Multi-Orchestrator Mobile Edge Learning: Designing Energy-Efficient Task Allocation and Incentive Schemes

by

Mhd Saria Al lahham

A thesis submitted to the
School of Computing
in conformity with the requirements for
the degree of Master of Science

Queen's University
Kingston, Ontario, Canada
May 2022

Abstract

Mobile Edge Learning (MEL) is a decentralized collaborative learning paradigm that features distributed training of Machine Learning (ML) models over resource-constrained edge devices (e.g., Internet of Things (IoT) devices). MEL enables such devices to either learn a shared model without sharing data, or distribute the learning model along with the data to other IoT devices and utilize their available resources. In the former case, IoT devices (aka learners) need to be assigned an orchestrator to facilitate decentralized learning and models' aggregation from different learners. Whereas in the latter case, IoT devices act as orchestrators and look for learners with available resources to distribute the learning task and utilize their resources.

However, in MEL, the coexistence of multiple learning tasks with different datasets may arise, which is referred to as multi-orchestrator MEL. The heterogeneity in edge devices' capabilities will require the joint optimization of the learners-orchestrators association and task allocation. Moreover, the performance of each learning task deteriorates without the availability of sufficient training data or computing resources. Therefore, it is crucial to motivate the edge devices to become learners and offer their computing resources, and either offer their private data or receive the needed data from the orchestrator and participate in the training process of a learning task.

To this end, we aim to develop an energy-efficient framework for orchestrators-learners association and learning task allocation, in which each orchestrator gets associated with a group of learners with the same learning task based on their communication channel qualities and computational resources, and allocate the tasks accordingly. Afterward, we propose an incentive mechanism, where we formulate the orchestrators-learners interactions as a two-round Stackelberg game to motivate the participation of the learners. In the first round, the learners decide which learning task to get engaged in, and then in the second round, the training parameters and the amount of data for training are decided in case of participation such that their utility is maximized. Finally, numerical experiments have been conducted to evaluate the performance of the proposed energy-efficient framework and the proposed incentive mechanism.

Acknowledgments

First of all, I would like to dedicate my work to my supervisor Prof. Sameh Sorour, who, although no longer with us, continues to inspire me by his passion and dedication towards his work, teaching, spreading knowledge, and helping others. I also would like to thank Prof. Amr Mohamed for his endless support and guidance throughout my master's journey. I also would like to express my gratitude to Prof. Hossam Hassanein for taking care of me and guiding me towards the end of my master's degree. I also thank Prof. Aiman Airbad, and Prof. Mohsen Guizani for their guidance and support.

I thank my family for their support and encouragement, and express my sincere appreciation to my fiance, Amani, who has been with me since the start of my journey, as without the guidance of Allah and her support I would not have been able to achieve all of this.

Lastly, I would like to thank my dear friends Naram Mhaisen and Sherif Azmy for being my destination whenever I needed a scientific and academic discussion, Ali Farhat, Abdelrahman Ramadan, Ibrahim Amer, and Mahmoud Abdelhadi for the great company and the laughs that I am grateful for.

Statement of Originality

I hereby declare that this thesis titled "Multi-Orchestrator Mobile Edge Learning: Designing Energy-Efficient Task Allocation and Incentive Schemes" is my own work, and to the best of my knowledge, contains no materials previously published or written by another person.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview and Motivation

The growing availability of various computing resources and abundant data have pushed the learning algorithms for this data towards the network edge rather than the cloud. Indeed, centralized cloud-based learning paradigms suffer from the huge latency overhead, and such paradigms are impractical for many real-time edge-based applications (e.g., health monitoring and surveillance) [1]. This has led to the emergence of the Mobile Edge Learning (MEL) framework [2]. MEL is a framework that combines two originally decoupled areas: Mobile Edge Computing (MEC) and Machine Learning (ML), where it distributes and executes learning tasks (i.e., ML models' training) on wireless edge nodes such as IoT devices, while taking into consideration the heterogeneity in these devices' communication and computation capabilities. In fact, the recent advancements in MEL would provide a platform for developing and deploying edge artificial intelligence (AI) in 5G-and-Beyond systems and solving large-scale problems in our society ranging from autonomous driving to personalized healthcare [3].

There are two main components in the MEL framework: 1) *orchestrators*, which are responsible for distributing the learning tasks along with aggregating and synchronizing the updates of the tasks, and 2) *learners*, each of which is responsible for training a local ML model using its possessed or received data. MEL is a generalized framework of both Parallelized Learning (PL) and Federated Learning (FL) (see Fig. 1.1). The former is orchestrator-oriented, where the orchestrator has the whole dataset needed for learning, but it lacks the needed computational resources to do so. Hence, it distributes the ML model with the data across trusted learners to utilize their available resources. On the other hand, the latter is learner-oriented, where each learner has its private data, but the goal is to learn a global model across all the learners in a distributed manner governed by an orchestrator, without sharing

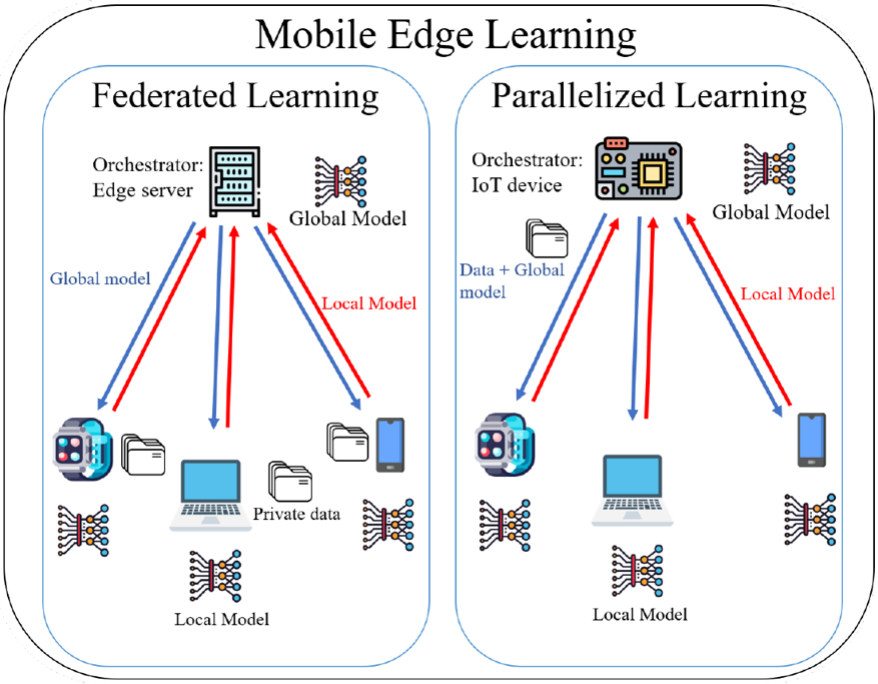

Figure 1.1: Federated Learning (left) and Parallelized Learning (right)

the private data of each learner. However, the heterogeneity of the learners' computing capacities causes the so-called "straggler's dilemma" issue, where the distributed learning process is throttled by the learner with the lowest capabilities [4]. Hence, the orchestrator in both cases needs to allocate the tasks according to the learners' capabilities. Task allocation in PL refers to the number of training data samples that will be sent along with the ML model to each learner, whereas in FL, it refers to the number of data samples from the private dataset that will be considered in the local training of each learner.

Multi-orchestrator MEL refers to the coexistence of multiple learning tasks with different datasets, each of which being governed by an orchestrator to facilitate the distributed training process. Examples of such environments are 1) multiple parallel FL jobs on different datasets stored at different groups of learners, or 2) multiple resource-constrained IoT devices parallelizing their learning tasks simultaneously on a set of nearby learners. Moreover, handling simultaneous learning tasks in the same edge environment with the heterogeneity of both learners' resources and channels quality represent a new challenge in the multi-task MEL setting, which was not addressed in previous works.

Nevertheless, the training performance deteriorates without the availability of sufficient training data or computing resources. Moreover, learners are reluctant to get engaged in the learning process without receiving benefits or compensation, either for offering their private data to train on, or for offering their computing resources and receive data from other incapable nodes to help in performing the learning task. In either case, the learning experience for the whole task increases with the amount and quality of data included in training [5]. Therefore, it is crucial to motivate edge

devices to become learners in an active and reliable manner, where they offer their computing resources and either offer their private data or receive the needed data from the orchestrator and participate in the training process of a learning task.

1.2 Objectives and Contributions

In this thesis, we first consider a multi-task MEL system to be administered simultaneously by multiple independent orchestrators. Examples of such environments are 1) multiple parallel FL jobs on different datasets stored at different groups of learners, or 2) multiple resource-constrained IoT devices parallelizing their learning tasks simultaneously on a set of nearby learners. Moreover, handling simultaneous learning tasks in the same edge environment with the heterogeneity of both learners' resources and channels quality represent a new challenge in multi-task MEL setting, which was not addressed in previous works. To this end, we propose energy-efficient learner-orchestrator association and task allocation techniques in this multi-task MEL system, while accounting for the learners' heterogeneity in terms of computation and communication capabilities. The contributions of the first part of the thesis can be summarized as follows:

1. First, we formulate a multi-objective optimization problem (MOP) for energy-efficient learner-orchestrator association and task allocation, that aims to minimize the total energy consumption in the system and maximize the learning accuracy at the orchestrators.

2. Being non-convex and NP-hard to solve, we propose an approach that employs an approximate solution to the relaxed and convexified formulated MOP.

3. To reduce the complexity of the optimization approach, we propose a set of lightweight heuristic algorithms that promote decentralization in the solution and utilize the solution of a simplified version of that formulated MOP. The performance of both proposed approaches is evaluated, analyzed and compared to a recent state-of-the-art technique in MEL.

Secondly, in multi-orchestrator MEL, each orchestrator has a different independent learning task. Accordingly, the learners get to choose which learning task to get engaged in (i.e., which orchestrator to associate with) such that their service costs are minimized and utility is maximized. To this end, we aim to design an incentive mechanism for the multi-orchestrator MEL system, where learners first decide the association, and then decide the amount of data that they will train on in the learning process based on the associated orchestrator's incentive, while the latter decides the incentive and the training parameters based on the learners' capabilities. The main contributions of the paper can be summarized as follows:

1. First, we formulate the learners-orchestrators interactions as a single 2-round Stackelberg game, where the associations are decided in the first round, and the training parameters and the incentives are decided in the second round.

2. In the first round, as the association problem turned out to be NP-hard, we propose a heuristic approach for the learners-orchestrators association. Whereas in the second round game, we prove the existence of a Nash equilibrium, where we derive the optimal strategies for both learners and orchestrators.

3. Finally, we carry out numerical results to show the performance of the proposed incentive mechanism while being compared to other techniques.

1.3 List of Publications

Journal Publications

1. Allahham, M. S., Sorour, S., Mohamed, A., Erbad, A., & Guizani, M. Energy-Efficient Multi-Orchestrator Mobile Edge Learning. IEEE Transaction on Green Communications and Networking (under review).

2. Allahham, M. S., Sorour, S., Mohamed, A., Erbad, A., & Guizani, M. Motivating Learners in Multi-Orchestrator Mobile Edge Learning: A Stackelberg Game Approach. IEEE Canadian Journal of Electrical and Computer Engineering (under review).

Conference Publications

1. Allahham, M. S., Sorour, S., Mohamed, A., Erbad, A., & Guizani, M. (2021, December). Energy-Efficient Device Assignment and Task Allocation in Multi-Orchestrator Mobile Edge Learning. In 2021 IEEE Global Communications Conference (GLOBECOM) (pp. 1-6). IEEE.

1.4 Organization of Thesis

The thesis is organized as follows: Chapter 2 introduces the concept of MEL and provides a comprehensive literature review of the design approaches of task allocation and incentive schemes. Chapter 3 presents the device association and the task allocation problem, where the system model is discussed first, then the proposed approaches, followed by the simulation results. Whereas in Chapter 4, the incentive mechanism formulation is introduced, followed by the proposed solution and the

simulation results. Finally, Chapter 5 concludes our work and outlines future work.

Chapter 2

Background and Literature Review

2.1 Mobile Edge Learning: Concepts and Design Approaches

In this section, We first present the learning preliminaries of MEL. After that, we proceed to present and discuss the design metrics and challenges, the proposed approaches and recent advances in MEL.

2.1.1 Preliminaries

Consider a dataset $\mathcal{D}$, consisting of $\{\boldsymbol{x}_i, y_i\}_i^{|D|} = 1$, where $|D|$ is the total number of samples in the dataset, $\boldsymbol{x}_i$ is the i^{th} feature vector (i.e., input sample), and y_i is the sample class or label. The objective function for the i^{th} sample will be denoted as $f_i(\boldsymbol{x}_i, y_i, \boldsymbol{w})$, and $f_i(\boldsymbol{w})$ for short, where $\boldsymbol{w}$ is the vector of the parameters of the ML or DL Model, which represents the classification error of the model for that sample. The overall objective function for ML training is defined as follows:

$$F(\boldsymbol{w}) = \frac{1}{|D|} \sum_{i=1}^{|D|} f_i(\boldsymbol{w}) \tag{2.1}$$

The aforementioned objective function can be convex for some ML models such as Support Vector Machines (SVMs), and not convex for other models such as Neural Networks. Most ML objective functions are usually optimized using a gradient descent optimizer, where the optimizer keeps updating the weights according to the direction of the gradient until convergence. The gradient descent update is as the following:

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - \eta \nabla F(\boldsymbol{w}) \tag{2.2}$$

where k is the update index and η is the step size hyperparameter.

However, in MEL, the dataset is distributed across L learners, where a learner l have a dataset $\mathcal{D}_l$, and $\cup_i^L \mathcal{D}_l = \mathcal{D}$. Each device's dataset can be either private and cannot be accessed by an orchestrator or any other device, or it can be offloaded from the orchestrator as part of the learning task. Since the data is not located at a centralized entity, the global objective function in (2.1) is not valid anymore and it cannot be computed, but it can be computed on each device according to its data only. Nevertheless, the authors in CITE defined the distributed training objective as the following:

$$F(\boldsymbol{w}) = \sum_{l=1}^{L} \frac{|D_l|}{|D|} F_n(\boldsymbol{w}_l) \tag{2.3}$$

where $F_l(\boldsymbol{w}_l)$ is the local objective function at learner l. Moreover, in order to update the learners' model parameters and maintain a unified global model across the learners, the authors have presented the *FedAvg* algorithm, which is the basic algorithm in any MEL system. In *FedAvg*, a centralized entity such as orchestrators first initialize a global model and sends it to the learners, and then the latter, in turn, run τ number of local iterations on its dataset and update the model. After the local iterations

are done, the models are sent back to the orchestrator, where it then aggregates the models and consolidates them into one global model, and sends back again the model consolidated model to the devices. This process is considered as one global cycle, and it is repeated G number of global cycles. The FedAvg algorithm was targeted for FL systems only, but in the PL case, the orchestrator distributes the data across the learners along with the global model parameters. The algorithm is detailed in Algorithm 1.

Algorithm 1: MEL Training Algorithm

At the orchestrator side:
initialization w_0
for *each global cycle* $g = 1, 2, ..., G$ **do**
  if *PL case* **then**
    $\mathcal{D}_L \leftarrow$ Split dataset $\mathcal{D}$ into L datasets where:
    $\mathcal{D}_L = \{D_1, D_2, ..., D_L\}$
  end
  else
    Set $\mathcal{D}_L = \{\varnothing, \varnothing, ..., \varnothing\}$
  end
  for *each learner* $l = 1, 2..., L$ **do**
    $\boldsymbol{w}_{t+1}^l \leftarrow \text{LearnerUpdate}(\boldsymbol{w}_t, D_l)$
  end
  $\boldsymbol{w}_{t+1} \leftarrow \sum_k \frac{|D_l|}{|D|} \boldsymbol{w}_{t+1}^l$
end
LearnerUpdate$(\boldsymbol{w}_t, D_l)$
if *FL case* **then**
  Set D_l to the local dataset
end
$\mathcal{B} \leftarrow$ (split D_l into batches with size B)
for *each local iteration* $\tau = 1, 2....$ **do**
  for *batch* $b \in \mathcal{B}$ **do**
    $\boldsymbol{w}_l \leftarrow \boldsymbol{w}_l - \eta \nabla F_l(\boldsymbol{w}_l)$
  end
end
return $\boldsymbol{w}_l$ to server

2.1.2 Literature Review

Since MEL is a distributed learning framework at the edge networks, there are multiple challenges that need to be addressed and aspects to be considered. On one hand, edge nodes (e.g., IoT devices) are heterogeneous in terms of computation and communication capabilities. Moreover, the learning task itself may be computationally exhaustive as the ML model can be huge, and therefore, it requires proper resource allocation from the learner's side and task allocation from the orchestrator's side. On the other hand, the quality of the learning experience (i.e., the learned model accuracy) depends on the amount of training data, and its distribution over the classes. In the case of FL, the problem of non-independent and identically distributed (Non-IID) data arises. In fact, the generated and collected data on each device is driven by the user or application behavior, and hence, each dataset on each device will not be representative of the whole data distribution. Moreover, the number of data points in each device may vary significantly, which is heavily dependent on the application or the users' behaviors. Moreover, the edge nodes are always on a budget (e.g., time or energy), and in synchronous settings, the learning process will always be throttled by the learner with the weakest capabilities, where this issue is referred to as straggler's dilemma. Indeed, this learner will limit the number of global cycles or local iterations in the learning process within its given budget, and hence, not granting the best possible learning experience.

As such, several works have addressed these challenges from different approaches. For example, The authors in [2] established an optimization paradigm to efficiently execute distributed learning tasks on wireless edge nodes (i.e., learners). The aim was to maximize the learning accuracy by maximizing the number of local iterations,

while the learners were limited to a preset duration. As the formulated problem turned out to be NP-hard, a solution is proposed that employs approximation and derived upper bounds using KKT conditions. A similar approach has been proposed in [6], but with considering power control and bandwidth allocation, where an optimization problem has been formulated to maximize the accuracy under time, bandwidth, and transmission powers constraints. With a different approach, Wang et al. in [7] have proposed a control algorithm that determines the best tradeoff between the number of local iterations and global cycles to minimize the loss function, under a given resource budget. Specifically, the authors have studied the relation analytically and defined the loss function as an upper bound to the divergence between the optimal learning model and the distributed learning model. The divergence refers to how much are the model parameters are deviating from the optimal parameters. As a result, minimizing the divergence upper bound will result in a model that is optimal or near-optimal in the worst cases. Afterward, an optimization problem has been formulated, where the aim is to minimize the divergence under a generic resources constraint. However, as the previous work has ignored the heterogeneity of the learners, this work has been extended in [8] to include also the problem of task allocation and consider the heterogeneous communication and computation capabilities of each learner. An optimization has been formulated where the objective is to minimize the divergence upper bounds under global time constraints, and the variables are the task allocation (the number of data samples to train on each device), the number of local iterations, and global cycles.

While previous works ignored the energy perspective, a similar approach has been proposed in [9], where an optimization problem has been formulated to maximize the

learning accuracy under energy and time constraint. From a different perspective, the authors in [10] explored a new direction of energy-efficient radio resource management including bandwidth allocation and scheduling. The proposed solution addresses the stragglers dilemma, as more bandwidth is allocated to devices with weaker channels or poorer computation capacities while granting them scheduling priority. A more general framework has been proposed in [11], where the authors address the tradeoff between the computation and the communication latency and energy which is determined by the desired accuracy level. More specifically, an optimization problem has been formulated to minimize the energy consumption and the total latency in the system, while considering different computation and communication capabilities for the learners. The optimization problem includes resources allocation in terms of learners' CPU frequency, power allocation, optimizing the number of local iterations and global cycles. As the optimization problem turned out to be NP-hard, the optimization problem has been broken down into three sub-problems, where first the CPU resource allocation is solved, then the uplink power control at the learners' side, and finally, the accuracy level for the learning task. Similar work has been done in [12], where a joint learning and communication problem is formulated as an optimization problem, and the goal is to minimize the total energy consumption of the system under a latency constraint. However, the proposed solution is an iterative algorithm, where at every step, closed-form solutions for time and bandwidth allocation, power control, computation frequency, and learning accuracy are derived.

Regarding the non-IID data challenge, the authors in [13] have proposed a strategy to improve the training on non-IID data, which is to create a small subset of the data which is globally shared between the devices. In that work, the authors

have shown that the original *FedAvg* algorithm can achieve much lower accuracy in the case of highly skewed non-IID data, where each device is trained on only one class. Moreover, the authors have shown that such reduction can be explained by the models' weights divergence, which can be quantified by the statistical distance between the distribution over classes on each device and the population distribution. The experiments show that if a small subset of data is shared globally, it can increase the accuracy by at least 30%. However, even though the shared data is relatively small, such an approach violates privacy in FL, which was the main motivation of FL in the first place. Another approach for addressing the non-IID challenge, the work in [14] has proposed an experience-driven control algorithm that selects learners to participate in each global cycle of federated learning to oppose the bias introduced by the non-IID nature of data and to speed up the convergence. In other words, a user selection scheme using Reinforcement Learning (RL) selects specific learners out of a pool of learners that should participate in each round of FL in an intelligent manner, leading to speeding up the convergence while maintaining high accuracy. With regard to the same challenge, the work in [15] has proposed clustered FL (CFL), which exploits the geometric properties of the FL objective function to group the devices into clusters, such that each cluster contains devices with similar data distributions. CFL is capable of distinguishing situations where a model can be trained from the devices' data from those in which this is not possible and only separates clients in the latter case. However, the clustering technique is achieved by multiple iterations of bi-partitioning, each requiring running the *FedAVg* algorithm until convergence.

Nevertheless, one important aspect has been ignored in the previous works, which is scalability. The question that comes to mind is, what if all the learners need to

learn the same global model in a very large-scale system? This question was the motivation for the new paradigm, the Hierarchical Federated Learning (HFL) which has been proposed in [16]. HFL is a client-edge-cloud hierarchy, where an edge server acts as an orchestrator for the clients connecting to that edge (i.e., learners), and the cloud acts as a global orchestrator for the edge servers. In fact, the edge servers which are the main facilitators enjoy more efficient communications with both clients and the cloud. However, since this work has disregarded the statistical challenge in FL, another work has been done to tackle this challenge in [17]. In the latter work, the authors aim to assign the clients to an edge and cluster them in an optimal way based on their data distributions. The presented results show that such an assignment can lead to faster convergence and higher accuracy. This work has been further extended in [18] to address the resource allocation while considering the heterogeneity of the learners in terms of the communication and computation capabilities.

There are also a plethora of works that try to improve the accuracy and the communication efficiency with different approaches such as client selection [19, 20, 21, 22, 23] and model parameters compression and quantization [24, 25, 26, 27, 28].

2.2 Incentive Mechanisms

To incentivize is to motivate or prompt individuals to do certain actions. Incentives can be either positive or negative. In the former case, incentives seek to motivate by a promised payoff or a reward. Whereas in the latter case, incentives aim to avoid unpleasant behaviors by penalizing the individuals. Incentive mechanisms are modeled by Game Theory. Game Theory is a mathematical framework that models the interactions between different players or agents. The goal of Game Theory is

to study and analyze the decisions of each agent to derive an optimal set of actions for each player. More specifically, we aim to find a set of solutions that satisfy all the players in the game, where no player is willing to change its behavior if someone else does, which is referred to as the Nash Equilibrium. Therein, we define the main components of Game Theory as follows:

- Players: aka agents. The individuals who make decisions in the considered environment.
- Strategy: aka policy. The set of possible actions that each agent can take in the environment.
- States: The information or the knowledge that each agent has when taking the decision.
- Utility: aka payoff/reward. It refers to what does each player gets in return for taking an action in a certain state.

The most commonly used approaches in Game Theory to design incentives are Stackelberg Game (SB) approach, or Auction Theory approach. In the SB approach, the incentive is modeled as follows: An agent (who is requesting the service) first publishes the service requirements with the incentive beforehand. Then the other agents (who provide the service) then take their decisions based on the payment and the requirements. Whereas in Auction Theory approach, an agent (who is providing the service) first publishes his service. The other agents (who request the service) bid to be served. The serving agent then decides who gets the service based on the bids. In what follows, we present and discuss previous works that exploited SB game or Auction Theory to design incentive mechanisms in MEC and MEL. Incentive

mechanisms in MEC are designed mainly for two applications: crowdsensing and computation offloading. In what follows, we discuss the incentive mechanisms in each application.

2.2.1 Incentive Mechanisms in Mobile Crowdsensing and Computation Offloading

Mobile crowdsensing requires a large number of participants (i.e., normal smartphone users) to sense the environment via various sensors (e.g., gyroscopes, accelerometers...etc). Employing such data enables the development of health care, traffic, and environment monitoring services [29]. The quality of these services relies heavily on the number of participants. As such, it is important to sufficiently motivate users to participate. Most incentives in crowdsensing fall into three categories: entertainment [30, 31, 32, 33, 34, 35], service [36, 37, 38, 39], and money [40, 41, 42, 43, 44]. Each incentive focus on some aspects of user need, such as: such as profit, enjoyment and comfort, fulfillment. In entertainment incentives, the crowd sensing task is turned into a sensing game, such that participants can offer computation or sensing capabilities of their mobile device when they play the game. By doing so, the participants feel enjoyable when they perform tasks and offer resources. As for service-based incentives, it is a manifestation of the mutual-benefit principle. Service consumers can also be providers, that is, if a user wants to be served, it also has to provide some kind of service. As for the monetary incentive category, the system pays the participant a certain amount of money to motivate them, such that devices' sensors or resources can be utilized.

While a plethora of works has approached the design of incentive mechanisms for

crowdsensing from different aspects, we mention a few which employed Auction theory or Stackelberg games. The authors in [45] have applied a two-stage Stackelberg game to analyze the level of participation of each mobile user and the optimal incentive mechanism using the backward induction algorithm. In order to motivate the formers, the incentive mechanism is designed such that it takes into consideration the social network effects from the mobile social domain. Whereas in [46], the work proposes a centralized framework where a platform provider can estimate participants' parameters very efficiently by sending and receiving a few messages. An optimization problem was formulated on the platform provider side as an integer non-linear program to optimize the payment and the resource usage with time and budget constraints. Since the optimization is NP-hard, heuristic algorithms were proposed that enable tradeoffs in the system such as optimality and scalability. A learning-based incentive mechanism was proposed in [47] to address the security issue in crowdsensing systems where it might be vulnerable to faked sensing attacks. More specifically, a Deep Reinforcement Learning (DRL) algorithm is employed to derive the optimal policy in terms of incentives against faked sensing attacks.

As for auction-based approaches, in [48] the authors propose a reverse auction-based incentive mechanism, which considers participants' potential contributions when employing new workers and retaining existing workers. The work aims to optimize the worker composition in the system while reducing the system cost. The potential contribution of a participant to the system is measured as the degree to which the user is joining or staying in the system. Whereas in [49], the authors propose a quality-driven incentive mechanism, where workers are paid off based on the quality

of sensed data instead of working time, as previous works adopted. Moreover, the incentive mechanism was applied in Wi-Fi fingerprint-based indoor localization system, where the system tries to incentivize the participants for fingerprint collection.

Computational offloading in MEC is a collaborative paradigm where an IoT device has a computational task and is incapable of performing it due to the lack of resources, so it sends this to nearby devices to perform the task. Collaborative offloading costs storage, computation, and communication resources from other devices. As such, service requesters have to incentivize other mobile devices to offer their resources to do the task. One of the earliest works to design an incentive mechanism for computation offloading was done in [50]. The authors formulated the interactions between cloud service operators (service requester) and edge servers as a Stackelberg game to maximize the utilities of both parties by obtaining optimal strategies for payment and offloading. Similar work was done in [51], wherein the formulated Stackelberg game, edge users act as followers, and edge cloud act as a leader. The proposed framework aims to maximize the revenue of the edge cloud while maximizing the edge users' utility under budget constraints. In line with the previous works' objective, similar approaches have been proposed in [52, 53, 54].

2.2.2 Previous Studies on Incentive Mechanisms in MEL

One of the earliest works of incentive mechanism in MEL is the one which was done in [55]. The incentive mechanism is modeled as an SB game, where the orchestrator aims to minimize the total training time given the learner's computation capacity. As a result, the orchestrator needs to incentivize learners to increase their allocated CPU power for the learning task. On the other side, the learners will determine the CPU

power according to its utility function such that their revenue is maximized, and the computation energy is minimized. A more generic approach has been proposed in [5], 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 leads to an increase in learning accuracy. As such, the orchestrator will try to motivate learners to include as much data as possible in the training process. Whereas at the learners' side, the utility is expressed in terms of revenue, communication, and computation energy consumption, where all the aforementioned terms depend on the number of data samples in the training. Consequently, the learners will determine the number of data samples such that their revenue is maximized, and the total energy consumption is minimized. A more sophisticated scenario has been investigated in [56], where a cooperative relay network is considered to support model transfer between the learners and the orchestrators to increase the energy efficiency at the edge network. However, the relay nodes have service costs, and during models' transmission, the channels will suffer from interference. As such, the learners need to determine the transmission power and the relay node in addition to the number of data samples. As for the orchestrator, the utility is also expressed in terms of the learning accuracy and the total payment to the learners, where the learning accuracy also depends on the number of training data samples. As for auction-based incentives, the authors in [57] developed a generic lightweight algorithm for encouraging learners' participation, taking into account the learner's multiple resources. The top K learners with the highest bids are then selected to be engaged in the learning process. Whereas in [58], the authors formulated the incentive mechanism with wireless

cellular network characteristics. The learners decide their bid and policy in terms of transmission power and computation capacity based on an allocated bandwidth and incentive. The learner selection is then formulated as a knapsack problem and solved via a greedy-based algorithm.

Chapter 3

Device Association and Task Allocation

3.1 System Model

In this chapter, we consider a multi-task multi-orchestrator MEL system as shown in Fig. (3.1). Each of these orchestrators can be considered as either 1) a governing node for learners that have private data for the same learning problem in the case of FL, or 2) an edge device that lacks the computational resources to execute the training of its learning problem due to its limited capabilities, or the computational resources are exhausted by another task in the case of PL. In this latter case, each of the orchestrators has to distribute its learning task and offload the data needed to the associated learners given their available resources. The learners are considered to be trusted nodes since orchestrators have to share private data with them. Without loss of generality, we focus in this work on the PL case only in the edge settings, as it can be considered as the general case with the fact that the orchestrators have to offload the data, which is not the case in FL. We will however point out how our formulations and solutions apply to FL whenever needed.

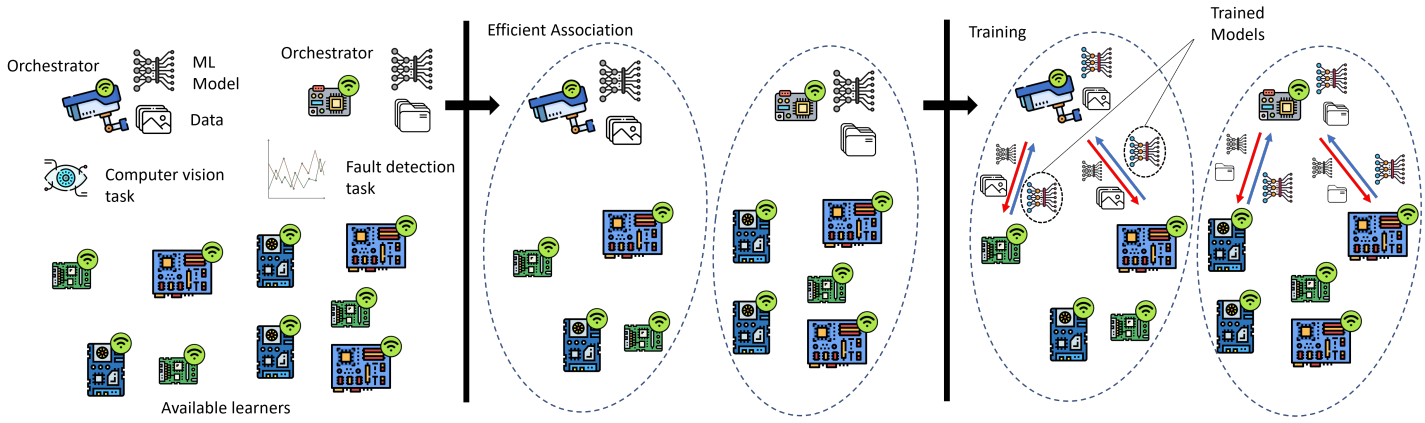

Figure 3.1: The considered system model

3.1.1 Learning Settings

We denote the set of orchestrators by $\mathcal{O}$, the set of learners by $\mathcal{L}$ and the set of learners that are associated with orchestrator o by $\mathcal{L}_o$. Each orchestrator $o \in \mathcal{O}$ has a dataset with N_o samples, and each data sample i can be represented by $\{x_i, y_i\}$, where x_i is the i^{th} feature vector (i.e., input data), and y_i is the i^{th} class or label. After the association with the learners is done, an orchestrator o sends the learning model parameters $\boldsymbol{w}_{l,o}$ and $n_{l,o} \times N_o$ data samples to learner l, where $n_{l,o}$ is the proportion of the data to send, and $\sum_{l=1}^{|\mathcal{L}_o|} n_{l,o} = 1$ since the orchestrators need to train on all the available data[1]. Each learner then performs τ_o local training iterations on its own or received data using Stochastic Gradient Descent (SGD) to minimize its loss function $f_l(\boldsymbol{w}_{l,o})$. Once done, the learners send back to the orchestrator the parameters of their locally trained models, where the latter aggregates these parameters by performing

[1]In the FL case, the constraint $\sum_l n_{l,o} = 1$ does not necessarily hold since each learner selects a proportion $n_{l,o}$ of its local dataset to perform the learning task. Instead, a constraint has to be added such that the selected samples from a learner represent the distribution of its local dataset.

weighted averaging as follows:

$$\boldsymbol{w}_o = \sum_{l \in \mathcal{L}_o} n_{l,o} \boldsymbol{w}_{l,o} \tag{3.1}$$

Each orchestrator then keeps sending back more data samples and the updated model to the learners, which repeat the same above process for G_o global cycles, until a stopping criteria is satisfied such as the exhaustion of a certain resource (e.g., energy, time) or the attainment of a given accuracy for the aggregated model. One can refer to [7] for more details about minimizing local loss functions and models aggregations.

3.1.2 Edge Settings

In this part, we introduce the communication and computation parameters of wireless edge learners. First, we define the number of bits that an orchestrator o sends to learner l as:

$$B_{l,o}^{\text{data}} = n_{l,o} N_o F_o \Gamma_o^d \tag{3.2}$$

$$B_o^{\text{weights}} = S_o^w \Gamma_o^w \tag{3.3}$$

where F_o is the feature vector length, S_o^w the total number of weights in the model, and Γ_o^d and Γ_o^w represent the bits/feature and bits/weight values, respectively. Therein, we can define the time needed for the orchestrator to send the data and the model weights by:

$$t_{l,o}^S = \frac{B_{l,o}^{\text{data}} + B_o^{\text{weights}}}{W \log_2(1 + \frac{h_{l,o} P_{l,o}}{\sigma^2})} \tag{3.4}$$

where W is the channel bandwidth, $P_{l,o}$ is the orchestrator's transmission power, σ^2 is the channel noise variance, and $h_{l,o}$ is the channel gain expressed as $h_{l,o} = d_{l,o}^{-\nu} g^2$

where $d_{l,o}$ is the distance between the orchestrator and the learner, ν is the path loss exponent, and g is the fading channel coefficient. Similarly, the time needed for a learner to send the updated model parameters is defined as follows:

$$t_{l,o}^{U} = \frac{B_o^{\text{weights}}}{W \log_2(1 + \frac{h_{l,o} P_{l,o}}{\sigma^2})} \tag{3.5}$$

Last, we define the time needed for a learner l to execute its allocated learning task by:

$$t_{l,o}^{C} = \frac{\tau_o n_{l,o} N_o C_o^w}{f_l} \tag{3.6}$$

where τ_o is the number of local iterations, C_o^w is the model computational complexity parameter, and f_l is the local processor frequency. Consequently, the total training time for a learner across G_o global cycles can be expressed as:

$$t_{l,o} = G_o(t_{l,o}^{S} + t_{l,o}^{U} + t_{l,o}^{C}) \tag{3.7}$$

We assume in the considered system a fixed bandwidth of W, fixed transmission power $P_{l,o} = P_{o,l} = P$ for all the nodes, and $h_{l,o} = h_{o,l}$ (i.e., channel reciprocity). Moreover, we assume that the orchestrator employs a controlled medium access control (MAC) protocol for the learners (e.g., the Reservation protocol).

Afterward, we define the energy consumption for both learners and orchestrators. Generally, the energy consumed for communications is the product of the transmission power with the transmission time and defined as $E = P \times t$. Therein, the orchestrator o energy consumption to send each cycles' data samples and the model weights can

be given by:

$$E_{l,o}^{S} = P_{l,o} t_{l,o}^{S} = \frac{P_{l,o}(B_{l,o}^{\text{data}} + B_{o}^{\text{weights}})}{W \log_2(1 + \frac{h_{l,o} P_{l,o}}{\sigma^2})} \tag{3.8}$$

Similarly, the learner's l energy consumption to send the updated model is given by:

$$E_{l,o}^{U} = P_{l,o} t_{l,o}^{U} = \frac{P_{l,o} B_{o}^{\text{weights}}}{W \log_2(1 + \frac{h_{l,o} P_{l,o}}{\sigma^2})} \tag{3.9}$$

As for the computation energy consumption at each learner l, we adopt the model in [59], which can be defined in our context as follows:

$$E_{l,o}^{C} = \mu \tau_o n_{l,o} N_o C_o^w f_l \tag{3.10}$$

where μ is the on-board chip capacitance. The total energy consumption for an orchestrator-learner pair during training can thus be expressed as:

$$E_{l,o} = G_o(E_{l,o}^{S} + E_{l,o}^{U} + E_{l,o}^{C}) \tag{3.11}$$

To simplify the notation in the reminder of the paper, we define the following time and energy coefficients:

$$A_{l,o}^{0} = \frac{2B_{o}^{\text{weights}}}{W \log_2(1+\frac{h_{l,o} P_{l,o}}{\sigma^2})}, \quad \zeta_{l,o}^{0} = P_{l,o} A_{l,o}^{0}$$

$$A_{l,o}^{1} = \frac{N_o F_o \Gamma_o^d}{W \log_2(1+\frac{h_{l,o} P_{l,o}}{\sigma^2})}, \quad \zeta_{l,o}^{1} = P_{l,o} A_{l,o}^{1}$$

$$A^2_{l,o} = \frac{N_o C^w_o}{f_l}, \qquad \zeta^2_{l,o} = \mu C^w_o f_l$$

such that the training time and energy consumption can be re-expressed respectively as:

$$t_{l,o} = G_o(A^2_{l,o}\tau_o n_{l,o} + A^1_{l,o} n_{l,o} + A^0_{l,o}) \tag{3.12}$$

$$E_{l,o} = G_o(\zeta^2_{l,o}\tau_o n_{l,o} + \zeta^1_{l,o} n_{l,o} + \zeta^0_{l,o}) \tag{3.13}$$

3.2 Problem Formulation

In this section, we first present the distributed learning objective formulation, then present the whole formulation in terms of energy consumption and learning objective in our MEL settings.

3.2.1 Learning Objective Formulation

For a given learners-orchestrator association, aligned with the previous literature [8], we first consider the model presented in [7] to define the learning objective in the resource constrained system:

$$\begin{aligned} &\min_{\tau_o, G_o} . \frac{1}{|\mathcal{O}|} \sum_{o \in \mathcal{O}} F_o(\boldsymbol{w}_o) \\ s.t. \quad & \\ & G_o(A^2_{l,o}\tau_o n_{l,o} + A^1_{l,o} n_{l,o} + A^0_{l,o}) \leq T_{\max}, \ \ \forall\, l, o \end{aligned} \tag{3.14}$$

where $F_o(.)$ is the global loss function for the learning task of an orchestrator o, and $T_{\max}$ is the maximum allowed training time for the whole learning process. It is generally impossible to find an exact analytical solution for the problem presented in (14)

that relates the optimization variables τ_o, G_o with $F_o(\boldsymbol{w}_o)$. Thus, the objective is typically re-formulated as a function of the distributed learning convergence bounds over the edge network. The convergence bounds in the formulation represents how much the trained model in the distributed learning is deviating from the optimal model. Interestingly, it has been shown that the convergence bound mainly depends on the optimization variables, namely, τ_o and G_o. The convergence bound for one learning task of one orchestrator has been derived in [7], and has been used to reformulate the learning problem objective. For the sake of brevity, we will only show the important findings that will be included in our main formulation. Similar to [7], we assume the following about the loss function at each learner l:

1. $F_l(\boldsymbol{w})$ is convex.
2. $F_l(\boldsymbol{w})$ is β-smooth, i.e., $||\nabla F_l(\boldsymbol{w}) - \nabla F_l(\boldsymbol{w}')|| \leq \beta_l ||\boldsymbol{w} - \boldsymbol{w}'||$ for any $\boldsymbol{w}'$.
3. The divergence between the gradients of the local loss and the aggregated loss function has a maximum of δ_l such that $||\nabla F_l(\boldsymbol{w}) - \nabla F(\boldsymbol{w})|| \leq \delta_l$

An auxiliary global model with weights $\boldsymbol{v}_o$ can be then defined, where the auxiliary model represents the centralized training model, which is considered as the optimal training model in the distributed training case. At each global cycle, the model will be updated by SGD as follows:

$$\boldsymbol{v}_o[g_o] = \boldsymbol{v}_o[g_o - 1] - \eta_o \nabla F_o(\boldsymbol{v}_o[g_o - 1]) \tag{3.15}$$

where η_o is the learning rate and g_o is the index of the global cycles. The difference between the distributed learning weights and the centralized learning at each global

update was shown to be upper bounded such that $||\boldsymbol{w}_o[g_o] - \boldsymbol{v}_o[g_o]|| \leq H_o(\tau_o)$, where:

$$H_o(\tau_o) = \frac{\delta_o}{\beta_o}\left[(\eta_o\beta_o + 1)^{\tau_o} - \eta_o\delta_o\tau_o\right] \tag{3.16}$$

where δ_o can be estimated at each global cycle by $\delta_o = \sum_{l\in\mathcal{L}_o} n_{l,o}\delta_l$, and β_o can be estimated by $\beta_o = \sum_{l\in\mathcal{L}_o} n_{l,o}\beta_l$, where β_l is given as:

$$\beta_l = \frac{||\nabla F_l(\boldsymbol{w}_o) - \nabla F_l(\boldsymbol{w}_l)||}{||\boldsymbol{w}_o - \boldsymbol{w}_l||} \tag{3.17}$$

As a result, the aim of the reformulated objective is to minimize the difference between optimal loss function $F_o(\boldsymbol{w}_o^*)$ (i.e., the centralized loss function $F_o(\boldsymbol{v}_o)$) and the distributed global loss function after G_o global cycles. This difference was shown to be upper bounded as follows:

$$F_o(\boldsymbol{w}_o) - F_o(\boldsymbol{w}_o^*) \leq \frac{1}{G_o\tau_o\left[\eta_o(1 - \frac{\beta_o\eta_o}{2}) - \phi\frac{H_o(\tau_o)}{\tau_o}\right]} \tag{3.18}$$

where ϕ is a control parameter. Moreover, the following conditions on the learning rate have to be satisfied to guarantee the convergence:

1. $\eta_o\beta_o \leq 1$
2. $\eta_o(1 - \frac{\beta_o\eta_o}{2}) - \phi\frac{H_o(\tau_o)}{\tau_o} > 0$

Even though the aforementioned convergence bound was shown to be convex when $\tau_o \in [1, \tau_{\max}]$, where $\tau_{\max}$ is the maximum allowed number of local iterations, minimizing over this bound will not have an exact analytical solution. Thus, to enable some solution analysis, we propose to approximate this bound with a simpler convex

expression instead. Since the parameters G_o and τ_o are the only active variables in the expression, and the other variables ϕ, η_o, δ_o and β_o are either fixed or empirically estimated during training, we can thus define the approximation function as:

$$U_o = \frac{c1}{G_o \tau_o^{c2}} \tag{3.19}$$

where $c1$ and $c2$ are the approximation parameters. Such approximation can be done with log transformation and Linear Regression [60]. Consequently, the learning objective is no longer a function of the number of local and global cycles only, but also preserves the convexity of the upper bound with the fact that $\tau_o, G_o \geq 1$. Furthermore, it can be noticed that, as we increase these two parameters, the convergence bound and its approximation in (19) is minimized, which leads to a better learning experience. Finally, although this approximation assumes the loss function is convex, our simulation results show that it also works well in practice for non-convex models (i.e., neural networks).

3.2.2 Multi-Objective Optimization Formulation

Recall that, in this work, we aim to associate each orchestrator o with a set of learners, and to optimize for each orchestrator with its associated learners set $\mathcal{L}_o$ the task allocation (i.e., $n_{l,o}$, the proportion of the data to be sent to each learner) along with the number of local iterations τ_o, and the number of global cycles G_o, such that both the energy consumption in the system and the average global loss functions at all orchestrators are minimized. To formulate this joint problem, we define the MOP as follows:

$$\mathbf{P1}: \quad \min_{\lambda_{l,o}, G_o, \tau_o, n_{l,o}} \alpha \sum_{l,o} \lambda_{l,o} E_{l,o} + (1-\alpha) \sum_{o} U_o \tag{3.20a}$$

$$s.t. \sum_{o} \lambda_{l,o} t_{l,o} \leq T_{\max}, \ \forall \ l \in \mathcal{L} \tag{3.20b}$$

$$\sum_{o} \lambda_{l,o} = 1, \ \forall \ l \in \mathcal{L} \tag{3.20c}$$

$$\sum_{l \in \mathcal{L}_o} n_{l,o} = 1, \ \forall \ o \in \mathcal{O} \tag{3.20d}$$

$$1 \leq \tau_o \leq \tau_{\max}, \ \forall \ o \in \mathcal{O} \tag{3.20e}$$

$$\lambda_{l,o} \in \{0,1\}, \ n_{l,o} \in [0,1], \ \forall \ l \in \mathcal{L}, \ \ o \in \mathcal{O} \tag{3.20f}$$

$$G_o, \tau_o \in \mathbb{Z}^+ \tag{3.20g}$$

where $\lambda_{l,o}$ is the association variable between an orchestrator o and a learner l, and α is a weighting coefficient that determines the importance of each objective with respect to the other. Constraint (20b) ensures that each learner does not exceed the global time limit for the whole training process, while constraint (20c) guarantees a learner can not associate with more than one orchestrator. Constraint (20d) conveys that the orchestrator offloads the whole dataset to its associated learners, and constraint (20e) guarantees that τ_o stays in the range where the original convergence bound is convex. Note that the energy and loss objectives in (20a) are averaged over all the learners and orchestrators and normalized between 0-1 by dividing by their maximum values $E_{\max}$ and $U_{\max}$, respectively, in order to enable a fair trade-off between the objectives.

It is readily obvious that the MOP **P1** is a Mixed Non-Linear Integer Program (MNLIP), due to the multiplication and division of the variables in the objective

function and the constraints. MNLIP's are known to be non-convex and NP-hard to solve [61]. Usually in such cases, a common approach is to relax the integer variables and consider it as a Geometric Program (GP), but this is not possible in our problem due to the existence of constraints (20c) and (20d). In fact, with the relaxation of the integer variables, the program presented in (20) is an uncommon special case of GP's and known as Signomial Programs (SP) [61]. Generally, SPs are not convex, but several works have proposed algorithms with different approaches to obtain a global solution for them through convexifications and successive solving of sub-optimization problems [62, 63, 64]. In this work, we adopt the presented approach in [64] and present some analysis from [62], where we convexify the problem and approximate the non-convex constraints by linear functions, and successively solve relaxed sub-optimization problems until we obtain a global solution. However, such approach is centralized and requires all the information about the environment, the orchestrators and the learners to be available at a single entity, which makes it not practical. Hence, we further present partially decentralized and fully decentralized light-weight heuristic algorithms that reduce the complexity of the solution.

3.3 Solution Approaches

In this section, we first present the centralized approach, where we solve the convexified and relaxed version of the presented MOP **P1** in (20). Afterward, we present the partially decentralized and fully decentralized approaches to simplify and solve the complex MOP.

3.3.1 Centralized Solution using Convex Relaxation

In order to convexify the problem, we first relax all other integer variables, namely, $\lambda_{l,o}, \tau_o$ and G_o, which will be later floored after finding the solution of the simplified convex problem. Since integer variable relaxation expands the feasible region, some undesirable values for the association variables might appear in the solution (e.g., $\lambda_{l,o} = \frac{1}{|\mathcal{O}|}$). Hence, we add the following constraint:

$$\sum_{i=1}^{|\mathcal{O}|-1} \sum_{j=i+1}^{|\mathcal{O}|} \lambda_{l,i}\lambda_{l,j} \leq \epsilon, \quad \forall l \in \mathcal{L} \tag{3.21}$$

where ϵ is a very small number, and $\epsilon > 0$. Along with constraint (20c), this constraint ensures that each learner can only associate with one orchestrator, where its association variable has higher value, and enforces the values of other orchestrators' association variables' to be close to 0.

Afterward, we perform an exponential variable transformation for all the variables such that:

$$x = \exp(\bar{x}) \tag{3.22}$$

where x can be either $\lambda_{l,o}, n_{l,o}, \tau_o$ or G_o. Such transformation will make the multi-objective function in (20a) as sum of exponential terms, which is known to be convex.

Consequently, the MOP in **P1** can be reformulated as:

$$\mathbf{P2}: \quad \min.\ \alpha \sum_{l,o} \sum_{k=0}^{2} \zeta_{l,o}^{k} \exp(X_k) + (1-\alpha) \sum_{o} c1 \exp(-c2\ \bar{\tau}_o - \bar{G}_o) \tag{3.23a}$$

$$s.t. \sum_{o} \sum_{k=0}^{2} A_{l,o}^{k} \exp(X_k) \leq T_{\max},\ \forall\ l \in \mathcal{L} \tag{3.23b}$$

$$\sum_{o} \exp(\bar{\lambda}_{l,o}) - 1 \leq 0,\ \forall\ l \in \mathcal{L} \tag{3.23c}$$

$$1 - \sum_{o} \exp(\bar{\lambda}_{l,o}) \leq 0,\ \forall\ l \in \mathcal{L} \tag{3.23d}$$

$$\sum_{i=1}^{|\mathcal{O}|-1} \sum_{j=i+1}^{|\mathcal{O}|} \exp(\lambda_{l,i} + \lambda_{l,j}) \leq \epsilon,\ \forall\ l \in \mathcal{L} \tag{3.23e}$$

$$\sum_{l \in \mathcal{L}_o} \exp(\bar{n}_{l,o}) - 1 \leq 0,\ \forall\ o \in \mathcal{O} \tag{3.23f}$$

$$1 - \sum_{l \in \mathcal{L}_o} \exp(\bar{n}_{l,o}) \leq 0,\ \forall\ o \in \mathcal{O} \tag{3.23g}$$

$$\bar{\lambda}_{l,o}, \bar{n}_{l,o}, \bar{\tau}_o, \bar{G}_o \in \mathcal{D} \tag{3.23h}$$

where $X_0 = \bar{\lambda}_{l,o} + \bar{G}_o$, $X_1 = \bar{\lambda}_{l,o} + \bar{G}_o + \bar{n}_{l,o}$, $X_2 = \bar{\lambda}_{l,o} + \bar{G}_o + \bar{n}_{l,o} + \bar{\tau}_o$, and $\mathcal{D}$ is the new domain of the MOP after the reformulation. This new domain can be defined by applying the transformation on the upper and lower bounds for each variable in constraints (20e)-(20g). It can be noticed that the reformulated MOP **P2** is convex in the objective and the constraints, except for constraints (23d) and (23g), where there exist concave terms. Hence, by utilizing the following linear function:

$$L(x) = \frac{x_{\max} e^{x_{min}} - x_{min} e^{x_{\max}}}{x_{\max} - x_{min}} + \frac{e^{x_{\max}} - e^{x_{min}}}{x_{\max} - x_{min}} x \tag{3.24}$$

where $x_{min}, x_{\max}$ represent the bounds for the variables $\lambda_{l,o}$ and $n_{l,o}$ in **P1** after the integer relaxation. We relax the MOP **P2** by underestimating each of the exponential terms in constraints (23d) and (23g) such that the constraints can be rewritten as

affine functions as follows:

$$1 - \sum_{o} L(\bar{\lambda}_{l,o}) \leq 0 \tag{3.25a}$$

$$1 - \sum_{l \in \mathcal{L}_o} L(\bar{n}_{l,o}) \leq 0 \tag{3.25b}$$

In fact, the linear underestimator represents a lower bound on the concave terms in (23d) and (23g), and the smaller the difference between the underestimation and the concave terms the closer the solution of the relaxed MOP will be to the solution of the MOP in (23). By making use of the analysis shown in [62], we can assess the quality of this lower bounding by examining the tightness of the underestimation of every concave term with the linear function (24) inside an interval $[x_{min}, x_{\max}]$.

Lemma 1. *Given the separation function $\Delta(x)$ as the difference between the concave term and its underestimator, $\Delta(x)$ is concave in x and its maximum can be given by:*

$$\Delta_{max} = e^{x_{min}} \left(1 - Z + Z \log(Z)\right) \tag{3.26}$$

where

$$Z = \frac{e^{\vartheta} - 1}{\vartheta}, \ \vartheta = x_{max} - x_{min} \tag{3.27}$$

Proof. The proof is detailed in Appendix A. We follow the same presented procedure in [62] with a slight modification.

It can be noticed that as the interval ϑ goes to zero, Z approaches one, and hence the maximum separation goes to zero. Furthermore, the rate at which $\Delta_{\max}$ goes to

zero can be examined using the Taylor series expansion of (26)[2] as follows:

$$\frac{\Delta_{\max}}{e^{x_{min}}} = \frac{\vartheta^2}{8} + \frac{\vartheta^3}{16} + \frac{11\vartheta^4}{576} + \frac{5\vartheta^5}{1152} + O(\vartheta^6) \tag{3.28}$$

Taking into consideration the first term only, it can be deduced that the rate at which $\Delta_{\max}$ goes to zero is:

$$\Delta_{\max} \approx O(\vartheta^2), \quad as \quad \vartheta \to 0. \tag{3.29}$$

We can conclude that if an effective relaxation is desired, a sufficiently small difference between the lower and the upper bounds is required.

After the convexification of **P2**, a Branch and Bound (BnB) algorithm is employed as in [64]. This approach solves a sequence of convex sub-problems of **P2** over partitioned subsets of $\mathcal{D}$ in order to obtain a global optimum solution. The BnB approach consists of k stages, where in each stage the set $\mathcal{D}^k$ is partitioned into subsets, each concerned with a node in the BnB-tree, and each node is associated with a sub-problem of **P2** in each subset. In each stage, the feasibility of each sub-problem in each node is checked and solved via interior point methods to obtain a lower bound on the optimal value of **P2**. Subsets that obtain a better lower bound than the previous stage are then partitioned again, each with a new node. This process is repeated until convergence, or the maximum number of stages is reached. Interested readers can refer to [64] for the full algorithm details and proof of convergence. Such BnB approaches for solving SPs are already available in optimization solvers such as GPkit [65]. Nevertheless, it is important to note that, while obtained solutions from the aforementioned approach can be optimal for the non-convex **P2**, but it might

[2]Taylor series expansion of (26) can be either derived by starting with $\frac{e^x-1}{x} = \sum_{n=0}^{\infty} \frac{x^n}{(n+1)!}$ and finding the derivatives of the other terms, or by using software such as MATLAB.

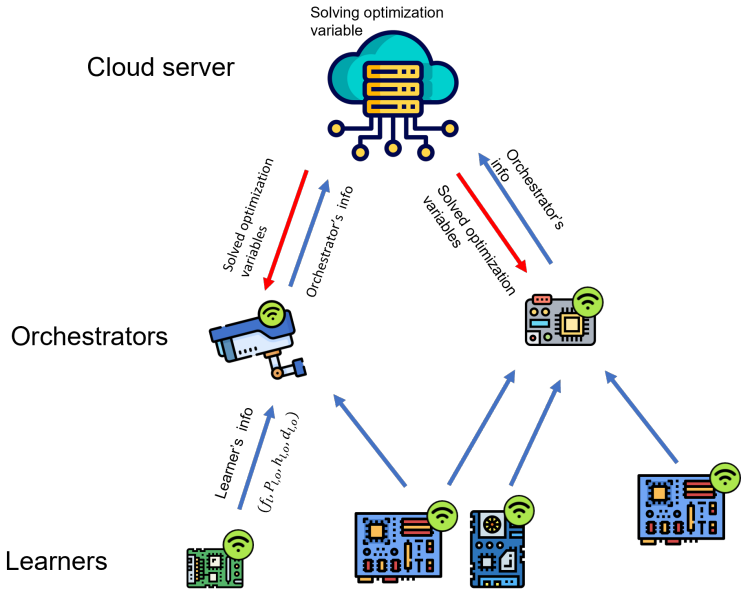

Figure 3.2: Optimization in MEL system

not be the case for **P1** due to the integer variable relaxation. Last but not least, the optimization algorithm is too computationally expensive to be performed at an IoT node, and it is impractical for an IoT node to have information about the whole system. Thus, as demonstrated in Fig 3.2, the optimization problem has to be solved at a cloud or an edge server, where orchestrators can gather their information about the system and the available learners and send it to the server to solve. Then the latter return the solution of the optimization problem to the orchestrators to start the training process.

3.3.2 Partially Decentralized Heuristics

In this work, we propose partially decentralized heuristics, where orchestrators need to cooperate and communicate by sharing the information (i.e., dataset size, channel qualities with learners...etc.) to realize the association with the learners. Once the

association is done, each orchestrator can optimize the task allocation, the number of local training iterations for each associated learner, and the number of the global cycles.

The Associate-Allocate-Train Decomposition Approach

First, we propose the Associate-Allocate-Train (AAT) algorithm, where the MOP **P1** is broken down into three simpler sub-problems as follows:

$$\begin{aligned} \textbf{SP1}: \quad & \min_{\lambda_{l,o}} \quad \sum_{l,o} \lambda_{l,o} E_{l,o} \\ & s.t. \quad \text{(20b)}, \text{(20c)}, \ \lambda_{l,o} \in \{0,1\} \end{aligned} \tag{3.30}$$

$$\begin{aligned} \textbf{SP2}: \quad & \min_{n_{l,o}} \quad \sum_{l \in \mathcal{L}_o} n_{l,o} G_o (\zeta^2_{l,o} \tau_o + \zeta^1_{l,o}) \\ & s.t. \quad \text{(20b)}, \text{(20d)}, \ n_{l,o} \in [0,1] \end{aligned} \tag{3.31}$$

$$\begin{aligned} \textbf{SP3}: \quad & \min_{\tau_o, G_o} \quad \alpha \sum_{l \in \mathcal{L}_o} E_{l,o} + (1-\alpha) \frac{c1}{G_o \tau_o^{c2}} \\ & s.t. \quad \text{(20b)}, \text{(20e)}, \text{(20g)} \end{aligned} \tag{3.32}$$

This decomposition is driven by the fact that both task allocation and training are dependent on the association. Hence, in the first sub-problem **SP1**, the orchestrators assume equal task allocation for all the learners (i.e., $n_{l,o} = \frac{1}{|\mathcal{L}|}$) and a fixed number of local iteration and global cycles so that the energy consumption for each association is known. **SP1** then optimizes the associations such that the total energy consumption is minimized. One can clearly see that **SP1** is a binary integer Linear Program (LP),

and efficient methods to obtain global solutions to such programs already exist in the literature [66, 67]. After obtaining the association variables, the task allocation sub-problem **SP2** can be solved. In fact, **SP2** is a simple LP that can be solved efficiently by each orchestrator to determine the task size $n_{l,o}$ for each learner. Finally, each orchestrator is left with the training sub-problem **SP3** to solve, which determines the number of local iterations and global cycles and also controls the energy-accuracy trade-off. Since **SP3** contains only two integer variables, we can employ exhaustive search to find the optimal values for τ_o and G_o. However, to achieve a faster search, we opt to find an optimal upper bound on both variables in order to reduce the search space. First, within a group of associated learners, we express l^* as the learner index with the maximum training time in that group such that $l^* = \arg \underset{l \in \mathcal{L}_o}{textmax} \ t_{l,o}$.

Lemma 2. *For $c2 = 1$ and $G_o \in [1, \frac{1}{\xi})$, the optimal upper bounds for G_o and τ_o can be given by:*

$$G_o^{max^*} = \left\lfloor \frac{1 - \sqrt{\frac{\xi a \theta^2}{b\xi - \theta c}}}{\xi} \right\rfloor \tag{3.33}$$

$$\tau_o^{max^*} = min\left(\left\lfloor \frac{1 - \xi G_o^{max^*}}{\theta G_o^{max^*}} \right\rfloor, \tau_{max} \right) \tag{3.34}$$

where

$$a = \frac{(1-\alpha)c1}{U_{max}}, \quad b = \frac{\alpha \sum_l \zeta^2_{l^*,o} n_{l^*,o}}{E_{max}|\mathcal{L}_o|}$$

$$c = \frac{\alpha \sum_l \left(\zeta^1_{l^*,o} n_{l^*,o} + \zeta^1_{l^*,o}\right)}{E_{max}|\mathcal{L}_o|}, \quad \theta = \frac{A^2_{l^*,o} n_{l^*,o}}{T_{max}}$$

$$\xi = \frac{(A^1_{l^*,o} n_{l^*,o} + A^0_{l^*,o})}{T_{max}}$$

Proof. The proof is detailed in Appendix B.

Lemma 2 shows that the maximum number of global cycles and the maximum local iterations are both dependent on the learner with the least capabilities, which thus takes the longest training time. Moreover, in practice, the regression parameter $c2$ depends on the learning parameters (i.e., β_o and η_o). We can thus set empirical upper bounds on these values which set the parameter $c2$ to 1. As a result, the optimal values for the number of local iterations and global cycles can be found by searching over the intervals $[1, \tau_o^{\max}]$ and $[1, G_o^{\max}]$ for τ_o and G_o, respectively.

Nevertheless, the two sub-problems **SP2** and **SP3** in the above heuristics are coupled together as the number of iterations and global cycles are optimized based on how much data each learner received, and vice versa, such that the training time does not exceed the limit. Moreover, solving each sub-problem separately might result in inefficient task allocation and poor choices for the number of local iterations and global cycles. As such, we propose an iterative procedure for jointly optimizing **SP2** and **SP3** by repeatedly alternating the minimization over the task allocation variables on one side, and the number of local iterations and global cycles on the other side. This procedure is as follows: 1) Initialize values for τ_o and G_o, solve the task allocation LP in **SP2** to obtain the task size for each learner; 2) Determine the optimal values for G_o and τ_o according to **SP3** by searching over the intervals that are defined by (33) and (34); 3) Repeatedly alternate the optimization between the two sub-problems until convergence (i.e., until the objective value of **P1** converges). The full details of the AAT algorithm are provided in Algorithm 1. Since the first

Algorithm 2: Assign-Allocate-Train (AAT) heuristic

Initialize: α, $T_{\max}, \tau_{\max}$, τ_o, G_o
At each orchestrator o **:**
Assume equal task allocation for all the learners $n_{l,o} = \frac{1}{|\mathcal{L}|}$;
Acquire other orchestrators information;
Solve the association problem (30);
Obtain the set of associated learners $\mathcal{L}_o$ and inform them;
while *no convergence* **do**
  Solve the task allocation problem in (31);
  Obtain the optimal upper bounds $G_o^{\max^*}$ and $\tau_o^{\max^*}$ according to (33) and (34);
  Perform exhaustive search to solve (32) and find the optimal number of iterations and global cycles;
end
return $n_{l,o}, \tau_o, G_o$

phase requires centralization, one of the orchestrators can facilitate the association phase by performing Algorithm 1, and then inform other orchestrators about their associations.

Factor-Based Association and Allocation

We present another heuristic approach that we refer to as the Factor-Based Association and Allocation (FBA). The FBA is based on an association factor (AF) $\Lambda_{l,o}$, which is expressed as:

$$\Lambda_{l,o} = \frac{\bar{f}_l}{\bar{d}_{l,o}} \tag{3.35}$$

where $\bar{f}_l$ and $\bar{d}_{l,o}$, both $\in [0, 1]$, are the normalized processor frequency of learner l and the distance between learner l and orchestrator o, respectively. The AF characterizes each learner's computing capability and its distance-based connection quality to each orchestrator. Each orchestrator can obtain this factor from all the available learners

and share it with other orchestrators. Based on these AFs, the FBA first performs a centralized turn-based association, where in each turn an orchestrator is selected fairly to get its chance to associate with a learner, and the learner with the maximum AF to that orchestrator gets associated with it. After an orchestrator is aware of its associated learners, each learner will be allocated a task size according to the following:

$$n_{l,o} = N_o \times \frac{\Lambda_{l,o}}{\sum_{l \in \mathcal{L}_o} \Lambda_{l,o}} \tag{3.36}$$

The rationale behind such task allocation is that learners with higher AF will get allocated larger amounts of data samples for training. Indeed, such learners will execute their tasks faster and have better channel quality for data and model transmissions with closer orchestrators. After determining the task size for each learner, orchestrators can do the same exhaustive search to solve (32) over the intervals that can be defined by (33) and (34) in order to obtain the optimal number of local iterations and global cycles. The FBA algorithm is detailed in Algorithm 2.

3.3.3 Fully Decentralized Heuristic

Lastly, we propose a fully decentralized approach, namely, the Learner-driven FBA (L-FBA). In L-FBA algorithm, the learners initiate the association by calculating their AFs for each orchestrator, selecting and associating with the orchestrator with the highest AF, and informing their selected orchestrators about their AF value. After an orchestrator receives the list of associated learners along with their AF values, it can determine the task sizes based on each learner AF similarly to the original FBA algorithm according to the task allocation equation (36). Finally, the orchestrator can perform the same exhaustive search with the specified bounds to find the optimal

number of local iterations and global cycles. The L-FBA algorithm is summarized in Algorithm 3.

Algorithm 3: Learner-driven FBA (L-FBA) heuristic

Initialize: α, $T_{\max}$, $\tau_{\max}$
At each learner:
Look for the available orchestrators;
Calculate the AF for each orchestrator $\Lambda_{l,o}$;
Associate with an orchestrator O_i where: $i = \arg\max_{o} \ \Lambda_{l,o}$;
At each orchestrator:
Receive the list of associated learners with their AF's $\Lambda_{l,o}$;
Perform the task allocation according to (36);
Obtain the optimal upper bounds $G_o^{\max^*}$ and $\tau_o^{\max^*}$ according to (33) and (34);
Perform exhaustive search to solve (32) and find the optimal number of iterations and global cycles;
return $n_{l,o}, \tau_o, G_o$

3.4 Complexity Analysis

In this section, we study the complexity of each proposed algorithm, namely, the centralized optimization (COPT), the AAT, the FBA and the L-FBA algorithms. For the ease of reading, some variables will be reused and redefined in this section.

3.4.1 The COPT Algorithm:

First, the COPT employ the BnB algorithm to find a global solution, which worst case complexity is known to be $O(b^k)$, where b is the number of branches per node, and k is the maximum number of iterations for the BnB algorithm, i.e., the depth of the constructed BnB tree. In each node, the COPT solves a convex sub-problem using the interior point method, which has a complexity of $O(\sqrt{n}\log\left(\frac{\mu_0 n}{\varepsilon}\right))$, where n

is the dimension of the domain $\mathcal{D}$ and expressed in our problem as $n = 2|\mathcal{O}|(|\mathcal{L}| + 1)$, ε is the tolerance variable, and μ_0 is a hyper parameter [68]. Thus, the complexity of the COPT is given by $O(\sqrt{n} \log\left(\frac{\mu_0 n}{\varepsilon}\right) \times b^k)$.

3.4.2 The AAT Algorithm:

As for the AAT, we first analyze each sub-problem complexity. For **SP1**, an integer linear program is solved, which has complexity $O(c + \log(c)\,\rho)$, where c is the number of constraints in the program, and ρ is the bit precision hyperparameter [67]. In fact, the number of constraints for **SP1** is $c = 2|\mathcal{L}|$. The sub-problem **SP2** is a simple linear program, and its solving complexity is given by $O(\mathcal{C}\sqrt{c})$, where $\mathcal{C}$ represents the bit complexity [3] [69]. The complexity of **SP3** is the complexity of the exhaustive search and can be expressed as $O(\tau_{\max} G_{\max})$, where $\tau_{\max}$ and $G_{\max}$ represents the upper bounds of the search interval. As a result, the overall complexity for the AAT algorithm can be given by $O(c + \log(c)\,\rho + k(\mathcal{C}\sqrt{c} + \tau_{\max} G_{\max}))$.

3.4.3 The FBA and L-FBA Algorithms:

The FBA algorithm first associates each learner to an orchestrator, then each orchestrator performs the task allocation for each associated learner, and the exhaustive search to obtain the number of local iterations and global cycles. Hence, its complexity can be expresses as $O(2|\mathcal{L}| + \tau_{\max} G_{\max})$. Lastly, the L-FBA approach is totally decentralized, where learners only do basic operations to determine the AF for each orchestrator, and then the orchestrator do the task allocation and the exhaustive search. Therefore, its complexity is given by $O(|\mathcal{L}| + \tau_{\max} G_{\max})$.

[3] Bit complexity is the number of single operations (of addition, subtraction, and multiplication) required to complete an algorithm.

Algorithm	Complexity
COPT	$O(\sqrt{n}\log\left(\frac{\mu_0 n}{\varepsilon}\right)\times b^k)$
AAT	$O(c+\log(c)\,\rho+k(\mathcal{C}\sqrt{c}+\tau_{\max}G_{\max}))$
FBA	$O(2\lvert\mathcal{L}\rvert+\tau_{\max}G_{\max})$
L-FBA	$O(\lvert\mathcal{L}\rvert+\tau_{\max}G_{\max})$

Table 3.1: Algorithms Complexity

One can see that the COPT approach is the most complex approach. In fact, as the number of learners and orchestrators increase in the system, its complexity can grow exponentially. This is due to the fact that the number of needed iterations or branches for the BnB algorithm can increase as the dimensions of the domain increase, which is not the case for the other algorithms. On the other extreme, the FBA and L-FBA are the least complex ones, as the orchestrator is only required to do the task allocation and search for the optimal number of iterations and global cycles, and their complexity scale linearly as the number of learners increases. All in all, the COPT approach is too impractical to be deployed in an MEL system with the absence of an edge server that is able solve the optimization problem with no computation overhead. Whereas the other approaches can be easily deployed as they are light-weight and can be decentralized, but with cost of losing the optimality in the solution.

3.5 Simulation Results

In this section, we first show the energy-accuracy trade-offs for the proposed algorithms, and then compare our approaches with the recent state-of-art MEL approach in [8], which we will refer to as the Energy-Unaware (EU) approach. Afterward, we show the performance of the algorithms in different scenarios where we: 1) Fix the

Table 3.2: Simulation parameters

Node Bandwidth W	5 MHz
Transmission Power $P_{l,o}$	200 mW
Distance range for $d_{l,o}$	$[5, 50]$ m
Processors frequencies f_l's	$[0.5, 0.7, 1.2, 1.8]$ GHz
On chip capacitance μ	1×10^{-19}
Learning parameters η_o, ϕ	[0.01,0.0001]
Maximum weights divergence δ_o	5
Maximum gradients divergence β_o	0.5
Bit precision for Γ_o^w and Γ_o^d	32 bits
Dataset size for all datasets	60,000

number of orchestrators and vary the number of learners; 2) Vary the number of orchestrators and fix the number of learners in the system. The comparison and the performance evaluation are done considering 3 orchestrators and 50 learners, where the orchestrators have similar multiple learning tasks (i.e., similar datasets and models architecture) for the sake of convenience in the results. We first utilize the MNIST dataset for performance comparison and evaluation. Subsequently, we evaluate the learning performance considering multiple and different tasks and datasets. The simulations were run considering the parameters shown in Table 3.2. The learning parameters values are taken from [7] as is.

3.5.1 Energy-Accuracy Tradeoff

Objectives Pareto trade-off curves are generated using different values for the MOP weight, namely, α. These curves help us to find Pareto optimal solutions for the weights such that it can balance the performance between the energy consumption and the accuracy. Each point on the curves represents a solution for a single value of the weight. The trade-off curves for the proposed algorithms are shown in Fig.

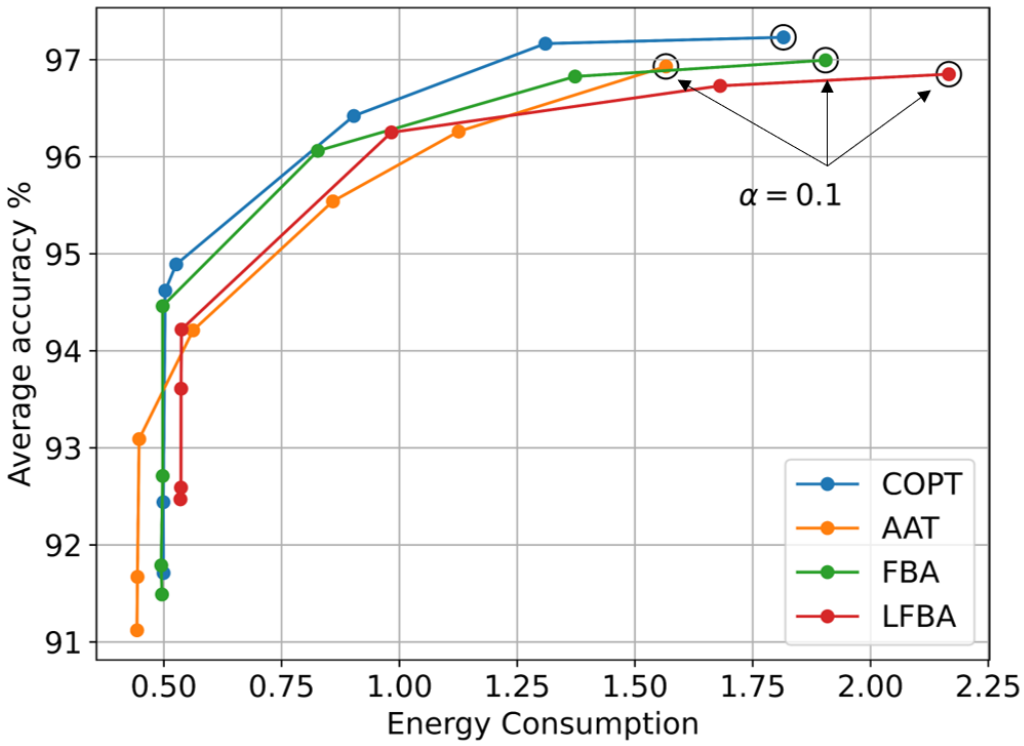

Figure 3.3: Energy-accuracy trade-offs curves considering $T_{\max} = 660s$ and

3.3. We can see that the COPT approach achieves the best trade-off as it achieves the highest accuracy with a low energy consumption. Moreover, the AAT approach is the best energy conservative approach, but it underperforms in terms of accuracy. This is due to the fact that the AAT algorithm optimizes the association and the task allocation energy consumption only first, and then it considers the energy-accuracy trade-off when deciding the number of local iterations and global cycles. As for the FBA, it performs slightly better than the LFBA approach, but they have similar performance where they have better accuracy than the AAT and worse than the COPT, but more energy consumption in general. Moreover, for $\alpha = 0$, the point is an outlier with 99% accuracy and very high energy consumption. Lastly, it can be noticed the Pareto optimal values will lay in the range of α between 0.2 and 0.4, where

decreasing the weight will increase the accuracy without significantly increasing the energy consumption.

3.5.2 Performance Comparison with the EU Approach

We compare our approach with the EU technique that is presented in [8] with distance based association. The EU approach optimizes the task allocation and the number of training iterations between heterogeneous learners such that the learning experience is maximized under global time constraints. We have conducted the comparative study using Monte Carlo simulation with 50-100 runs. The comparisons in terms of energy consumption and accuracy are depicted in Fig. 3.4 (a) and (b), respectively, considering different values of the time constraint $T_{\max}$. In Fig. 3.4 (a), we can see that, as we increase $T_{\max}$, all approaches consumes more energy, since increasing $T_{\max}$ adds more degree of freedom to do more local iterations and global cycles. However, the energy consumption for all the proposed approaches is significantly lower than the EU approach, and the optimization approach consumes slightly less energy than both the FBA and LFBA approaches, while the AAT approach consumes the least energy in the system. Fig. 3.4 (b) shows that the EU approach achieved the highest accuracy, but the proposed COPT approach only falls behind within 2% accuracy range for the different values of $T_{\max}$. On the other hand, the proposed heuristics underperform for smaller $T_{\max}$ values, but it only falls behind within 3% range for larger $T_{\max}$ values. In addition, for the same accuracy (e.g., 96%), the COPT approach also consumes less energy consumption but it takes more time to achieve the same accuracy.

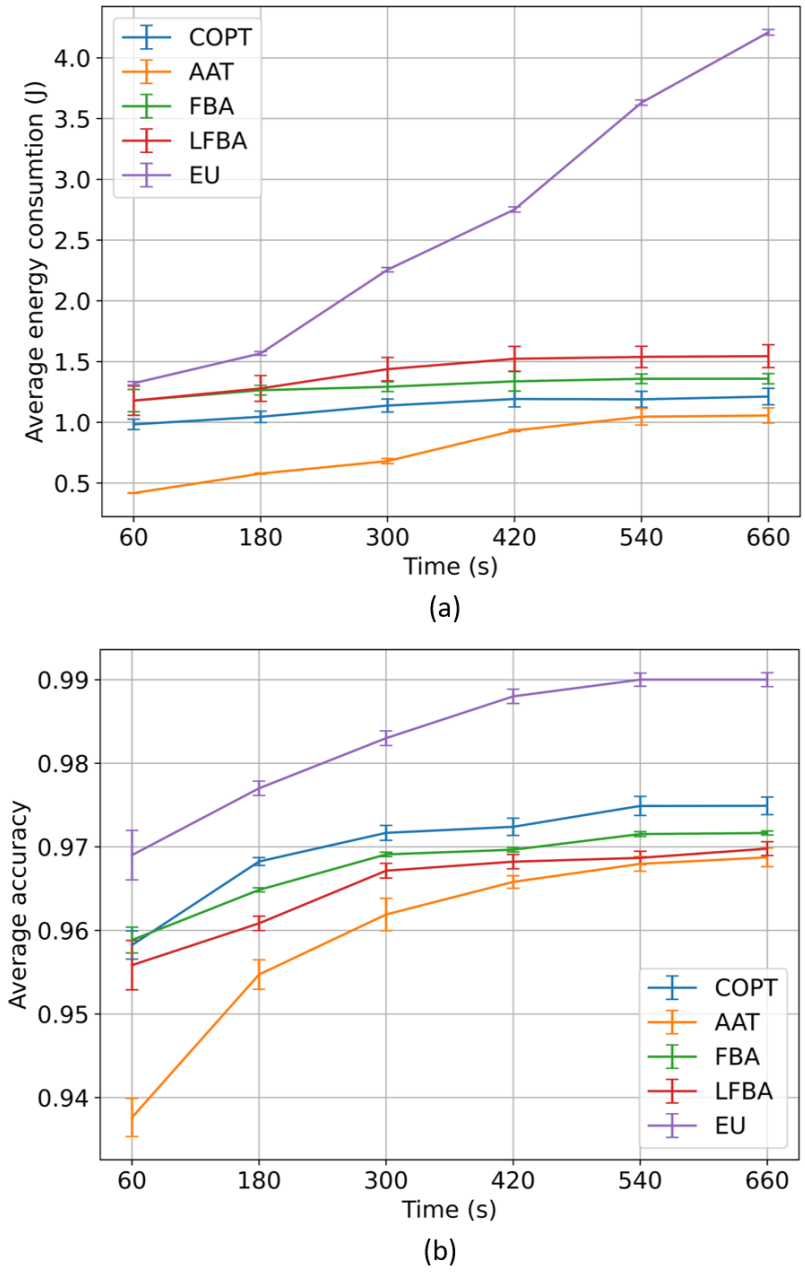

Figure 3.4: Performance comparison between approaches in terms of (a) energy consumption (b) learning accuracy with 3 orchestrators and 50 learners

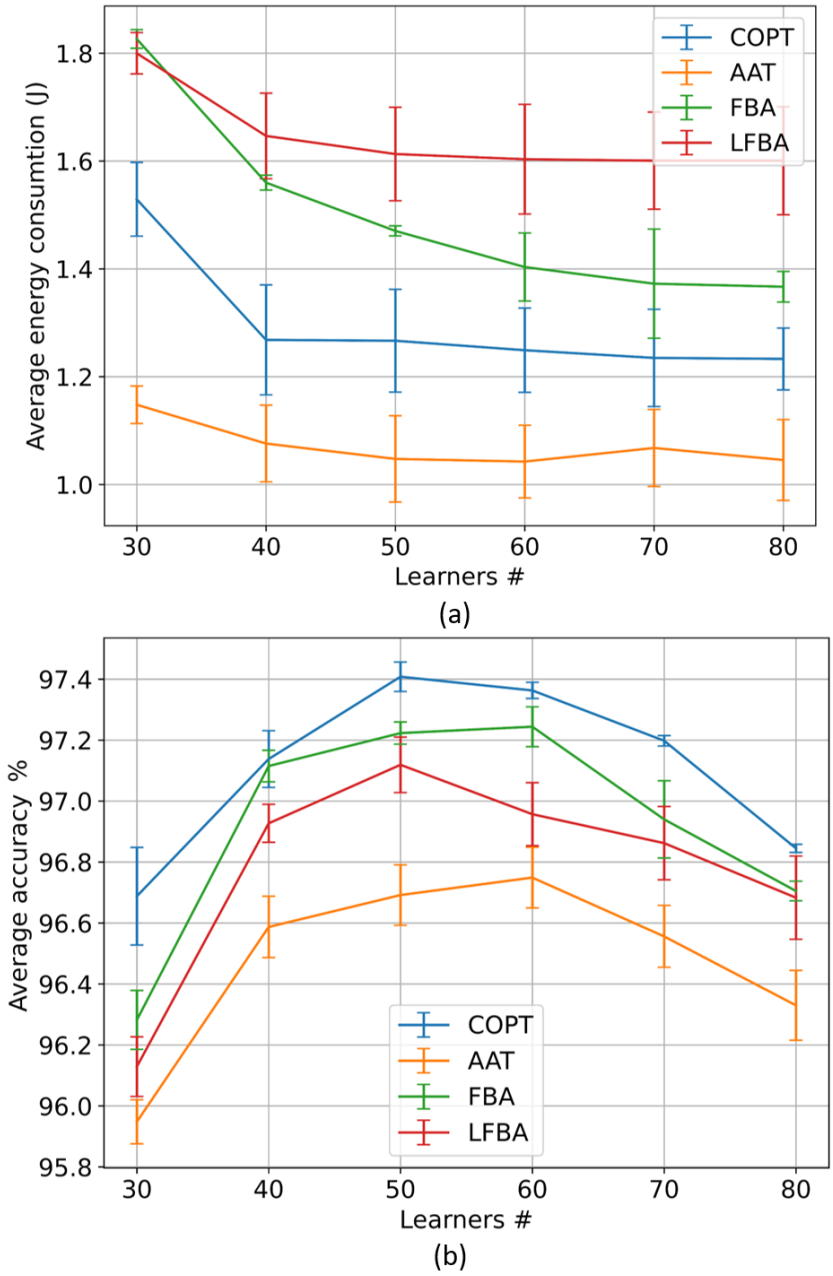

Figure 3.5: Performance evaluation with different number of learners in terms of (a) energy consumption (b) learning accuracy while considering 3 orchestrators and $T_{\max} = 660s$.

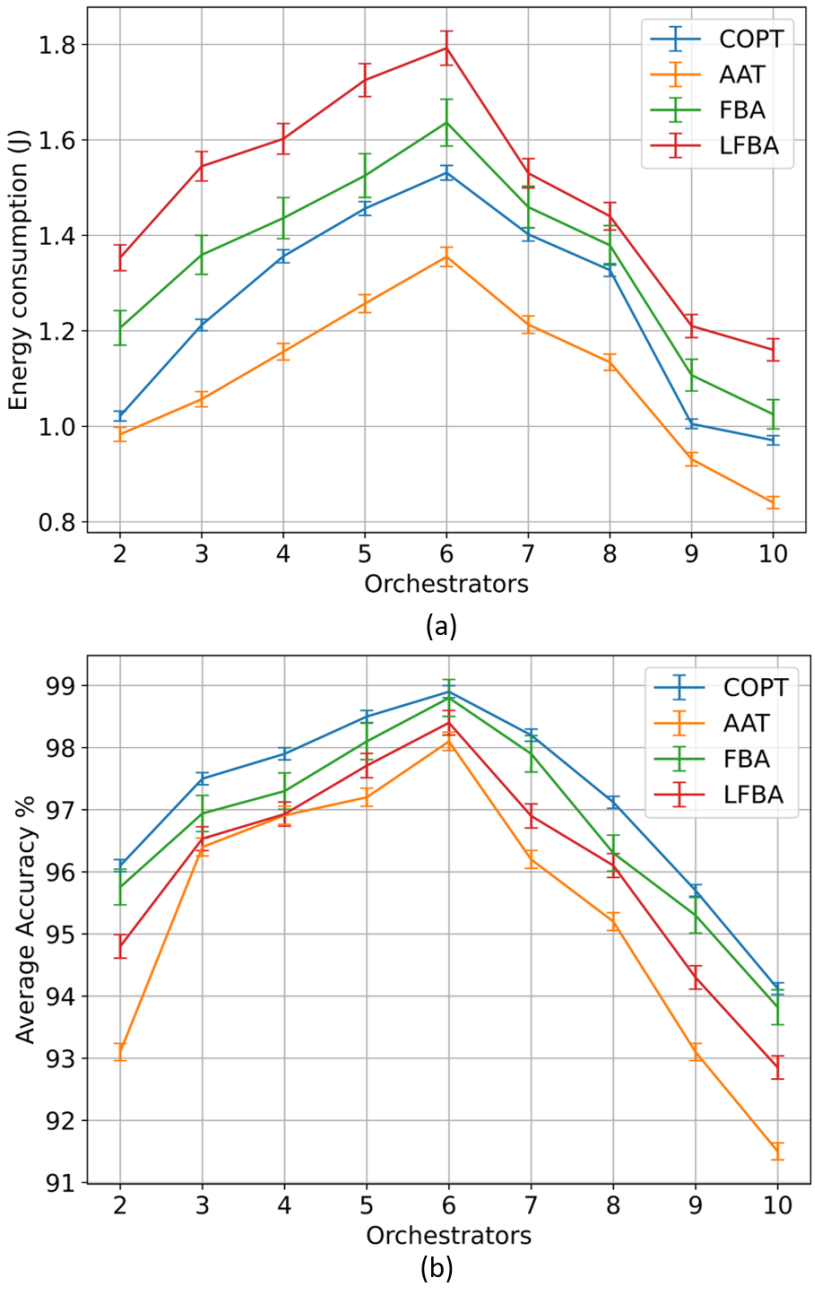

Figure 3.6: Performance evaluation with different number of orchestrators in terms of (a) energy consumption (b) learning accuracy while considering 50 learners and $T_{\max} = 660s$.

3.5.3 Performance Evaluation in Different Scenarios

We first show the performance of our proposed algorithms in terms of energy consumption and accuracy when varying the number of learners while fixing the number of orchestrators in Fig. 3.5. Then in Fig. 3.6 we show the performance when varying the number of orchestrators while fixing the number of learners. In Fig. 3.5 (a), we can see that as the number of learners is increased, the energy consumption decreases gradually for all the algorithms. Since increasing the number of learners can potentially allow orchestrators to associate with more learners, the learning task is distributed to more learners and each learner will have a smaller task size. Hence, the communication and computation energy consumption will be less for each learner. As for the accuracy, as shown in Fig. 3.5 (b), it starts to increase but then it decreases gradually as we increase the number of learners. In fact, smaller task sizes lead to less data transmission time between the orchestrator and the learners and less computation time at the learner, which can allow for more local iterations and global cycles. However, if the task size for each learner (i.e., the number of data samples) is small, it might not be sufficient to do the learning task and results in lower learning accuracy. On the other side, increasing the number of orchestrators means increasing the amount of data available for learning, which leads to a larger task size for each learner. As we can see in Fig. 3.6 (a), the energy consumption increases at first due to the aforementioned fact, which leads to more computing and communication energy consumption. However, the task size for each learner can become large enough to throttle the learning process, since increasing the task size results in increasing the time needed for data transmission between the orchestrators and the learners, and more training time at the learner side. Therefore, the number of local iterations

and global cycles will be decreased so that the total training time does not exceed its limit, which explains the sharp drop in the energy consumption after we increase the number of orchestrators. Similarly, in Fig. 3.6 (b), we can see that the accuracy at first increases since larger task sizes for the learners result in a better learning experience. However, by limiting the number of local iterations and global cycles, the accuracy drops abruptly since the learners can not train sufficiently.

3.5.4 Evaluation with Different Learning Tasks

Lastly, we consider different multiple learning tasks with different datasets, namely, MNIST [70], FMNIST [71], and CIFAR-10 [72]. We show the performance metrics for each learning task, specifically, the global accuracy, the global loss value, the weights, and the gradients divergence between the orchestrators' models and the associated learners' models. The global accuracy and loss value are plotted in Fig. 3.7 (a) and (b), respectively. We can notice for the MNIST and FMNIST datasets, the global models started to converge after 4 global cycles. Being a more complex learning task, the CIFAR-10 model did not converge within the same number of global cycles. However, if compared to the centralized training with the same number of total learning iterations which has an accuracy $\sim 73\%$, it only falls behind by $\sim 4\%$. As for the weights and gradients divergence, we claimed that these parameters can be empirically fixed to an upper bound to facilitate the analysis as provided in Table 1, where during training the actual values do not exceed this bound. The weights and gradients divergence are depicted in Fig. 3.7 (c) and (d), respectively. We noticed that during training, the values for both weights and gradients divergence are always below the upper bound for all the learning tasks, and the divergence gets smaller as

the training progresses.

3.5.5 Federated Learning Evaluation and Discussion

In this section, we evaluate the FL in our model using the COPT approach for task allocation and FedAvg algorithm for distributed learning [73] with different cases as follows: 1) **case 1**: The data is independent and identically distributed (IID) among all the learners, 2) **case 2**: The local datasets are not IID, with different amounts of data for each learner (e.g., randomly distributed between 30%-80% per class), 3) **case 3**: The data is completely non-IID and the distributions are skewed among learners. The accuracy evaluation of the FL cases compared to the PL is depicted in Fig 3.8. It can be noticed that the FL with IID data performs similarly to the PL. In fact, since the orchestrator in the PL can control the data distribution, it can allocate the learning task to the learners such that the data is IID among them, which makes it identical to the FL in the first case. In the second case, it can be seen that the performance of FL drops at first, but as the training progresses the performance is improved and gets closer to the IID case. In the last case, it can be seen that the performance suffers from a sharp drop in accuracy with respect to the other cases. In fact, the FedAvg algorithm in the complete non-IID case fails to deliver any learning experience to the learners. Such downfall is already discussed in the literature [13, 14], and can be mitigated by improving the distributions of the learners' local datasets by sharing a small portion of data from each local dataset between the learners or selecting a subset of the associated learner to participate in the training. This can come at the cost of violating the data privacy the main feature of FL, or missing learners with high capabilities that can speed up the training.

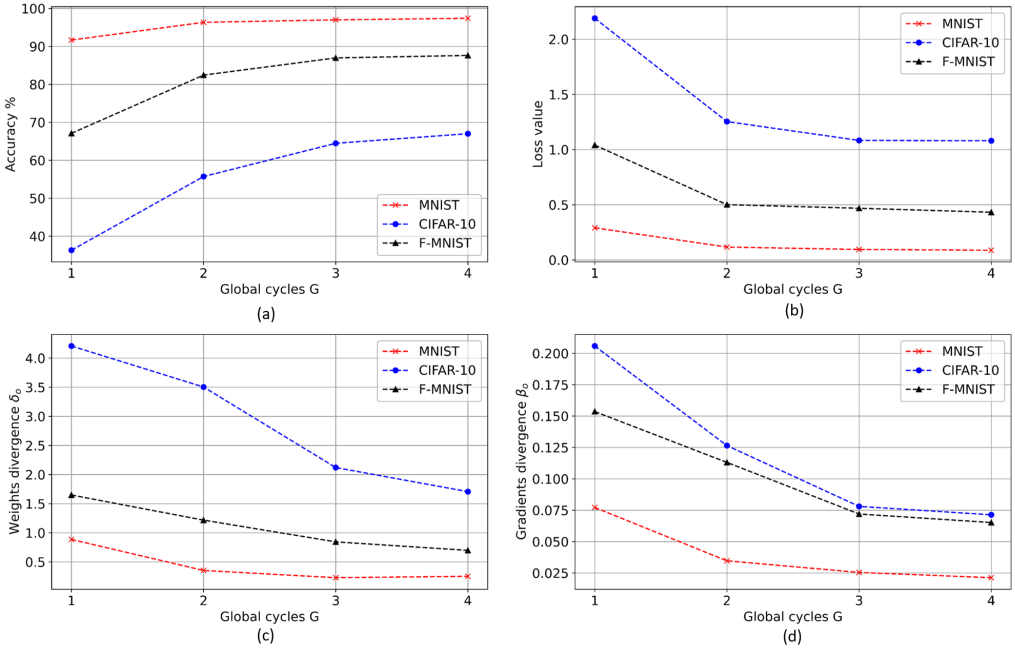

Figure 3.7: Learning metrics evaluation in terms of (a) global accuracy (b) global loss value (c) weights divergence (d) gradients divergence

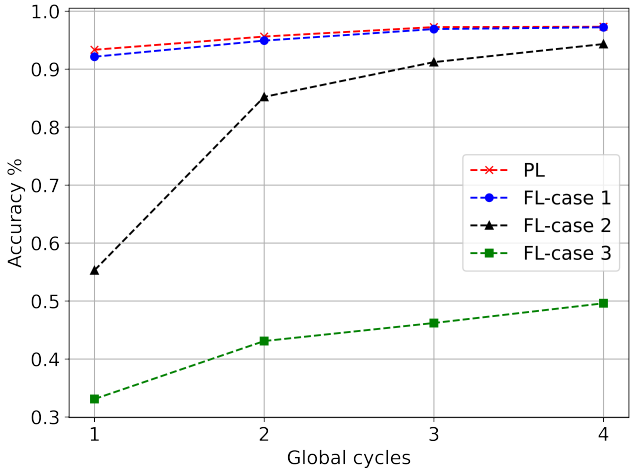

Figure 3.8: Accuracy evaluation of PL and FL with different cases.

3.6 Summary

In this chapter, we studied the problem of an energy-aware, multi-task and multi-orchestrator MEL system. We first formulated a multi-objective optimization problem for learners-orchestrator association and task allocation that aims to minimize the total energy consumption and maximize the learning accuracy. Being NP-hard and non-convex problem, the problem is relaxed and covexified via exponential variable transformation and linear approximations for the non-convex terms, and then solved via the BnB algorithm. Since the optimization approach is centralized and computationally expensive, we then proposed a set of lightweight partially and fully decentralized heuristics for the association and task allocation. The proposed heuristics let the orchestrators simply solve the association and the task allocation problems via convex optimization, and then determines the number of local iterations and global cycles via exhaustive search. To reduce the complexity and achieve a faster search, optimal upper bounds are derived for the number of local iterations and global cycles. The conducted experiments show that the proposed approaches reduce the energy consumption significantly while executing multiple learning tasks compared to recent state-of-art methods while falling behind the benchmark by 2%-3% in terms of accuracy.

Chapter 4

Motivating Learners in Mobile Edge Learning

4.1 Designing the Incentives Mechanism: Formulating A Stackelberg Game

In this paper, we consider a multi-orchestrator multi-task MEL system. Each of these orchestrators can be either 1) a governing node for learners that have private data for the same learning task in the case of FL, or 2) an edge node that is incapable of doing the training of its learning problem due to the lack of its computing resources in the case of PL. Nevertheless, our formulations are generic and applicable for both cases, but we will focus on the PL case in this work and point out the difference in the formulation whenever needed.

4.1.1 Learning Settings

We denote the set of learners by $\mathcal{L}$ and the set of learners that are associated with orchestrator o by $\mathcal{L}_o$, and each orchestrator has a dataset with N_o samples. After the association, an orchestrator o sends to a participating learner l the learning task

in terms of the model parameters $\boldsymbol{w}_{l,o}$, and $n_{l,o}$ data samples to train on [1]. Afterward, each participating learner l performs $\tau_{l,o}$ local training iterations employing Stochastic Gradient Descent (SGD) to minimize its local loss function. Once done, the learners send back to the orchestrator the training model parameters, where the latter aggregates these parameters by performing weighted averaging on the received models.

Each orchestrator then keeps sending back the data and the updated model for G_o global cycles until a stopping criterion is satisfied such as the exhaustion of the resources, e.g., energy. Moreover, we consider the system is globally synchronous and locally asynchronous as in [9]. In other words, at each global cycle, all the models from all the learners are collected and aggregated, but between the global cycles, each learner performs a different number of local training iterations on their local models. By considering so, the effect of the "straggler's dilemma" is reduced, which represents how the learning process is throttled by the learner with the least capabilities [4]. Last but not least, we adopt the presented model in [74] to define the learning objective as a function of the local iterations and global cycles as follows:

$$\tilde{F}_o(\tau_{l,o}, G_o) = \frac{c1}{G_o \tau_{l,o}^{c2}} \tag{4.1}$$

where $\tau_{l,o} \in [1, \tau_{max}]$ to ensure the convexity of the objective, c1 and c2 are constants that depend on learning parameters, namely, the divergence between the learners' local models and the global model, and the learning rates. The objective function (4.1) represents the distributed learning convergence bounds over the edge network. The convergence bounds refer to how much the trained global model in the distributed

[1] In case of FL, $n_{l,o}$ represents the number of data samples from the local dataset of the learner.

learning is deviating from the optimal model. Therein, we denote the distributed learning quality as $F_o(\tau_{l,o}, G_o) = -\tilde{F}_o(\tau_{l,o}, G_o)$. Since the objective function is obviously convex for $c1, c2 > 0$, it follows that the learning quality $F_o(\tau_{l,o}, G_o)$ is a concave function.

4.1.2 Mobile Edge Settings

We introduce the communication and computation models for wireless edge learning in this part. First, the number of bits that a learner l receives from an orchestrator o can be defined as:

$$B_{l,o}^{data} = n_{l,o} X_o \Gamma_o^d \tag{4.2}$$

$$B_o^{weights} = S_o^w \Gamma_o^w \tag{4.3}$$

where X_o is the feature vector length, S_o^w the total number of weights in the model, and Γ_o^d and Γ_o^w represent the bits/feature and bits/weight values, respectively. Consequently, the transmission time needed for both the data and the model weights from the orchestrator, and the updated model weights from the learner can be given by, respectively[2]:

$$t_{l,o}^S = \frac{B_{l,o}^{data} + B_o^{weights}}{W \log_2(1 + \frac{h_{l,o} P}{\sigma^2})} \tag{4.4}$$

$$t_{l,o}^U = \frac{B_o^{weights}}{W \log_2(1 + \frac{h_{l,o} P}{\sigma^2})} \tag{4.5}$$

where P is the devices' transmission power, W is the channel bandwidth, σ^2 is the channel noise variance, and $h_{l,o}$ is the channel gain expressed as $h_{l,o} = d_{l,o}^{-\nu} g^2$ where ν is the path loss exponent, $d_{l,o}$ is the distance between the orchestrator and the

[2]In case of FL, the term $B_{l,o}^{data}$ is set to 0.

learner, and g is the fading channel coefficient. It is worth noting that we assume the bandwidth and the transmission power are fixed during the training, as well as channel reciprocity. We then define learner's training time as follows:

$$t_{l,o}^{C} = \frac{\tau_{l,o} n_{l,o} C_o}{f_l} \tag{4.6}$$

where f_l is the learner's CPU frequency and C_o is the model computational complexity (i.e., the number of CPU cycles needed to train on a single data sample).

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:

$$E_{l,o}^{U} = P t_{l,o}^{U} = \frac{P_{l,o} B_o^{weights}}{W \log_2(1 + \frac{h_{l,o} P_{l,o}}{\sigma^2})} \tag{4.7}$$

We also denote the reception energy consumption as $E_{l,o}^{S}$, which we consider it as a constant and is coupled with the number of data samples $n_{l,o}$ to be received from the orchestrator[3]. As for the learner's computation energy consumption, it can be defined in our system as:

$$E_{l,o}^{C} = \frac{\mu \tau_{l,o} n_{l,o} C_o}{f_l^{\xi}} \tag{4.8}$$

where μ and ξ are hardware-related constants.

[3]In case of FL, the term $E_{l,o}^{S}$ is set to 0.

4.1.3 Incentive Mechanism Formulation

We formulate the incentive mechanism for multi-orchestrator MEL as a static 2-rounds Stackelberg game for the whole learning process, where the players are the orchestrators and the learners. There are two stages in each round, where the orchestrators are the leaders which act in the first stage, and the learners are the followers which act in the second stage.

In the first round, the orchestrators first decide the initial payment and broadcast their learning task details including: 1) the time frame of the whole learning process T_{max}, 2) the initial monetary service price ρ_o^i (i.e., \$/CPU cycle), 3) the minimum number of local iterations τ_{min} and global cycles G_{min} to guarantee a learning experience for the task. The learners then choose which orchestrator to associate with, and send their computing and communication capabilities information. In the second round, each orchestrator o then announces its strategy including the final monetary service price for each learner $\rho_{l,o}$ in the first stage, followed by the second stage where each learner determines its strategy in terms of the number of data samples $n_{l,o}$ such that their utility is maximized. We express an orchestrator's strategy in the second round as the number of local iterations to be performed by each associated learner and the monetary service price. Consequently, we can define the utility of the orchestrator o as:

$$U_o(\tau_{l,o}, \rho_{l,o}) = \sum_l \lambda_{l,o} \left(F_o(\tau_{l,o}, G_o) - G_o C_o \rho_{l,o} \tau_{l,o} n_{l,o} \right) \tag{4.9}$$

where the $\lambda_{l,o}$ is the association variable, and the first term represents the task learning quality, while the second term represents the total payment to the learners. However, the orchestrators cannot determine the number of global cycles since the considered MEL system is globally synchronous, and the learners have heterogeneous capabilities.

Hence, the learning time will be dependent on the weakest learner that causes the maximum computation and communication delay. As a result, the orchestrator has to set the number of global cycles according to that learner to accommodate all the learners in the training process and increase the level of participation. In fact, by considering the weakest learner, the number of global cycles will be limited, but the orchestrator will have more degrees of freedom in determining its strategy in terms of the monetary service price and the number of local iterations for the other learners. After determining the number of global cycles, an orchestrator needs to determine its strategy by maximizing its utility, i.e., $\max_{\tau_{l,o},\rho_o} . \; U_o(\tau_{l,o}, \rho_{l,o})$, while being constrained to its budget as follows:

$$\sum_{l} \rho_{l,o} C_o n_{l,o} = b_o \tag{4.10}$$

where b_o is the orchestrator's o budget. We assume that orchestrators have similar budgets or vary a little, and no orchestrator has a huge budget. In fact, even though this might be a limitation to our approach, but generally, IoT devices usually are not expected to have huge budgets. As for the learners, their strategy is expressed as the association with an orchestrator, which will be done in the first round, and the amount of participation if decided in terms of the data samples in the second round. Moreover, we assume the distribution of the decided training data is representative of the whole data distribution, and the data at each learner have the same quality, i.e., independently and identically distributed. We define the service cost for the learners as their computation and communication energy consumption, which are proportional to the amount of training data. Therein, we can define a learner l utility as follows:

$$U_l(\lambda_{l,o}, n_{l,o}) = \sum_{o} \lambda_{l,o} G_o \left(C_o \rho_{l,o} \tau_{l,o} n_{l,o} - (E^S_{l,o} + E^U_{l,o} + E^C_{l,o}) \right) \tag{4.11}$$

where the first term represents the total revenue from the learning process. We assume that a learner can associate with only one orchestrator for a single learning task, and it has to finish the training within the learning task time frame. Accordingly, the learners can decide their optimal strategy by solving the following optimization problem:

$$\mathbf{P1}: \quad \max_{\lambda_{l,o}, n_{l,o}} . U_l(\lambda_{l,o}, n_{l,o}) \tag{4.12a}$$

$$s.t. \sum_o \lambda_{l,o} \left(t^S_{l,o} + t^U_{l,o} + t^C_{l,o} \right) \leq T_{max} \tag{4.12b}$$

$$\sum_o \lambda_{l,o} = 1 \tag{4.12c}$$

$$0 \leq n_{l,o} \leq n_{max} \tag{4.12d}$$

$$\lambda_{l,o} \in \{0, 1\} \tag{4.12e}$$

where T_{max} is the learning time period, and n_{max} is a hyperparameter that indicates the maximum allowed number of data samples to receive from the orchestrator if any, where $n_{max} \in (0, 1]$. Lastly, to simplify the formulations in the reminder of the paper, we define the following coefficients:

$$A^0_{l,o} = \frac{2B^{weights}_o}{T_{max} \times W \log_2(1 + \frac{h_{l,o} P_{l,o}}{\sigma^2})}, \quad \zeta^0_{l,o} = \frac{P_{l,o} A^0_{l,o}}{E_{max}}$$

$$A^1_{l,o} = \frac{N_o F_o \Gamma^d_o}{T_{max} \times W \log_2(1 + \frac{h_{l,o} P_{l,o}}{\sigma^2})}, \quad \zeta^1_{l,o} = E^S_{l,o}$$

$$A^2_{l,o} = \frac{N_o C^w_o}{T_{max} \times f_l}, \qquad \zeta^2_{l,o} = \left(\frac{C_o \rho_{l,o}}{R_{max}} - \frac{\mu C_o}{f^{\xi}_l E_{max}}\right)$$

where E_{max} and R_{max} are the maximum energy consumption and revenue for the learners, respectively.

4.2 Solution Approach

In this section, we first present a heuristic approach for the learners-orchestrator association problem. Afterward, we analyze the learners' and orchestrators' behavior and derive the optimal learner strategy in the second round.

4.2.1 First round: Factor-Based Association

Since the learners in the first round determine the association given the learning task details, the optimization problem **P1** can be first solved to determine the association by assuming a fixed arbitrary $n_{l,o}$ for all the learners. Herein, we define the association sub-problem **SP1** as follows:

$$\mathbf{SP1}: \quad \max_{\lambda_{l,o}} . \quad \sum_o \lambda_o G_o \left(n_{l,o}(\zeta^2_{l,o}\tau_{l,o} - \zeta^1_{l,o}) - \zeta^0_{l,o}\right) \tag{4.13a}$$

$$s.t. \quad \sum \lambda_o G_o (A^2_{l,o}\tau_{l,o} n_{l,o} + A^1_{l,o} n_{l,o} + A^0_{l,o}) \leq 1 \tag{4.13b}$$

$$\sum_o \lambda_{l,o} = 1 \tag{4.13c}$$

$$\lambda_o \in \{0, 1\} \tag{4.13d}$$

It can be noticed that **SP1** is a binary integer linear program (BILP), which is eventually a minimization knapsack problem, and is known to be NP-hard. As a result, employing algorithms to derive an equilibrium based on estimating the learners'

strategy is not feasible in the first round. Thus, we first present a heuristic algorithm for learners-orchestrators association that we refer to as the Factor-Based Association (FBA). We define the association factor (AF) with learner l and orchestrator o as follows:

$$\Gamma_{l,o} = \frac{\tilde{\rho}_o^i \tilde{C}_o}{\tilde{d}_{l,o}} \tag{4.14}$$

where the term $\tilde{\rho}_o^i \tilde{C}_o$ represents the normalized initial payment per data sample from the orchestrator , and $\tilde{d}_{l,o}$ is the normalized distance between learner l and orchestrator o, respectively, and both $\in [0, 1]$. The AF is calculated at the learners' side in order for them to decide the association with which orchestrator. The AF characterizes the potential revenue for the learner, and since the game is played only once and does not capture the communication channel dynamics, the AF also characterizes the connectivity by the distance. Each learner then associate with orchestrator $\tilde{o}$ such that $\tilde{o} = \arg\max_o \Gamma_{l,o}$. After the association is done, each learner sends to its associated orchestrator information about its computation and communication capabilities.

4.2.2 Second round: Deriving the Learners and Orchestrators' Strategies

In the second round, we employ the backward-induction method to derive the Stackelberg equilibrium, where the second stage in this round is solved to obtain the learners' optimal strategy, which is then used for solving the first stage to obtain the associated orchestrator's optimal strategy. Since the variable $n_{l,o}$ is an integer, we relax it to a continuous variable to solve the problem, then we round it to the nearest integer in the solution.

For a given association $\lambda_{l,o}$, monetary service price $\rho_{l,o}$, local training iterations

$\tau_{l,o}$ and global cycles G_o, the learner can determine its strategy by solving the participation sub-problem **SP2** which can be given as the following :

$$\mathbf{SP2}: \quad \max_{n_{l,o}} . \quad n_{l,o}(\zeta^2_{l,o}\tau_{l,o} - \zeta^1_{l,o}) \tag{4.15a}$$

$$s.t. \quad G_o(A^2_{l,o}\tau_{l,o}n_{l,o} + A^1_{l,o}n_{l,o} + A^0_{l,o}) \leq 1 \tag{4.15b}$$

$$0 \leq n_{l,o} \leq n_{max} \tag{4.15c}$$

Lemma 1. *The problem **P2** is concave and the learner's optimal strategy is given by:*

$$n^*_{l,o} = \begin{cases} 0 & \zeta^2_{l,o}\tau_{l,o} - \zeta^1_{l,o} < 0 \\ \frac{\zeta^2_{l,o}(1-G_oA^0_{l,o})}{G_o(\zeta^1_{l,o}A^2_{l,o}+\zeta^2_{l,o}A^1_{l,o})} & n_{l,o} \in [n_{min}, n_{max}) \\ n_{max} & n_{l,o} \geq n_{max} \end{cases} \tag{4.16}$$

where:

$$n_{min} = \frac{\zeta^0_{l,o}}{\zeta^2_{l,o}\tau_{l,o} - \zeta^1_{l,o}} \tag{4.17}$$

Proof. It is readily obvious that the optimization problem **P2** is a Linear Program, and hence, it follows directly that it is concave. Moreover, the term $\zeta^2_{l,o}\tau_{l,o} - \zeta^1_{l,o}$ in the objective is the variable linear coefficient, and it can represent the net utility given the orchestrator's strategy. As a result, if the net utility is positive, the learner will try to maximize it by maximizing the amount of participation until the time frame period is finished. The rest of the proof can be found in Appendix C.

Recall that the number of global cycles will be set according to the weakest learner in the group, given the fact that it utilizes its full-time period. Consequently, the

constraint (14b) becomes equality, and the number of global cycles can be given by:

$$G_o = \max\left(G_{min}, \frac{1}{A^2_{\tilde{l},o}\tau_{l,o}n^*_{\tilde{l},o} + A^1_{\tilde{l},o}n^*_{\tilde{l},o} + A^0_{\tilde{l},o}}\right) \tag{4.18}$$

where $\tilde{l}$ is the index of the weakest learner, and it can be determined as follows $\tilde{l} = \arg\min_{l} \frac{\tilde{f}_l}{\tilde{d}_{l,o}}$, as the learner with the less CPU frequency and greater distance to its associated orchestrator causes more delay in the learning process.

Afterward, we derive the orchestrator's strategy set for the service monetary price. To ensure learners participation, the net utility term from (15) has to be positive, and it follows that:

$$\tau_{l,o} \geq \frac{\zeta^1_{l,o}}{\left(\frac{C_o\rho_{l,o}}{R_{max}} - \frac{\mu C_o f_l}{E_{max}}\right)} \tag{4.19}$$

Since we know that $\tau_{min} \leq \tau_{l,o} \leq \tau_{max}$, we can thus define the lower and upper bounds for the monetary price as follows:

$$\underline{\rho_{l,o}} = \frac{\zeta^1_{l,o}}{\tau_{max}} - \frac{R_{max}\mu f_l}{E_{max}}, \quad \overline{\rho_{l,o}} = \frac{\zeta^1_{l,o}}{\tau_{min}} - \frac{R_{max}\mu f_l}{E_{max}} \tag{4.20}$$

According to the above analysis, the orchestrator, which is the leader in the second round of the Stackelberg game, knows that there exists a Nash equilibrium among learners given any monetary service price and the number of local iterations within its strategy set. Therefore, by considering the learners' participation from (15) as a function of the service price, i.e., $n^*_{l,o}(\rho_{l,o})$, and setting the number of global cycles according to (17), the orchestrator can maximize its utility and determines its strategy

by solving the following optimization:

$$\mathbf{P3}: \max_{\tau_{l,o},\rho_{l,o}} \frac{-1}{F_{max}|\mathcal{L}_o|}\sum_{l}\frac{c1}{G_o\tau_{l,o}^{c2}} - \frac{G_oC_o}{P_{max}}\sum_{l}\rho_{l,o}\tau_{l,o}n_{l,o}^* \tag{4.21a}$$

$$s.t. \quad \tau_{min} \leq \tau_{l,o} \leq \tau_{max} \tag{4.21b}$$

$$\underline{\rho_{l,o}} \leq \rho_{l,o} \leq \overline{\rho_{l,o}} \tag{4.21c}$$

$$\sum_{i}\rho_{l,o}C_on_{l,o}^* \leq b_o \tag{4.21d}$$

where P_{max} is the maximum possible payment and F_{max} is the maximum loss value.

Lemma 2. *For $G_oC_o\rho_{l,o}\tau_{l,o}n_{l,o} \geq 2$, the problem **P3** is concave, and hence, there exists a Nash equilibrium for the second round of the game $(\tau_{l,o}^*, \rho_{l,o}^*, n_{l,o}^*)$, where $(\tau_{l,o}^*, \rho_{l,o}^*)$ is the maximizer of the orchestrator's utility.*

Proof. The proof can be found in Appendix D.

It can be noticed that the term in the condition represents the participation revenue of the learner, and is verified to hold practically and shown in the result. Nevertheless, the optimization problem **P3** has no closed-form solution for $\tau_{l,o}^*$ and $\rho_{l,o}^*$. Hence, the orchestrator can find the optimal strategy using any optimization technique (e.g., gradient ascent, interior point method...etc.).

4.3 Simulation Results

4.3.1 Environment Setup

We have conducted the experiments with 3 orchestrators and 50 learners. The learners' distances to the orchestrators were distributed uniformly randomly in the range of [5-50]m, and each learner can have one of the following processor frequencies

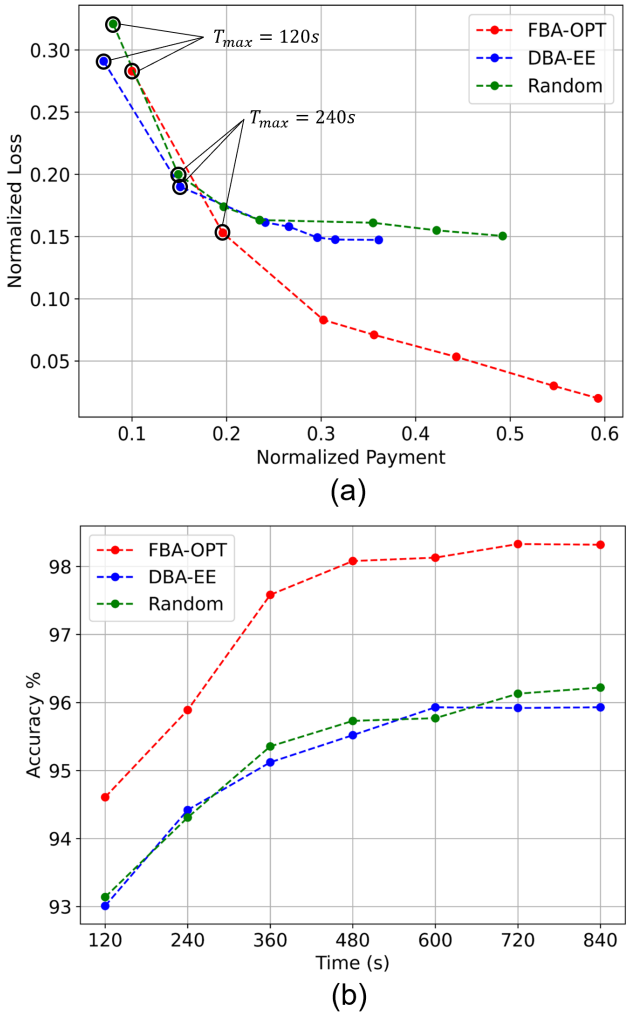

Figure 4.1: Orchestrators' performance comparison in terms of (a) utility trade-off (b) learning accuracy

[2.4,1.4,1.0] GHz. We have utilized the MNIST dataset for all the learning tasks. The learning models are Convolutional Neural Networks with [2,3,4] convolution layers, where higher number of layers represents higher model complexity. In the results, we show the average performance, e.g., the average learner utility and the average energy consumption.

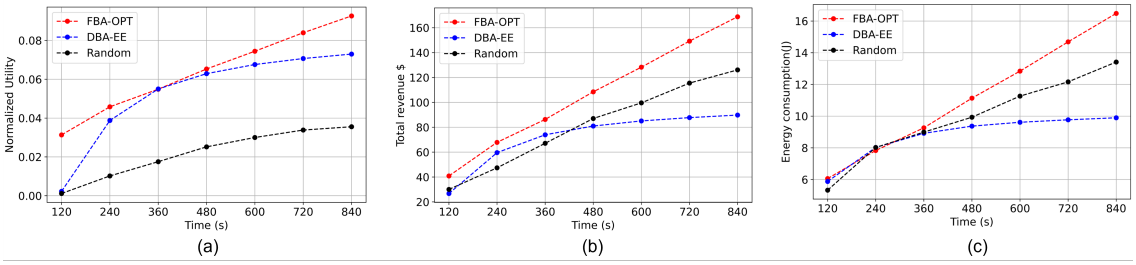

Figure 4.2: Learners' performance comparison in terms of (a) utility (b) total revenue (c) energy consumption

4.3.2 Performance comparison

We compare our approach which employs factor-based association and the optimal strategy (FBA-OPT) with the random strategy, and an energy-efficient technique that minimizes the energy consumption by considering the minimum possible number of iterations and global cycles, and employs a distance-based association, which we will refer to (DBA-EE). We compare the performance with respect to different time constraints, i.e., T_{max}. The orchestrators' performance in terms of utility trade-off and learning accuracy is depicted in Fig. 4.1. In Fig. 4.1 (a), we observe the trade-off curves for the utility function between the payment and the learning loss. Each point on the curves represents a different T_{max}. We can notice that, as we increase the time constraint, the total payment increases for all the approaches with a slight overpayment for our approach. However, the learning loss for the heuristic DBA-EE and the random strategies improve but then saturate with no improvements, while our approach keeps minimizing the learning loss. In Fig. 4.1 (b), the learning accuracy of the tasks is shown. Our approach achieves higher accuracy and outperforms the other approaches for all the time constraints since it minimized the learning loss much more than the other approaches.

The learners' performance is depicted in Fig 4.2. In Fig 4.2 (a), the normalized utility is shown. It can be seen that for all approaches, the utility increases as T_{max} increases. In fact, with more available time, the learner can increase the amount of participation to maximize its utility. However, our approach grants the best utility for the learners, while the random strategy is the worst. The heuristic DBA-EE approach at first starts increasing the utility, but then it starts to saturate. In fact, it limits the amount of participation in order to preserve its energy. The total revenue is shown in Fig 4.2 (b). It can be noticed that our approach attains the best revenue for the learners, while the random approach attains better than the heuristic, since the latter limits its participation to preserve more energy. Lastly, the energy consumption is presented in Fig 4.2 (c). Similarly, the energy consumption increases as T_{max} increases. However, the heuristic DBA-EE outperforms the other approaches and preserves energy the most, while our approach is opportunistic in terms of the revenue and consumes energy the most.

4.4 Summary

In this chapter, we proposed an incentive mechanism for multi-orchestrator MEL by employing a 2-round Stackelberg game approach. In the first round, we employed a heuristic algorithm for the learners-orchestrators association. Whereas in the second round, we proved the existence of the Nash equilibrium in the game and derived a learner's optimal policy based on the associated orchestrator's incentive. Finally, numerical experiments have been conducted to show the performance of the proposed mechanism while being compared to other heuristic techniques.

Chapter 5

Future Work and Conclusion

5.1 Conclusion

In this work, we first studied the problem of an energy-aware, multi-task and multi-orchestrator MEL system. We first formulated a multi-objective optimization problem for learners-orchestrator association and task allocation that aims to minimize the total energy consumption and maximize the learning accuracy. Being NP-hard and non-convex problem, the problem is relaxed and convexified via exponential variable transformation and linear approximations for the non-convex terms, and then solved via the BnB algorithm. Since the optimization approach is centralized and computationally expensive, we then proposed a set of lightweight partially and fully decentralized heuristics for the association and task allocation. The proposed heuristics let the orchestrators simply solve the association and the task allocation problems via convex optimization, and then determines the number of local iterations and global cycles via exhaustive search. To reduce the complexity and achieve a faster search, optimal upper bounds are derived for the number of local iterations and global cycles. The conducted experiments show that the proposed approaches reduce the

energy consumption significantly while executing multiple learning tasks compared to recent state-of-art methods while falling behind the benchmark by 2%-3% in terms of accuracy.

Afterward, we proposed an incentive mechanism for multi-orchestrator MEL by employing a two-round Stackelberg game approach. In the first round, we employed a heuristic algorithm for the learners-orchestrators association. Whereas in the second round, we proved the existence of the Nash equilibrium in the game and derived a learner's optimal policy based on the associated orchestrator's incentive. Finally, numerical experiments have been conducted to show the performance of the proposed mechanism while being compared to other heuristic techniques.

5.2 Future Work

First, considering the device association and task allocation, one promising direction is to consider the D2D data offloading between the learners to reduce the overhead on the weak learners, and to achieve a better learning performance in the system. Moreover, experience-driven algorithms that learn from the real-time experience can be a better option, as it reduces the complexity overhead of optimization-based approaches, and keeps learning as time progresses. This applies to both the task allocation and the incentive mechanism. However, such algorithms may need more time to converge on an acceptable performance, but afterwards, it will reduce the complexity overhead and can achieve better performance in the long run.

Bibliography

[1] D. Gündüz, D. B. Kurka, M. Jankowski, M. M. Amiri, E. Ozfatura, and S. Sreekumar, "Communicate to learn at the edge," *IEEE Communications Magazine*, vol. 58, no. 12, pp. 14–19, 2020.

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

[3] M. Bennis, M. Debbah, K. Huang, and Z. Yang, "Guest editorial: Communication technologies for efficient edge learning," *IEEE Communications Magazine*, vol. 58, no. 12, pp. 12–13, 2020.

[4] X. Cai, X. Mo, J. Chen, and J. Xu, "D2d-enabled data sharing for distributed machine learning at wireless network edge," *IEEE Wireless Communications Letters*, vol. 9, no. 9, pp. 1457–1461, 2020.

[5] 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.

[6] A. Abutuleb, S. Sorour, and H. S. Hassanein, "Joint task and resource allocation for mobile edge learning," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pp. 1–6, IEEE, 2020.

[7] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[8] U. Mohammad, S. Sorour, and M. Hefeida, "Optimal task allocation for mobile edge learning with global training time constraints," in *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, pp. 1–4, IEEE, 2021.

[9] U. Mohammad, S. Sorour, and M. Hefeida, "Task allocation for asynchronous mobile edge learning with delay and energy constraints," *arXiv preprint arXiv:2012.00143*, 2020.

[10] Q. Zeng, Y. Du, K. Huang, and K. K. Leung, "Energy-efficient radio resource allocation for federated edge learning," in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 1–6, IEEE, 2020.

[11] N. H. Tran, W. Bao, A. Zomaya, M. N. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 1387–1395, IEEE, 2019.

[12] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 3, pp. 1935–1949, 2020.

[13] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv*, 2018.

[14] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing Federated Learning on Non-IID Data with Reinforcement Learning," *Proceedings - IEEE INFOCOM*, vol. 2020-July, pp. 1698–1707, 2020.

[15] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[16] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2020.

[17] N. Mhaisen, A. Awad, A. Mohamed, A. Erbad, and M. Guizani, "Optimal user-edge assignment in hierarchical federated learning based on statistical properties and network topology constraints," *IEEE Transactions on Network Science and Engineering*, 2021.

[18] A. A. Abdellatif, N. Mhaisen, A. Mohamed, A. Erbad, M. Guizani, Z. Dawy, and W. Nasreddine, "Communication-efficient hierarchical federated learning for iot heterogeneous systems with imbalanced data," *Future Generation Computer Systems*, vol. 128, pp. 406–419, 2022.

[19] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–7, IEEE, 2019.

[20] Y. J. Cho, J. Wang, and G. Joshi, "Client selection in federated learning: Convergence analysis and power-of-choice selection strategies," *arXiv preprint arXiv:2010.01243*, 2020.

[21] J. Xu and H. Wang, "Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective," *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 1188–1200, 2020.

[22] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "Oort: Efficient federated learning via guided participant selection," in *15th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 21)*, pp. 19–35, 2021.

[23] T. Huang, W. Lin, W. Wu, L. He, K. Li, and A. Y. Zomaya, "An efficiency-boosting client selection scheme for federated learning with fairness guarantee," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1552–1564, 2020.

[24] N. Shlezinger, M. Chen, Y. C. Eldar, H. V. Poor, and S. Cui, "Federated learning with quantization constraints," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8851–8855, IEEE, 2020.

[25] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization," in *International Conference on Artificial Intelligence and Statistics*, pp. 2021–2031, PMLR, 2020.

[26] N. Shlezinger, M. Chen, Y. C. Eldar, H. V. Poor, and S. Cui, "Uveqfed: Universal vector quantization for federated learning," *IEEE Transactions on Signal Processing*, vol. 69, pp. 500–514, 2020.

[27] M. M. Amiri, D. Gunduz, S. R. Kulkarni, and H. V. Poor, "Federated learning with quantized global model updates," *arXiv preprint arXiv:2006.10672*, 2020.

[28] J. Xu, W. Du, Y. Jin, W. He, and R. Cheng, "Ternary compression for communication-efficient federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[29] X. Zhang, Z. Yang, W. Sun, Y. Liu, S. Tang, K. Xing, and X. Mao, "Incentives for mobile crowd sensing: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 54–67, 2015.

[30] C. Magerkurth, A. D. Cheok, R. L. Mandryk, and T. Nilsen, "Pervasive games: bringing computer entertainment back to the real world," *Computers in Entertainment (CIE)*, vol. 3, no. 3, pp. 4–4, 2005.

[31] N. M. Avouris and N. Yiannoutsou, "A review of mobile location-based games for learning across physical and virtual spaces.," *J. Univers. Comput. Sci.*, vol. 18, no. 15, pp. 2120–2142, 2012.

[32] S. Matyas, "Playful geospatial data acquisition by location-based gaming communities.," *Int. J. Virtual Real.*, vol. 6, no. 3, pp. 1–10, 2007.

[33] L. Barkhuus, M. Chalmers, P. Tennent, M. Hall, M. Bell, S. Sherwood, and B. Brown, "Picking pockets on the lawn: the development of tactics and strategies in a mobile game," in *International Conference on Ubiquitous Computing*, pp. 358–374, Springer, 2005.

[34] M. Bell, M. Chalmers, L. Barkhuus, M. Hall, S. Sherwood, P. Tennent, B. Brown, D. Rowland, S. Benford, M. Capra, *et al.*, "Interweaving mobile games with everyday life," in *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pp. 417–426, 2006.

[35] C. Schlieder, P. Kiefer, and S. Matyas, "Geogames: Designing location-based games from classic board games," *IEEE Intelligent Systems*, vol. 21, no. 5, pp. 40–46, 2006.

[36] T. Luo and C.-K. Tham, "Fairness and social welfare in incentivizing participatory sensing," in *2012 9th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pp. 425–433, IEEE, 2012.

[37] R. T. Ma, S. C. Lee, J. C. Lui, and D. K. Yau, "Incentive and service differentiation in p2p networks: a game theoretic approach," *Ieee/ACM Transactions on networking*, vol. 14, no. 5, pp. 978–991, 2006.

[38] B. Hoh, T. Yan, D. Ganesan, K. Tracton, T. Iwuchukwu, and J.-S. Lee, "Trucentive: A game-theoretic incentive platform for trustworthy mobile crowdsourcing

parking services," in *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pp. 160–166, IEEE, 2012.

[39] K.-c. Lan, C.-M. Chou, and H.-Y. Wang, "An incentive-based framework for vehicle-based mobile sensing," *Procedia Computer Science*, vol. 10, pp. 1152–1157, 2012.

[40] W. Mason and D. J. Watts, "Financial incentives and the" performance of crowds"," in *Proceedings of the ACM SIGKDD workshop on human computation*, pp. 77–85, 2009.

[41] M. Musthag, A. Raij, D. Ganesan, S. Kumar, and S. Shiffman, "Exploring micro-incentive strategies for participant compensation in high-burden studies," in *Proceedings of the 13th international conference on Ubiquitous computing*, pp. 435–444, 2011.

[42] S. Reddy, D. Estrin, M. Hansen, and M. Srivastava, "Examining micro-payments for participatory sensing data collections," in *Proceedings of the 12th ACM international conference on Ubiquitous computing*, pp. 33–36, 2010.

[43] J. Jia, Q. Zhang, Q. Zhang, and M. Liu, "Revenue generation for truthful spectrum auction in dynamic spectrum access," in *Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*, pp. 3–12, 2009.

[44] C. Wu, B. Li, and Z. Li, "Dynamic bandwidth auctions in multioverlay p2p streaming with network coding," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 6, pp. 806–820, 2008.

[45] J. Nie, J. Luo, Z. Xiong, D. Niyato, and P. Wang, "A stackelberg game approach toward socially-aware incentive mechanisms for mobile crowdsensing," *IEEE Transactions on Wireless Communications*, vol. 18, no. 1, pp. 724–738, 2018.

[46] H. Sedghani, D. Ardagna, M. Passacantando, M. Z. Lighvan, and H. S. Aghdasi, "An incentive mechanism based on a stackelberg game for mobile crowdsensing systems with budget constraint," *Ad Hoc Networks*, vol. 123, p. 102626, 2021.

[47] L. Xiao, Y. Li, G. Han, H. Dai, and H. V. Poor, "A secure mobile crowdsensing game with deep reinforcement learning," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 1, pp. 35–47, 2017.

[48] G. Ji, Z. Yao, B. Zhang, and C. Li, "A reverse auction-based incentive mechanism for mobile crowdsensing," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8238–8248, 2020.

[49] Y. Wen, J. Shi, Q. Zhang, X. Tian, Z. Huang, H. Yu, Y. Cheng, and X. Shen, "Quality-driven auction-based incentive mechanism for mobile crowd sensing," *IEEE transactions on vehicular technology*, vol. 64, no. 9, pp. 4203–4214, 2014.

[50] Y. Liu, C. Xu, Y. Zhan, Z. Liu, J. Guan, and H. Zhang, "Incentive mechanism for computation offloading using edge computing: A stackelberg game approach," *Computer Networks*, vol. 129, pp. 399–409, 2017. Special Issue on 5G Wireless Networks for IoT and Body Sensors.

[51] Y. Chen, Z. Li, B. Yang, K. Nai, and K. Li, "A stackelberg game approach to multiple resources allocation and pricing in mobile edge computing," *Future Generation Computer Systems*, vol. 108, pp. 273–287, 2020.

[52] F. Yang, J. Yan, Y. Guo, and X. Luo, "Stackelberg-game-based mechanism for opportunistic data offloading using moving vehicles," *IEEE Access*, vol. 7, pp. 166435–166450, 2019.

[53] F. Li, H. Yao, J. Du, C. Jiang, and Y. Qian, "Stackelberg game-based computation offloading in social and cognitive industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5444–5455, 2020.

[54] Y. Qiao, Y. Li, and J. Li, "An economic incentive for d2d assisted offloading using stackelberg game," *IEEE Access*, vol. 8, pp. 136684–136696, 2020.

[55] 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.

[56] 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)*, pp. 815–820, IEEE, 2019.

[57] 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), pp. 278–288, IEEE, 2020.

[58] 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.

[59] 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.

[60] N. R. Draper and H. Smith, *Applied regression analysis*, vol. 326. John Wiley & Sons, 1998.

[61] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[62] C. A. Floudas, *Deterministic global optimization: theory, methods and applications*, vol. 37. Springer Science & Business Media, 2013.

[63] G. Xu, "Global optimization of signomial geometric programming problems," *European Journal of Operational Research*, vol. 233, no. 3, pp. 500–510, 2014.

[64] "Global optimization of signomial geometric programming using linear relaxation," *Applied Mathematics and Computation*, vol. 150, no. 1, pp. 99–114, 2004.

[65] E. Burnell, N. B. Damen, and W. Hoburg, "Gpkit: A human-centered approach to convex optimization in engineering design," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pp. 1–13, 2020.

[66] M. J. Kochenderfer and T. A. Wheeler, *Algorithms for optimization*. Mit Press, 2019.

[67] F. Eisenbrand, "Fast integer programming in fixed dimension," in *Algorithms - ESA 2003* (G. Di Battista and U. Zwick, eds.), (Berlin, Heidelberg), pp. 196–207, Springer Berlin Heidelberg, 2003.

[68] "Interior-point methods," *Journal of Computational and Applied Mathematics*, vol. 124, no. 1, pp. 281–302, 2000. Numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations.

[69] Y. T. Lee and A. Sidford, "Path finding i: Solving linear programs with o(sqrt(rank)) linear system solves," *arXiv preprint arXiv:1312.6677*, 2013.

[70] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[71] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[72] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research),"

[73] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, pp. 1273–1282, PMLR, 2017.

[74] M. S. Allahham, S. Sorour, A. Mohamed, A. Erbad, and M. Guizani, "Energy-efficient device assignment and task allocation in multi-orchestrator mobile edge

learning," in *2021 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2021.

Appendix A

Lemma 1

First, we define the separation function between the concave term and its under-estimator as follows:

$$\Delta(x) = -e^x - (-L(x)) \tag{A.1}$$

$$= -e^x + \left(\frac{x_{max} e^{x_{min}} - x_{min} e^{x_{max}}}{x_{max} - x_{min}} + \frac{e^{x_{max}} - e^{x_{min}}}{x_{max} - x_{min}} x \right) \tag{A.2}$$

The, we derive the first and second derivatives:

$$\frac{d\Delta}{dx} = -e^x + \frac{e^{x_{max}} - e^{x_{min}}}{x_{max} - x_{min}} \tag{A.3}$$

$$\frac{d^2\Delta}{dx^2} = -e^x \tag{A.4}$$

It is readily obvious that $\Delta(x)$ is concave since its second derivative is always negative, and by setting $\frac{d\Delta}{dx} = 0$, its maximum point can be given as:

$$x^* = \log \left(\frac{e^{x_{max}} - e^{x_{min}}}{x_{max} - x_{min}} \right) \tag{A.5}$$

by plugging it in the separation function, we can get the maximum separation value as follows:

$$\Delta(x^*) = -\frac{e^{x_{max}} - e^{x_{min}}}{x_{max} - x_{min}} + \frac{x_{max}e^{x_{min}} - x_{min}e^{x_{max}}}{x_{max} - x_{min}} + \frac{e^{x_{max}} - e^{x_{min}}}{x_{max} - x_{min}} \times \log\left(\frac{e^{x_{max}} - e^{x_{min}}}{x_{max} - x_{min}}\right) \tag{A.6}$$

$$= e^{x_{min}}\left(\frac{1 - e^{x_{max}-x_{min}}}{x_{max} - x_{min}} + \frac{x_{max} - x_{min}e^{x_{max}-x_{min}}}{x_{max} - x_{min}} + \frac{e^{x_{max}-x_{min}} - 1}{x_{max} - x_{min}} \log\left(\frac{e^{x_{min}}(e^{x_{max}-x_{min}} - 1)}{x_{max} - x_{min}}\right)\right) \tag{A.7}$$

By considering $\vartheta = x_{max} - x_{min}$ we can have:

$$= e^{x_{min}}\left(\frac{1 - e^{\vartheta}}{\vartheta} + \frac{x_{max} - x_{min}e^{\vartheta} + (e^{\vartheta} - 1)x_{min}}{\vartheta} + \frac{e^{\vartheta} - 1}{\vartheta} \log\left(\frac{e^{\vartheta} - 1}{\vartheta}\right)\right) \tag{A.8}$$

$$= e^{x_{min}}\left(\frac{1 - e^{\vartheta}}{\vartheta} + 1 + \frac{e^{\vartheta} - 1}{\vartheta} \log\left(\frac{e^{\vartheta} - 1}{\vartheta}\right)\right) \tag{A.9}$$

Finally, by considering $Z = \frac{e^{\vartheta}-1}{\vartheta}$, we can have:

$$\Delta(x^*) = \Delta_{max} = e^{x_{min}}\big(1 - Z + Z\log(Z)\big) \quad \square \tag{A.10}$$

Appendix B

Lemma 2

The sub-problem **SP3** in (32) can be explicitly expressed as:

$$\min_{G_o,\tau_o} \cdot \quad \frac{\alpha}{E_{max}|\mathcal{L}_o|}\sum_{l\in\mathcal{L}_o} G_o(\zeta^2_{l,o}\tau_o n_{l,o} + \zeta^1_{l,o} n_{l,o} + \zeta^0_{l,o}) + \frac{(1-\alpha)c1}{U_{max}\tau_o G_o} \tag{B.1a}$$

$$s.t. \quad G_o(A^2_{l,o}\tau_o n_{l,o} + A^1_{l,o} n_{l,o} + A^0_{l,o}) \leq T_{max},\ \forall\, l \in \mathcal{L}_o \tag{B.1b}$$

$$1 \leq \tau_o \leq \tau_{max} \tag{B.1c}$$

$$G_o \geq 1 \tag{B.1d}$$

Constraint (47b) represents the time constraints for each learner. However, we can substitute these constraint by a single one by considering the learner l^* with the maximum training time $l^* = \arg\ \max_{l\in\mathcal{L}_o}\ t_{l,o}$. Then, by using the following notations :

$a = \frac{(1-\alpha)c1}{U_{max}}$, $\quad b = \frac{\alpha\sum_l \zeta^2_{l^*,o} n_{l^*,o}}{E_{max}|\mathcal{L}_o|}$

$c = \frac{\alpha\sum_l \left(\zeta^1_{l^*,o} n_{l^*,o} + \zeta^1_{l^*,o}\right)}{E_{max}|\mathcal{L}_o|}$, $\quad \theta = \frac{A^2_{l^*,o} n_{l^*,o}}{T_{max}}$

$\xi = \frac{(A^1_{l*,o} n_{l*,o} + A^0_{l*,o})}{T_{max}}$

The sub-problem **SP3** in (32) can be re-written as the following:

$$\min_{G_o, \tau_o} . \quad \frac{a}{\tau_o G_o} + b\tau_o G_o + cG_o \tag{B.2a}$$

$$s.t. \;\; \theta\tau_o G_o + \xi G_o \leq 1 \tag{B.2b}$$

$$1 \leq \tau_o \leq \tau_{max} \tag{B.2c}$$

$$G_o \geq 1 \tag{B.2d}$$

Afterwards, we assume the learner with the maximum training time takes his full time to train such that:

$$\theta\tau_o G_o + \xi G_o = 1 \tag{B.3}$$

So we can have the following equation:

$$\tau_o G_o = \frac{1 - \xi G_o}{\theta} \tag{B.4}$$

By utilizing the above equation, the problem in (48) can be re-expresses as a single variable optimization problem $\mathcal{F}(G_o)$ as follows:

$$\min . \;\; \mathcal{F}(G_o) = \frac{a\theta}{1 - \xi G_o} + (c - \frac{b\xi}{\theta}) G_o \tag{B.5}$$

$$s.t. \;\; G_o \geq 1 \tag{B.6}$$

Next, we derive the first and second derivatives of the objective function:

$$\frac{d\mathcal{F}}{G_o} = \frac{a\theta\xi}{(1-\xi G_o)^2} + c - \frac{b\xi}{\theta} \tag{B.7}$$

$$\frac{d^2\mathcal{F}}{G_o^2} = \frac{2a\theta\xi^2}{(1-\xi G_o)^3} \tag{B.8}$$

From the above derivation, we can see that the second derivative is positive when $G_o < \frac{1}{\xi}$, and hence, the objective function is convex. The optimal point can be found by setting the first derivative to zero as follows:

$$\frac{a\theta\xi}{(1-\xi G_o^*)^2} + c - \frac{b\xi}{\theta} = 0 \tag{B.9}$$

$$\frac{a\theta^2\xi}{(1-\xi G_o^*)^2} + c\theta - b\xi = 0 \tag{B.10}$$

$$(1-\xi G_o^*)^2(c\theta - b\xi) + a\theta^2\xi = 0 \tag{B.11}$$

$$(1-\xi G_o^*)^2 = \frac{a\theta^2\xi}{b\xi - c\theta} \tag{B.12}$$

$$G_o^* = \left\lfloor \frac{1 - \sqrt{\frac{\xi a\theta^2}{b\xi - \theta c}}}{\xi} \right\rfloor \tag{B.13}$$

where we ignored the negative root since the value of $(1-\xi G_o^*)$ has to be positive, and floored the value since the number of global cycles is an integer. Moreover, one can see that for this solution to be feasible the following conditions must be satisfied $\beta\xi - \theta c > \xi a\theta^2$. Lastly, the number of local training iterations can found using (50) and considering its maximum as follows:

$$\tau_o^* = \min\left(\left\lfloor \frac{1-\xi G_o^*}{\theta G_o^*} \right\rfloor, \tau_{max}\right) \quad \square \tag{B.14}$$

Appendix C

Lemma 3

It is readily obvious that the optimization problem **P2** is a Linear Program, and hence, it follows directly that it is concave. Moreover, the term $\zeta^2_{l,o}\tau_{l,o} - \zeta^1_{l,o}$ in the objective is the variable linear coefficient, and it can represent the net utility given the orchestrator's strategy. As a result, if the net utility is positive, the learner will try to maximize it by maximizing the amount of participation until the time frame period is finished. Conversely, if the net utility is negative, the learner will not participate in the learning process. Thus, we can have constraint (14b) as equality in case of participation, and with some rearrangements we have the following:

$$\tau_{l,o} = \frac{1 - G_o(A^1_{l,o} n_{l,o} + A^0_{l,o})}{n_{l,o} G_o A^2_{l,o}} \tag{C.1}$$

However, we know from that in order to participate the following condition $\tau_o \geq \frac{\zeta^1_{l,o}}{\zeta^2_{l,o}}$ must hold. Therein, we have:

$$\frac{1 - G_o(A^1_{l,o} n_{l,o} + A^0_{l,o})}{n_{l,o} G_o A^2_{l,o}} \geq \frac{\zeta^1_{l,o}}{\zeta^2_{l,o}} \tag{C.2}$$

and by rearranging the terms, we can have the following upper bound on the amount of participation:

$$n_{l,o} \leq \frac{\zeta_{l,o}^2(1 - G_o A_{l,o}^0)}{G_o(\zeta_{l,o}^1 A_{l,o}^2 + \zeta_{l,o}^2 A_{l,o}^1)} \tag{C.3}$$

Finally, since we know the utility function (14a) for the learner has to be positive, we can have the following lower bound:

$$n_{l,o} \geq \frac{\zeta_{l,o}^0}{\zeta_{l,o}^2 \tau_{l,o} - \zeta_{l,o}^1} \tag{C.4}$$

If the amount of participation is less than the lower bound, it will result in a negative utility, hence the learner will not participate. On the other hand, the learner cannot participate with more than n_{max}, which is the maximum amount specified by the orchestrator. □

Appendix D

Lemma 4

In order to prove the concavity of the problem **P3**, we have to show that the Hessian matrix of the utility function is negative semi-definite. In practice, we can be set set the parameter $c2$ to 1, and assume $P_{max} = 1$ and $\frac{c1}{F_{max}|\mathcal{L}_o|} = 1$. For the ease of reading, we will present the variables without the subscripts, i.e., $\tau_{l,o}$ as τ and $n_{l,o}$ as n...etc. Moreover, we present $\frac{dn_{l,o}}{d\rho_{l,o}}$ as n', and to avoid confusions we introduce the following variables:

$\mathrm{a} = A^2$, $\mathrm{b} = A^1$, $\mathrm{c} = A^0$

$\mathrm{A} = \zeta^2$, $\mathrm{B} = \zeta^1$

$\alpha = \frac{C_o}{R_{max}}$, $\beta = \frac{\mu C_o f_l}{E_{max}}$ such that: $\zeta^2 = \alpha\rho - \beta$

Therein, we can have the following first derivatives:

$$\frac{\partial U}{\partial \tau} = -GC\rho n + \frac{1}{\tau^2 G} \tag{D.1}$$

$$\frac{\partial U}{\partial \rho} = -GC\tau(\rho n' + n) \tag{D.2}$$

where:

$$n = \frac{A(1 - Gc)}{G(aB + bA)} \tag{D.3}$$

$$n' = \frac{Ba\alpha(1 - Gc)}{G(aB + bA)^2} \tag{D.4}$$

Afterward, we can have the second derivatives as follows:

$$\frac{\partial^2 U}{\partial \tau^2} = -\frac{2}{\tau^3 G} \tag{D.5}$$

$$\frac{\partial^2 U}{\partial \rho^2} = -GC\tau(2n' + \rho n'') \tag{D.6}$$

$$\frac{\partial U}{\partial \tau \rho} = \frac{\partial U}{\partial \rho \tau} = -GC(\rho n' + n) \tag{D.7}$$

where:

$$n'' = \frac{-2a\alpha^2 bB(1 - Gc)}{G(aB + bA)^3} \tag{D.8}$$

We can notice that $n' \geq 0$ and $n'' \leq 0$. After that, we can express the Hessian matrix as follows:

$$\mathbf{H} = \begin{bmatrix} -\frac{2}{\tau^3 G} & -GC(\rho n' + n) \\ -GC(\rho n' + n) & -GC\tau(2n' + \rho n'') \end{bmatrix} \tag{D.9}$$

For $\mathbf{H}$ to be negative semi-definite, its determinant and sub-determinant have to be

negative, where the sub-determinant $-\frac{2}{\tau^3 G}$ is obviously negative. Therein:

$$\begin{aligned} det(\mathbf{H}) &= det \begin{vmatrix} -\frac{2}{\tau^3 G} & -GC(\rho n' + n) \\ -GC(\rho n' + n) & -GC\tau(2n' + \rho n'') \end{vmatrix} \\ &= \frac{2C}{\tau^2}(2n' + \rho n'') - G^2C^2(\rho n' + n)^2 \underset{?}{\leq} 0 \\ \frac{2C}{\tau^2}(2n' + \rho n'') &\underset{?}{\leq} G^2C^2(\rho n' + n)^2 \\ 2(2n' + \rho n'') &\underset{?}{\leq} \tau^2 G^2 C \left((\rho n')^2 + 2\rho n' n + n^2\right) \\ 4n' &\underset{?}{\leq} \tau^2 G^2 C \left((\rho n')^2 + 2\rho n' n + n^2\right) - 2\rho n'' \end{aligned} \tag{D.10}$$

Since $n' \geq 0$, $n'' \leq 0$ and $\tau, G \geq 1$, it is sufficient to show the following:

$$4n' \leq \tau^2 G^2 C 2\rho n' n \tag{D.11}$$

or simply:

$$\tau GC\rho n \geq 2 \tag{D.12}$$

which completes the proof. □