumption and accuracy for mid-iid data.

in absolute accumulated energy consumption when compared to the standard selection of all clients and the LeastEnergy strategy, respectively.

6 Conclusion

The investigation conducted in this work allowed for the evaluation of the impact of resource optimization strategies in Federated Learning networks. The results demonstrate that, although strict control of processing time favors a reduction in energy consumption per round, this approach imposes a critical trade-off, as fixing a time limit reduces the total number of epochs, significantly impairing the quality and convergence of the final model.

On the other hand, the proposed frequency optimization algorithm, based on Lagrangian Relaxation and KKT Conditions, proved to be effective. It was evidenced that, for a previously defined client selection, controlling the computational effort via frequency adjustment is capable of optimizing energy consumption without leading to losses in model performance or accuracy. This effectiveness was experimentally validated for both partially unbalanced data distributions (mid-iid) and scenarios where certain clients lack specific categories (non-iid). Furthermore, the combination of the LeastEnergy selection methodology with the optimization algorithm enhanced the accumulated energy savings compared to the standard training mode.

As perspectives for future work, we suggest evolving the optimization model to include the number of local epochs as a decision variable, in addition to evaluating communication-related costs. Furthermore, we propose integrating client selection criteria directly into the frequency optimization model to enhance the trade-off between execution time and energy efficiency in low-power networks.

Another suggestion for future optimization involves incorporating suitable methods to measure the computational effort of hardware accelerators, such as GPUs, NPUs (Neural Processing Units), and TPUs (Tensor Processing Units). These architectures are becoming increasingly common in machine learning and neural network applications, even in edge computing scenarios. Additionally, scaling the experimental evaluation to larger device counts, potentially through distributed emulation hosts or hardware testbeds, remains as future work.

Acknowledgments. The authors acknowledge partial funding from the Coordination for the Improvement of Higher Education Personnel (CAPES) - Funding Code 001; the National Council for Scientific and Technological Development (CNPq) - Funding Codes 442344/2023-6, 444602/2024-0, and 407466/2025-8; the Espírito Santo Research and Innovation Foundation (FAPES) - Grants No. 2026-6NX7B, 2026-B74MN, 2024-9G1WP, 2023-RWXSZ; and Fapesp/CGI.br (2020/05182-3).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

Declaration of Generative AI and AI-Assisted Technologies in the Writing Process. During the preparation of this work the authors used Gemini/Google in

order to improve language and readability. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

References

1. Ali, N., Aloi, G., Gravina, R., Savaglio, C., Sodhro, A.H., Fortino, G.: FedHeur: multi-heuristic client selection for task offloading in federated learning. In: 2025 21st International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT), pp. 816–821 (2025). https://doi.org/10.1109/DCOSS-IoT65416.2025.00124
2. Bastos, J., Batista, J., Fontes, R., Cerqueira, E., Villaça, R., Mota, V.F.S.: A lightweight emulation framework for energy-aware federated learning. In: Proceedings of the ACM SIGCOMM 2025 Posters and Demos, ACM SIGCOMM Posters and Demos '25, pp. 130–131. Association for Computing Machinery, New York, NY, USA (2025). https://doi.org/10.1145/3744969.3748395
3. Bastos, J.J.S., Batista, J.P.C., dos Reis Fontes, R., Cerqueira, E., Villaça, R.S., Mota, V.F.S.: Otimizando energia no aprendizado federado em redes de baixa potência e com alta taxa de perda de pacotes. In: Anais do XLIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos. pp. 43–56. SBC, Porto Alegre, RS, Brasil (2025). https://doi.org/10.5753/sbrc.2025.5786, https://sol.sbc.org.br/index.php/sbrc/article/view/35121
4. Boyd, S.P., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
5. Burd, T.D., Brodersen, R.W.: Processor design for portable systems. J. VLSI Sig. Process. Syst. **13**(2–3), 203–221 (1996)
6. Daly, K., Eichner, H., Kairouz, P., McMahan, H.B., Ramage, D., Xu, Z.: Federated learning in practice: reflections and projections. In: 2024 IEEE 6th International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications (TPS-ISA), pp. 148–156 (2024). https://doi.org/10.1109/TPS-ISA62245.2024.00026
7. De Vogeleer, K., Memmi, G., Jouvelot, P., Coelho, F.: The energy/frequency convexity rule: modeling and experimental validation on mobile devices. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Waśniewski, J. (eds.) PPAM 2013. LNCS, vol. 8384, pp. 793–803. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55224-3_74
8. Efthymiadis, F., Karras, A., Karras, C., Sioutas, S.: Advanced optimization techniques for federated learning on non-IID data. Fut. Internet **16**(10) (2024). https://doi.org/10.3390/fi16100370, https://www.mdpi.com/1999-5903/16/10/370
9. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016). http://www.deeplearningbook.org
10. Guendouzi, B.S., Ouchani, S., Assaad, E.L., H., EL Zaher, M.: A systematic review of federated learning: challenges, aggregation methods, and development tools. J. Netw. Comput. Appl. **220**, 103714 (2023). https://doi.org/10.1016/j.jnca.2023.103714, https://www.sciencedirect.com/science/article/pii/S1084804523001339
11. Kim, J., Kim, D., Lee, J., Hwang, J.: A novel joint dataset and computation management scheme for energy-efficient federated learning in mobile edge computing. IEEE Wirel. Commun. Lett. **11**(5), 898–902 (2022). https://doi.org/10.1109/LWC.2022.3147236

12. LeCun, Y., Cortes, C., Burges, C.: MNIST handwritten digit database. ATT Labs [Online]. http://yann.lecun.com/exdb/mnist**2** (2010)
13. Li, M.: Generalized Lagrange multiplier method and KKT conditions with an application to distributed optimization. IEEE Trans. Circ. Syst. II Exp. Briefs **66**(2), 252–256 (2019). https://doi.org/10.1109/TCSII.2018.2842085
14. Luo, S., Chen, X., Wu, Q., Zhou, Z., Yu, S.: HFEL: joint edge association and resource allocation for cost-efficient hierarchical federated edge learning. IEEE Trans. Wirel. Commun. **19**(10), 6535–6548 (2020). https://doi.org/10.1109/TWC.2020.3003744
15. McMahan, B., Moore, E., Ramage, D., Hampson, S., Aguera y Arcas, B.: Communication-efficient learning of deep networks from decentralized data. In: Singh, A., Zhu, J. (eds.) Proceedings of the 20th International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 54, pp. 1273–1282. PMLR (2017). https://proceedings.mlr.press/v54/mcmahan17a.html
16. Nishio, T., Yonetani, R.: Client selection for federated learning with heterogeneous resources in mobile edge. In: 2019 IEEE International Conference on Communications (ICC), ICC 2019, pp. 1–7 (2019). https://doi.org/10.1109/ICC.2019.8761315
17. Sarmento, E.M., Bastos, J.J., Villaca, R.S., Comarela, G., Mota, V.F.: MininetFed: a tool for assessing client selection, aggregation, and security in federated learning. In: 2024 IEEE 10th World Forum on Internet of Things (WF-IoT), pp. 1–6. IEEE (2024)
18. Tahir, M., Ali, M.I.: Multi-criterion client selection for efficient federated learning. Proc. AAAI Symp. Ser. **3**(1), 318–322 (2024). https://doi.org/10.1609/aaaiss.v3i1.31227, https://ojs.aaai.org/index.php/AAAI-SS/article/view/31227
19. Tran, N.H., Bao, W., Zomaya, A., Nguyen, M.N.H., Hong, C.S.: Federated learning over wireless networks: optimization model design and analysis. In: IEEE Conference on Computer Communications, IEEE INFOCOM 2019, pp. 1387–1395 (2019). https://doi.org/10.1109/INFOCOM.2019.8737464
20. Wei, X., Ye, K., Shi, X., Xu, C.Z., Wang, Y.: Joint participant and learning topology selection for federated learning in edge clouds. IEEE Trans. Parallel Distrib. Syst. **35**(8), 1456–1468 (2024). https://doi.org/10.1109/TPDS.2024.3413751
21. Zhou, Y., Liu, X., Lei, L.: Multi-objective optimization for bandwidth-limited federated learning in wireless edge systems. IEEE Open J. Commun. Soc. **4**, 954–966 (2023). https://doi.org/10.1109/OJCOMS.2023.3266389