



HAL
open science

Optimal Time and Energy-Aware Client Selection Algorithms for Federated Learning on Heterogeneous Resources

Alan Lira Nunes, Cristina Boeres, Lúcia Maria de A. Drummond, Laércio Lima Pilla

► **To cite this version:**

Alan Lira Nunes, Cristina Boeres, Lúcia Maria de A. Drummond, Laércio Lima Pilla. Optimal Time and Energy-Aware Client Selection Algorithms for Federated Learning on Heterogeneous Resources. 2024. hal-04690494

HAL Id: hal-04690494





<https://hal.science/hal-04690494>

Preprint submitted on 6 Sep 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Optimal Time and Energy-Aware Client Selection Algorithms for Federated Learning on Heterogeneous Resources

Alan L. Nunes^{1,2} , Cristina Boeres¹ , Lúcia M. A. Drummond¹ , Laércio L. Pilla² 

¹Fluminense Federal University, Niterói, Brazil
alan_lira@id.uff.br, {boeres, lucia}@ic.uff.br

²University of Bordeaux, CNRS, Bordeaux INP, Inria, LaBRI, Talence, France
alan.lira-nunes@u-bordeaux.fr, laercio.pilla@inria.fr

Abstract—Federated Learning systems allow training machine learning models distributed across multiple clients, each one using private local data. Iteratively, the clients send their training contributions to a server, which performs a merge to produce an enhanced global model. Due to resource and data heterogeneity, client selection is crucial to optimize the system efficiency and improve the global model generalization. Selecting more clients is likely to increase the overall energy consumption, while a small number of clients may decline the performance of the trained model or require longer training time. We propose two time- and energy-aware client selection algorithms, MEC and ECMTC, which are proven regarding their optimality and evaluated against state-of-the-art algorithms on an extensive series of experiments in both simulation and HPC platform scenarios. The results indicate the benefits of jointly optimizing the time and energy consumption metrics using our proposals.

Index Terms—Federated learning, client selection, optimal schedule, makespan optimization, energy consumption

I. INTRODUCTION

The advent of Federated Learning (FL) systems has provided opportunities for improved data protection and performance for myriad industrial, scientific, and healthcare applications [9], [30]. FL acts as a distributed machine learning technique, where a centralized server chooses a subset of available clients to help with training [16]. Each client trains the same model with its local data (which is never shared) and sends the updated model’s weights to the server. The server combines all updates to the model, tests the new model (locally or again with the clients), and starts a new communication round until a stopping condition is achieved, for instance, expected accuracy or number of rounds.

The client selection for training plays a paramount role in the effectiveness of FL systems solution quality due to the heterogeneity of computing capacity and data [1], [3], [17], [20], [27]–[29], [32]. Selecting more clients is likely to increase the overall energy consumption. In contrast, a small number of clients may affect the accuracy of the trained model. Therefore, careful client selection reduces the training efforts, improves the model performance, and enhances fairness [9].

Our focus in this work lies in two dimensions of client selection: time and energy. The time taken by a training round is determined by the slowest participating device, known as the straggler [20], [25], [26]. Given the clients’ heterogeneity and the exponentially-increasing computing demands of machine learning models [4], this issue may only tend to increase in importance. Meanwhile, the energy consumption and carbon footprint of FL are seen as primary concerns due to limited batteries on devices [13], energy availability [2], and for environmental reasons [23]. Nonetheless, limited work concerns the concurrent optimization of both dimensions, especially regarding optimal solutions.

In this context, we present *optimal client selection algorithms for the joint optimization of time and energy in Federated Learning*. By defining how much data each client should use for training, we propose two algorithms: *Minimal Makespan and Energy Consumption FL Schedule* (MEC) and *Minimal Energy Consumption and Makespan FL Schedule under Time Constraint* (ECMTC). The first minimizes training time and total energy consumption, in that order, while the second reverses the priority between the two metrics while also meeting a deadline. Besides proposing these novel algorithms, we prove their optimality and provide an extensive experimental analysis using simulation and a real computing platform using an open-source FL framework named Flower [6]. Our results illustrate the benefits of joint time and energy optimization against state-of-the-art algorithms [20], [28], [32].

The remainder of this paper is structured as follows. Section II summarizes state-of-the-art client selection strategies for FL. Section III specifies the optimization problems addressed in this work. Section IV introduces MEC and ECMTC and shows their optimality. Section V performs a series of experiments comparing the proposed algorithms with strategies from the literature. Section VI concludes and outlines future work.

II. RELATED WORK

Client selection methods aim to tackle the challenges present in FL systems regarding system and statistical hetero-

generality. The former is characterized by significant variability of hardware, network connectivity, and battery capacity, and the latter is expressed by highly non-identically distributed data across devices, which adds complexity to problem modeling, theoretical analysis, and empirical evaluations [9], [12].

FedCS [17], the **pioneering work** in this sense, selects clients based on their resource conditions, such as wireless channel states, computational capacities, and the size of relevant data, to avoid inefficient clients that take longer time to train or to communicate with the server.

Several works aim to **reduce the number of communication rounds** needed to reach a desired **accuracy**. FedPNS [27] dynamically changes the probability of clients being selected based on the quality of local updates, preferentially selecting clients that propel faster model convergence. ECSM [22] selects clients based on their historical accuracy and reputation while employing random sampling to ensure model generalization. AdaFL [11] evaluates clients' contributions by combining their performance metrics from current and historical rounds through a flexible weighted average function. FedMCCS [1] selects clients by predicting whether they can successfully train the model in an environment with limited communication and computation. AUCTION [8] encodes the client selection policy into an attention-based neural network using reinforcement learning when considering factors like the size and quality of data and the resource price. In [19], the contribution threshold and the data batch size are optimized based on the client's contribution to the convergence rate. FANS [14] evaluates clients with a universally standardized dataset to quantify their efficacy based on contextual information.

Other works **optimize energy consumption**. ELASTIC [28] selects clients by dynamically adjusting the trade-off between maximizing the number of selected clients and minimizing the total energy consumption. GREED [3] ensures that selected clients have sufficient battery power to upload their local updates before a deadline. FedAECS [32] optimizes the trade-off between energy consumption and accuracy to handle clients with dissimilar amounts of data. ESCS [15] considers decision criteria such as battery level, training time, and network quality during client selection. FedAEB [31] balances overall energy consumption, model performance, and latency through a dynamic optimization method that uses entropy-regularized reinforcement learning. EACS-FL [33] jointly optimizes energy consumption and training time by employing the combinatorial multi-armed bandit method.

Few works find the **optimal selection of clients** regarding a given optimization target. OLAR [20] minimizes the duration of a communication round by greedily controlling how much data each client uses for training. Wang et al [26] propose a binary search-based solution for the same problem with independent and identically distributed (IID) data and another algorithm for non-IID data. Although not proved in the paper, the former algorithm seems optimal. Both works require monotonically increasing execution times for their algorithms to work. Finally, a solution based on the multiple-choice minimum-cost maximal knapsack packing

problem ((MC)²MKP) has been proposed to minimize energy consumption while controlling the workload distribution on heterogeneous resources [21].

Despite the contributions of the related work, to the best of our knowledge, this work is the first to propose algorithms that make optimal selections of clients with heterogeneous resources by optimizing the execution time and energy consumption jointly while defining how much data each should use locally. We also go beyond simulations by providing experimental results in a real platform, which incurs additional challenges regarding time and energy profiling, and the effects of scheduling decisions over multiple communication rounds.

III. PROBLEM EXAMPLE AND DEFINITIONS

A. Problem Example

Consider an FL system consisting of one server and three clients with heterogeneous resources, illustrated in Fig. 1. The server must choose how to distribute six mini-batches of work (tasks) during a training round. When the server requests that client 1 use two mini-batches for training, the client will take part of its local data (that is never shared) to compose the mini-batches, train with them, and send its updated model back to the server.

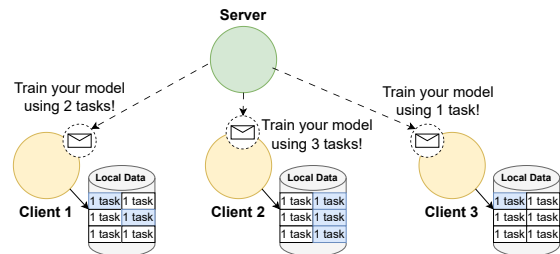


Fig. 1. Interaction between the server and the selected clients concerning the distribution of tasks. Tasks represent slices of local data to be used by clients.

In FL systems, the standard manner for distributing work (FedAvg [16]) is to split the work evenly among the clients (i.e., 2 tasks/client). Suppose that the server has an estimation of the execution time in seconds (P_i) and the energy consumption in joules (E_i) for training each client i with a certain number of mini-batches (Fig. 2). With these estimations, we can notice that the resulting schedule would result in a training time (C_{\max}) of 10 s, given by the slowest client, and total energy consumption (ΣE) of 13.32 J, given by the sum of the energy consumed by all clients. This schedule is illustrated in the left-hand side of Fig. 3.

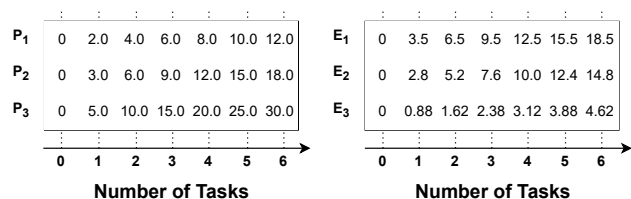


Fig. 2. Performance and energy costs per client for each number of tasks.

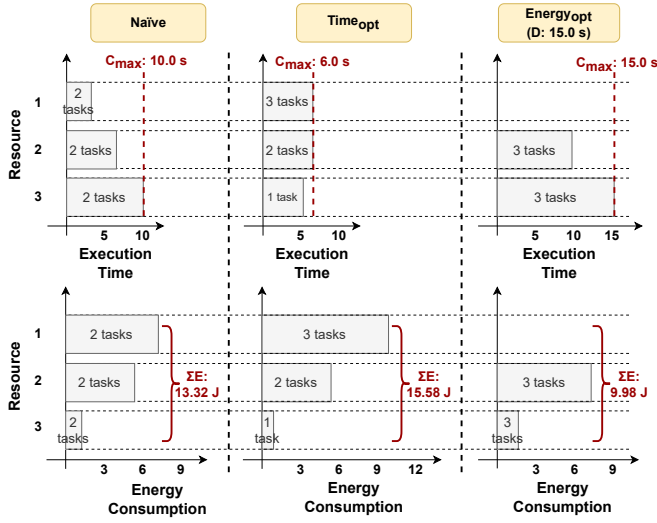


Fig. 3. Examples of training times and energy consumption when scheduling six mini-batches (tasks) among the devices.

Now consider the situation where the server aims to optimize the training time and the energy consumption, in order. An optimal schedule would distribute three tasks for the first, two for the second, and one for the third client, leading to $C_{\max} = 6$ s and $\Sigma E = 15.58$ J (Fig. 3, center).

Finally, consider the scenario where the server aims to optimize, in order, the energy consumption and the training time, while also meeting a given deadline $D = 15$ seconds. In this situation, no tasks would be given to the first client, while the other two would receive three tasks each, leading to $C_{\max} = 15$ s and $\Sigma E = 9.98$ J (Fig. 3, right-hand side). Meanwhile, if no deadline were present, a schedule with all tasks on the third client would lead to better energy consumption ($\Sigma E = 4.62$ J) but worse training time ($C_{\max} = 30$ s).

B. Scheduling Problem Definitions

The previous example illustrated the importance of efficient client selection in FL. Consider the case of n heterogeneous resources organized in a set $\mathcal{R} = \{1, 2, \dots, n\}$. These resources are required to train machine learning models with a workload of total size $T \in \mathbb{N}$. This workload contains identical, independent, and atomic tasks, which represent slices of local data. Each resource $i \in \mathcal{R}$ can be assigned $A_i \subset \mathbb{N}$ tasks to compute. Two functions, $P_i : A_i \rightarrow \mathbb{R}_{\geq 0}$ and $E_i : A_i \rightarrow \mathbb{R}_{\geq 0}$, represent the execution time (or performance) and energy consumption, respectively, of resource i when computing its assigned number of tasks. For now, no assumptions are made regarding the behavior or shape of these functions. Finally, let $X = \{x_1, \dots, x_n\}$ be the schedule that assigns $x_i \in A_i$ tasks to each resource $i \in \mathcal{R}$. Based on these functions, we define two distinct optimization targets: the *makespan* C_{\max} (Eq. (1)) and the *total energy consumption* ΣE (Eq. (2)).

$$C_{\max} := \max_{i \in \mathcal{R}} P_i(x_i) \quad (1) \quad \left| \quad \Sigma E := \sum_{i \in \mathcal{R}} E_i(x_i) \quad (2)$$

Table I summarizes the notation used throughout this text. Let $\mathcal{A} = \{A_1, \dots, A_n\}$, $\mathcal{P} = \{P_1, \dots, P_n\}$, and $\mathcal{E} = \{E_1, \dots, E_n\}$ denote the sets of possible assignments, time functions, and energy functions. Given a problem instance $(\mathcal{R}, T, \mathcal{A}, \mathcal{P}, \mathcal{E})$, the goal of the *Minimal Makespan and Energy Consumption FL Schedule* (MEC) problem is to find an optimal schedule X^* that minimizes C_{\max} and ΣE , in that order (Eq. (3)). On the other hand, given a problem instance $(\mathcal{R}, T, \mathcal{A}, \mathcal{P}, \mathcal{E}, D)$ with deadline $D \in \mathbb{R}_{\geq 0}$, the goal of the *Minimal Energy Consumption and Makespan FL Schedule under Time Constraint* (ECMTC) problem is to find an optimal schedule X^* that minimizes ΣE and C_{\max} , in that order, while meeting the deadline (Eq. (4)).

TABLE I
NOTATION SUMMARY FOR THE OPTIMIZATION PROBLEMS.

| Symbol | Meaning |
|---------------|----------------------------------------------------------------|
| n | Number of resources. |
| \mathcal{R} | Set of resources. |
| T | Size of the workload. |
| A_i | Set of numbers of tasks that can be assigned to resource i . |
| x_i | Number of tasks assigned to resource i . |
| P_i | Set of execution times of resource i . |
| E_i | Set of energy consumptions of resource i . |
| D | User-defined deadline (ECMTC only). |
| X | Schedule that assign all tasks to resources. |
| X^* | Optimal schedule. |
| C_{\max} | Makespan (maximum execution time) of a schedule. |
| ΣE | Total energy consumption of a schedule. |

$$\begin{array}{l|l} \text{lex min}_X C_{\max}, \Sigma E & (3a) \quad \text{lex min}_X \Sigma E, C_{\max} & (4a) \\ \text{subject to } \sum_{i \in \mathcal{R}} x_i = T, & (3b) \quad \text{subject to } \sum_{i \in \mathcal{R}} x_i = T, & (4b) \\ & x_i \in A_i, \forall i \in \mathcal{R} & (3c) \quad C_{\max} \leq D, & (4c) \\ & & & x_i \in A_i, \forall i \in \mathcal{R} & (4d) \end{array}$$

With the lexicographic method (lex), a type of multi-objective optimization, the objective functions are arranged in order of importance. Once the order is set, each function is solved such that each subsequent problem does not degrade any of the previous solutions [5].

Fig. 4 illustrates the interactions between the different problems by comparing the makespan (horizontal axis) and total energy consumption (vertical axis) of varied solutions. The solution for MEC (red-solid point) finds the minimal makespan and the minimal energy consumption that goes with it. The solution for ECMTC with $D \approx \infty$ (green-solid point) finds the minimal energy consumption and the minimal makespan for it. In between, solutions for ECMTC with specific time limits (blue-solid points) find the minimal energy consumption in a given time limit and the minimal makespan for such energy consumption. All these points represent solutions belonging to the Pareto frontier — by definition, no solutions can be improved for one metric without compromising the other.

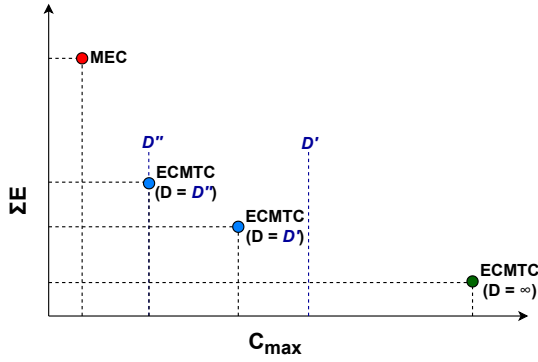


Fig. 4. Relation between MEC and ECMTC for a given problem instance.

IV. TIME AND ENERGY-AWARE SCHEDULING ALGORITHMS

The solutions to the MEC and ECMTC problems are based on a dynamic programming algorithmic structure (Algorithm 1). It computes partial solutions that optimize the problem with worst-case complexity of $O(T^2n)$ and includes five actions that can be executed differently depending on the problem being solved: (I) *Filtering*, (II) *Initialization*, (III) *Find Solutions for the First Resource*, (IV) *Test New Solution*, and (V) *Organize Final Solution*.

Algorithm 1 Dynamic Programming Skeleton

Input: $\mathcal{R}, T, \mathcal{A}, \mathcal{P}$, and/or \mathcal{E}, D (default: $+\infty$).
Output: X^*, C_{\max} , and/or ΣE .

- 1: (I) *Filtering*
- 2: **for** $i = 1, \dots, n$ **do** \triangleright Initialize the minimal costs and partial solutions matrices.
- 3: **for** $t = 0, \dots, T$ **do**
- 4: $I[i][t] \leftarrow \emptyset$
- 5: (II) *Initialization*
- 6: **for** $j \in A_1$ **do**
- 7: $I[1][j] \leftarrow j$
- 8: (III) *Find Solutions for the First Resource*
- 9: **for** $i = 2, \dots, n$ **do** \triangleright Find the solutions for other resources.
- 10: **for** $j \in A_i$ **do** \triangleright Test all assignments to resource i .
- 11: **for** $t = j, \dots, T$ **do**
- 12: (IV) *Test New Solution*
- 13: $t \leftarrow T$
- 14: **for** $i = n, \dots, 1$ **do** \triangleright Extract the optimal schedule X^* .
- 15: $j \leftarrow I[i][t]; x_i \leftarrow j; t \leftarrow t - j$
- 16: (V) *Organize Final Solution*

A. Solving MEC (time first, energy second)

A two-step approach is required to solve MEC (Algorithm 2). Firstly, the minimal makespan is computed (Algorithm 3) and then used as the deadline D when computing the minimal energy consumption (Algorithm 4). The second step provides the optimal schedule with minimal makespan and energy consumption, as proven in Theorem 2.

Algorithm 2 MEC

- 1: $-, C_{\max}, - \leftarrow \text{MinMaxTime}(\mathcal{R}, T, \mathcal{A}, \mathcal{P})$
- 2: $X^*, -, \Sigma E \leftarrow \text{MinSumEnergy}(\mathcal{R}, T, \mathcal{A}, \mathcal{P}, \mathcal{E}, C_{\max})$
- 3: **return** $X^*, C_{\max}, \Sigma E$

1) *Step 1: Computing the Minimal Makespan.*: Consider $\mathcal{Z}_r^P(\tau)$ as an optimal solution value for a partial problem with the first r resources that schedule τ tasks. Assume that $\mathcal{Z}_r^P(\tau) := \infty$ if no solution exists and that $\mathcal{Z}_0^P(0) := 0$. Thus, $\mathcal{Z}_r^P(\tau)$ is defined in Eq. (5) and can be recursively computed following Eq. (6).

$$\mathcal{Z}_r^P(\tau) := \min \left\{ \max_{i \in [1, r]} P_i(x_i) \mid \sum_{i=1}^r x_i = \tau \right\} \quad (5)$$

$$\mathcal{Z}_r^P(\tau) = \min_{j \in A_r, j \leq \tau} \max \left(\mathcal{Z}_{r-1}^P(\tau - j), P_r(j) \right) \quad (6)$$

Algorithm 3 employs the properties of Eqs. (5) and (6). It computes all possible solutions for \mathcal{Z}_1^P in action (III), then computes optimal solutions for an increasing number of resources on the main loop (Algorithm 1, line 9). The best solution for a given resource i and t tasks is defined considering the previous best solution with $i - 1$ resources (based on Eq. (6)). The makespan of this partial solution is computed and stored in $P[i][t]$ by comparing the maximum between the previous best solution and the execution time for a given number of tasks in resource i . The minimal makespan is found by keeping the minimal solution in action (IV). The solution for $\mathcal{Z}_n^P(T)$ is returned in action (V).

Algorithm 3 MinMaxTime (Minimal Makespan)

- 1: (I) *Filtering*: Nothing to do.
- 2: (II) *Initialization*: $P[i][t] \leftarrow \infty$
- 3: (III) *Find Solutions for the First Resource*: $P[1][j] \leftarrow P_1(j)$
- 4: (IV) *Test New Solution*:
- 5: $P_{\text{new}} \leftarrow \max(P[i-1][t-j], P_i(j))$
- 6: **if** $P_{\text{new}} < P[i][t]$ **then**
- 7: $P[i][t] \leftarrow P_{\text{new}}; I[i][t] \leftarrow j$ \triangleright Best solution so far.
- 8: (V) *Organize Final Solution*: $C_{\max} \leftarrow P[n][T]$

The optimality of Algorithm 3 can be demonstrated by induction, and its proof will be based on Eqs. (5) and (6), since row r in matrix P represents the optimal solutions found for \mathcal{Z}_r^P . Notice that the maximum and minimum operations are commutative and associative, so the order in which the resources and number of tasks are considered does not impact the optimal makespan found.

Lemma 1. *Solutions in \mathcal{Z}_1^P are optimal.*

Proof. The only possible solutions for \mathcal{Z}_1^P are $\mathcal{Z}_1^P(j) = P_1(j)$ for all $j \in A_1$, therefore they are optimal. \square

Lemma 2. *If solutions in \mathcal{Z}_i^P are optimal, solutions in \mathcal{Z}_{i+1}^P are also optimal.*

Proof. The value of $\mathcal{Z}_{i+1}^P(\tau)$ for $\tau \in [0..T]$ is minimum among all maximums between $\mathcal{Z}_i^P(\tau - j)$ and $C_{i+1}(j)$ for $j \in A_{i+1}, j \leq \tau$ (Eq. (6)). To consider that there would be another schedule with a smaller value is a contradiction, as it would require having a sub-optimal \mathcal{Z}_i^P or a value that is smaller than the minimum of all possible optimal solutions for $\mathcal{Z}_{i+1}^P(\tau)$. Hence, \mathcal{Z}_{i+1}^P is optimal. \square

Theorem 1. $\mathcal{Z}_n^P(T)$ yields optimal solution to the Minimal Makespan Problem.

Proof. Lemmas 1 and 2 prove the optimality of the base case and the inductive step, so $\mathcal{Z}_n^P(T)$ is optimal. \square

2) *Step 2: Computing the Minimal Energy Consumption.*: Consider $\mathcal{Z}_r^E(\tau, D)$ as an optimal solution value for a partial problem with the first r resources that schedule τ tasks while meeting the deadline D . Similar to the previous case, assume that $\mathcal{Z}_r^E(\tau, \cdot) := \infty$ if no solution exists and that $\mathcal{Z}_0^E(0, \cdot) := 0$. Thus, $\mathcal{Z}_r^E(\tau, D)$ is defined in Eq. (7) and can be recursively computed following Eq. (8).

$$\mathcal{Z}_r^E(\tau, D) := \min \left\{ \sum_{i=1}^r E_i(x_i) \mid \begin{array}{l} \sum_{i=1}^r x_i = \tau, \\ P_i(x_i) \leq D, i \in [1, r] \end{array} \right\} \quad (7)$$

$$\mathcal{Z}_r^E(\tau, D) = \min_{j \in A_r, j \leq \tau, P_r(j) \leq D} \mathcal{Z}_{r-1}^E(\tau - j, D) + E_r(j) \quad (8)$$

Algorithm 4, which is based on (MC)²MKP [21], employs the properties of Eqs. (7) and (8) and discards in action (I) any task assignments that could surpass the deadline D . Moreover, only solutions with minimal energy consumption are kept in action (IV). Since its difference compared with (MC)²MKP lies in its filtering, which does not impact the general behavior of the algorithm nor its optimality, and since (MC)²MKP has been proven optimal for finding minimal accumulated costs, we will refrain from proving Algorithm 4's optimality.

Algorithm 4 MinSumEnergy (Minimal Energy Consumption)

```

1: (I) Filtering:
2: for  $i = 1, \dots, n$  do ▷ Allow only assignments that respect  $D$ .
3:    $A_i \leftarrow \{j \mid j \in A_i \wedge P_i(j) \leq D\}$ 
4: (II) Initialization:  $E[i][t] \leftarrow \infty$ 
5: (III) Find Solutions for the First Resource:  $E[1][j] \leftarrow E_1(j)$ 
6: (IV) Test New Solution:
7:  $E_{\text{new}} \leftarrow E[i-1][t-j] + E_i(j)$ 
8: if  $E_{\text{new}} < E[i][t]$  then
9:    $E[i][t] \leftarrow E_{\text{new}} ; I[i][t] \leftarrow j$  ▷ Best solution so far.
10: (V) Organize Final Solution:  $\Sigma E \leftarrow E[n][T]$ 

```

Theorem 2. *The schedule computed by MEC is optimal.*

Proof. *MinMaxTime* (Algorithm 3) finds a schedule with minimal makespan (Theorem 1), which is used to discard any task assignments that could surpass its value at action (I) of *MinSumEnergy* (Algorithm 4). Consequently, no schedule computed by *MinSumEnergy* can have a higher (due to filtering) or lower (due to the optimality of *MinMaxTime*) makespan. By definition, *MinSumEnergy* finds a schedule with minimal energy consumption [21, Theorem 1]. Thus, the schedule computed by MEC (Algorithm 2) achieves minimal makespan and energy consumption and, therefore, it is optimal. \square

B. Solving ECMTC (energy first, time second)

Unlike MEC, ECMTC can be computed in a single pass. Consider $\mathcal{Z}_r^{EP}(\tau, D)$ as an optimal solution pair $(\Sigma E, C_{\max})$ for a partial problem with the first r resources that schedules τ

tasks while meeting the deadline D . Assume that $\mathcal{Z}_r^{EP}(\tau, \cdot) := (\infty, \infty)$ if no solution exists and that $\mathcal{Z}_0^{EP}(0, \cdot) := (0, 0)$. Thus, $\mathcal{Z}_r^{EP}(\tau, D)$ is defined in Eq. (9) and can be recursively computed following Eq. (10).

$$\mathcal{Z}_r^{EP}(\tau, D) := \text{lex min} \left\{ \sum_{i=1}^r E_i(x_i), \max_{i \in [1, r]} P_i(x_i) \mid \sum_{i=1}^r x_i = \tau \wedge P_i(x_i) \leq D, i \in [1, r] \right\} \quad (9)$$

$$\mathcal{Z}_r^{EP}(\tau, D) = \text{lex min}_{j \in A_r, j \leq \tau, P_r(j) \leq D} e + E_r(j), p + P_r(j) \quad \text{where } (e, p) = \mathcal{Z}_{r-1}^{EP}(\tau - j, D) \quad (10)$$

Algorithm 5 employs the properties of Eqs. (9) and (10). It does the same filtering as *MinSumEnergy*, and it combines actions (II), (III), and (V) of *MinMaxTime* and *MinSumEnergy*. Its main difference lies in action (IV). The best solution for a given resource i and t tasks is still defined based on the previous best solution with $i - 1$ resources. However, two conditions can lead to a better solution: (i) the energy consumption of the new solution is smaller than that of the current solution, or (ii) they both have the same energy consumption, and the new solution has a shorter execution time. By comparing the energy consumption and execution time of the previous best and current solutions and keeping their lexicographic minimum, $\mathcal{Z}_i^{EP}(t, D)$ is computed, and the minimum energy consumption and makespan pair is found.

Algorithm 5 ECMTC

```

1: (I) Filtering:
2: for  $i = 1, \dots, n$  do ▷ Allow only assignments that respect  $D$ .
3:    $A_i \leftarrow \{j \mid j \in A_i \wedge P_i(j) \leq D\}$ 
4: (II) Initialization:  $E[i][t] \leftarrow \infty ; P[i][t] \leftarrow \infty$ 
5: (III) Find Solutions for the First Resource:  $E[1][j] \leftarrow E_1(j) ; P[1][j] \leftarrow P_1(j)$ 
6: (IV) Test New Solution:
7:  $E_{\text{new}} \leftarrow E[i-1][t-j] + E_i(j)$ 
8:  $P_{\text{new}} \leftarrow \max(P[i-1][t-j], P_i(j))$ 
9: if  $(E_{\text{new}} < E[i][t])$  or  $(E_{\text{new}} = E[i][t] \text{ and } P_{\text{new}} < P[i][t])$  then
10:    $E[i][t] \leftarrow E_{\text{new}} ; P[i][t] \leftarrow P_{\text{new}} ; I[i][t] \leftarrow j$  ▷ Best solution so far.
11: (V) Organize Final Solution:  $\Sigma E \leftarrow E[n][T] ; C_{\max} \leftarrow P[n][T]$ 

```

The optimality of Algorithm 5 is proved analogously to *MinMaxTime*.

Lemma 3. *Solutions in \mathcal{Z}_1^{EP} are optimal.*

Proof. The only possible solutions for \mathcal{Z}_1^{EP} are $\mathcal{Z}_1^{EP}(j, D) = (E_1(j), P_1(j))$ for all $j \in A_1, P_1(j) \leq D$, therefore they are optimal. \square

Lemma 4. *If solutions in \mathcal{Z}_i^{EP} are optimal, solutions in \mathcal{Z}_{i+1}^{EP} are also optimal.*

Proof. By the definition in Eq. (10), the value of $\mathcal{Z}_{i+1}^{EP}(\tau, D)$ for $\tau \in [0, T]$ is the minimum pair (energy consumption, execution time) among all possible combinations of $(e + E_{i+1}(j), p + P_{i+1}(j))$ for $(e, p) = \mathcal{Z}_i^{EP}(\tau - j, D)$ and $j \in A_{i+1}, j \leq \tau, P_{i+1}(j) \leq D$.

Using the notation $(e^*, p^*) = \mathcal{Z}_i^{EP}(\tau, D)$ for $\tau \in [0, T]$, assume there is another solution (e', p') such that $e' < e^*$ or $e' = e^*$ and $p' < p^*$. This situation would require one of following: (I) having a sub-optimal \mathcal{Z}_i^{EP} ; (II) having a solution with e' smaller than the minimum of all possible combinations of solutions of \mathcal{Z}_i^{EP} and E_{i+1} ; or (III) having a solution with $e' = e^*$ and p' smaller than the minimum of all possible combinations of solutions of \mathcal{Z}_i^{EP} and P_{i+1} .

All of the above-stated possibilities are contradictions: (I) \mathcal{Z}_i^{EP} is optimal by the definition of Lemma 4; (II) e' cannot be smaller than its minimum possible value; and (III) p' cannot be smaller than its minimum possible value (or $e' < e^*$, which is also a contradiction). Therefore, \mathcal{Z}_{i+1}^{EP} is optimal. \square

Theorem 3. $\mathcal{Z}_n^{EP}(T)$ yields an optimal solution to the ECMTC problem.

Proof. Lemmas 3 and 4 prove the optimality of the base case and the inductive step, so $\mathcal{Z}_n^{EP}(T)$ is optimal. \square

V. EXPERIMENTAL EVALUATION

This section presents an extensive evaluation of MEC and ECMTC algorithms compared to four state-of-the-art client selection strategies for FL: ELASTIC [28], FedAECS [32], OLAR [20], and $(MC)^2MKP$ [21]. The evaluations were carried out in both simulation-based and real-world scenarios. For more information about the data and code required to run and analyze the experiments, see [reproducibility instructions](#).

A. Simulation-Based Experiments

1) *Description:* The resource heterogeneity was simulated by generating arbitrary values for execution time, energy consumption, and accuracy for each possible task assignment per resource, using different functions as summarized in Table II. An incremental seed basis was used to ensure the reproducibility of the experiments and the randomness of the generated values.

TABLE II
FUNCTIONS USED TO SIMULATE THE RESOURCE HETEROGENEITY.

| Function | Behavior |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>linear</i> | $f(x) = \alpha + \beta \cdot x$, where $x = t, \forall t \in [0..T]$ $\alpha, \beta \in [1, 10], \forall i \in \mathcal{R}$ (time) $\alpha, \beta \in [0.32, 3.2], \forall i \in \mathcal{R}$ (energy) $\alpha, \beta \in [0.2, 2), \forall i \in \mathcal{R}$ (accuracy) |
| $n \cdot \log(n)$ | $f(x) = \alpha + \beta \cdot x \cdot \log(x + 1)$, where $x = t, \forall t \in [0..T]$ $\alpha, \beta \in [1, 10], \forall i \in \mathcal{R}$ (time) $\alpha, \beta \in [0.32, 3.2), \forall i \in \mathcal{R}$ (energy) $\alpha, \beta \in [0.2, 2), \forall i \in \mathcal{R}$ (accuracy) |
| <i>random</i> | $f(x) = x$, where $x \in [0..T]$ |

The generated values for accuracy were normalized through the *min-max scaling* procedure to prevent the sum of the accuracy of the selected clients from exceeding the theoretical

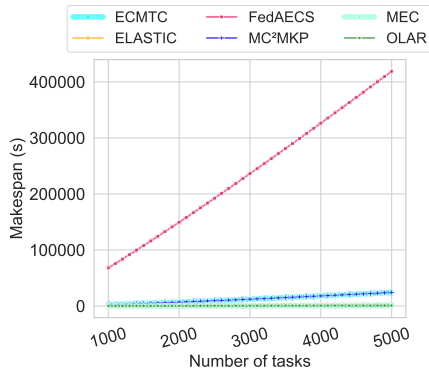
maximum value of 1. Thus, the lowest value was transformed to 0 and the highest to 1. Three sets of simulation-based experiments were considered: ES_1 , ES_2 , and ES_3 , which are described in Table III.

TABLE III
DESCRIPTION OF THE SIMULATION-BASED EXPERIMENTS.

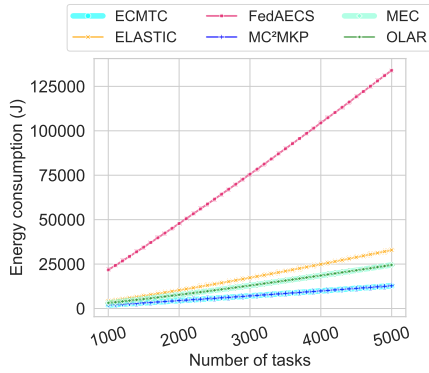
| Experiment Set | Description |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ES_1 | Purpose: Evaluation of the makespan, energy consumption, and accuracy metrics based on the schedule produced by each algorithm. Organization <ul style="list-style-type: none"> ◆ Number of resources: 10 and 100. ◆ Number of tasks: from 1,000 to 5,000, with step of 100. ◆ Resource heterogeneity: <i>linear</i>, $n \cdot \log(n)$, and <i>random</i> functions. |
| ES_2 | Purpose: Evaluation of the elapsed time taken by each algorithm to find a schedule. Organization <ul style="list-style-type: none"> ◆ First group (G_1): <ul style="list-style-type: none"> ◊ Number of resources: 100. ◊ Number of tasks: from 200 to 2,000, with a step of 200. ◆ Second group (G_2): <ul style="list-style-type: none"> ◊ Number of resources: from 10 to 100, with a step of 15. ◊ Number of tasks: 2,000. ◆ Resource heterogeneity: <i>linear</i> function. ◆ Measurements: 20 samples of five executions per algorithm. |
| ES_3 | Purpose: Evaluation of the trade-off between time and energy consumption with ECMTC. Organization <ul style="list-style-type: none"> ◆ Algorithms: MEC and ECMTC. ◆ Number of resources: 10 and 100. ◆ Number of tasks: from 1,000 to 5,000, with a step of 100. ◆ Resource heterogeneity: <i>linear</i>, $n \cdot \log(n)$, and <i>random</i> functions. ◆ Reference deadline: optimal makespan of MEC. ◆ Relaxed deadlines: 25% to 200% percentage increases of the reference deadline with a step of 25%. |

2) ES_1 Results: Fig. 5 displays the obtained makespan, energy consumption, and accuracy when varying the number of tasks as described in ES_1 . Table IV outlines the minimum and maximum values obtained for each metric by the algorithms, for 1,000 (min.) and 5,000 (max.) scheduled tasks, respectively. Concerning the makespan, the schedules for MEC and OLAR achieved the leading results, which are optimal (curves are overlapped in Fig. 5(a)). The makespan associated with the MEC schedule was 40.08% smaller than the ELASTIC one, the second-best outcome amongst the literature algorithms. Regarding energy consumption, the schedules for ECMTC and $(MC)^2MKP$ achieved the leading results, which are optimal (curves are overlapped in Fig. 5(b)). Compared to the best results produced by the algorithms from the literature for 5,000 tasks, the ECMTC schedule consumed 52.14% less energy than OLAR, the second-best result in energy consumption terms. For accuracy, the schedules for FedAECS outperformed the others for any number of tasks. In particular, a single resource processed all tasks differently from the other algorithms. Nonetheless, this gain in accuracy for FedAECS when using just one resource significantly impacted the makespan and energy metrics.

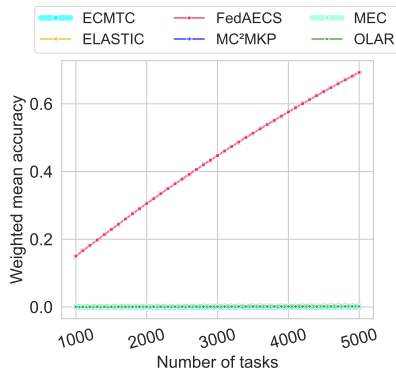
3) ES_2 Results: Table V presents the minimum and maximum execution times required for each algorithm to schedule 2,000 tasks for 100 resources, which was the worst-case configuration considering the experiments held in this work. Note that MEC and ECMTC algorithms demanded a longer time than the others. Particularly compared to $(MC)^2MKP$, the two proposed strategies took respectively 40.13 (+63.84%)



(a) Makespan variation



(b) Energy consumption variation



(c) Accuracy variation

Fig. 5. ES_1 results for 100 $n \cdot \log(n)$ resources regarding makespan (a), energy consumption (b), and accuracy (c) (1,000 to 5,000 tasks scheduled).

TABLE IV
 ES_1 RESULTS FOR 100 $n \cdot \log(n)$ RESOURCES.

| Client Selection Algorithm | 1,000 Scheduled Tasks | | | 5,000 Scheduled Tasks | | |
|----------------------------|-----------------------|------------------------|----------|-----------------------|------------------------|----------|
| | Makespan (s) | Energy Consumption (J) | Accuracy | Makespan (s) | Energy Consumption (J) | Accuracy |
| MEC | 105.83 | 3,144.66 | 0.0002 | 777.98 | 24,526.49 | 0.0018 |
| ECMTTC | 2,423.59 | 2,003.96 | 0.00003 | 24,178.32 | 12,787.26 | 0.00003 |
| OLAR | 105.83 | 3,144.66 | 0.0002 | 777.98 | 24,526.49 | 0.0018 |
| (MC) ² MKP | 2,423.59 | 2,003.96 | 0.00003 | 24,178.32 | 12,787.26 | 0.00003 |
| ELASTIC | 244.34 | 4,173.18 | 0.0006 | 1,941.03 | 32,878.40 | 0.0046 |
| FedAECS | 67,899.87 | 21,727.96 | 0.1502 | 418,898.89 | 134,047.64 | 0.6930 |

and 130.95 (+208.31%) more seconds to find a schedule for the same number of tasks. Anyway, one should recall that both

MEC and ECMTTC optimize two targets and, in real-world scenarios, where the performance of resources is usually not known for all possible task assignments, their overheads can be drastically reduced to feasible times, as we will see in the subsequent subsection V-B.

TABLE V
 ES_2 MINIMUM AND MAXIMUM EXECUTION TIMES IN SECONDS FOR SCHEDULING 2,000 TASKS TO 100 *linear* RESOURCES.

| Client Selection Algorithm | First Group (G_1) | | Second Group (G_2) | |
|----------------------------|-----------------------|--------|------------------------|-------|
| | Min. | Max. | Min. | Max. |
| MEC | 86.42 | 102.99 | 86.61 | 90.78 |
| ECMTTC | 159.52 | 193.81 | 159.99 | 161.8 |
| OLAR | 0.012 | 0.015 | 0.012 | 0.013 |
| (MC) ² MKP | 52.13 | 62.86 | 52.03 | 53.17 |
| ELASTIC | 0.005 | 0.006 | 0.005 | 0.007 |
| FedAECS | 0.53 | 0.71 | 0.53 | 0.54 |

4) ES_3 Results: Table VI presents the analysis of the trade-off between the makespan and energy consumption metrics for 100 *linear* resources for 5,000 tasks, the most relevant configuration. The percentage increases (+) or decreases (-) for ECMTTC are relative to the values obtained by MEC. Fig. 6 illustrates the convex curve obtained from these solutions. The dashed blue vertical line marks the point of maximum curvature (knee point), which shows the solution for ECMTTC with the most balanced tradeoff. For this particular solution, by employing a deadline of 150% relative to MEC's optimal makespan, ECMTTC reduced the energy consumption by 37.32% compared to MEC.

TABLE VI
 ES_3 RESULTS FOR 100 *linear* RESOURCES.

| Client Selection Algorithm | Relaxed Deadline | Makespan (s) | Energy Consumption (J) |
|----------------------------|----------------------------|--------------------|------------------------|
| MEC | — | 196.0 | 6,188.65 |
| ECMTTC | 125% $\rightarrow D = 245$ | 244.89 (+24.94%) | 4,607.88 (-25.54%) |
| | 150% $\rightarrow D = 294$ | 294.0 (+50.0%) | 3,878.75 (-37.32%) |
| | 175% $\rightarrow D = 343$ | 342.97 (+74.98%) | 3,505.1 (-43.36%) |
| | 200% $\rightarrow D = 392$ | 391.94 (+99.97%) | 3,238.61 (-47.67%) |
| | 225% $\rightarrow D = 441$ | 440.81 (+124.9%) | 3,064.85 (-50.48%) |
| | 250% $\rightarrow D = 490$ | 489.89 (+149.94%) | 2,936.56 (-52.55%) |
| | 275% $\rightarrow D = 539$ | 538.77 (+174.88%) | 2,835.75 (-54.18%) |
| 300% $\rightarrow D = 588$ | 587.87 (+199.93%) | 2,756.81 (-55.45%) | |

B. Real-World Experiments

1) *Description of the Emulated Environment:* As a proof of concept, the client selection algorithms were implemented on Flower [6], currently one of the most prominent open-source frameworks for managing FL systems. The investigations considered an FL system formed by a Server and 50 Clients, each running on an exclusive node that featured an Intel Xeon Gold 5220 CPU (18 cores), 96 GiB RAM, and no accelerators (e.g., GPUs). All the nodes belonged to a particular cluster of Grid'5000 [7], a large-scale and flexible testbed environment focused on parallel and distributed computing experiment-driven research. Despite that, the number of CPU cores used by

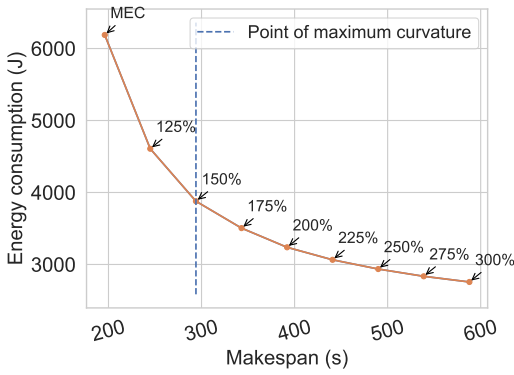


Fig. 6. ES_3 trade-off curve for 100 linear resources.

each client was deliberately restricted, allowing the emulation of resource heterogeneity. The overall number of clients was 50 heterogeneous clients, where each subset of 5 clients was associated with i cores, $i = 1, 2, 4, 6, 8, 10, 12, 16, 16, 18$.

The clients were requested to train the MobileNetV2 [24] model, a lightweight convolutional neural network, using a balanced and distinct partition of CIFAR-10 [10], a popular multi-class dataset of colored images (32×32 pixels) arranged in ten classes, each labeling a different object. The balanced partitions, generated through the dataset-splitter¹ tool, were accessible exclusively by each client on its local storage, allowing the emulation of the data privacy aspect of FL systems. Hence, each client had 1,000 local images available for training and 200 for testing the model, totaling 50,000 and 10,000 images, respectively.

The experiments comprised 100 communication rounds. The batch size, *i.e.*, the number of local samples processed before a local model update, was set to 32, and the number of epochs, *i.e.*, complete passes through the local training dataset, to 5. FedAvg [16] was used on the server to fusion the partial contributions of each round. PowerJoular [18] was employed on the selected clients to measure their energy consumption during the training and testing phases.

The client selection algorithms executed during the training phase were Random, MEC, ECMTC, ECMTC_D15, OLAR, (MC)²MKP, ELASTIC, and FedAECS. During the testing phase, only Random was employed, as this phase was executed exclusively to evaluate the quality of the models trained by the clients selected by each algorithm for the training phase. Regarding the Random Strategy, 50% and 20% of the available clients were selected for the training and testing phases, respectively. In the case of the ECMTC and (MC)²MKP algorithms, as preliminary experiments had pointed out a fixed subset of clients was selected in all communication rounds², a random sampling of 80% of the available clients was applied in each round, causing a diversification of the selection client

¹dataset-splitter - <https://github.com/alan-lira/dataset-splitter>

²Both algorithms optimize a single round. Given that there were no changes in the clients between rounds, the same energy-optimal subset of clients was always chosen.

process. Opposite to the *unlimited* deadline used by ECMTC, the ECMTC_D15 assumed a training deadline of 15 seconds.

The overhead of the client selection process can force all clients to an idle state until its completion, resulting in a waste of resources. In this sense, from round r onward ($r \geq 2$), the selection of clients for round $r + 1$ was concomitantly performed (in the server) with the execution of round r . In this way, suppose TT and CT are the maximum training time (makespan) and the maximum communication time of the selected clients during the round r , respectively. The communication time can be characterized by the actions performed before and after a client's local training, such as the model parameters (weights) exchange between the server and him. Let ST be the duration of the client selection process by an algorithm to select clients for round $r + 1$. Let IT be the idle time of clients before the round $r + 1$ starts. Therefore, IT can be calculated as follows: if $ST > TT + CT$, $IT = ST - (TT + CT)$. Otherwise, $IT = 0$.

Unlike the simulation-based experiments, the scheduling expenses were unknown before the FL execution. Consequently, the first communication round targeted the performance profiling of all 50 clients by scheduling an equal number of tasks for each client. From the second round onward, the client selection algorithms based their scheduling decision on the achieved performance of the clients in previous rounds, particularly for the execution time, energy consumption, and accuracy metrics, when applicable. Nevertheless, since each round would report the scheduling expense of a specific number of tasks for each selected client, the performance for most assignment possibilities per client would still be uncertain to the algorithms. Therefore, in each round, a linear interpolation/extrapolation of the past metrics values was computed, providing varied estimated performances for each client as input to the algorithms. The linear estimations assumed the range from zero to the number of available local images in each client, with a step of 100.

In the first set of experiments, 12,500 training tasks and 2,500 testing tasks were scheduled per round, while in the second set of experiments, 25,000 and 5,000 were scheduled, respectively. Each task symbolized a local image used by a client. In each round, each selected client randomly sliced a subset of data based on its number of assigned tasks. The data slice implies that a client selected in multiple rounds could have used a distinct number of images and eventually used repeated images throughout the FL execution.

2) *Scheduling Overhead*: Aiming to evaluate the overhead of the algorithms regarding all 100 communication rounds, Table VII presents the mean values, in seconds, for the client selection duration (ST), the makespan (TT), the maximum communication time (CT), and the clients idle time (IT). Concerning the first set of experiments, no client suffered from idleness. However, for the second set of experiments, all clients waited 1.9 seconds per round before the MEC algorithm could finish its selection, which represented a 12.31% overhead. Nonetheless, for models that are more complex than MobileNetV2 or resources with lower bandwidth capacity,

this overhead is likely to be masked due to a longer training duration or communication.

TABLE VII
MEAN CLIENT SELECTION OVERHEAD IN SECONDS.

| Client Selection Algorithm | First Set of Experiments | | | | Second Set of Experiments | | | |
|----------------------------|--------------------------|-------|------|----|---------------------------|-------|------|-----|
| | ST | TT | CT | IT | ST | TT | CT | IT |
| Random | 0.01 | 15.79 | 6.06 | 0 | 0.01 | 26.88 | 7.58 | 0 |
| MEC | 8.55 | 7.21 | 6.87 | 0 | 17.34 | 8.73 | 6.71 | 1.9 |
| ECMTC | 6.08 | 12.98 | 6.11 | 0 | 17.94 | 14.6 | 6.37 | 0 |
| ECMTC_D15 | 5.28 | 11.53 | 5.79 | 0 | 16.02 | 12.45 | 5.82 | 0 |
| OLAR | 0.86 | 7.96 | 6.67 | 0 | 1.72 | 10.19 | 6.41 | 0 |
| (MC) ² MKP | 2.28 | 13.22 | 6.0 | 0 | 6.12 | 15.09 | 6.42 | 0 |
| ELASTIC | 0.82 | 10.0 | 6.63 | 0 | 1.49 | 15.8 | 6.47 | 0 |
| FedAECS | 0.99 | 10.63 | 5.93 | 0 | 1.54 | 16.12 | 7.44 | 0 |

3) *Results*: Having in hand the first and second sets of experiments, Tables VIII and IX summarize, respectively, the following metrics considering all of the 100 communication rounds: i) for the training phase, the mean number of selected clients, the total time (in seconds), and the total energy consumption (in joules), and ii) for the testing phase, the final accuracy for the trained model (last round).

TABLE VIII
SUMMARY OF METRICS FOR THE FIRST SET OF EXPERIMENTS.

| Client Selection Algorithm | Mean Number of Selected Clients | Total Time (s) | Total Energy Consumption (J) | Final Accuracy |
|----------------------------|---------------------------------|----------------|------------------------------|----------------|
| Random | 25 | 1,579.19 | 1,039,519.74 | 0.10 |
| MEC | 50 | 721.22 | 1,486,976.73 | 0.10 |
| ECMTC | 13.58 | 1,298.15 | 763,300.76 | 0.40 |
| ECMTC_D15 | 13.62 | 1,152.81 | 767,812.95 | 0.38 |
| OLAR | 50 | 795.65 | 1,493,421.76 | 0.10 |
| (MC) ² MKP | 13.58 | 1,322.24 | 764,808.14 | 0.40 |
| ELASTIC | 50 | 1,000.28 | 1,502,038.71 | 0.10 |
| FedAECS | 22.93 | 1,062.69 | 953,130.32 | 0.10 |

TABLE IX
SUMMARY OF METRICS FOR THE SECOND SET OF EXPERIMENTS.

| Client Selection Algorithm | Mean Number of Selected Clients | Total Time (s) | Total Energy Consumption (J) | Final Accuracy |
|----------------------------|---------------------------------|----------------|------------------------------|----------------|
| Random | 25 | 2,687.71 | 1,615,671.34 | 0.45 |
| MEC | 50 | 872.96 | 2,030,212.43 | 0.10 |
| ECMTC | 25.25 | 1,460.16 | 1,522,079.18 | 0.41 |
| ECMTC_D15 | 25.58 | 1,244.79 | 1,532,521.77 | 0.45 |
| OLAR | 50 | 1,019.37 | 2,051,372.70 | 0.10 |
| (MC) ² MKP | 25.25 | 1,508.98 | 1,519,950.96 | 0.43 |
| ELASTIC | 50 | 1,579.53 | 2,078,821.69 | 0.10 |
| FedAECS | 25.25 | 1,611.62 | 1,541,478.76 | 0.41 |

In the *first set of experiments* (Table VIII), recall that 12,500 training and 2,500 testing tasks were scheduled per communication round, amounts that equate to 25% of the respective total data. These reduced amounts influenced the poor quality of the model obtained by the algorithms that selected more clients since each one trained its local model with lesser data than the clients selected by the remaining algorithms. MEC and OLAR, the best algorithms for optimizing makespan, reduced the total training time by 54.33% and 49.62%, respectively, compared with Random. Still, MEC reduced the required energy for training by 0.43% compared to OLAR. ECMTC and (MC)²MKP, the best algorithms for optimizing energy

consumption, reduced the total energy consumed during the training by 26.57% and 26.43%, respectively, compared with Random. Yet, ECMTC reduced the required time for training by 1.82% compared to (MC)²MKP.

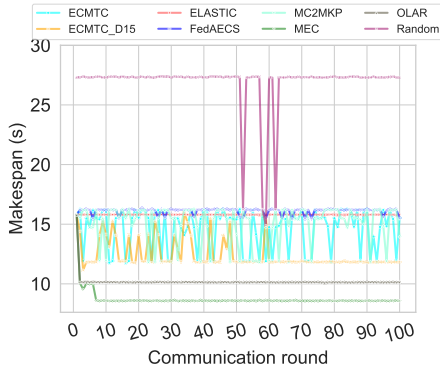
Concerning the *second set of experiments* (Table IX), recall that 25,000 training tasks and 5,000 testing tasks were scheduled per communication round, in this case, amounts that equate to 50% of the total data. Although the data used was double that of the first set of experiments, the algorithms that selected all clients still obtained low-quality models. MEC and OLAR, the best algorithms for optimizing makespan, reduced the total training time by 67.52% and 62.07%, respectively, compared with Random. Moreover, MEC reduced the required energy for training by 1.03% compared to OLAR. ECMTC and (MC)²MKP, the best algorithms for optimizing energy consumption, reduced the total energy consumed during the training by 5.79% and 5.92%, respectively, compared with Random. Although ECMTC consumed slightly more energy than (MC)²MKP, it required 3.24% less time for training.

Due to a more representative case, the remaining breakdown focused on the second set of experiments. The results of Random selection, the default client selection strategy for FedAvg, were established as the baseline during the analysis. Fig. 7 illustrates the makespan, energy consumption, and accuracy obtained during the training phase of each round for the second set of experiments, and Fig. 8 shows the accuracy obtained during their respective testing phase.

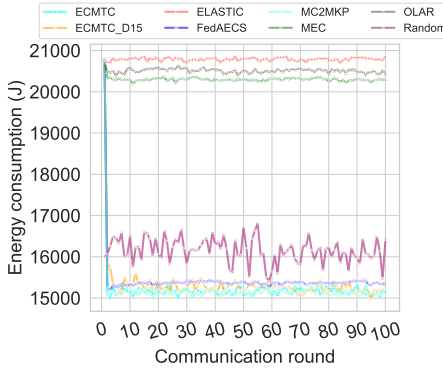
Regarding the *makespan* (Fig. 7(a)), MEC and OLAR outperformed all other algorithms, Random being the one that produced the worst results. On average, they required 18.15 and 16.68 fewer seconds than the Random strategy to finish the training phase, respectively. The observed difference in the makespan between MEC and OLAR is assumed to be caused by a performance variation on the client nodes during the execution of the experiments and requires further investigation. Particularly for the ECMTC_D15, while the algorithm produced schedules respecting the 15-second deadline, likely due to performance variation of the selected clients and imprecise estimations, the established 15-second deadline was transgressed in 7 out of 100 rounds.

Regarding the *energy consumption* (Fig. 7(b)), ECMTC and (MC)²MKP outperformed all of the other algorithms. On average, they required 935.92 and 957.20 fewer joules than Random to finish the training phase, respectively. Although the difference in energy consumed per round may appear small compared to Random, both ECMTC and (MC)²MKP selected, on average, 25.25 clients, in contrast to the fixed number of 25 clients by Random, particularly demonstrating the importance of energy-aware algorithms, whose selection decisions are driven not only by the suitable amount of clients but also by their actual efficiency.

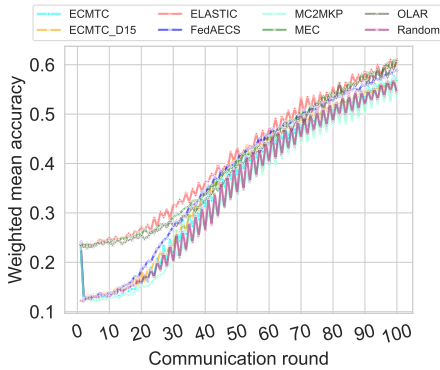
Regarding the *training accuracy* (Fig. 7(c)), MEC, OLAR, and ELASTIC (which selected all of the 50 available clients in all rounds) slightly outperformed the other algorithms, suggesting that they would require fewer communication rounds to achieve a particular training accuracy value. Nevertheless,



(a) Makespan variation



(b) Energy consumption variation



(c) Accuracy variation

Fig. 7. Results for the second set of experiments regarding makespan (a), energy (b), and accuracy (c) during the training phase (25,000 tasks scheduled).

the *testing accuracy* progression curves (Fig. 8) reveal these three algorithms likely suffered from overfitting. Recall that selecting all the available clients forced each chosen client to train its local model with fewer data samples than the other algorithms. Moreover, the testing accuracy progression for the remaining algorithms followed a comparable behavior among them for most rounds. Considering the fully balanced data scenario being investigated, excluding MEC, OLAR, and ELASTIC, these results suggest that the selected clients by each algorithm did not impact the global model quality.

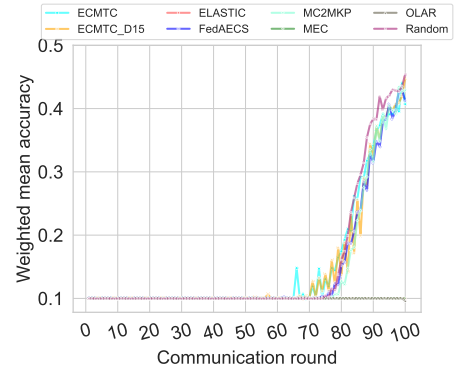


Fig. 8. Results for the second set of experiments regarding accuracy during the testing phase (5,000 tasks scheduled).

4) *Discussion:* The results demonstrate the importance of employing algorithms that efficiently select clients regarding one or more metrics of interest. In some cases, multi-objective optimization can achieve better outcomes than single optimization algorithms, as revealed by MEC and ECMTC. Nevertheless, the amount of data used locally by the clients is crucial for the model quality, in the sense that optimizing makespan as the primary optimization target is not always beneficial. Therefore, optimizing metrics related to the model quality, such as accuracy or loss, should be considered when dealing with clients with low data or unbalanced distribution.

VI. CONCLUSION AND FUTURE WORK

The client selection procedure has attained great importance in the efficiency of Federated Learning systems throughout the model training due to the heterogeneity of resources and data on the clients. This work introduces two novel algorithms, MEC and ECMTC, which jointly optimize the execution time and energy consumption during the selection of clients by defining how much data each should use locally.

Extensive experiments revealed the usefulness of our proposals compared to state-of-the-art algorithms. Regarding real-world experiments, compared to FedAvg's client selection (Random) for varied amounts of data, MEC reduced the total training time by up to 67.52%, and ECMTC reduced the total energy consumed during the training by up to 26.57%. Moreover, leveraging their multi-objective optimization distinctive, MEC required up to 1.03% less energy than OLAR, while ECMTC required up to 3.24% less time than (MC)²MKP.

Future directions include investigating the effect of selecting clients when their local data are partially or fully unbalanced and proposing a generic optimization framework that supports varied joint optimization targets, such as having accuracy as the first target and energy consumption as the second.

ACKNOWLEDGMENT

This work was funded by CAPES-PrInt (n^o 88887.310261/2018-00), CNPq/AWS 64/2022 BioCloud2 (n^o 421828/2022-6), and Inria Center at the University of Bordeaux.

Experiments presented in this work were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER, and several Universities as well as other organizations (see <https://www.grid5000.fr>).

REFERENCES

- [1] Abdulrahman, S., Tout, H., Mourad, A., Talhi, C.: FedMCCS: Multicriteria Client Selection Model for Optimal IoT Federated Learning. *IEEE Internet of Things Journal* **8**(6), 4723–4735 (2021)
- [2] Albaseer, A., Seid, A.M., Abdallah, M., Al-Fuqaha, A., Erbad, A.: Novel Approach for Curbing Unfair Energy Consumption and Biased Model in Federated Edge Learning. *IEEE Transactions on Green Communications and Networking* **8**(2), 865–877 (2024)
- [3] Albelaihi, R., Yu, L., Craft, W.D., et al.: Green Federated Learning via Energy-Aware Client Selection. In: *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*. pp. 13–18 (2022)
- [4] Amodei, D., Hernandez, D.: AI and Compute (2018), <https://openai.com/blog/ai-and-compute/>, last accessed: 2024/07/01
- [5] Andersson, J.: A survey of multiobjective optimization in engineering design. Department of Mechanical Engineering, Linköping University (2000)
- [6] Beutel, D.J., Topal, T., Mathur, A., Qiu, X., Parcollet, T., de Gusmão, P.P., Lane, N.D.: Flower: A Friendly Federated Learning Research Framework. *arXiv* (2020)
- [7] Bolze, R., Cappello, F., Caron, E., et al.: Grid'5000: A Large Scale And Highly Reconfigurable Experimental Grid Testbed. *The International Journal of High Performance Computing Applications* **20**(4), 481–494 (2006)
- [8] Deng, Y., Lyu, F., Ren, J., et al.: AUCTION: Automated and Quality-Aware Client Selection Framework for Efficient Federated Learning. *IEEE Transactions on Parallel and Distributed Systems* **33**(8), 1996–2009 (2022)
- [9] Fu, L., Zhang, H., Gao, G., et al.: Client Selection in Federated Learning: Principles, Challenges, and Opportunities. *IEEE Internet of Things Journal* **10**(24), 21811–21819 (2023)
- [10] Krizhevsky, A.: Learning Multiple Layers of Features from Tiny Images. Tech. rep., University of Toronto (2009)
- [11] Li, Q., Li, X., Zhou, L., Yan, X.: AdaFL: Adaptive Client Selection and Dynamic Contribution Evaluation for Efficient Federated Learning. In: *IEEE International Conference on Acoustics, Speech and Signal Processing*. pp. 6645–6649 (2024)
- [12] Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Processing Magazine* **37**(3), 50–60 (2020)
- [13] Li, Y., Liang, W., Li, J., Cheng, X., Yu, D., Zomaya, A.Y., Guo, S.: Energy-Aware, Device-to-Device Assisted Federated Learning in Edge Computing. *IEEE Transactions on Parallel and Distributed Systems* **34**(7), 2138–2154 (2023)
- [14] Ma, X., Shi, W., Wen, J.: An Enhanced Combinatorial Contextual Neural Bandit Approach for Client Selection in Federated Learning. In: *Proceedings of the 2024 European Interdisciplinary Cybersecurity Conference*. pp. 171–178 (2024)
- [15] Maciel, F., de Souza, A.M., Bittencourt, L.F., Villas, L.A., Braun, T.: Federated learning energy saving through client selection. *Pervasive and Mobile Computing* **103** (2024)
- [16] McMahan, B., Moore, E., Ramage, D., et al.: Communication-Efficient Learning of Deep Networks from Decentralized Data. In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. vol. 54, pp. 1273–1282 (2017)
- [17] Nishio, T., Yonetani, R.: Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge. In: *IEEE International Conference on Communications (ICC)*. pp. 1–7 (2019)
- [18] Noureddine, A.: PowerJoular and JoularJX: Multi-Platform Software Power Monitoring Tools. In: *2022 18th International Conference on Intelligent Environments (IE)*. pp. 1–4. IEEE (2022)
- [19] Ouyang, J., Liu, Y.: Learning Efficiency Maximization for Wireless Federated Learning With Heterogeneous Data and Clients. *IEEE Transactions on Cognitive Communications and Networking* pp. 1–1 (2024)
- [20] Pilla, L.L.: Optimal Task Assignment for Heterogeneous Federated Learning Devices. In: *IEEE International Parallel and Distributed Processing Symposium*. pp. 661–670 (2021)
- [21] Pilla, L.L.: Scheduling Algorithms for Federated Learning With Minimal Energy Consumption. *IEEE Transactions on Parallel and Distributed Systems* **34**(4), 1215–1226 (2023)
- [22] Putra, M.A.P., Sampedro, G.A., Kim, D.S., Lee, J.M.: ECSM: An Ensembled Client Selection Mechanism for Efficient Federated Learning. In: *IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology*. pp. 13–17 (2023)
- [23] Qiu, X., Parcollet, T., Fernandez-Marques, J., Gusmao, P.P.B., Gao, Y., Beutel, D.J., Topal, T., Mathur, A., Lane, N.D.: A First Look into the Carbon Footprint of Federated Learning. *Journal of Machine Learning Research* **24**(129), 1–23 (2023)
- [24] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: Inverted Residuals and Linear Bottlenecks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 4510–4520 (2018)
- [25] Sprague, M.R., Jalalirad, A., Scavuzzo, M., Capota, C., Neun, M., Do, L., Kopp, M.: Asynchronous Federated Learning for Geospatial Applications. In: *ECML PKDD 2018 Workshops*. pp. 21–28. Springer International Publishing (2019)
- [26] Wang, C., Yang, Y., Zhou, P.: Towards Efficient Scheduling of Federated Mobile Devices Under Computational and Statistical Heterogeneity. *IEEE Transactions on Parallel and Distributed Systems* **32**(2), 394–410 (2021)
- [27] Wu, H., Wang, P.: Node Selection Toward Faster Convergence for Federated Learning on Non-IID Data. *IEEE Transactions on Network Science and Engineering* **9**(5), 3099–3111 (2022)
- [28] Yu, L., Albelaihi, R., Sun, X., et al.: Jointly Optimizing Client Selection and Resource Management in Wireless Federated Learning for Internet of Things. *IEEE Internet of Things Journal* **9**(6), 4385–4395 (2021)
- [29] Zhang, R., Xu, Z., Yin, H.: Scout: An Efficient Federated Learning Client Selection Algorithm Driven by Heterogeneous Data and Resource. In: *IEEE International Conference on Joint Cloud Computing*. pp. 46–49 (2023)
- [30] Zhang, T., Gao, L., He, C., et al.: Federated Learning for the Internet of Things: Applications, Challenges, and Opportunities. *IEEE Internet of Things Magazine* **5**(1), 24–29 (2022)
- [31] Zheng, F., Sun, Y., Ni, B.: FedAEB: Deep Reinforcement Learning Based Joint Client Selection and Resource Allocation Strategy for Heterogeneous Federated Learning. *IEEE Transactions on Vehicular Technology* **73**(6), 8835–8846 (2024)
- [32] Zheng, J., Li, K., Tovar, E., Guizani, M.: Federated Learning for Energy-balanced Client Selection in Mobile Edge Computing. In: *International Wireless Communications and Mobile Computing*. pp. 1942–1947 (2021)
- [33] Zhu, K., Zhang, F., Jiao, L., Xue, B., Zhang, L.: Client selection for federated learning using combinatorial multi-armed bandit under long-term energy constraint. *Computer Networks* **250** (2024)