Incentive-based Resource Allocation for Mobile Edge Learning

Mhd Saria Allahham[1], Amr Mohamed[2] and Hossam Hassanein[1]
[1]School of Computing, Queen's University, Kingston, ON, Canada
[2]College of Engineering, Qatar University, Qatar.
Email: {20msa7, hossam.hassanein}@queensu.ca and amrm@qu.edu.qa

2022 IEEE 47th Conference on Local Computer Networks (LCN) | 978-1-6654-8001-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/LCN53696.2022.9843405

***Abstract*—Mobile Edge Learning (MEL) is a learning paradigm that facilitates training of Machine Learning (ML) models over resource-constrained edge devices. MEL consists of an orchestrator, which represents the model owner of the learning task, and learners, which own the data locally. Enabling the learning process requires the model owner to motivate learners to train the ML model on their local data and allocate sufficient resources. The time limitations and the possible existence of multiple orchestrators open the doors for the resource allocation problem. As such, we model the incentive mechanism and resource allocation as a multi-round Stackelberg game, and propose a Payment-based Time Allocation (PBTA) algorithm to solve the game. In PBTA, orchestrators first determine the pricing, then the learners allocate each orchestrator a timeslot and determine the amount of data and resources for each orchestrator. Finally, we evaluate the PBTA performance and compare it against a recent state-of-the-art approach.**

***Index Terms*—edge learning, distributed learning, incentive mechanism, stackelberg game.**

I. Introduction

The emerging technologies and applications at the network edge have generated vast amounts of data. Such data enables the deployment of Machine Learning (ML)-based services such as crowdsensing [1] on resource-constrained edge devices (e.g., Internet of Things (IoT) devices). Previously, the data was sent to the cloud to utilize its resources and perform the training of ML models. However, due to the latency overhead and the limitations of communication resources along with privacy concerns, moving such data to the cloud has become impractical. Instead, edge-based schemes, such as Mobile Edge Learning (MEL) [2], were shown to be more practical, that is, learning tasks (i.e., the training of ML models) are distributed across the edge devices to utilize their available computational resources. In MEL, there are two main entities: 1) Orchestrator (a.k.a. parameter server) which owns the ML model, governs the distributed training, and synchronizes it, and 2) learners, which have local datasets and available computational resources.

However, edge devices are hesitant to become learners and share their resources as well as offer their private data for training without being compensated. Moreover, the performance of the distributed training deteriorates without sufficient data and computational resources [3]. Hence, it is crucial for orchestrators to incentivize and motivate as many edge devices as possible to become learners and engage in the learning process in an active and reliable manner.

Incentive mechanism design in the literature have been modeled with different approaches such as: Stackelberg Game (SB) [4], [5], [3], [6], Auction Theory [7], [8], [9], [10] and Contract Theory [11], [12], [13]. For instance, in [4], the SB formulation was adopted, and the aim was to minimize the total training time given the learner's computation capacity. In order to do so, the orchestrator tries to incentivize learners to increase their allocated computational resources for the learning task. On the other side, the learners will determine the computational resources such that their revenue is maximized, and the computation energy is minimized. Similarly, the authors in [5] aimed at optimizing the total training time for the orchestrator, and the energy consumption for the learner. A Deep Reinforcement Learning (DRL) algorithm was proposed to capture the network dynamics and adapt the orchestrator's and learners' strategies accordingly. A more generic approach has been proposed in [3], where the orchestrator utility has been expressed in terms of the learning accuracy and the payment. The modeled learning accuracy function depends on the number of data samples that will be included in the training process, where training on more unique data samples results in increasing the learning accuracy. The learners' utility is expressed in terms of revenue, communication, and computation energy consumption, and their strategy is to determine the number of data samples in the training. As for auction-based incentives, a generic approach was proposed in [7], where the learners first bid in terms of their resources, then the orchestrator collects the bids and determines the top K bidders that will be considered in the training. Whereas in [8], the authors formulated the auction problem in the context of cellular networks. The learners first determine their strategy in terms of transmission power and computation capacity based on an allocated bandwidth and incentive, then at the orchestrator's side, the learner selection problem is formulated as a knapsack problem and solved via a greedy-based algorithm.

While previous works have either considered optimizing only the amount of data or their resources, none of them considered the case of multiple learning tasks within the same environment. In our work, we consider the case of a multi-orchestrator MEL system, where each orchestrator has a different learning task, and learners have multiple datasets each of which is associated with a learning task. It is important to notice that our work is not multi-task learning with a

single orchestrator as referred to in [14] and [15], but it considers multiple single learning tasks, where each task is administrated by a unique orchestrator. To this end, we aim to address the following research questions: (1) How much data and resources should each learner allocate to a learning task? (2) Should the learners engage in a learning task in the first place? (3) What is the optimal number of training iterations and pricing from an orchestrator to motivate as many learners as possible to engage in its learning task? As such, the main contributions are summarized as follows:

1) We formulate the orchestrators-learners interactions in the learning process as a multi-round SB game, where each round represents a global cycle, the orchestrators act as leaders and the learners act as followers.
2) We propose a Payment-based Time Allocation (PBTA) algorithm where learners divide their given time frame based on the orchestrators' pricing and determine their strategy accordingly, whereas the orchestrators' strategy is determined by adopting the backward induction algorithm.
3) We assess the performance of the proposed algorithm and compare it against a recent state-of-the-art technique.

The rest of the paper is organized as follows: we present the system model in Section II. Then the game formulation is described in Section III. The proposed algorithm is then presented and discussed in Section IV. Finally, we show the simulation results in Section V before we conclude in Section VI.

II. System Model

In this work, we consider a MEL system with multiple orchestrators, where each orchestrator has a different learning task with a different learning model and dataset.

A. Learning model

1) Preliminaries: In our system, we consider O orchestrators and L learners, where each learner has O local datasets with N_o data samples. Each orchestrator o sends the learning task to the set of participating learners, denoted by $\mathcal{L}_o$, in terms of the ML model parameters. Afterward, each participating learner runs τ_o local training iterations of Stochastic Gradient Descent (SGD) on $n_{l,o}$ data samples to minimize its local loss function. Thereafter, the learners send back the trained ML model parameters to each orchestrator, where the latter aggregate these parameters by performing weighted averaging on the received models. Each orchestrator then keeps sending the model back and forth synchronously for a fixed number of global cycles. Interested readers can refer to [16] for more details.

2) Accuracy function: Designing a function that accounts for the model learning accuracy is challenging since it depends on numerous parameters such as the number of local iterations and global cycles, the amount of training data, and the data quality in terms of the statistical distribution of the dataset, the learning rate...etc. Within one global cycle, some of the previous works have adopted an empirical accuracy function by either considering the number of iterations for a given local accuracy [8], [11], the amount of training data only [3], or the amount of data with its quality in [9]. In our work, we conducted experiments on the FMNIST dataset [17] to measure the global accuracy for an aggregated model within one global cycle, considering a different number of local iterations and statistical distributions characterized by Kullback-Leibler divergence (KLD). The KLD metric measures how a data distribution is far from the optimal data distribution, i.e., the uniform distribution. According to the results shown in Figure 1, we observe that the empirical accuracy function can be given by:

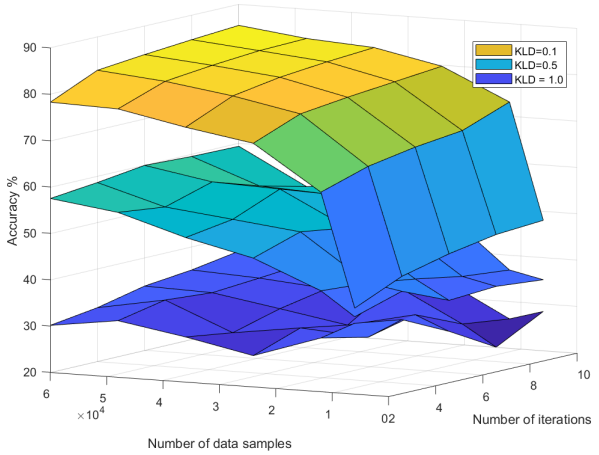

Fig. 1. Test accuracy with varying number of iterations and amount of training data considering different KLDs

$$A_o = \delta_1 \kappa_1 \times \tau_o \log(1 + \delta_2 \kappa_2 \times \sum_{l} n_{l,o}), \qquad (1)$$

where $n_{l,o}$ is the amount of data from a learner l, δ_1, δ_2 are regression parameters, κ_1 and κ_2 are parameters that account for the statistical distribution of the data across learners. Such function can be a generalization of the one that was proposed in [3], where if we considered an ideal statistical distribution and fixed number of training iterations both functions will be identical. It is worth noting that the accuracy function is a concave function with respect to $n_{l,o}$ and τ_o. However, if the data distribution across learners is highly skewed and completely non independent and identically distributed (non-IID), the concavity is no longer maintained since the parameters κ_1 and κ_2 play a big role in shaping the function surface.

B. Communication and Computation models

For the communication model, an orthogonal frequency-division multiple access (OFDMA) is considered in the system for the uplink and downlink access. We assume each orchestrator operates on a different bandwidth W_o, and each bandwidth is divided into sub-channels, where each participated learner l is allocated a sub-channel with bandwidth $W_{l,o}$ such that

$\sum_{l \in \mathcal{L}_o} W_{l,o} \leq W_o$. Therein, the achievable uplink rate for a learner l to an orchestrator o can be expressed as:

$$r_{l,o}^{up} = W_{l,o} \log_2(1 + \gamma_{l,o}^{up}), \tag{2}$$

where $\gamma_{l,o}^{up}$ is the Signal-to-Noise ratio (SNR) and is given by $\gamma_{l,o}^{up} = \frac{P_{l,o} h_{l,o}}{W_{l,o} N_o}$, $P_{l,o}$ is the learner's transmission power, $h_{l,o}$ is the channel gain and N_o is the noise power density. Similarly, with the assumption of channel reciprocity (i.e., $h_{l,o} = h_{o,l}$), $P_{o,l}$ denoting the orchestrator fixed transmission power and and SNR $\gamma_{l,o}^{d} = \frac{P_{o,l} h_{o,l}}{W_{l,o} N_o}$, the achievable downlink rate can be expressed as:

$$r_{l,o}^{d} = W_{l,o} \log_2(1 + \gamma_{l,o}^{d}). \tag{3}$$

In every global cycle, the number of bits that a learner l receives from an orchestrator o can be defined as:

$$B_o^{weights} = S_o^w V_o^w, \tag{4}$$

where S_o^w denotes the total number of weight parameters in the model and V_o^w denotes the bit per weight value. Accordingly, the total time required to send and receive the models can be express as:

$$T_{l,o}^{comm} = \frac{B_o^{weights}}{r_{l,o}^{up}} + \frac{B_o^{weights}}{r_{l,o}^{d}}. \tag{5}$$

As for the computation, the time required to train the ML model for τ_o local iterations is given by:

$$T_{l,o}^{comp} = \frac{\tau_{l,o} n_{l,o} C_o}{f_{l,o}}, \tag{6}$$

where C_o is the model computational complexity (i.e., CPU cycles per data sample) and $f_{l,o}$ is the assigned CPU resources to an orchestrator o.

Subsequently, we define the energy consumption models in the system. As the energy consumed for communications is the product of the transmission power with the transmission time, a learner's communication energy consumption can be given by:

$$E_{l,o}^{comm} = \frac{P_{l,o} B_o^{weights}}{r_{l,o}^{up}}. \tag{7}$$

As for the computation energy consumption, we adopt the energy model in [18], which can be defined in our system as:

$$E_{l,o}^{comp} = \mu \tau_o n_{l,o} C_o f_{l,o}^2, \tag{8}$$

where μ is the on-chip capacitance constant. Consequently, we can express a learner's total time and energy consumption for a learning task as follows:

$$T_{l,o} = T_{l,o}^{comp} + T_{l,o}^{comm}, \tag{9}$$

$$E_{l,o} = E_{l,o}^{comp} + E_{l,o}^{comm}. \tag{10}$$

It is important to mention that since the values of $n_{l,o}, f_{l,o}$ and $P_{l,o}$ are always positive, it can be easily proven that the energy consumption $E_{l,o}$ is a convex term.

III. Stackelberg Game Formulation

We formulate the incentive mechanism problem as a multi-round SB game [19]. Each round in the game represents a global cycle in MEL with two types of players: the orchestrators and the learners. There are two stages in each round. In the first stage, the orchestrators act first as leaders, and in the second stage, the learners act as followers. Before the start of each round, we assume some information about the learners' resources is already available on the orchestrators' side.

In the first stage, each orchestrator o broadcasts to the learners its learning task (e.g., the ML model complexity) and its strategy in terms of 1) the monetary service price $\rho_{l,o}$ (i.e., \$/CPU Cycle), and 2) the number of local training iterations τ_o. Thereafter, in the second stage, the learners decide their strategy in terms of 1) the amount of data to participate in each learning task $n_{l,o}$, 2) the allocated CPU frequency to each task $f_{l,o}$, and 3) the uplink transmission power to each orchestrator $P_{l,o}$. In our MEL system, each global cycle in the game for all the players is time-limited by T_{max}. After each player determines its strategy, the set of learners $\mathcal{L}_o$ that have decided to participate with an orchestrator o is formed for each orchestrator to perform the distributed ML training process. Afterward, the game is repeated for a predetermined number of global cycles.

A. First Stage: the Orchestrators' Side

First, we define the payment function for each orchestrator o as follows:

$$\Gamma_o(\boldsymbol{\rho}_o, \tau_o) = \sum_l C_o \rho_{l,o} \times \tau_o \times n_{l,o} \times c_l, \tag{11}$$

where $\rho_{l,o}$ is the monetary price, $\boldsymbol{\rho}_o$ represents the price vector for all the learners, and $c_l \in [0, 1]$ represents the capabilities factor of a learner l, where learners with higher capabilities get paid more than the weak ones. Therein, we can define the orchestrator o utility function as follows:

$$U_o(\tau_o, \boldsymbol{\rho}_o) = A_o(\tau_o) - \Gamma_o(\boldsymbol{\rho}_o, \tau_o), \tag{12}$$

where $A_o(\tau_o)$ is defined in Eq. (1). Each orchestrator tries to determine the pricing for each learner and the number of local iterations such that their utility function is maximized. Accordingly, each orchestrator strategy is determined by solving the following optimization problem:

$$\begin{aligned} \max_{\boldsymbol{\rho_o}, \tau_o} \quad & U_o(\tau_o, \boldsymbol{\rho_o}) \\ \text{s.t.} \quad & \Gamma_o \leq B_o \\ & \tau_{l,o} \geq 1 \\ & \rho_{min} \leq \rho_{l,o} \leq \rho_{max} \end{aligned} \tag{13}$$

where B_o is the budget of the orchestrator.

B. Second Stage: the Learners' Side

We first define the revenue function for a learner l from an orchestrator o as follows:

$$R_{l,o}(n_{l,o}) = C_o \rho_{l,o} \times \tau_o \times n_{l,o} \tag{14}$$

Consequently, the learners' utility function can be defined as follows:

$$U_l(n_{l,o}, f_{l,o}, P_{l,o}) = \sum_o R_{l,o}(n_{l,o}) - E_{l,o}(n_{l,o}, f_{l,o}, P_{l,o}). \tag{15}$$

Each learner seeks to maximize its utility via participating in the best profitable learning task and allocating the least possible resources to endure the least energy consumption, while respecting the round time constraint T_{max}. Such strategy can be determined by solving the following optimization problem for each learner as follows:

$$\begin{aligned} \max_{n_{l,o}, f_{l,o}, P_{l,o}} \quad & U_l(n_{l,o}, f_{l,o}, P_{l,o}) \\ \text{s.t.} \quad & \sum_o T_{l,o} \leq T_{max} \\ & n_{l,o} \in [n_{min}, N_o] \\ & f_{l,o} \in [0, f_{max}] \\ & P_{l,o} \in [0, P_{max}] \end{aligned} \tag{16}$$

where n_{min} is the minimum number of training data samples, f_{max} and P_{max} are the maximum CPU frequency and transmission power, respectively. The value of n_{min} is determined by the orchestrators since it is a learning requirement to guarantee a minimum learning experience for the ML model. One can notice that even though the learner's utility function is concave, it is still intractable to analyze and derive closed-form solutions for the strategy. However, closed-form solutions are required for the backward induction algorithm that solves SB games. In the backward induction algorithm, the learner optimal strategy (e.g., the amount of training data $n_{l,o}$) has to be obtained and expressed in terms of the orchestrator's strategy (e.g., the pricing $\rho_{l,o}$) such that the tradeoff optimization in the orchestrator's utility is introduced. After that, each orchestrator solves its optimization problem with any optimizer (e.g., interior-point methods, branch-and-bound) to obtain its optimal strategy. In what follows, we aim to simplify the learners' optimization problem such that it becomes tractable to analyze and derive closed-form expressions for the learners' strategy.

IV. The Payment-based Time Allocation Algorithm

In this section, we propose a heuristic algorithm to simplify the optimization problem. More specifically, we propose the Payment-based Time Allocation (PBTA) algorithm, where learners split the round time frame T_{max} between the orchestrators based on what each orchestrator pays per data sample. Precisely, the time allocated for a learning task from an orchestrator o for learner l is defined as follows:

$$T_{l,o} = \frac{C_o \rho_{l,o}}{\sum_o C_o \rho_{l,o}} \times T_{max} \tag{17}$$

where the term $C_o \rho_{l,o}$ represents the pricing per data sample (i.e., \$/data sample). The allocated time for each task is dependent on the relative ratio between the payment for this task and the total payments from all tasks. As a result, the learners will allocate more time to the learning tasks which have better payments than the others. Moreover, we know that for each learning task there are communication times, where learners download and upload the ML models, and computation time, where learners perform the training iterations on the ML model. As such, we denote θ^{comp} and θ^{comm} as the proportion of the time frame $T_{l,o}$ given for computation and communication, respectively, such that $\theta^{comp} + \theta^{comm} = 1$. As a result, the computation and communication time can be given by, respectively:

$$T^{comp}_{l,o} = \theta^{comp} T_{l,o} \ , \ T^{comm}_{l,o} = \theta^{comm} T_{l,o} \tag{18}$$

Generally, the values for θ^{comp} or θ^{comm} have a two-way dependency relationship with many parameters such as the amount of training data, the time frame of the learning task, the CPU frequency, the transmission power...etc. In other words, to determine one value of these parameters, the other values have to be known. Since our main aim is to optimize the learner's strategy, we deal with θ^{comp} and θ^{comm} as hyperparameters in the system, and then study the behavior of the learners while varying their values in the simulation to obtain their optimal ones.

After the communication and computation time for each task is now given, we can now determine the required CPU frequency and the transmission power for a learning task, respectively, as follows:

$$f_{l,o} = \begin{cases} \frac{n_{l,o} \tau_o C_o}{T^{comp}_{l,o}} & f_{l,o} \leq f_{max} \\ 0 & f_{l,o} > f_{max} \end{cases} \tag{19}$$

$$P_{l,o} = \begin{cases} \frac{\sigma^2}{h_{l,o}} \left(2^{\left(\frac{B^{weights}_o r_{l,o}}{W_{l,o}\left(r^D_{l,o} T^{comm}_{l,o} - B^{weights}_o\right)} \right)} - 1 \right), & c \geq 0 \\ 0, & c < 0 \end{cases} \tag{20}$$

where $c = P_{max} - P_{l,o}$. The terms are derived directly from Eq. (6) and (5), respectively. Nevertheless, if a learning task requires CPU frequency or transmission power greater than f_{max} and P_{max}, respectively, the learner will disregard the learning task as it asks for resources more than the available, and might endure higher energy consumption. We can notice that the CPU frequency term is now expressed in terms of the amount of data $n_{l,o}$, while the transmission power does not depend on $n_{l,o}$, but rather depends on the communication channel and time parameters only. Therefore, the communication energy term can be dropped from the learner optimization problem, and the computation energy term is expressed in terms of $n_{l,o}$ only.

***Lemma 1**: Given the time required for the communication $T^{comm}_{l,o}$ and computation $T^{comp}_{l,o}$ for a learning task, the optimal number of training data samples can be given by:*

$$n_{l,o} = \begin{cases} 0 & n_{l,o} < n_{min} \\ \frac{\phi_l \sqrt{\rho^3_{l,o}}}{\tau_o \sum_o C_o \rho_{l,o}} & n_{min} \leq n_{l,o} < N_o \\ N_o & n_{l,o} \geq N_o \end{cases} \tag{21}$$

where:

$$\phi_l = \frac{\theta^{comp} T_{max}}{\sqrt{3\mu}} \tag{22}$$

Proof. The proof can be found in Appendix A.

In Lemma 1, we can see that the amount of data for a task scales with the relative ratio between its payment and the other payments, where if the payment is relatively higher it gets allocated more training data. Moreover, the lower bound n_{min} is specified by the orchestrator, where if the learner decides to contribute with training data less than required, the strategy is rejected and it will not be considered by the orchestrator. If a task is rejected from either the learner or the orchestrator side, the learner will distribute its allocated time to other tasks equally. As such, the research questions (1) and (2) have been answered for each learner strategy. In what follows, we answer the last research question for the orchestrators' strategy.

Since the learners' strategy is now represented as a function of the orchestrator strategy, i.e., $n_{l,o}(\tau_o, \rho_{l,o})$, the orchestrators utility function can now be re-written as follows:

$$U_o(\tau_o, \rho_{l,o}) = \delta_1 \kappa_1 \times \tau_o \log(1 + \delta_2 \kappa_2 \times \sum_l n_{l,o}(\tau_o, \rho_{l,o})) - \sum_l C_o \rho_{l,o} \times \tau_o \times n_{l,o}(\tau_o, \rho_{l,o}) \times c_l \tag{23}$$

Lemma 2*: The amount of training data $n_{l,o}$ possesses a concave relationship with τ_o and $\rho_{l,o}$ jointly, and hence, the utility function $U_o(\tau_o, \rho_{l,o})$ is concave and has a unique maximum $(\tau_o^*, \rho_{l,o}^*)$.*

Proof. The proof is detailed in Appendix B.

One can notice that $n_{l,o}$ for a task depends on other tasks' payments, which requires each orchestrator to have the knowledge of other payments from other orchestrators. As such, we can make use of the concavity property of the utility function to employ an iterative optimization procedure, that is, in each iteration, each orchestrator passes its payment information to the other orchestrators as a simple message, and then optimizes its utility function locally based on the passed information. The orchestrators keep passing information and optimizing until all the orchestrators obtain the optimal strategy. The PBTA algorithm with backward induction is detailed in Algorithm 1.

Algorithm 1: Payment-based Time Allocation with backward induction

Input: $\theta^{comp}, \theta^{comm}, T_{max}, \epsilon$
At the orchestrators' side:;
Distribute the learning task details across the learners;
$t = 0$;
while $|U_o^t - U_o^{t-1}| > \epsilon, \ \ \forall\ o \in O$ **do**
  foreach *orchestrator* **do**
    Receive payment vector $\boldsymbol{\rho}_o^t$ from orchestrators;
    Optimize U_o^t ;
    Evaluate U_o^t and obtain τ_o^t and $\boldsymbol{\rho}_o^t$;
    Send $\boldsymbol{\rho}_o^t$ to other orchestrators ;
  $t = t + 1$;
Send the payments $\rho_{l,o}$ and number of iterations τ_o for each learner;
————————————————————————————;
At each learner side:;
Receive the payments $\rho_{l,o}$ and number of iterations τ_o;
$T_{l,o} = \frac{C_o \rho_{l,o}}{\sum_o C_o \rho_{l,o}} \times T_{max}$;
$T_{l,o}^{comm} = \theta^{comm} T_{l,o}$; $T_{l,o}^{comp} = \theta^{comp} T_{l,o}$;
$\mathcal{P} = \{\rho_{l,o}\}_{o \in O}$;
for *i = 1:O* **do**
  $o = \arg\max_o \mathcal{P}$;
  $f_{l,o} = \frac{n_{l,o} \tau_o C_o}{T_{l,o}^{comp}}$;
  $P_{l,o} = \frac{\sigma^2}{h_{l,o}} \left(2^{\left(\frac{B_o^{weights} r_{l,o}}{W\left(r_{l,o} T_{l,o}^{comm} - B_o^{weights}\right)} \right)} - 1 \right)$;
  $n_{l,o} = \phi_{l,o} \frac{C_o \sqrt{\rho_{l,o}^3}}{\tau_o \sum_o C_o \rho_{l,o}}$;
  if $n_{l,o} < n_{min}$ *or* $P_{l,o} > P_{max}$ *or* $f_{l,o} > f_{max}$ **then**
    $n_{l,o} = 0$, $f_{l,o} = 0$, $P_{l,o} = 0$;
    for *k = i+1:O* **do**
      $T_{k,o} = T_{k,o} + \frac{T_{l,o}}{O - i}$;
  $\mathcal{P} \leftarrow \mathcal{P} \setminus \rho_{l,o}$;

V. Simulation Results

A. Simulation Setup

We have conducted the simulation considering 15 learners and 3 orchestrators. The considered ML models are Convolutional Neural Networks (CNNs). The first two orchestrators have similar datasets and models, while the third orchestrator has a less complex model, where complexity refers to the depth of the CNN. All the simulation parameters are detailed in Table I. In the results, we run multiple simulations and show the mean values for the utility functions and parameters across all learners and orchestrators.

TABLE I
Simulation parameters

Time limit T_{max}	500s
Bandwidth $W_{l,o}$	1 MHz
Maximum transmission powers P_{max}	[30, 3, 0.3]mW
Distance range	$[5, 50]$ m
Maximum CPU frequencies f_{max}	[250, 175, 150]MHz
On chip capacitance μ	1×10^{-22}
Maximum number of iterations τ_{max}	6
Bit precision V_o^w	64 bits
Learner dataset size N_o	6000
Models complexity C_o	[7.5, 7.5, 5]$\times 10^5$
Number of weights S_o^w	[850, 850, 350]$\times 10^6$

B. Performance Evaluation

We first show the convergence of the PBTA algorithm in Fig 2. We can see that for all the orchestrators, the algorithm converges within 15 iterations. Moreover, the orchestrators

with more complex models achieved a higher utility value. In order to explain this behavior, we examine the learners' strategy for a whole training process with 10 global cycles in terms of task participation percentage to each orchestrator, the amount of allocated training data, the transmission power percentage, and the computational resources percentage, which are depicted in Figure 3. In Figure 3 (a), we can see that a higher percentage of learners preferred to participate with orchestrators that have more complex models. Similarly, in Figure 3 (b) and (d), it can be seen that the learners have participated with fewer data to the orchestrator with the less complex model, and allocated less computational resources. This is due to the fact that in participation with that model, learners do not attain higher revenue, and hence, it allocates less CPU frequency and data in order to consume less energy. Whereas with the complex models, learners allocate more resources since the gained revenue will be higher and cover the cost of the energy. However, we can observe in Figure 3 (c) that all the learners have similar strategies in terms of transmission power for all the orchestrators. In fact, unlike the other strategies, the transmission power does not depend on nor affect the revenue explicitly, and it heavily depends on the communication channel behavior and the allocated bandwidth from the orchestrator.

Afterward, to justify the choice of the hyperparameter θ_{comp} in the system, we study the learners' behavior as θ_{comp} changes, which represents the proportion of time for computation. Figure 4 depicts the average utility function value across the learners. We can notice that for low values of θ_{comp}, the utility function is at zero since the learners do not have enough time to participate in any task. As θ_{comp} increases, the utility function value keeps increasing until it reaches a nominal point at $\theta_{comp} \sim 0.76$ and then decreases sharply afterward. This shows that the optimal proportion of time allocated for computation is around 76%, which results in obtaining the highest utility value for the learners. In fact, if this proportion keeps increasing, the time allocated for communication keeps decreasing, which results in enforcing higher transmission power to upload the models on time, and eventually results in more energy consumption.

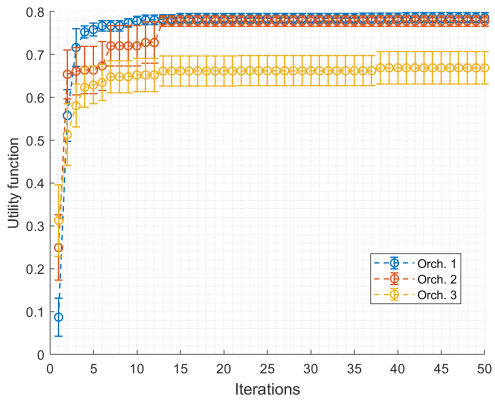

Fig. 2. Convergence behavior of the proposed PBTA algorithm

C. Performance Comparison

We compare the proposed PBTA algorithm with a random strategy as the baseline that selects random values which respect the constraints, and the proposed approach in [3] as the benchmark. The latter approach is similar in terms of modeling the SB game, where the orchestrator's utility is represented in terms of the accuracy function and the payment, and the learners' utility in terms of the revenue and energy consumption. However, unlike our approach, it only optimized for the number of data samples at the learner side and considered a fixed number of local training iterations, while our approach optimizes them along with the resources of each learner. In Figure 5, we show a statistical comparison in terms of the Cumulative Function Distribution (CDF) of the orchestrator and learner utility function value across 50 global cycles. In Figure 5 (a), the CDF of the average learners utility function is shown. We can see that the random strategy is the worst as its range is way below the other two with a median of ~ 0.28. As for the PBTA, it clearly outperforms the benchmark with a median of ~ 0.64, while the latter has a median of ~ 0.4. Similarly, in Figure 5 (b), the corresponding CDF of the average orchestrators utility function is shown. Our PBTA algorithm achieves better utility function values with a median of ~ 0.7 than the benchmark which achieves a median of ~ 0.61, and the random strategy is the worst with a median of ~ 0.28.

VI. Conclusion

In this work, we aimed to design a joint incentive mechanism and resource allocation for a multi-orchestrator MEL system. We modeled the whole learning process as a multi-round SB game, where each round represents a global cycle. Due to the intractability of the utility functions optimization, we proposed a heuristic algorithm, namely, the payment based time allocation (PBTA), where learners allot their time to each

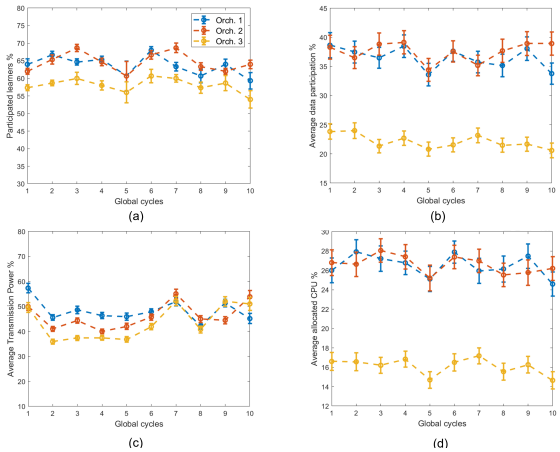

Fig. 3. The learners' strategy in terms of (a) task participation (b) data participation for a task (c) transmission power (d) CPU frequency

orchestrator based on its payment and then determine the optimal number of data samples and the assigned resources to each orchestrator based on the allocated time. Finally, we evaluated the performance of the algorithm in terms of convergence and learners' behavior and then compared our results to a recent state-of-the-art approach. The conducted simulations show that our proposed PBTA algorithm outperforms the other approach in terms of orchestrators and learners' utility functions.

ACKNOWLEDGEMENT

This research is supported by a grant from the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant number: ALLRP 549919-20. The work of Amr Mohamed was partially supported by NPRP grant # NPRP13S-0205-200265. The findings achieved herein are solely the responsibility of the authors.

REFERENCES

[1] Z. Zhou, H. Liao, B. Gu, K. M. S. Huq, S. Mumtaz, and J. Rodriguez, "Robust mobile crowd sensing: When deep learning meets edge computing," *IEEE network*, vol. 32, no. 4, pp. 54–60, 2018.

[2] U. Mohammad and S. Sorour, "Adaptive task allocation for mobile edge learning," in *2019 IEEE Wireless Communications and Networking Conference Workshop (WCNCW)*. IEEE, 2019, pp. 1–6.

[3] Y. Zhan, P. Li, Z. Qu, D. Zeng, and S. Guo, "A learning-based incentive mechanism for federated learning," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6360–6368, 2020.

[4] Y. Sarikaya and O. Ercetin, "Motivating workers in federated learning: A stackelberg game perspective," *IEEE Networking Letters*, vol. 2, no. 1, pp. 23–27, 2019.

[5] Y. Zhan and J. Zhang, "An incentive mechanism design for efficient edge learning by deep reinforcement learning approach," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 2489–2498.

[6] S. Feng, D. Niyato, P. Wang, D. I. Kim, and Y.-C. Liang, "Joint service pricing and cooperative relay communication for federated learning," in *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2019, pp. 815–820.

[7] R. Zeng, S. Zhang, J. Wang, and X. Chu, "Fmore: An incentive scheme of multi-dimensional auction for federated learning in mec," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2020, pp. 278–288.

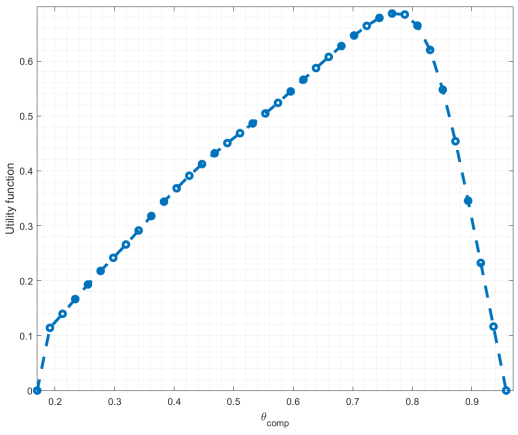

Fig. 4. The average learner utility function as the proportion of time for computation θ_{comp} increases

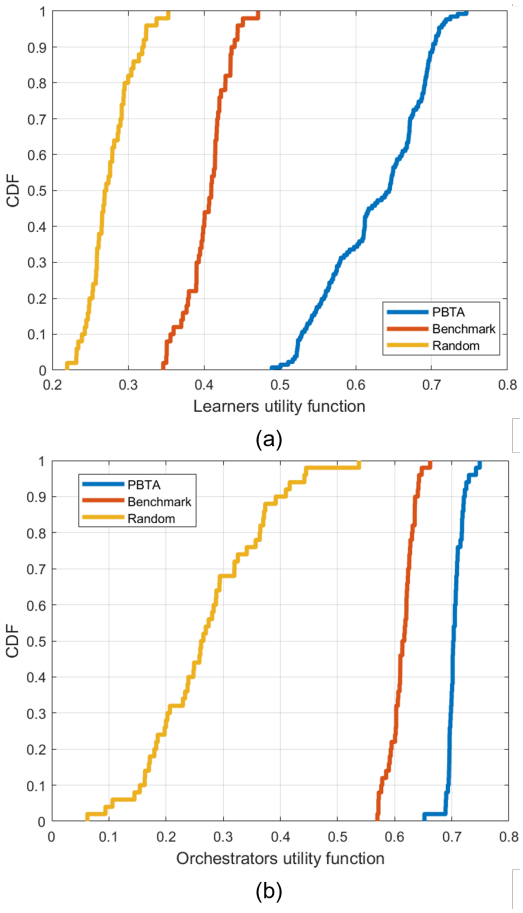

Fig. 5. The comparison in terms of the CDF for (a) learners utility (b) orchestrator utility

[8] T. H. T. Le, N. H. Tran, Y. K. Tun, M. N. Nguyen, S. R. Pandey, Z. Han, and C. S. Hong, "An incentive mechanism for federated learning in wireless cellular network: An auction approach," *IEEE Transactions on Wireless Communications*, 2021.

[9] J. S. Ng, W. Y. B. Lim, Z. Xiong, X. Cao, D. Niyato, C. Leung, and D. I. Kim, "A hierarchical incentive design toward motivating participation in coded federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 1, pp. 359–375, 2021.

[10] Y. Jiao, P. Wang, D. Niyato, B. Lin, and D. I. Kim, "Toward an automated auction framework for wireless federated learning services market," *IEEE Transactions on Mobile Computing*, vol. 20, no. 10, pp. 3034–3048, 2020.

[11] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, "Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 700–10 714, 2019.

[12] J. Kang, Z. Xiong, D. Niyato, H. Yu, Y.-C. Liang, and D. I. Kim, "Incentive design for efficient federated learning in mobile networks: A contract theory approach," in *2019 IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS)*. IEEE, 2019, pp. 1–5.

[13] D. Ye, R. Yu, M. Pan, and Z. Han, "Federated learning in vehicular edge computing: A selective model aggregation approach," *IEEE Access*, vol. 8, pp. 23 920–23 935, 2020.

[14] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[15] J. Mills, J. Hu, and G. Min, "Multi-task federated learning for personalised deep neural networks in edge computing," *IEEE Transactions on*

Parallel and Distributed Systems, vol. 33, no. 3, pp. 630–641, 2021.
[16] U. Mohammad, S. Sorour, and M. Hefeida, "Dynamic task allocation for mobile edge learning," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2021.
[17] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.
[18] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE communications surveys & tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.
[19] R. B. Myerson, *Game theory: analysis of conflict*. Harvard university press, 1997.
[20] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

APPENDIX A
LEMMA 1

Since we have the expression $f_{l,o}$ in terms of $n_{l,o}$ in (19), the utility function can be reformulated as follows:

$$U_l(n_{l,o}) = C_o\rho_{l,o} \times \tau_o \times n_{l,o} - \frac{\mu\tau_o^3 C_o^3 n_{l,o}^3}{(T_{l,o}^{comp})^2} - \frac{P_{l,o}B_o^{weights}}{r_{l,o}^{up}} \tag{24}$$

and the optimization problem in (16) becomes unconstrained single variable optimization. Consequently, to find the optimal value of $n_{l,o}$, we find the solution that satisfies the equation $\frac{\partial U_l}{\partial n_{l,o}} = 0$ as follows:

$$C_o\rho_{l,o}\tau_o - \frac{3\mu\tau_o^3 C_o^3 n_{l,o}^2}{(T_{l,o}^{comp})^2} = 0 \tag{25}$$

and by rearranging the terms we have:

$$n_{l,o}^2 = \frac{(T_{l,o}^{comp})^2\rho_{l,o}}{3\mu\tau_o^2 C_o^2} \tag{26}$$

$$n_{l,o} = \frac{T_{l,o}^{comp}\sqrt{\rho_{l,o}}}{\sqrt{3\mu}\tau_o C_o} \tag{27}$$

where we disregard the negative root since $n_{l,o}$ has to be positive. Since $T_{l,o}^{comp} = \frac{\theta^{comp}C_o\rho_{l,o}}{\sum_o C_o\rho_{l,o}} \times T_{max}$ we can express $n_{l,o}$ as follows:

$$n_{l,o} = \frac{\theta^{comp}T_{max}\rho_{l,o}\sqrt{\rho_{l,o}}}{\sqrt{3\mu}\tau_o\sum_o C_o\rho_{l,o}} = \frac{\phi_l\sqrt{\rho_{l,o}^3}}{\tau_o\sum_o C_o\rho_{l,o}} \tag{28}$$

with $\phi_l = \frac{\theta^{comp}T_{max}}{\sqrt{3\mu}}$. Since the amount of data cannot exceed the total dataset size N_o and cannot go below task requirements n_{min}, the optimal amount of training data is as follows:

$$n_{l,o} = \begin{cases} 0 & n_{l,o} < n_{min} \\ \frac{\phi_l\sqrt{\rho_{l,o}^3}}{\tau_o\sum_o C_o\rho_{l,o}} & n_{min} \le n_{l,o} < N_o \\ N_o & n_{l,o} \ge N_o \end{cases} \quad \blacksquare \tag{29}$$

APPENDIX B
LEMMA 2

First, we disregard the positive term ϕ_l since it is a constant. In order to prove the concavity of the term $n_{l,o}$, the hessian matrix $\mathbf{H}$ with respect to τ_o and $\rho_{l,o}$ has to be negative semi-definite, i.e., $\det(\mathbf{H}) \le 0$. We first find the partial second derivatives as follows:

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 U_{l,o}}{\partial\rho_{l,o}^2} & \frac{\partial^2 U_{l,o}}{\partial\rho_{l,o}\tau_o} \\ \frac{\partial^2 U_{l,o}}{\partial\rho_{l,o}\tau_o} & \frac{\partial^2 U_{l,o}}{\partial\tau_o^2} \end{bmatrix} \tag{30}$$

$$\frac{\partial^2 U_{l,o}}{\partial\rho_{l,o}^2} = \frac{2C_o^2\rho_{l,o}^3 + 3\rho_{l,o}(\sum_o C_o\rho_{l,o})^2 - 3C_o\rho_{l,o}^2(\sum_o C_o\rho_{l,o})}{\sqrt{\rho_{l,o}^3}\tau_o(\sum_o C_o\rho_{l,o})^3} - \frac{9}{4\sqrt{\rho_{l,o}}\tau_o\sum_o C_o\rho_{l,o}} \tag{31}$$

$$\frac{\partial^2\tau_o}{\partial\rho_{l,o}^2} = \frac{2\sqrt{\rho_{l,o}^3}}{\tau_o^3\sum_o C_o\rho_{l,o}} \tag{32}$$

$$\frac{\partial^2\tau_o}{\partial\tau_o\rho_{l,o}} = \frac{2C_o\sqrt{\rho_{l,o}^3} - 3\sqrt{\rho_{l,o}}\sum_o C_o\rho_{l,o}}{2\tau_o^2(\sum_o C_o\rho_{l,o})^2} \tag{33}$$

Therein,

$$\det(\mathbf{H}) = -\frac{3\rho_{l,o}((\sum_{j\neq o} C_j\rho_{l,j})^2 + 6C_o\rho_{l,o}\sum_{j\neq o} C_j\rho_{l,j})}{4(\tau_o\sum_o C_o\rho_{l,o})^4} + \frac{3\rho_{l,o}(C_o\rho_{l,o})^2}{4(\tau_o\sum_o C_o\rho_{l,o})^4} \tag{34}$$

since $\det(\mathbf{H})$ is always negative, the term $n_{l,o}$ is concave with respect to τ_o and $\rho_{l,o}$. As for the orchestrators' utility function, the accuracy function A_o is concave, and the concave function of a concave term is also concave [20]. The payment function for a single learner:

$$\Gamma_o(\rho_{l,o}) = \frac{\phi_l C_o\sqrt{\rho_{l,o}^5}}{\sum_o C_o\rho_{l,o}} \times c_l \tag{35}$$

is a single variable function and can be shown to be convex as follows:

$$\frac{d\Gamma_o}{d\rho_{l,o}} = \frac{\rho_{l,o}^4(3C_o\rho_{l,o} + 5\sum_{j\neq o} C_j\rho_{l,j})}{2\sqrt{\rho_{l,o}^5}(\sum_o C_o\rho_{l,o})^2} \tag{36}$$

$$\frac{d^2\Gamma_o}{d\rho_{l,o}^2} = \frac{\rho_{l,o}^3(3(C_o\rho_{l,o})^2 + 10C_o\rho_{l,o}\sum_{j\neq o} C_j\rho_{l,j})}{4\sqrt{\rho_{l,o}^5}(\sum_o C_o\rho_{l,o})^3} + \frac{15\rho_{l,o}^3(\sum_{j\neq o} C_j\rho_{l,j})^2)}{4\sqrt{\rho_{l,o}^5}(\sum_o C_o\rho_{l,o})^3} \tag{37}$$

and since $\frac{d^2\Gamma_o}{d\rho_{l,o}^2}$ is positive, the payment function is convex. Therefore, it follows that the utility function $U_o = A_o - \Gamma_o$ is also concave. $\blacksquare$