JOINT TASK PLANNING AND RESOURCE ALLOCATION IN MOBILE EDGE LEARNING SYSTEMS

By

AMR ABUTULEB

A thesis submitted to the Graduate Program in Department of Electrical and Computer Engineering in conformity with the requirements for the Degree of Master of Applied Science

Queen's University

Kingston, Ontario, Canada

August, 2021

Dedication

FOR THE SAKE OF ALLAH

to my parents

Abstract

Mobile Edge Learning (MEL) has recently emerged as a paradigm to enable distributed parallelized learning (PL) and federated learning (FL) on resource-constrained wireless edge devices. The development of distributed learning is a result of the fact that the number of devices connected to IP networks is increasing exponentially. Though the individual computing powers of these devices may be limited, their collective power is potentially unlimited. This unlimited yet under-utilized computing power coupled with MEL is the future technology for application and host serving. In this work, we aim to jointly optimize the planning of the learning process and tasks and the allocation of physical resources for mobile edge learning scenarios with global training time constraints. The term emlearning task planning refers to the number of local iterations in each of these global cycles and the number of data samples to be used for training by each device within each local iteration. The allocation of physical resources involves the determination of the computing speeds of each device and its communications resources to the MEL orchestrator. We discuss the problem of jointly optimizing the learning task planning and physical resources with two different objectives. In objective 1, we provide a solution to maximize the number of local training cycles each device executes within a given time constraint, which was shown to achieve a faster convergence to the desired learning accuracy. In objective 2, we provide a solution to minimize the global loss function of the training process by the end of the global training time constraint. Where we propose two novel algorithms, the dynamic and the static algorithm to solve the problem. We finally show the performance of each objective compared to the results of optimizing the Task Planning (TP). Where we optimize the planning parameters and the allocation of tasks across the system and Equal Data Allocation (EDA), where we optimize the planning parameters and the physical resources allocation across the system.

Acknowledgments

First, I would like to thank my Lord gratefully, I would never accomplish this work without his endless blessings. I would like to express my gratitude to Prof. Hossam Hassanein and Dr. Sameh Sorour for their unrelenting support, supervision, mentorship, and guidance in teaching me the intricacies of research. I would like to thank Prof. Hossam Hassanein for offering me the chance to come to Queen's University as an intern in my undergraduate years before giving me the chance to come as a full time MASc student. I would also like to thank Dr. Sameh Sorour specifically for keeping up with my mistakes and always guiding me to the right direction whenever i was on the wrong track.

Dr. Magdy Abutaleb and Dr. Amany Negm, my role models in everything I do in my life. You are my parents, my friends and my teachers. Thank you for teaching me everything I know in life, for keeping up with my mistakes and always encouraging me to be a better person and for always having my back whether you are in the room next to me or thousand of miles away. Thank you for making me the man I'm today.

My brother, Ahmed Aboutaleb. Thank you for having my back in the last two years, for helping me in my masters journey and for encouraging me whenever I was stressed out or disappointed. Thank you for all the lovely meals that you cooked throughout the last two years whenever I was too busy, sick or frustrated to cook (I will forever love your Bamya). Could not have asked for a better brother/flatmate.

My Fiancé, Norhan Amgad. Thank you for being there through the rough sailing before being there in the calm waters, thank you for your support, assistance and your encouragement through this journey.Your presence in my life has always been a great blessing (Alhamdullah), Love you Coco.

My friends, Mohamed Ibrahim and Mohamed Waly for their continuous support and lovely phone calls throughout the last two years.

I would also like to thank all my TRL colleagues for making my academic experience awesome. Special thanks to Sherif Azmy for his help in bouncing ideas, always having a solution to my LaTeX problems and for the long beautiful walks by the lake. I also wish to thank Basia Palmer for her patient and accurate review of this thesis and her useful suggestions.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

4G	Fourth Generation Communication Technology
5G	Fifth Generation Communication Technology
BS	Base Station
CPU	Central Processing Unit
DL	Distributed Learning
DNN	Deep Neural Networks
DP	Data Parallelism
EDA	Equal Data Allocation
FL	Federated Learning
GD	Gradient Descent
GPR	Ground Penetrating Radar
ILPQC	Integer Linear Program with Quadratic Constraints
IoT	Internet of Things
KKT	Karush-Kuhn-Tucker
LTE	Long-Term Evolution
MEL	Mobile Edge Learning
ML	Machine Learning
MNIST	Modified National Institute of Standards and Technology

MP	Model Parallelism
NLCLP	Linear Integer Program with Nonlinear Constraints
NN	Neural Networks
NP	Non-deterministic Polynomial-time
OFDMA	Orthogonal Frequency Division Multiple Access
OPTI	Optimization Interface
PARTEL	Partitioned Edge Learning
PL	Parallelized Learning
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
TP	Task Planning
Wi-Fi	Wireless Fidelity

List of Symbols

K	Number of Learners
D	Total number of samples available
x_j	Number of features in sample j
y_j	Target value for sample j
$f(x_j, y_j, w)$	Loss function of sample j
d_k	Number of samples used by learner k
F	Feature vector size of sample j
P_d	Number of bits representing each feature
S_d	Number of data-size dependent model parameters
S_m	Number of data-size independent model parameters
P_m	Number of bits representing each parameter
τ	Number of iterations performed in one global cycle
L	Total iterations performed in the learning process
ω	Parameter used to apply the convergence bounds
$w(l)$	Local model parameter vector after the l^{th} iteration
C_m	Number of processor flops consumed per iteration
N	Number of subcarriers in the system
n	Number of subcarriers allocated to learner k

$c_{k,n}$	Number of bits needs to be delivered on subcarrier n
BER	Target bit error rate
N_o	Noise power spectral density
$h_{k,n}$	Channel gain between the orchestrator and learner k
T_s	OFDMA symbol duration
f_k	CPU flop speed of learner k
G	Total number of global cycles performed
η	Learning rate of the model
ϵ	Lower bound on $F(w[L]) - F(w^*)$
ρ	Meta parameter related to ρ-Lipschitz assumption
β	Meta parameter related to β-smoothness assumption

Chapter 1

Introduction

1.1 Overview and Motivation

The Cisco annual internet report [1] projects that the number of devices connected to IP networks (e.g., sensors and wearable devices) will be more than three times the global population, growing from 22 billion in 2020 to 29.3 billion by 2023. Though the individual computing powers of these devices may be limited, their collective power is potentially unlimited, in most cases, highly under-utilized. On the other hand, these devices typically produce large amounts of data [2], a large portion of which is private not to shared thus exposing it to a barrage of privacy breaches [3]. Even for non-private data, the cost of sending them to cloud data centers for analytics is prohibitive bandwidth and delay wise standpoint. As a result, it is estimated that over 90% of this data will be stored and processed among these devices themselves. These facts motivated the emergence of several novel mobile edge learning (MEL) paradigms at the network edge.

The idea behind MEL is the distribution of the learning process over multiple mobile edge nodes (a.k.a. as learners), by training a common model on these learners and aggregating the model parameters obtained from them at one master node (a.k.a. orchestrator) to compute the final aggregated model. MEL is thus clearly the solution to enable learning across devices baring private datasets without sharing these datasets with other nodes, thus maintaining their privacy. This defines the sub-paradigm of MEL known as federated learning (FL). On the other hand, MEL is also most suited for resource-constrained devices to offload their non-private data to neighboring devices (or even private date to trusted neighboring nodes) to parallelize the learning over multiple of these resource-constrained nodes, thus speeding up the learning process and reducing their consumed resources in it. This introduces the other sub-paradigm of MEL known as parallelized learning (PL). Though being clearly different in nature and purpose, it can be inferred that FL can be procedure-wise viewed as a special case of PL in which datasets are already hosted at the learners, and need not be conveyed to these learners along with the common learning model (as in PL). Otherwise, the learning procedure is very similar.

Figure 1 illustrates the two sub-paradigms and shows the differences between the PL and FL systems.

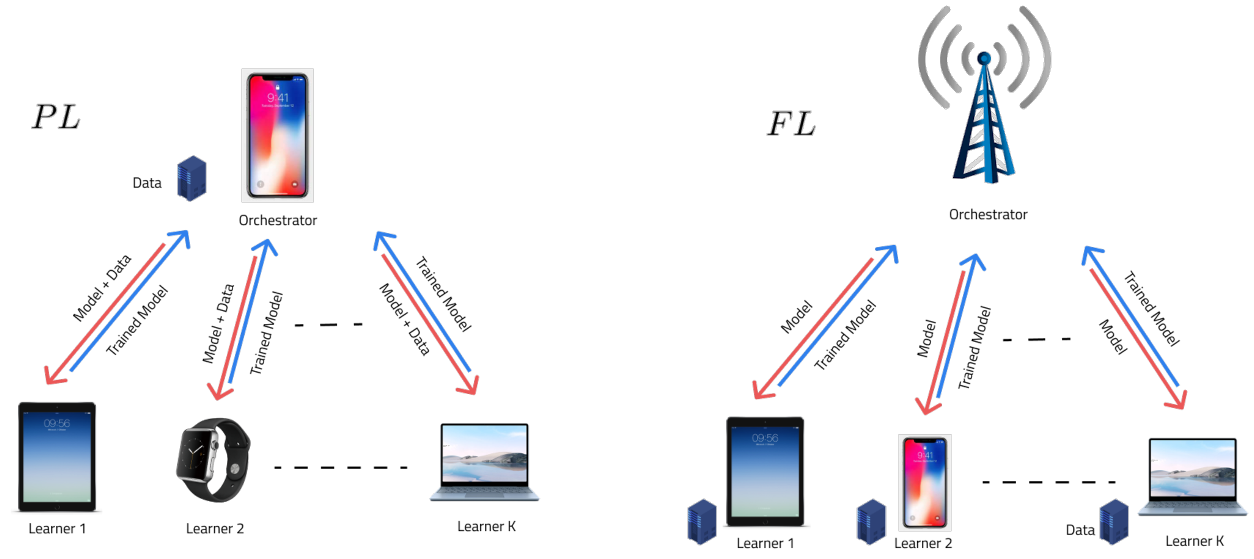

Figure 1.1: The architecture of a MEL system in the two different sub-paradigms

1.2 MEL Learning Problem

MEL can be viewed as a distributed learning (DL) system that has been moved to the heterogeneous wireless and mobile edge setting. First, the orchestrator initiates the learning process by distributing the machine learning (ML) model on a set of learners (distributes ML model and data in PL). Next, each learner runs a number of local training iterations (which we will refer to local iterations for short), in each of which it applies the gradient descent approach to their local model using a number of data samples (whether sent by the orchestrator in PL or taken from the stored data points at the learner in FL). After a certain duration and/or number of local iterations, each learner sends that model back to the orchestrator to compute the aggregated model. These aforementioned steps constitute one global cycle. These global cycles is repeated until the time constraint is met or until the desired accuracy/desired loss function target is achieved. The time needed to send the model and receive the weights is known as the transmission time, while the time needed to apply the gradient descent approach is known as the execution time.

The total size of tasks allocated to each learner is usually preset by the orchestrator based on it's computational capabilities, the desired accuracy/loss function target and the time constraint of the training/learning process. The total number of local iterations and global cycles (a.k.a planning parameters) performed by all the learners, and the number of data samples each learner will use in its training per global cycle (a.k.a task allocation) will impact both the computation time and the transmission time. On the other hand, the communication bandwidth and compute power (a.k.a resource allocation) will impact the transmission time. The MEL learning problem is defined as the problem of evaluating the planning parameters and the allocation of tasks and resources to achieve the desired accuracy or to minimize the global loss function.

1.3 Challenges

MEL comes with a wide range of challenges, most of these challenges are shared between MEL and edge computing. We next describe three core challenges associated with MEL, including bottleneck, heterogeneity, and privacy.

1. **Costly Communication:** Communication is a critical bottleneck in MEL networks especially in federated learning. Where networks consists of a massive number of devices, this causes sluggishness in the communication and also makes it slower than local computation by many orders of magnitude [4]. In order to successfully fit a machine learning model to data on these devices, it would be efficient to have model updates or small messages sent throughout the learning process. Also, in parallelized learning concerns regarding the cost of sending a complete batch of the allocated data across the network to every learner arise.

2. **Systems Heterogeneity:** In a practical MEL system, heterogeneity is a challenge due to the variability in every aspect of the network. For example, each device would have differences in hardware, power needs and consumption, and connectivity causing incompatibility. Heterogeneity rise up as a problem when trying to allocate the tasks across the learners, this means that each learner needs to get the exact number of tasks that they can process in a given time constraint. Additionally, allocation of resources is important and also depends on the learners heterogeneity. Availability of the connected devices is another aspect to consider, for example hundreds of active devices in a million-device network [5]. Therefore, these aspects must be taken into consideration when designing a fully functional MEL system.

3. **Privacy Concerns:** Moving around sensitive data to run a centralized learning process has always been one of the challenges facing MEL. Privacy issues are of major concern in edge learning, with parallelized learning this issue becomes even greater as raw data is sent from the orchestrator to the learners across the system. On the other hand, federated learning mitigates this issue by sending only model updates across the networks, as in gradient information and model parameters [6, 7].

1.4 Objectives and Contributions

This work aims to solve the MEL learning problem by optimizing the planning of the learning process and tasks, and the allocation of physical resources for mobile edge learning in both FL and PL. We will compare the joint optimization performance with the performance of optimizing the task planning (TP), where we optimize the planning parameters and the allocation of tasks across the system and equal data allocation (EDA), where we optimize the planning parameters and the physical resources allocation across the system respectively. Our two main objectives are summarized below.

Objective 1: We consider the problem of joint task and resource allocation for both PL and FL, where we set a time constraint on the time of one global cycle. The aim of these adaptive allocations is to maximize the number of local updates within the constrained duration of a global cycle, which was shown to achieve a faster convergence to the desired learning accuracy. We evaluate our solution's performance with the TP or EDA solutions of the same problem, for the MNIST dataset.

Objective 2: Given a global training time constraint, we consider the problem of jointly optimizing the planning of the learning tasks and the allocation of physical resources over a network of MEL devices using a multicarrier scheme (e.g., OFDMA) for multiple access. We chose OFDMA because of its dominance as a multiple access scheme in the majority of current wireless communication networks (e.g., 5G, Wi-Fi 6). With the aim of minimizing the overall loss function within this global time constraint, the planning of the learning task involves the determination of the number of global learning cycles within the constrained global training time, the number of local iterations in each of these global cycles, and the number of data samples to be used for training by each device in each of these local iterations. On the other hand, the allocation of physical resources involves the determination of the computing speeds of each device and its allocated OFDMA subcarriers and bit loading on them. We evaluate the performance of our solution through simulations covering the MNIST dataset, various machine learning models, and different system configurations with varied numbers of learners. We also show that the proposed joint algorithms outperform the TP or EDA solutions of the same problem.

1.5 Thesis Outline

The thesis is organized as follows: Chapter 2 provides a comprehensive literature review of MEL, in particular the MEL system model and architecture. Chapter 3 provides our system model, problem formulation, proposed solution and simulation results for objective 1. Chapter 4 provides our system model, problem formulation, proposed solution and simulation results for objective 2. Chapter 5 concludes and sheds light on future work.

Chapter 2

Mobile Edge Learning: System Model, Entities, and Literature Review

In this section we present the MEL system model, describe the different entities of a MEL system and review the current literature on MEL and it's limitations.

2.1 MEL System Model

In this subsection, we elaborate on the system model of MEL. Firstly, we talk briefly about machine learning (ML). Secondly, we extend the ML discussion to distributed learning (DL) and lastly, we elaborate on how DL when moved to the edge settings is known as MEL.

2.1.1 Machine Learning

Machine learning (ML) algorithms are designed to automatically solve a problem, based on having been already trained on an available data set. There are three types of problems that ML can solve, which can be described as:

1. **Supervised Learning:** Supervised learning is learning a function where there is a set of inputs that relates to a set of outputs [8–10]. Supervised learning algorithms receive a dataset with a labeled feature column, where each sample in the dataset has a corresponding label or ground truth. Supervised learning can be further divided into classification and regression:

    - **Classification:** Classification models approximate a mapping from the input variables to a discrete output variable [11–13]. A classification model classifies the inputs into one of two or more classes. The performance of the model is often measured by classification accuracy.

- **Regression:** Regression models approximate a mapping from the input variables to a continuous output variable [14–16]. The model's performance is measured in terms of errors made in the model's predictions.

2. **Unsupervised Learning:** Unsupervised learning algorithms receive a set of input variables with no output variable or label [17–19]. The objective of unsupervised learning is to learn the underlying structure of the data and find an efficient representation of the data. Two unsupervised learning tasks are clustering and dimensionality reduction:

   - **Clustering:** The task of clustering is the collection of datapoints that share similar features into one cluster [20–22]. This learning task also helps in identifying and matching a new datapoint to a cluster.
   - **Dimensionality Reduction:** The idea of dimensionality reduction is to project samples from a high-dimensional space onto a lower-dimensional space without losing more than 10% of the projected information [23–25], reducing the complexity of the data while retaining its structure.

3. **Reinforcement Learning:** Reinforcement learning is a class of machine learning where an agent interacts with the environment [26–28]. The goal of reinforcement learning is to train the agent to choose the optimum action for a given environmental state, that yields the highest reward

In general, every supervised learning task consists of a training phase, validation phase and a testing phase.

At the beginning of a ML learning process, the data is usually distributed as 70% for training, 15% for validation and 15% for testing. For training, the ML model parameters keep being updated using an iterative procedure to minimize the loss function. Before the testing phase, there is usually a validation phase. In this phase, part of the training dataset is hidden from the model while training. The hidden part is given to the model after it has finalized its training. How well the model performs in the validation phase, is generally a good indicator of how well this model will perform with unseen data. Usually a threshold is set for the validation accuracy, if the model does not achieve that threshold, the model retrains and test again until it's validation accuracy is above the desired accuracy. Finally, in the testing phase, the model is given the test dataset and the output of the model is compared to the actual targets(The true value of each data sample). The model's performance in the testing phase is gauged by the level of accuracy or the F1-Score.

In a typical ML process, the model will be trained on a data set comprised of D samples. Each sample will have its own set of features, which acts as an input to the ML model. In a typical ML process, the model will be trained on each data sample d_n for $n = 1, ..., D$, where each sample will have its own set of F features that can be denoted by x_j where $j = 1, ..., F$. The set of features that belong to each sample n can be denoted as $x_n = \{x_1, ..., x_j, ..., x_F\}$.

In ML, the objective can be summarized as using a set of parameters w to find a relation between x_n and the target value y_n such that the loss function $F(x_n, y_n, w)$ or $F_n(w)$ is minimized. From this, the total loss over the entire dataset can be defined as:

$$F(w) = \frac{1}{D}\sum_{n=1}^{D} F_n(w) \tag{2.1}$$

It's often difficult to find an analytical solution to the loss equation. Therefore, an iterative gradient descent approach is used so that the model parameter set at any discrete time-step l, for $l = 1, ..., L$ relates to the previous time-step and the gradient of the loss as:

$$w[l] = w[l-1] - \eta \nabla F(w[l-1]) \tag{2.2}$$

The learning rate represented by η is usually set on the interval (0, 1) and influences the convergence rate and the final accuracy. Typically, the ML model will pass over the entire dataset and this complete pass is known as an epoch.

The output of the ML algorithm differs depending on the application, it may be a real number, a set of integers, or a binary number. In binary classification, the output is 0 or 1 (zero or one), which tells us whether a data sample belongs to a category or class. In multi-class classification, the output may be an integer representing the class label or a set of real numbers representing the probability of a data sample belonging to a class. In regression, curve fitting or time-series prediction, the output is simply a real number. Typically, a loss function is used to quantify the difference between the actual target and the ML model output.

2.1.2 Distributed Learning

Many ML techniques, including regression, support vector machine (SVM) and neural networks (NN) built on gradient-based learning. Because iterative approachs can be computationally intensive for a single device, especially if it has low computation power, DL has been introduced. Training an ML model on a large dataset in a distributed manner where subsets of the data are located across multiple learners or data parallelism (DP) is one plausible scenario for DL. Another scenario is, model parallelism (MP) which is training very large models in a distributed manner on one dataset co-located at each learner. Although both options apply to the wireless edge, most of the discussions focus on the DP scenario

In DL, there is usually one centralized controller or orchestrator that has a specific problem that it solves (e.g. classification, prediction, image segmentation). First, the orchestrator distributes the ML model on a set of learners $\mathcal{K} = \{1, .., k, ..., K\}$, where each learner k may have a locally owned dataset of size d_k or it may be supplied by the orchestrator. Next, the orchestrator initiates the learning process, sending a global model w to each learner k. Each learner then applies the gradient descent approach to their local model w_k, then sends that model back to the orchestrator to start the aggregation process. One such cycle is known as a global cycle.

$$w_k[l] = w_k[l-1] - \eta \nabla F_k(w_k[l-1]) \tag{2.3}$$

The local model parameter set at learner k is given by w_k, the local loss is given by F_k, and η is the learning rate. At time-step l, the local model $w_k[l]$ depends on the model $w_k[l-1]$ at the previous time-step l, the gradient of the local loss $\nabla F_k(w)\ \forall k \in \mathcal{K}$ can be calculated using the local dataset of size d_k as [29]:

$$F_k(w_k) = \frac{1}{d_k}\sum_{n=1}^{d_k} F_n(w_k) \tag{2.4}$$

The global model parameters will only be visible to learners after a global aggregation is performed, this can happen at any arbitrary time-step l. For that particular time-step, $w_k = w \forall k \in \mathcal{K}$. There are two scenarios for aggregation, the synchronous case where all learners return their model parameters after τ time-steps, and the asynchronous case where the global aggregation will occur after different τ_k for each learner k. Our work focuses on the synchronous case as it has been proven that it usually yields the better results [30]. The global optimal model for both scenarios can be obtained by applying the following aggregation mechanism [29]:

$$w[l] = \frac{1}{D}\sum_{k=1}^{K} d_k w_k[l] \tag{2.5}$$

The orchestrator can perform multiple global cycles until the objective is reached, whether it is a specific loss function value or a pre-set accuracy threshold or even if it keeps on working until the resources such as multi-core processors are no longer available.

2.1.3 MEL Description

Motivated by having a highly computational job on a low computational device and privacy concerns among data owners, the concept of MEL was introduced which refer to the transitioning of the DL setting to the heterogeneous device setting. This transition will be performed by defining the parameters related to the heterogeneity of the computing and communication capacities of wireless edge nodes (a.k.a. learners), and how they relate to the steps of the global update clock duration. Two scenarios for DL's data parallelism are Federated Learning (FL) and parallelized learning (PL).

- **Federated Learning**: Different datasets are already stored at the different learners. This scenario is motivated by the high cost and privacy risks of moving data across the network [31].

- **Parallelized Learning**: The orchestrator initially possesses the entire dataset and decides to distribute fragments of it to learners for learning purposes. This scenario is motivated by the limited computational capabilities of edge IoT devices that need to learn from own collected data. This limitation drives them to parallelize their learning jobs on multiple near-by and usually trusted learners (e.g., a home local network).

Figure 2 illustrates PL and FL and shows the difference between them [32].

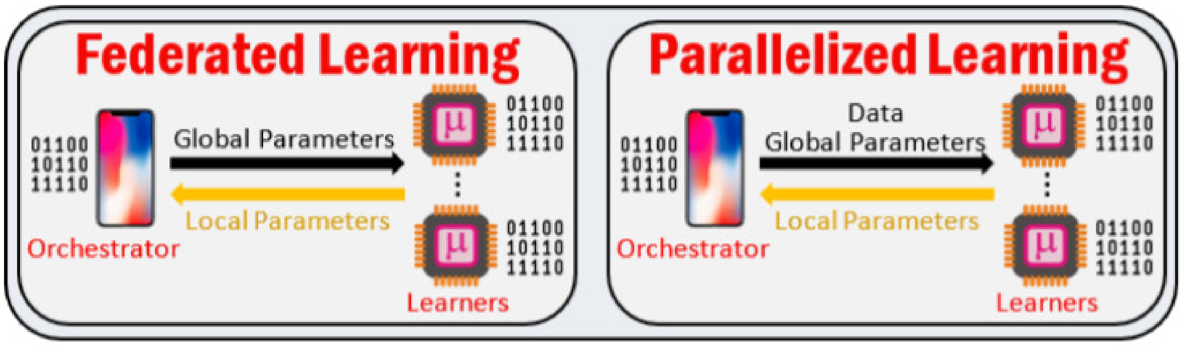

Figure 2.1: The difference between FL and PL.

Though being different in nature and purpose, it can be inferred that FL can be procedure-wise viewed as a special case of PL in which datasets are already hosted at the learners, and need not be conveyed to these learners along with the common learning model (as in PL). Otherwise, the learning procedure is very similar. Therefore, we will be considering PL in our work and mentioning the variations in the model if FL was the paradigm chosen.

Consider a MEL system comprising of K learners, where each learner trains its local learning model to minimize the loss function on a batch size of d_k samples. The total size of all batches is $D = \sum_{k=1}^{K} d_k$, which is usually preset by the orchestrator based on it's computational capabilities, the desired accuracy and the time constraint of the training/learning process. Number of local iterations performed by all the learners in the synchronous system is equivalent. There are many variables that can impact the time and may also impact the accuracy:

- The number of local updates will directly impact the execution time
- The dataset size will impact both the execution time and the transmission time
- The resource allocation will impact the transmission time.

To summarize, the global update process in MEL occurs in periodic cycles, that we will refer to as the global update cycles. This process should include the following steps:

1. Transmission of the global model to each learner $k \in \mathcal{K}$
2. Computation of τ local update cycles at each learner k and sending back their locall model to the orchestrator
3. Global aggregation at the orchestrator.

The orchestrator will typically demand the results within a pre-set duration in which all three steps are completed. To this end, in the following two subsections we discuss the entities of a typical MEL system and discuss some of the previous work done on MEL and its limitations.

2.2 MEL Entities

MEL systems for both FL and PL consists of the same entities which is MEL orchestrator and MEL learner, both of which are defined as:

1. **MEL Orchestrator:** The MEL orchestrator is responsible for assigning tasks and physical resources to the MEL learners. Also, retrieving and aggregating the model

parameters produced by each MEL learner, and finally, sending back the aggregated model to the MEL learners to continue the MEL learning process.

2. **MEL Learner:** The MEL learner is responsible for running a machine learning model on it's assigned datapoints. It's usually a device registered as a MEL learner on the system and has it's own set of data in FL or receives the data that it needs to work on from the MEL Orchestrator in PL.

Revisiting Figure 1, figure 3 illustrates the two entities and also shows the differences between the PL and FL systems.

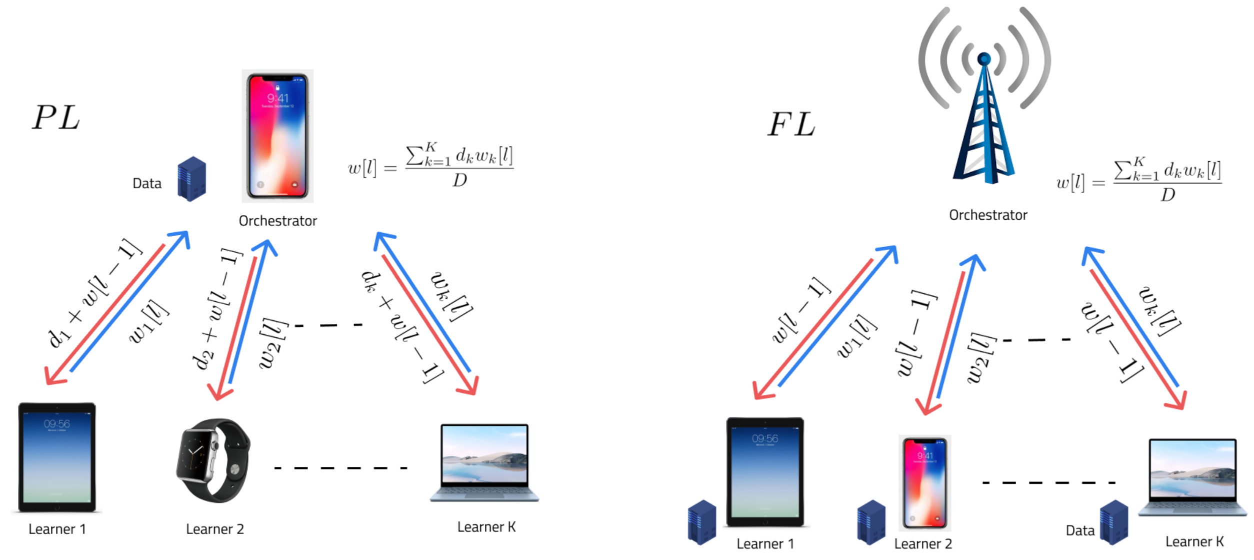

Figure 2.2: The architecture of the PL and FL algorithms that is executed over a wireless network in iteration l

2.3 Literature Review

Though distributed learning has been widely investigated in wired and non-heterogeneous computing and communication environments [33, 34], recently attention has shifted towards leveraging distributed learning in wireless, heterogeneous, edge communication and computing environments. This led to the emergence of FL, PL, and MEL. In [35], comprehensive surveys discussed the design of FL algorithms and introduced various solutions to enhance

FL effectiveness. Yet, these works did not optimize the distribution of learning tasks nor have they addressed the relationship between the computing/communication resources and learning accuracy.

The work in [36] introduced partitioned edge learning (PARTEL), an MEL paradigm that partitions the learning global model into local models, is sent to different learners while considering their computing and communication heterogeneities. Yet, this model partitioning approach is limited only to learning models with decomposable loss functions (e.g., logistic regression), and is not easily applicable to models lacking such property (e.g., convolutional neural networks).

Another interesting study [37] focused on client scheduling and resource block allocation, where this problem was formulated to minimize the training loss and the channel state information uncertainties. They resorted to GPR-based channel prediction methods and derived an upper bound for the loss of accuracy in FL which minimized the loss. However, the analysis does not investigate whether training times are affected by different channel statistics.

The authors in [38] propose a federated learning-based optimization model design and analysis for the wireless network. they considered the sum learning and transmission energy minimization problem for FL, for a case in which all users transmit learning results to the BS. However, their solution requires all users to upload their learning model synchronously and also they did not provide any convergence analysis for FL.

In [39], to maximize the data rates of users authors used FL algorithms for traffic estimation. While interesting, they assumed that wireless networks can readily integrate FL algorithms. However, in practice, wireless channels and wireless resources are not always available where we not only can encounter training errors due to the wireless links, but also wireless resource limitations arises (e.g., in terms of bandwidth and power).

The global aggregation frequency varied dynamically in a networked system which was presented in [40]. Therefore, the authors tried to minimize the network traffic by doing the aggregation step only when the model parameters changed beyond an empirically chosen threshold. However, there is no theoretical analysis performed to understand how the threshold values affects the learning. this approach only aims at reducing the network traffic, which is insufficient in MEL systems where the computation resource is also limited.

The work in [41] focused on the number of local iterations in resource-constrained edge environments to maximize the accuracy. This work was the first to jointly optimize the number of local iterations and global update cycles made in the learning process. They started by formulating a problem to minimize the loss function, where it was generally impossible to find an exact analytical expression to relate τ which is the number of local iterations in one global cycle and T which is the total time available for the learning process with $F(w(T))$ because it depends on the convergence property of gradient descent. Although it is generally difficult, they were able to analyze the convergence rate upper bound of the gradient descent by making the following assumptions:

1. $F_i(w)$ is convex
2. $F_i(w)$ is ρ-Lipschitz, i.e., $\|F_k(w) - F(w)\| \leq \rho\|w - w'\|$ for any w, w'
3. $F_i(w)$ is β-smooth i.e., $\|\delta F_k(w) - \delta F(w)\| \leq \beta\|w - w'\|$ for any w, w'

These assumptions facilitated their solution, and they were able to get a closed form expression for the problem. However, they overlooked the heterogeneities in the system and assumed having an identical learner system, this is never the case in MEL as heterogeneity is one of the core challenges that this paradigm faces.

The works in [42,43], aimed at jointly optimizing the learning and communication parameters to minimize the training completion time and the energy consumed in [42] and the loss function for the learning process in [43]. Convergence time minimization was also investigated in [44] for FL in wireless edge environments. Nonetheless, all these works focused only on FL and assumed that each learner must use all its stored data samples to train its model. They did not adapt the number of data samples used for training by the different learners according to their physical capabilities, which may be of critical importance especially in learning tasks with local or global training time deadlines. They also did not adopt realistic physical layer settings of modern wireless networks (e.g., OFDMA systems).

Another thread of research considering the adaptation component, mentioned above as part of task planning for both FL and PL, was presented in [45]. This work aimed to maximize the number of local iterations and adapt the number of samples used for training in each time-constrained global cycle. The work in [46] extended the work in [45] to a more general global training time-constrained setting, to optimize task planning (including data size adaptation) to minimize the training loss function. Details of both works can be presented as:

- In [45], the authors were able to formulate an integer linear program with quadratic constraints (ILPQC), which is well known to be NP-hard. They were able to calculate an upper bound on the relaxed problem using KKT conditions by calculating the Lagrangian function of the relaxed problem. They were able to prove their solution to the problem matches the original problem through extensive simulations.

- In [46], the authors proved that their problem was convex by using the convergence bounds derived in [41] and applying those convergence bounds to their problem, this solved the problem using basic numerical solvers. Furthermore, they were able to dynamically update their allocation by relating the loss function to the NN model updated after each global cycle.

Although these works [45,46] addressed the heterogeneity of the learners while allocating their tasks, they were limited in the sense that they adapted to only the task planning of the learners and did not try to jointly optimize the physical resources alongside the task planning.

Chapter 3
Objective 1: Maximizing the Number of Local Iterations

In this work, we consider the joint task and resource allocation problem for both time-constrained PL and FL to maximize the number of local updates τ within the constrained duration of a global aggregation cycle. To this end, we first study the system settings, problem formulation and then present our proposed solution. Finally, we illustrate our simulation results and compare them to our baselines TP and EDA.

3.1 System Settings

This work considers a system consisting of one orchestrator and K learners that perform one FL or PL job in a mobile edge environment. We will first start by describing the learning and data settings and parameters and then discuss the parameters and implications of executing this learning job in the considered mobile edge environment.

3.1.1 Learning and Data Model

The learning setting considered in this work consists of any arbitrary DL model (e.g., linear regression, support vector machine, K-means, deep neural network) that can employ a stochastic gradient descent (SGD) approach to train the K local models deployed by the orchestrator on the network's K learners. Unlike Gradient Descent (GD) approaches that require exhaustive iteration on all stored data samples to do one single update of the learners' parameter (a.k.a. weight) vectors, SGD enables cycles of training using randomly selected samples from the training set to update these vectors in each of these cycles [47]. SGD is more suitable to both FL and PL as it allows the adaptation of allocated tasks to the learners according to their capabilities and resources. It also fits the PL concept as the orchestrator can distribute randomly selected sets of data samples to each learner in each cycle. Luckily, it has been proven that SGD often converges much faster than GD, though the error function may not be as well minimized as in the case of GD. However, the close approximation obtained using SGD for the parameter values is usually enough in most cases, particularly in mobile edge environments where speed is typically more critical than absolute accuracy [48].

The learning process for each global aggregation cycle occurs in three steps:

Step 1: The orchestrator conveys the global learning model parameters to the learners. For PL, this step also involves sending a set of d_k data samples to each learner $k \in \{1, \ldots, K\}$. Defining F as the number of features in the data set, and P_d as the precision of each data feature (i.e., the number of bits representing each feature), the total number of sent data bits in PL can be expressed as:

$$B_k^{data} = d_k F P_d \tag{3.1}$$

note that in FL, these bits are not sent by the orchestrator, but rather each learner k selects a random set of d_k data samples from its own stored data set to use them in the training process explained in Step 2. Each data sample i assigned/selected by learner k is defined by the input-output pair $\{x_i^k, y_i^k\}_{i=1}^{d_k}$, $k \in \{1, \ldots, K\}$. In addition, defining S_d and S_m as the data-size dependent and data-size independent model parameters, and P_m as the precision of the learning model (i.e., number of bits representing each parameter/weight), the total number of sent bits to convey the employed FL or PL model is:

$$B_k^{model} = P_m(d_k S_d + S_m) \tag{3.2}$$

Step 2: Each learner starts training the received model with the d_k received (in PL) or selected (in FL) data samples for τ local cycles. The goal of the training is to minimize the global loss function of the model expressed as:

$$L_{global} = \sum_{k=1}^{K} \sum_{i=1}^{d_k} f\left(\underline{w}_k, x_i^k, y_i^k\right) \tag{3.3}$$

where $\underline{w}_k \in \mathbb{R}^{B_k^{model}}$ is the local model parameter vector and $f(.)$ is a loss function in building the relationship between an x_i^k and y_i^k through $\underline{w}_k$. To achieve this target, each learner needs to C_m flops to execute the training calculations per data sample in each local cycle, resulting in a total of:

$$X_k = d_k C_m \tag{3.4}$$

flop computations per local cycle. In a typical SGD algorithm, it has been shown that L_{global} is a decreasing function of τ, i.e., minimizing the loss function can be efficiently achieved by maximizing the number of learning iterations [49], which translates in our setting to maximize the number of local cycles.

Step 3: At the end of each global cycle of duration T, the orchestrator collects the local model parameter vectors from all learners and aggregates them to build the global model parameter vector. One popular method for such aggregation is the weighted averaging approach expressed as:

$$\underline{w} = \frac{\sum_{k=1}^{K} d_k \underline{w}_k}{D} \tag{3.5}$$

where $D = \sum_{k=1}^{K} d_k$ defines the total number of samples that need to be analyzed in each global cycle, this is usually imposed by the orchestrator given the considered learning job.

Once these three steps are completed, the orchestrator chooses to stop the process, typically, if it converged to the desired level of accuracy, or start another cycle. Interested readers in the local/global loss function minimization and local parameter aggregations are referred to [50] for more details.

3.1.2 Mobile Edge Settings

From the mobile edge environment viewpoint, the three learning steps and settings must be performed by wireless/mobile edge devices. These steps physically translates into three different time epochs to complete the above three steps of each global update cycle. The first epoch represents the time needed to send the model and data (in PL) to each learner given their channel characteristics. If learner k is assigned forward bandwidth B_k^F and transmit power P_k^F. This expression is for PL. For FL, the value of d_k in the first term of the numerator is set to zero. This epoch will take:

$$t_k^S = \frac{d_k F P_d + P_m(d_k S_d + S_m)}{B_k^F log_2(1 + \frac{P_k^F h_k}{N_o})} \tag{3.6}$$

for learner k, where h_k is the power gain of the channel between the orchestrator and learner k. The second epoch represents the duration taken by each learner k to finish all its assigned computations to generate $\underline{w}_k$. If learner k has a CPU flop speed of f_k, this duration is equal to:

$$t_k^C = \frac{\tau X_k}{f_k} = \frac{\tau d_k C_m}{f_k} \tag{3.7}$$

the third and final epoch represents the time needed for each learner to send back its $\underline{w}_k$ to the orchestrator. If the assigned reverse bandwidth and power to learner k are B_k^R and P_k^R, the duration of this epoch is:

$$t_k^R = \frac{P_m(d_k S_d + S_m)}{B_k^R log_2(1 + \frac{P_k^R h_k}{N_o})} \tag{3.8}$$

3.2 Problem Formulation

This work aims is to minimize the global loss function in each global cycle by maximizing the number of local cycles in each global aggregation cycle, which should result in the maximum possible learning accuracy at the end of this global cycle [49]. This goal will be achieved by jointly optimizing the tasks allocated to each learner (i.e., $d_k \ \forall \ k$) and its assigned physical resources (i.e., $f_k, B_k^F, B_k^R, P_k^F, P_k^R \ \forall \ k$) to maximize τ. Thus, the general form of this optimization problem can be expressed as follows:

$$\max_{\substack{\tau, d_k, f_k, B_k^F, B_k^R, \\ P_k^F, P_k^R, \ \forall \ k}} \quad \tau \tag{3.9}$$

$$
\begin{aligned}
\text{s.t.} \qquad & t_k^S + t_k^C + t_k^R \leq T, \quad \forall\, k && \text{(3.9a)}\\
& \sum_{k=1}^{K} d_k = D && \text{(3.9b)}\\
& \sum_{k=1}^{K} B_k^F \leq B && \text{(3.9c)}\\
& \sum_{k=1}^{K} P_k^F \leq P && \text{(3.9d)}\\
& 0 \leq f_k \leq f_k^{max}, \quad \forall\, k && \text{(3.9e)}\\
& 0 \leq B_k^R \leq B_k^{R,max}, \quad \forall\, k && \text{(3.9f)}\\
& 0 \leq P_k^R \leq P_k^{R,max}, \quad \forall\, k && \text{(3.9g)}
\end{aligned}
$$

The constraints in (3.9a) guarantee that the total time of the three process steps will not exceed the preset global cycle duration T for any of the learners. Constraint (3.9b) ensures that the total no. of samples analyzed by all learners conforms with the bound D set by the orchestrator for each global cycle. Constraints (3.9c) and (3.9d) assures that the total forward bandwidths and powers used by the orchestrator to complete Step 1 do not exceed its total bandwidth B and power budgets (denoted by B and P, respectively). Finally, the K constraints in (3.9e) ensure that each learner does not exceed its maximum flop speed given its computational capabilities or allowance given its other loads. Similarly, the constraints in (3.9f) and (3.9g) ensure that each learner does not exceed its maximum reverse bandwidth nor transmit power, respectively, when returning its local model parameter vector to the orchestrator.

As per the above description, the considered problem is a linear integer program with nonlinear constraints (NLCLP), which is well known to be NP-Hard [51]. Therefore, solving this optimization problem is challenging, even when using numerical solvers, some simplification or reduction of variables is required.

3.3 Proposed Solution

As in several prior works [43,52], the optimal values of several of the optimization parameters can be directly obtained from the formulation. For instance, maximizing τ is directly impacted by setting any variable to minimize each time expression t_k^S, t_k^C, and t_k^R in Constraints (3.9b). This simple fact can result in the following determinations of the optimal values of some variables, thus eliminating them and their constraints from the problem:

- By examining the expression of t_k^C in (3.7), we can see that it is minimized for every learner k by setting its f_k to its maximum possible value. By looking at the constraints in (3.9e), we can conclude that the optimal value for f_k is to set it to f_k^{max} $\forall$ k.
- By examining the expression of t_k^R in (3.8), it can be inferred that it is minimized for every Learner k by setting its B_k^R and P_k^R to the maximum possible values, which are defined in Constraints (3.9f) and (3.9g) to be $B_k^{R,max}$ and $P_k^{R,max}$, respectively.

The above facts enable the removal of the parameters from the set of optimization variables (3.9) and the set of constraints (3.9e), (3.9f), and (3.9g); decreasing the size of the problem as follows:

$$\max_{\substack{\tau, d_k, B_k^F, \\ P_k^F \ \forall\ k}} \quad \tau \tag{3.10}$$

$$\text{s.t.} \quad t_k^S + t_k^C + t_k^R \leq T, \quad \forall k \tag{3.10a}$$

$$\sum_{k=1}^{K} d_k = D \tag{3.10b}$$

$$\sum_{k=1}^{K} B_k^F \leq B \tag{3.10c}$$

$$\sum_{k=1}^{K} P_k^F \leq P \tag{3.10d}$$

clearly, Problem (3.10) represents the joint optimization of allocated tasks to learners and their assigned resources from the orchestrator to achieve the maximum possible τ. Constraints (3.9a), (3.9b), (3.9c), and (3.9d) are the immediate equivalent to those in (3.10a), (3.10b), (3.10c), and (3.10d), respectively.

Though this problem is simpler than the one in (3.9), it is still very combinatorial. Finding closed-form expressions or approximate methods was not feasible for a simpler version of the problem in which only resources were optimized [42]. Thus, we use a numerical solver, namely the OPTI solver [53], to find the solution for the problem and identify its gains compared to the two related works TP and EDA.

3.4 Simulation Results

In this section, we present the simulation results of our joint task and resource allocation solution. We also compare our solution's performance against two recent optimizations proposed in the literature.

- **Task Planning (TP)**: This scheme optimizes task planning only (i.e., optimizes τ and $d_k \; \forall \; k$) given that each learner has a fixed heterogeneous physical resource.
- **Equal Data Allocation (EDA)**: In this scheme, it assumed that all learners will analyze the same number of data samples (i.e., $d_k = d \; \forall \; k$). Given this setting, the scheme still optimizes the other task planning parameters (i.e., τ) along with the physical resources.

The two figures of merit that we use in our comparisons are the maximum achievable number of local cycles per global aggregation cycle (i.e., τ) and the achieved model accuracy at the end of each global aggregation cycle.

The dataset chosen to test our proposed scenarios is the MNIST [54] dataset which consists of 60,000 images where each image consists of 784 features. The employed neural network consists of three hidden layers with the following configuration [784, 300, 124, 60, 10] For this network, the model size was calculated to be 8,974,080 bits and the required

Table 3.1: Simulation parameters

Parameter	Value
System Bandwidth B	100 MHz
Node Bandwidth B_k^R	5 Mhz
Maximum B_k^F	$5K$ Mhz
Device Proximity	50 m
Node Power	23 dBm
Maximum P_k^F	$23K$ dBm
Noise Power Density N_o	-174 dBm/Hz
Attenuation Model	7+2.1log(R)dB[]
Computation Capability f_k	2.4 GHz and 700 MHz
MNIST dataset size D	60,000 images
MNIST dataset Features F	784 features

floating-points operations were 1,123,736. To ensure a fair comparison between all three schemes, the neural network was constructed from scratch in the simulation environment without using any predefined functions. This guarantees our ability to control different parameters and obtain the most accurate results for each of the three schemes without any impact from any hidden settings or variables.

From the physical perspective, the edge learners were divided into two groups, one simulating the computational capabilities of portable computing devices and the other simulating those of commercial micro-controllers. In addition, random distances (with a maximum distance of 50m) and fading conditions were generated for each learner with respect to the orchestrator. The employed channel model was chosen to emulate 802.11 links between the learners and the orchestrator. Table I summarizes the simulation parameters for both the physical resources/setup and the employed data set.

In Fig 3.1, τ is tested at different values of K for T=30s and T=60s. We observe that both sub-figures show that the gain between the three schemes remains almost the same as K increases. For instance, at K=10, T=30s the joint scheme performs five updates, the TP scheme performs three updates, and the EDA scheme performs two updates, resulting in a gain of 166% and 250%, respectively for the joint scheme over the TP and EDA schemes. When K was increased to 20, the gains remained in the range of 150% and 225%, respectively.

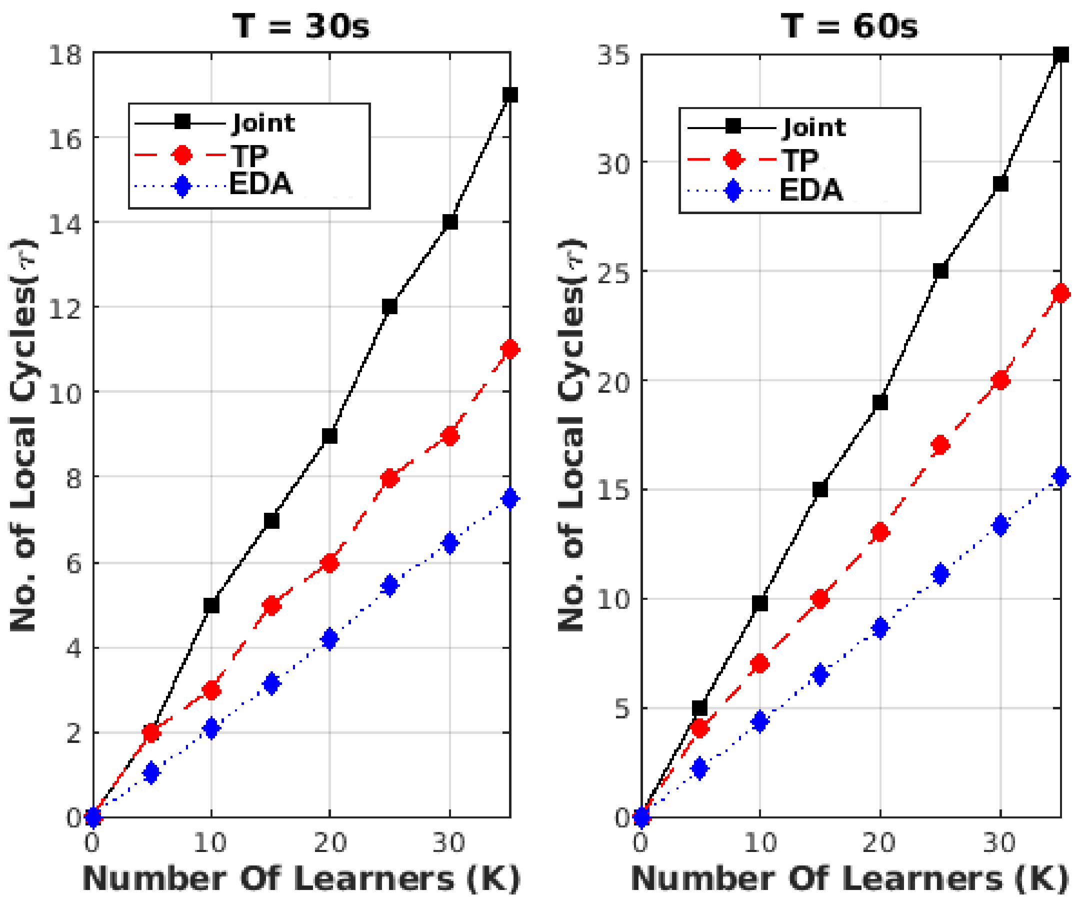

Figure 3.1: Number of local cycles for all schemes against K for T = 30 and 60s.

Same gain ranges were also obtained for T=60s. These results show the consistency of the gains of our joint scheme for different values for K. Another interesting observation is that the performance of the joint scheme at K=20 and T=30s exceeds the performance for the resource scheme at K=20 and T=60s, which means that the joint scheme can achieve better performance at less duration than the EDA scheme.

In Fig 3.2, τ is tested at different values of T for K=10 and K=20. Similar to Fig 3.1, the gains between the different schemes is almost the same as T increases. One key observation is that at K=10 and T=10, the resource-only scheme was not able to perform even one local update, while the other two schemes were able to perform the same number of updates. However, the joint scheme outperforms the task-only scheme as T increases.

In Fig 3.3, the progression of learning accuracy achieved by all three schemes at the end of

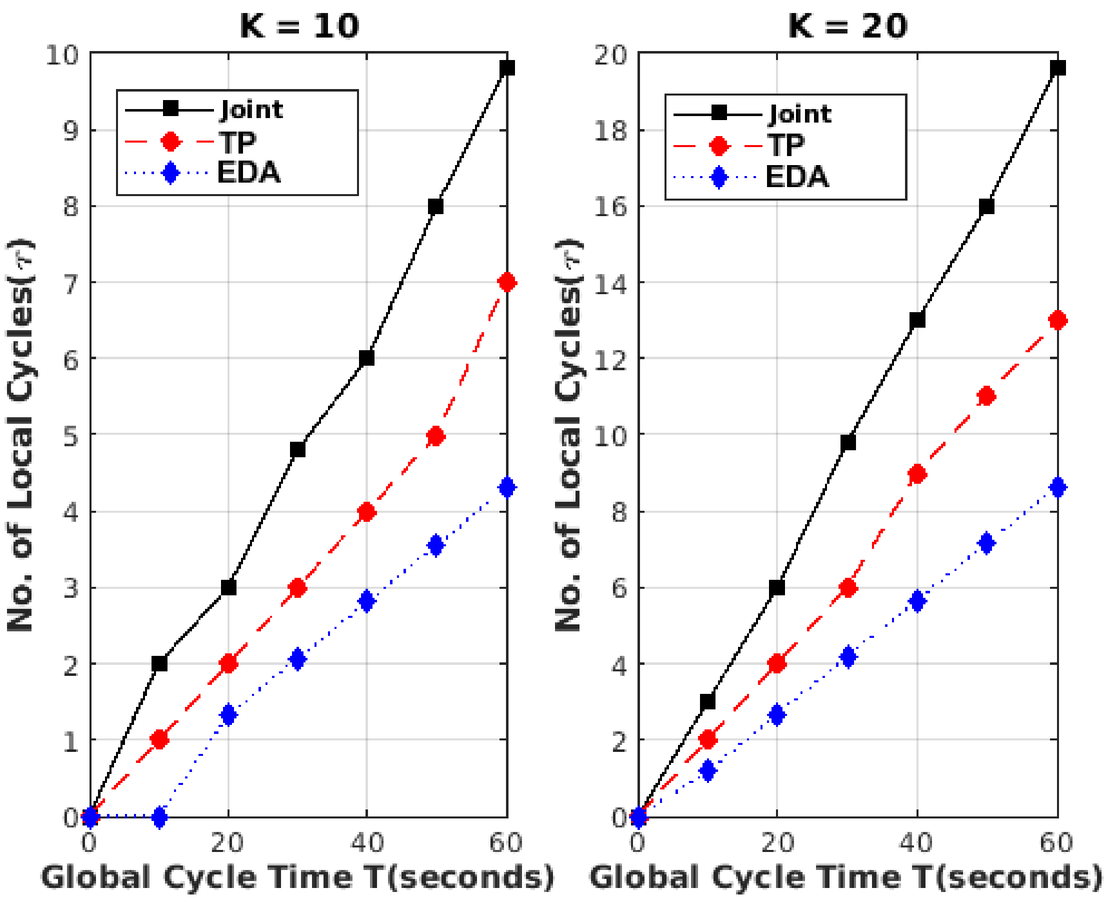

Figure 3.2: Number of local cycles for all schemes against T for K = 10 and 20.

each global cycle are plotted for T=30s and K = 10 and 20. The figure shows higher accuracy for the joint scheme, especially for low global cycle indices. As the learning progresses, the accuracy of all the schemes starts to be equivalent when K=20, but not K= 10. Yet, even for K = 20, the joint, TP, and EDA schemes reach 98% accuracy after three, four, and seven global cycles, thus, achieving a reduction of 25% (i.e., 30s) and 58% (i.e., 120s) as opposed to the other two schemes, respectively.

Finally, Fig 3.4 depicts the same progression of learning accuracy for K=20 and T = 12s and 30s. Again, the joint scheme achieves better accuracy than the other schemes at low global indices for both cycle durations. It also converges to 98% faster than the other two schemes. For T=12s, the joint, TP, and EDA schemes exceeded 98% accuracy after four, seven, and nine cycles, resulting in a reduction of 43% (i.e., 36s) and 56% (i.e., 60s) for

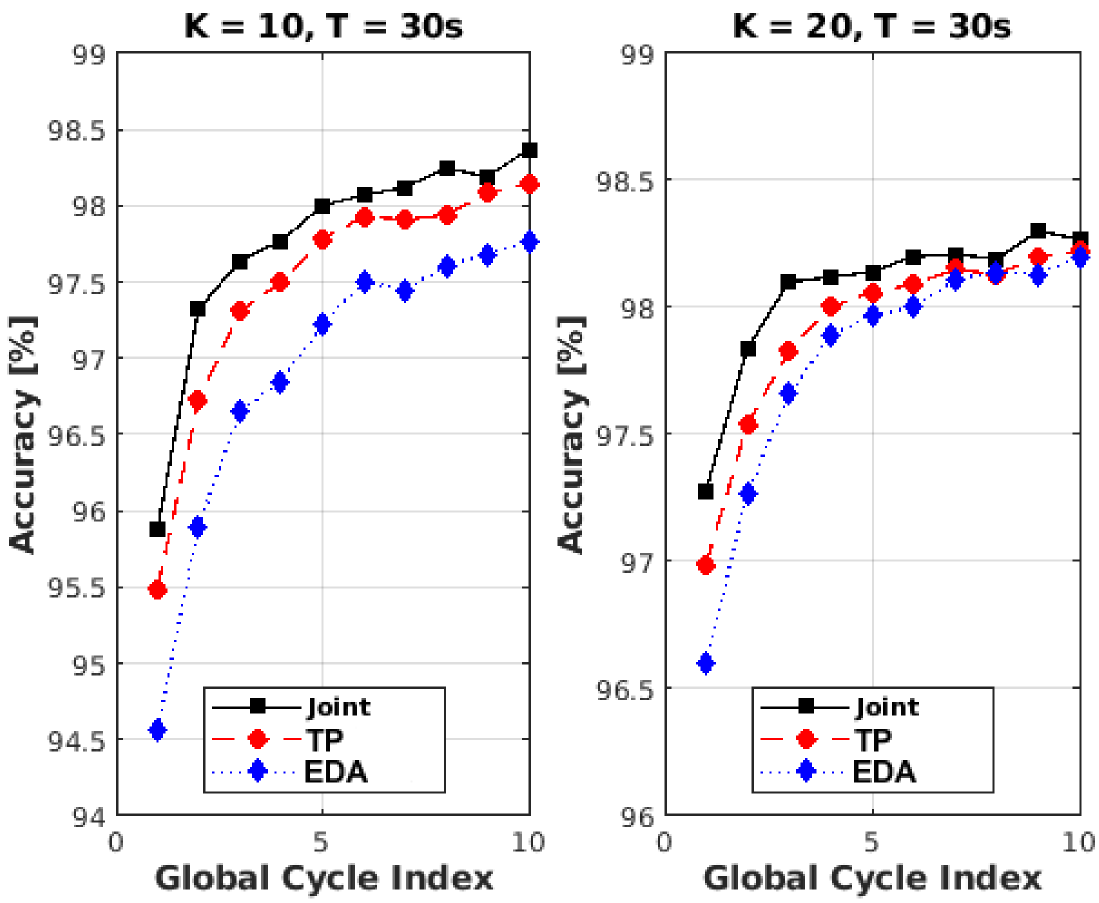

Figure 3.3: Learning accuracy achieved at the end of each global cycle for T=30s and K = 10 and 20.

the joint scheme over the TP and EDA schemes, respectively. More interestingly, the joint scheme needed 4 T=12s cycles to reach 98% accuracy, whereas the TP and EDA schemes needed four and seven T=30s cycles to reach the same accuracy. Thus, the joint scheme is more practical when the amount of available time for the learning is restricted.

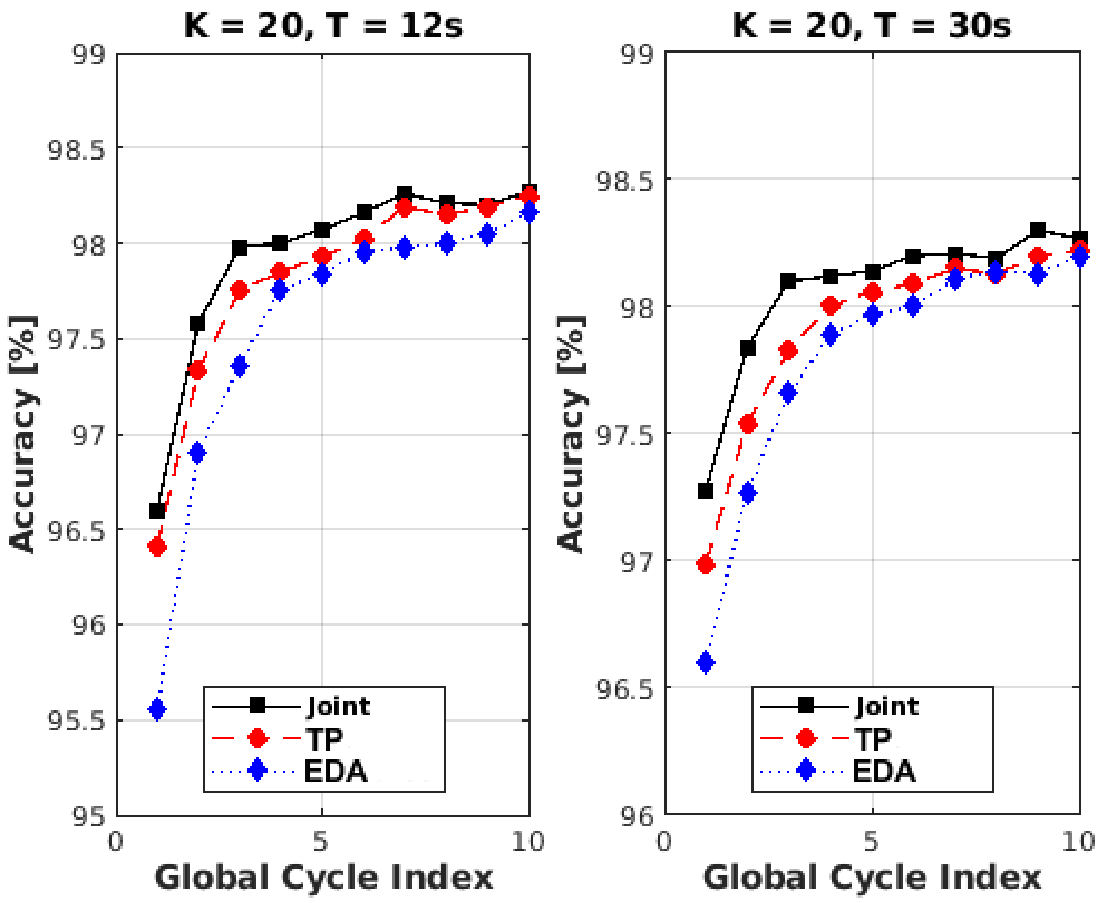

Figure 3.4: Learning accuracy achieved at the end of each global cycle for K=20 and T = 12s and 30s.

Chapter 4

Objective 2: Minimizing the Global Loss Function

This work considers jointly optimizing the planning of the learning tasks and allocating physical resources over a network of MEL devices using a multicarrier scheme for multiple access within a global training time constraint. We chose OFDMA as the multicarrier scheme because of its dominance as a multiple access scheme in current wireless communication networks (e.g., 5G, Wi-Fi 6). To minimize the overall loss function within a global time constraint, we begin by discussing the system model, formulating our problem, and proposing a solution. Next, we illustrate our simulation results and compare them to the two baselines TP and EDA. Lastly, we focus on the joint scheme in FL, comparing its performance using our proposed algorithms.

4.1 System Model and Parameters

The considered system consists of one edge orchestrator that aims to train a learning model in a distributed manner with the help of K heterogeneous and resource-constrained mobile edge learners. Due to the time critically of edge applications and the limited connectivity time between edge nodes (due to their high mobility), the available time to train this model is assumed to be constrained by a global learning time T. The training is done on distributed subsets of data. These subsets are either already collected and stored at the different learners, or distributed to them by the orchestrator to help it in the training. These two scenarios delineate the boundary between the two considered MEL approaches, namely FL and PL, respectively.

4.1.1 Model Training Preliminaries

A learning model usually consists of parameters (a.k.a. weights) that are calculated through a recursive training process. In each recursion, dataset samples are inserted into the model, and are used with the model parameters to compute an output (e.g., classification of objects or prediction of future values in a time series). The resulting error margin (a.k.a. the loss function) between this output and the ground-truth of the data samples (e.g., actual object classes or actual future values in the time series) is then used to update the model parameters before starting the next recursion. The training recursions are typically stopped whenever a target value of the loss function or (in our case) a global training time is reached.

Most recently, machine learning models, especially in edge environments, resorted to employing the more practical stochastic gradient descent (SGD) methods [55]. SGD enables the use of randomly selected samples from the dataset for training in each of these parameter update rounds. SGD is thus more suitable to MEL (for both FL and PL) as SGD allows the adaptation of the allocated tasks to the learners according to their capabilities and resources [47]. Consequently, many distributed learning machine learning models at the edge (e.g., linear regression, support vector machine, K-means, deep neural network) employ a stochastic gradient descent (SGD). Our work in this thesis thus can apply to all edge models.

The loss function is one of the most important merit metrics in machine learning, as it is used to determine how well the model is trained or more precisely how far it is from accurately identifying the target values of the dataset. For example, consider a dataset consisting of D samples that can be used to train a machine learning model, where each sample j, $j = 1, ..., D$, has a set of features denoted by x_j and a target value y_j. Throughout the learning process, the set of parameters w are updated to minimize the loss function, denoted by $f(x_j, y_j, w)$ or $f(w)$ for short.

4.1.2 Learning and Data Model

In this section, we use the same distributed learning process used in Chapter 3 to train the system's desired model. The utilized model has a few adjustments to adapt itself to our second objective, that is why we present the three steps and discuss the relevant changes.

Step 1: The orchestrator transmits the (initial or current) global learning model parameters to the K learners. In PL, a new set of d_k randomly selected data samples from the complete dataset are also sent to learner $k \in \{1, \dots, K\}$ along with the learning model. Define F as the feature vector size of x_j where $j = 1, ..., d$, and P_d as the precision of each data feature (i.e., the number of bits representing each feature). In addition, let S_d and S_m be the number of data-size dependent and data-size independent model parameters, and P_m as the precision of the learning model (i.e. number of bits representing each weight). Consequently, the number of bits sent to express the data and the model can be expressed as:

$$B_k^{data} = d_k F P_d \tag{4.1}$$

$$B_k^{model} = P_m(d_k S_d + S_m) \tag{4.2}$$

In FL, the datasets are already hosted at the learners, and thus Equation (1) becomes $B_k^{data} = 0$. Yet, each learner can choose d_k random sample from its hosted dataset for each global cycle, as advised by the task allocated to it by the orchestrator. Note that Equation (2) is the same for both PL and FL.

Step 2: After receiving the global model parameters (and data samples in PL), each learner executes τ local training iterations to parameterstrain these parameters initially or further on its local model. In each of these iterations, the model parameters are trained using all the d_k received or selected data samples in PL or FL, respectively, to minimize the local loss function by the end of this iteration, which is expressed as follows:

$$F_k(w[l]) = \sum_{j=1}^{d_k} f\left(w[l], x_j, y_j\right) \tag{4.3}$$

where $w[l] \in \mathbb{R}^{B_k^{model}}$ is the local model parameter vector after the l-th iteration and $f(.)$ is the loss function based on one feature. It is important to note here that, in most state-of-the-art MEL and FL works, the following assumptions are made about $F_k(w[l])$ with respect to l, to facilitate the analysis where l and l' are any two different training iteration indices. [41–43]:

- $F_k(w[l])$ is convex
- $F_k(w[l])$ is ρ-Lipschitz, that is, $\|F_k(w[l]) - F_k(w[l'])\| \leq \rho \| \, w[l] - w[l'] \, \|$
- $F_k(w[l])$ is β-smooth, that is, $\|\nabla F_k(w[l]) - \nabla F_k(w[l'])\| \leq \beta \| \, w[l] - w[l'] \, \|$

These assumptions hold for smooth-SVM and linear regression, which are ML models with a convex loss function. We show that these assumptions also hold for neural networks with ReLU activation, which is a model that has a non-convex loss function.

For a learner to minimize $f(.)$ in each iteration, it needs to perform training calculations per data samples, which consumes C_m flops of its processor per iteration. Consequently, the resulting total number of flops at learner k per iteration can be expressed as:

$$X_k = d_k C_m \tag{4.4}$$

Step 3: At the end of each global cycle, the learners send back their updated model parameters to the orchestrator, which are then aggregated by the latter to update the global model. There are several approaches to aggregate the received model parameters into the global parameters [56, 57]. We consider the weighted averaging approach, which can be expressed as:

$$w[l] = \frac{\sum_{k=1}^{K} d_k w_k[l]}{D} \tag{4.5}$$

where l is the index of the last iteration performed by each learner in the global cycle, and $D = \sum_{k=1}^{K} d_k$ defines the total number of samples considered in the training across all learners in each global cycle, D can be set as the size of either the entire dataset or, more generally, a subset of its samples whose size is pre-defined by the orchestrator given the stringency of the global time constraint. Afterwards, the orchestrator calculates the global loss function, which can also be expressed as:

$$F(w[l]) = \sum_{k=1}^{K} \sum_{n=1}^{d_k} f\left(w_k[l], x_j^k, y_j^k\right) = \frac{\sum_{k=1}^{K} d_k F_k(w[l])}{D} \tag{4.6}$$

After the end of the three steps of each global cycle, the orchestrator can start a new global cycle, by resuming Step 1. This process continues until the global training time T is reached, at this moment the orchestrator ends the training process. Let L be the total number of training iterations each learner performed from start to end of the entire training process (i.e., across all global cycles within the duration T).

All learners belong to the set $\mathcal{K} = \{1, \ldots, k, \ldots, K\}$.

4.1.3 Network Model

We assume OFDMA-based communications between the orchestrator and the learners. The N subcarriers of this OFDMA system must be partitioned between the different learners so that each learner receives the model (plus the data in PL) in Step 1 and deliver its parameters in Step 3 of each global cycle. The allocation indicator of subcarrier $n \in \{1, \ldots, N\}$ to learner k $n \in \{1, \ldots, N\}$ is defined as:

$$\alpha_{k,n} = \begin{cases} 1 & \text{If } n \text{ is allocated to } k \\ 0 & \text{Otherwise} \end{cases} \tag{4.7}$$

When subcarrier n is allocated to learner k, the orchestrator in Step 1 and learner k in Step 3 load $c_{k,n}$ bits of the information they need to deliver one another on this subcarrier. The power used at the sending party to transmit these $c_{k,n}$ bits on this subcarrier must be determined to achieve a target received bit error rate (BER) at the receiving party. If only square constellations are used for bit loading on this subcarrier (e.g., QPSK, 16-QAM, 64-QAM), this transmit power on subcarrier n by the orchestrator or learner k to achieve a target BER can be expressed as [**?**]:

$$p(c_{k,n}) = \frac{N_0}{3}\left[Q^{-1}(\frac{BER}{4})\right]^2 (2^{c_{k,n}} - 1) \tag{4.8}$$

where N_0 is the noise power spectral density, and $h_{k,n}$ is the channel gain of the n^{th} between the orchestrator and learner k. The total power used for the communication between the orchestrator and learner k is:

$$P_k = \sum_{n=1}^{N} \frac{p(c_{k,n})}{h_{k,n}} \alpha_{k,n} \tag{4.9}$$

and the total bit rate for this communication is:

$$r_k = \frac{1}{T_s} \sum_{n=1}^{N} c_{k,n}\alpha_{k,n} \tag{4.10}$$

where T_s is the OFDMA symbol duration. Assuming channel reciprocity, the same allocated subcarriers and bit loading levels will be used for all transmissions from the orchestrator to each learner k and vice versa, thus the transmission rate r_k will be identical in both transmission directions.

4.1.4 MEL Settings

This section, examines the joint implication of both the learning and network settings in terms of time. For every learner k, the time needed to execute the three steps of the global cycle consists of three main epochs each corresponding to one of the three steps.

Epoch 1 t_k^S encompasses the time needed for the orchestrator to send the model parameters, plus the new d_k randomly selected data samples in PL to each learner k. This time epoch can thus be expressed for PL as:

$$t_k^S = \frac{d_k F P_d + P_m(d_k S_d + S_m)}{r_k} \tag{4.11}$$

Note that, for FL, the value of d_k in the first term of the numerator is set to zero.

Epoch 2 t_k^C represents the duration taken by each learner k to finish all its assigned computations across all the τ local iterations of one global cycle in order to update its model parameters. If learner k has a CPU flop speed of f_k, this time epoch is equal to:

$$t_k^C = \frac{\tau X_k}{f_k} = \frac{\tau d_k C_m}{f_k} \tag{4.12}$$

Epoch 3 t_k^R is the amount of time required for each learner to send back its trained model parameters to the orchestrator after completing its local iterations. This epoch can be represented as:

$$t_k^R = \frac{P_m(d_k S_d + S_m)}{r_k} \tag{4.13}$$

4.2 Problem Formulation

4.2.1 Basic Formulation

As mentioned earlier, we aim to minimize the final loss function of MEL training, thus leading to higher accuracy, given a constraint on the global training time T. This goal is achieved by jointly optimizing both the planning parameters of the learning tasks, namely L, τ, and d_k $\forall$ $k \in \{1, \ldots, K\}$, and the parameters allocating resources to these learners, namely f_k, $\alpha_{k,n}$, and $c_{k,n}$ $\forall$ $k \in \{1, \ldots, K\}$ and $n \in \{1, \ldots, N\}$.

Let $G = L/\tau$ be the total number of global cycles performed in the learning process in the duration T. To simplify the analysis, let us assume that L is an integer multiple of τ, which means that G is an integer. The case where L deviates from being an integer multiple of τ and the case of time-varying computation and communication parameters is considered in our proposed algorithms. Furthermore, to simplify the problem formulation, we assume that the computation and communication-related parameters do not change throughout the learning process. Consequently, the times t_k^C, t_k^S and t_k^R $\forall$ k does not change along the duration T, thus the time needed for each learner k to perform all its training cycles within the global time constraint is:

$$t_k = \frac{L}{\tau}\left(t_k^S + t_k^C + t_k^R\right) \tag{4.14}$$

One important constraint of our considered problem is to have $max(t_k) \leq T$, which can be guaranteed when $t_k \leq T, \forall\, k$. Given these assumptions, the basic form of our optimization problem of interest can be expressed as:

$$\min_{\substack{L,\tau,d_k,f_k,c_{k,n}\\ \alpha_{k,n}\ \forall\ k}} \quad F(w[L]) \tag{4.15}$$

$$\text{s.t.} \quad \frac{L}{\tau}\left(t_k^S + t_k^C + t_k^R\right) \leq T, \quad \forall\ k \tag{4.15a}$$

$$\sum_{k=1}^{K} d_k = D \tag{4.15b}$$

$$\sum_{n=1}^{K} \alpha_{k,n} = 1, \quad \forall\ n \tag{4.15c}$$

$$\sum_{k=1}^{K}\sum_{n=1}^{N} \alpha_{k,n} = N \tag{4.15d}$$

$$\sum_{k=1}^{K}\sum_{n=1}^{N} \frac{p(c_{k,n})}{h_{k,n}}\alpha_{k,n} \leq P_{max} \tag{4.15e}$$

$$0 \leq f_k \leq f_k^{max}, \quad \forall\ k \tag{4.15f}$$

$$\tau \in \mathbb{Z}_+ \tag{4.15g}$$

$$L \in \mathbb{Z}_+ \tag{4.15h}$$

$$d_k \in \mathbb{Z}_+, \quad \forall\ k \tag{4.15i}$$

$$\alpha_{k,n} \in \{0,1\}, \quad \forall\ k,n \tag{4.15j}$$

$$c_{k,n} \in \mathcal{Q}, \quad \forall\ k,n \tag{4.15k}$$

where $F(w[L])$ is the loss function of the model after evaluating its parameters after the L training cycles. Constraint (4.15a) guarantees that the total time taken for the learning process does not exceed the total available time T. Constraint (4.15b) guarantees that the sum of the distributed data samples must be equal to the total number D of considered data samples for training across all learners in each of the global cycles. Constraints (4.15c) and (4.15d) are subcarrier allocation constraints, where the former (4.15c) guarantees that one subcarrier can be allocated to only one learner. Whereas the latter (4.15d) ensures that the total number of allocated subcarriers is equal to the number N of available subcarriers in the system. Constraint (4.15e) assures that the sum of the power used for transmission from the orchestrator to the different learners does not exceed its maximum total transmission power. Since the same subcarrier allocation and bit loading is used on the reverse communication (i.e., from the learners to the orchestra tors), this guarantees that their maximum transmit power is not exceeded either. Constraint (4.15f) represents the bounds on each learner's computation frequency. Constraints (4.15g), (4.15h), and (4.15i) are non-negativity and integer constraints for τ, L, and d_k $\forall$ k respectively, whereas Constraint (4.15j) imposes binary constraints on all subcarrier allocation indicators. Finally, Constraint (4.15k) ensures that the number of bits loaded on any subcarrier must be from among the set $\mathcal{Q}$ of bit/carrier values corresponding to the allowed square constellations in the system.

Unfortunately, it is not possible to derive an exact expression for the objective function of problem (4.15) for most machine learning models. Therefore, we will reformulate this objective function in the following subsection to express it as a function of the "convergence bound" [41].

4.2.2 Formulation Using Convergence Bound

It has been shown in [41] that, with an imposed global number L of iterations (which is in our case in direct relation to the global time constraint), a more suitable objective function to optimize for many machine learning models is the deviation of their global loss function after these L iterations (denoted by $F(w[L])$) from the loss function corresponding to the optimal model (denoted by $F(w*)$). With some manipulation of the different learning parameters to fit the imposed global number of iterations, it can be shown that this deviation is upper bounded by the following convergence bound [41]:

$$F(w[L]) - F(w^*) \leq \frac{1}{L\left(\omega\eta(1-\frac{\beta\eta}{2}) - \frac{\rho h(\tau)}{\epsilon^2\tau}\right)} \tag{4.16}$$

where η denotes the learning rate of the model, ρ and β are meta-parameters related to the ρ-Lipschitz and β-smoothness assumptions on $F_k(w)$ illustrated in Section 2.2 respectively, ϵ is defined as the lower bound on $F(w[L]) - F(w^*)$, ω is a parameter that inversely depends on the deviation of the distributed learning parameter vector from an auxiliary parameter vector that follows a centralized gradient descent, and $h(\tau)$ is defined as:

$$h(\tau) = \frac{\delta}{\beta}[(\eta\beta+1)^\tau - 1] - \eta\delta\tau, \tag{4.17}$$

where

$$\delta = \frac{\sum_{k=1}^{K} d_k\delta_k}{D} \tag{4.18}$$

such that δ_k is the upper bound on the divergence of the loss function of learner k compared to the global learning model (i.e., $\parallel F_k(w) - F(w) \parallel \leq \delta_k$).

Given the above convergence bound, our optimization problem can be re-formulated as:

$$\min_{\substack{L,\tau,d_k,f_k,c_{k,n}\\ \alpha_{k,n}\ \forall\ k}} \quad \frac{1}{L\left(\omega\eta(1-\frac{\beta\eta}{2}) - \frac{\rho h(\tau)}{\epsilon^2\tau}\right)} \tag{4.19}$$

$$\text{s.t.} \quad \frac{L}{\tau}\left(t_k^S + t_k^C + t_k^R\right) \leq T, \quad \forall\ k \tag{4.19a}$$

$$\sum_{k=1}^{K} d_k = D \tag{4.19b}$$

$$\sum_{n=1}^{K} \alpha_{k,n} = 1, \quad \forall\ n \tag{4.19c}$$

$$\sum_{k=1}^{K}\sum_{n=1}^{N} \alpha_{k,n} = N \tag{4.19d}$$

$$\sum_{k=1}^{K}\sum_{n=1}^{N} \frac{p(c_{k,n})}{h_{k,n}}\alpha_{k,n} \leq P_{max} \tag{4.19e}$$

$$0 \leq f_k \leq f_k^{max}, \quad \forall\ k \tag{4.19f}$$

$$\tau \in \mathbb{Z}_+ \tag{4.19g}$$

$$L \in \mathbb{Z}_+ \tag{4.19h}$$

$$d_k \in \mathbb{Z}_+, \quad \forall\ k \tag{4.19i}$$

$$\alpha_{k,n} \in \{0, 1\}, \quad \forall\ k, n \tag{4.19j}$$

$$c_{k,n} \in \mathcal{Q}, \quad \forall\ k, n \tag{4.19k}$$

Constraints (4.19a)-(4.19k) are identical to constraints (4.15a)-(4.15k), respectively. As can be seen from the above formulation, the problem is an integer non-linear problem with linear and non-linear constraints. It is well known that solving such a problem is NP-hard.

In the next section, we use decomposition approach to solve the problem efficiently. One important insight that drives several actions in the proposed decomposition solution is that the objective function of (4.19) is a decreasing function of L. This decreasing function suggests that the larger the number of total iterations performed by the learners within the global time constraint, the smaller the convergence bound on the loss function. Consequently, all measures on any problem parameters that maximize L would be exploited in our path towards simplifying the problem and deriving our proposed solutions.

4.3 Proposed Solution

The proposed solution is based on decomposing the optimization problem into three steps and designing efficient algorithms to solve these steps. The philosophy behind this decomposition stems from the observation that the only non-linear constraint is related to the subcarrier allocation process. It also follows the trend used in recent related works (e.g., [43], [52]) in terms of separating the optimization of the physical parameters while reflecting their impact on optimizing the learning parameters.

Given the above, a decomposition of the problem into three steps is proposed to simplify and solve it:

1. Separately solve the subcarrier allocation and bit loading problem among the learners to maximize their total sum rate from the orchestrator. The idea behind this step is to derive the maximum possible rate that the orchestrator can communicate the model parameters and data samples in PL, given its power constraint. Using a maximum rate will reduce delivery time these data blocks, thus giving more time for executing a larger number L of iterations within the short time imposed by the global training time constraint, which was shown to minimize our problem's objective function.

2. Modify and solve the main optimization of the problem in (4.19) by changing the subcarrier allocation and bit loading variables of every learner by a rate allocation variable, whose sum across all learners is bounded by the maximum rate derived from Step 1. We show that this change of variables simplifies the problem significantly. We also show that, for FL, such simplification yields a strictly convex problem.

3. Re-adjust the sub-carrier allocation and bit loading parameters of the learners obtained in Step 1 so as to match or get the closest possible to the derived optimal rates from Step 2 while preserving the maximum power constraint.

The details of these three steps and the proposed algorithms implementing them to solve the main problem will be illustrated in the reminder of this section.

4.3.1 Initial Subcarrier Allocation and Bit Loading

The subcarrier allocation and bit loading problem for rate maximization given a power constraint in OFDMA systems is a well-investigated problem. In our solution, we will consider the approach presented in [58–60] while making some changes to simplify the solution and adapt it our problem. This rate maximization problem can be expressed as:

$$\max_{c_{k,n},\alpha_{k,n}} \quad \sum_{k=1}^{K}\sum_{n=1}^{N} c_{k,n}\alpha_{k,n} \tag{4.20}$$

$$\text{s.t.} \quad \sum_{n=1}^{K} \alpha_{k,n} = 1, \quad \forall\, n \tag{4.20a}$$

$$\sum_{k=1}^{K}\sum_{n=1}^{N} \alpha_{k,n} = N \tag{4.20b}$$

$$\sum_{k=1}^{K}\sum_{n=1}^{N} \frac{p(c_{k,n})}{h_{k,n}}\alpha_{k,n} \leq P_{max} \tag{4.20c}$$

$$\alpha_{k,n} \in \{0,1\}, \quad \forall\, k,n \tag{4.20d}$$

$$c_{k,n} \in \mathcal{Q}, \quad \forall\, k,n \tag{4.20e}$$

The problem remains a non-linear problem due to both the product of the objective variables in the objective function and the expression of $p(c_{k,n})$ defined in Equation (4.8). Yet, this problem can be converted into a linear optimization problem by exploiting the fact that $c_{k,n}$ can only take a value from a small set $\mathcal{Q}$ of integers as defined in constraint (4.20e).

To do so, let $c \in \mathcal{Q}$ be an arbitrary bit loading value. Consequently, the power $p_k(c)$ allocated to any subcarrier using the bit loading value c can be easily pre-calculated using 4.8. Due to the small size of $\mathcal{Q}$, all the values $p_k(c)\ \forall\ c \in \mathcal{Q}$ can be pre-calculated. Now, let's define a new indicator variable $\gamma_{k,n,c}$ as follows:

$$\gamma_{k,n,c} = \begin{cases} 1 & \alpha_{k,n} = 1 \quad \text{AND} \quad c_{k,n} = c \\ 0 & \text{, Otherwise} \end{cases} \tag{4.21}$$

In other words, $\gamma_{k,n,c}$ is a joint subcarrier allocation and bit loading indicator that is only set to one when subcarrier n is assigned to learner k to transmit c bits on it.

Using this new indicator, the problem in (4.20) can be re-written as an integer linear problem as follows:

$$\max_{\gamma_{k,n,c}} \quad \sum_{k=1}^{K}\sum_{n=1}^{N}\sum_{c\in\mathcal{Q}} c\ \gamma_{k,n,c} \tag{4.22}$$

$$\text{s.t.} \quad \sum_{n=1}^{K}\sum_{c\in\mathcal{Q}} \gamma_{k,n,c} = 1, \quad \forall\ n \tag{4.22a}$$

$$\sum_{k=1}^{K}\sum_{n=1}^{N}\sum_{c\in\mathcal{Q}} \gamma_{k,n,c} = N \tag{4.22b}$$

$$\sum_{k=1}^{K}\sum_{n=1}^{N}\sum_{c\in\mathcal{Q}} \frac{p_k(c)}{h_{k,n}}\ \gamma_{k,n,c} \leq P_{max} \tag{4.22c}$$

$$\gamma_{k,n,c} \in \{0,1\}, \quad \forall\ k,n,c \tag{4.22d}$$

It can thus be solved using many efficient solvers or can simply be relaxed to its non-integer version and solved using linear programming and greedy rounding. Once the solution is found, we can easily derive the initial rate r_k^i of each learner and the maximum rate R_{max} from the orchestrator as:

$$r_k^i = \sum_{n=1}^{N}\sum_{c\in\mathcal{Q}} c\ \gamma^*_{k,n,c} \tag{4.23}$$

$$R_{max} = \sum_{k=1}^{K}\sum_{n=1}^{N}\sum_{c\in\mathcal{Q}} c\ \gamma^*_{k,n,c} \tag{4.24}$$

where $\gamma^*_{k,n,c}$ is the optimal solution for the problem in (4.22).

4.3.2 Simplifications of the Main Problem

In this section, we simplify our main problem interest in (4.19) by both introducing some modifications based on insights from prior works (e.g., [43], [52]), and leveraging the maximum rate expression derived in (4.24).

Below are three modifications we will apply to the problem in (4.19):

1. It can be easily shown that minimizing our main problem's objective function will

be achieved when the computation frequency f_k of every learner k is set to its maximum value f_k^{max} (i.e., $f_k^* = f_k^{max}\ \forall\ k$. Indeed, setting f_k^* to $f_k^{max}\ \forall\ k$ will enable all learners to train their models using a larger total number L of iterations within the constrained global training time, which directly translates into a lower convergence bound on the loss function. This above determination of f_k^* removes these variables from the optimization problem in (4.19) as well as the constraints in (4.19f)

2. We perform a change of the $c_{k,n}$ and $\alpha_{k,n}$ variables, expressing the allocated communication resources to each learner k, by a rate variable r_k. This r_k variable directly relates to $c_{k,n}$ and $\alpha_{k,n}$ as expressed in (4.10), and directly impacts both t_k^S and t_k^R as defined (4.11) and (4.13), respectively. Unlike $\alpha_{k,n}$ and $c_{k,n}$, the new r_k variables are real, which would reduce the complexity of the problem significantly. To guarantee the restoration of the main variables $\alpha_{k,n}$ and $c_{k,n}$ in Step 3 from the r_k values obtained in Step 2, given the constraints on the network's physical resources, we impose the constraint $\sum_{k=1}^{K} r_k = R_{max}$ to the simplified problem in Step 2, where R_{max} is the value obtained in (4.22). The proposed change of variable will thus simplify the expressions in Constraint (4.19a), will replace constraints (4.19c), (4.19d), and (4.19e) by the above constraint on the sum rates by R_max, and will finally replace the integer constraints in (4.19j) and (4.19k) by simple linear non-negativity constraints on $r_k\ \forall\ k$.

3. To further simplify the problem, we relax the integer constraints on the variables τ, L, and $d_k\ \forall_k$ in (4.19g), (4.19h), and (4.19i), respectively, by assuming that they can take real positive values. This is a common practice to simply integer optimization problems. Once the simplified problem is solved, the integer values of τ L, and d_k can be restored by simple rounding approaches that satisfy the main problem's constraints.

By applying the above changes on the optimization problem in (4.19), the simplified problem we consider in this step can be expressed as follows:

$$\min_{L,\tau,d_k,r_k\ \forall\ k} \quad \frac{1}{L\left(\omega\eta(1-\frac{\beta\eta}{2})-\frac{\rho h(\tau)}{\epsilon^2\tau}\right)} \tag{4.25}$$

$$\text{s.t.} \quad \frac{L}{\tau}\left(t_k^S+t_k^C+t_k^R\right)\leq T, \quad \forall\ k \tag{4.25a}$$

$$\sum_{k=1}^{K} d_k = D \tag{4.25b}$$

$$\sum_{k=1}^{K} r_k \leq R_{max} \tag{4.25c}$$

$$\tau > 0 \tag{4.25d}$$

$$L > 0 \tag{4.25e}$$

$$d_k > 0, \quad \forall\ k \tag{4.25f}$$

$$r_k > 0, \quad \forall\ k \tag{4.25e}$$

Although (4.25) is indeed simpler than (4.19), finding a closed-form expression for its solution is still not possible as the problem is still combinatorial in nature [51].

4.3.3 Special Case of Simplified Problem in FL

As discussed earlier, FL is a special case of the above general and simplified formulation considered in (4.19) and (4.25), in which the value of d_k in the first term of t_k^S $\forall\ k$ (expanded in (4.11) are set to zero. From this fact, it can be seen from (4.11) and (4.13) that $t_k^S = t_k^R$ $\forall\ k$ in FL. This plays an important role in simplifying Constraints (4.25a) (as well as (4.19a) but we focus now on the simplified problem), as it can be re-written as:

$$\frac{L}{\tau}\left(t_k^S+t_k^C+t_k^R\right)=\frac{L}{\tau}\left(\frac{C_m d_k\tau}{f_k^{max}}+\frac{2P_mS_m}{r_k}\right)\leq T, \quad \forall\ k \tag{4.26}$$

Re-arranging (4.26) to separate L, and applying the fact that our main problem's objective function will be minimized if L is set to its maximum possible value, we must have:

$$L = \frac{T\tau}{\frac{C_m d_k \tau}{f_k^{max}} + \frac{2P_m S_m}{r_k}}, \quad \forall\ k \tag{4.27}$$

The above expression of L is a set of K equations, each of which being a function of the real values of τ, d_k, r_k $\forall\ k \in \{1, \ldots, K\}$, such that the value of L is equal to the right-hand side for all K equations.

Afterwards, we can involve the equality constraint in (4.25b) to further simplify this set of equations. By re-arranging (4.27) to separate d_k, we get:

$$d_k = \frac{f_k^{max}}{C_m}\left(\frac{T}{L} - \frac{2P_m S_m}{\tau r_k}\right) \quad \forall\ k \tag{4.28}$$

By summing both sides over all values of $k \in \{1, \ldots, K\}$, the left-hand side of this summation will yield the constant D as per Constraint (4.25b). Re-arranging the resulting equation after this summation to separate L on the left-hand side, we get

$$L(\tau, r_k) = \frac{T \sum_{k=1}^{K} a_k}{d + \frac{1}{\tau}\sum_{k=1}^{K} b_k} \tag{4.29}$$

where

$$a_k = \frac{f_k^{max}}{C_m} \tag{4.30}$$

$$b_k = \frac{2P_m S_m f_k^{max}}{\tau r_k C_m} \tag{4.31}$$

Finally, by defining $P(\tau)$ as:

$$P(\tau) = \frac{1}{\omega\eta(1 - \frac{\beta\eta}{2}) - \frac{\rho}{\epsilon^2}\frac{h(\tau)}{\tau}} \tag{4.32}$$

which is a function of τ only, we can re-write the objective function of (4.25) for the FL case as:

$$O(\tau, r_k) = \frac{P(\tau)}{L(\tau, r_k)} \tag{4.33}$$

Theorem 1 *$O(\tau, r_k)$ is strictly convex in the domain $\tau > 0$ and $r_k > 0$*

Given that the objective function in the FL case is strictly convex, the optimal τ and r_k $\forall$ k can be obtained by solving the following problem:

$$\min_{\tau, r_k \ \forall \ k} \quad O(\tau, r_k) \tag{4.34}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} r_k \leq R_{max} \tag{4.34a}$$

$$\tau > 0 \tag{4.34b}$$

The purpose of this part is to prove the convexity of the FL case in Objective 2. To this end, our objective function $O(\tau, r_k) = \frac{P(\tau)}{L(\tau, r_k)}$ can be divided into two terms $O(\tau, r_k) = P(\tau) \times \frac{1}{L(\tau, r_k)} =$, where the reciprocal of $L(\tau, r_k)$ can be defined as $M(\tau, r_k)$ and the reciprocal of $P(\tau)$ can be defined as $Q(\tau)$ as follows:

$$M(\tau, r_k) = \frac{d}{T\sum_{k=1}^{K} a_k} + \frac{\sum_{k=1}^{K} b_k}{T\tau \sum_{k=1}^{K} a_k} \tag{4.35}$$

$$Q(\tau) = A - B\frac{C^{\tau} - 1 - (C-1)\tau}{\tau} \tag{4.36}$$

Let us recall that $a_k = \frac{f_k^{max}}{C_m}$ and $b_k = \frac{2P_m S_m f_k^{max}}{\tau r_k C_m}$, while also defining $A = \omega\eta(1 - \frac{\beta\eta}{2})$, $B = \frac{\delta}{\beta}\frac{\rho}{\epsilon^2}$, and $C = \eta\beta + 1$. Note that B can be re-written as $B = \frac{\delta}{\beta}B_0$, where $B_0 = \frac{\rho}{\epsilon^2} > 0$ is a control parameter that can be set empirically. In the reminder of the proof, we will designate $O(\tau, r_k)$, $M(\tau, r_k)$, $N(\tau) = \frac{1}{Q(\tau)}$, and $Q(\tau)$, as O, M, N and Q for simplicity of the notation.

To prove that the objective function is strictly convex, we must derive the Hessian matrix, defined as:

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 O}{\partial \tau^2} & \frac{\partial^2 O}{\partial \tau \partial r_1} & \cdots & \frac{\partial^2 O}{\partial \tau \partial r_k} & \cdots & \frac{\partial^2 O}{\partial \tau \partial r_K} \\ \frac{\partial^2 O}{\partial r_1 \partial \tau} & \frac{\partial^2 O}{\partial r_1^2} & \cdots & \frac{\partial^2 O}{\partial r_1 \partial r_k} & \cdots & \frac{\partial^2 O}{\partial r_1 \partial r_K} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial^2 O}{\partial r_k \partial \tau} & \frac{\partial^2 O}{\partial r_k \partial r_1} & \cdots & \frac{\partial^2 O}{\partial r_k^2} & \cdots & \frac{\partial^2 O}{\partial r_k \partial r_K} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial^2 O}{\partial r_K \partial \tau} & \frac{\partial^2 O}{\partial r_K \partial r_1} & \cdots & \frac{\partial^2 O}{\partial r_K \partial r_k} & \cdots & \frac{\partial^2 O}{\partial r_K^2} \end{bmatrix}, \tag{4.37}$$

and show that it is positive definite. This means that we first have to prove that all the elements in the matrix are positive elements. There are three different combinations for taking the second derivative of $O = MN$, which can be described as follows:

$$\frac{\partial^2 O}{\partial \tau^2} = \frac{\partial^2 M}{\partial \tau^2} N + 2\frac{\partial M}{\partial \tau}\frac{\partial N}{\partial \tau} + M\frac{\partial^2 N}{\partial \tau^2} \tag{4.38}$$

$$\frac{\partial^2 O}{\partial r_k \partial \tau} = \frac{\partial^2 O}{\partial \tau \partial r_k} = \frac{\partial^2 M}{\partial r_k \partial \tau} N + \frac{\partial M}{\partial r_k}\frac{\partial N}{\partial \tau} \tag{4.39}$$

$$\frac{\partial^2 O}{\partial r_k^2} = \frac{\partial^2 M}{\partial r_k^2} N \tag{4.40}$$

(54-56) is defined $\forall\ k \in \mathcal{K}$. For simplicity let us define $\frac{\partial Q}{\partial \tau}$ and $\frac{\partial^2 Q}{\partial \tau^2}$ as Q' and Q'' respectively, we thus need to prove that $\frac{\partial^2 M}{\partial \tau^2}$, $\frac{\partial^2 M}{\partial r_k \partial \tau}$, $\frac{\partial^2 M}{\partial r_k^2}$ and $\frac{\partial^2 N}{\partial \tau^2}$ are positive, while also proving that $\frac{\partial M}{\partial \tau}$, $\frac{\partial M}{\partial r_k}$ and $\frac{\partial N}{\partial \tau}$ are all either positive or negative. This will guarantee that the Hessian matrix contains only positive elements. These partial derivatives can be calculated as follows:

$$\frac{\partial M}{\partial \tau} = -\frac{\sum_{k=1}^{K} \frac{b_k}{a_k}}{\tau^2} \tag{4.41}$$

$$\frac{\partial M}{\partial r_k} = -\frac{\sum_{k=1}^{K} d_k}{\tau \sum_{k=1}^{K} r_k^2 a_k} \tag{4.42}$$

$$\frac{\partial N}{\partial \tau} = -\frac{Q'}{Q^2} \tag{4.43}$$

$$\frac{\partial^2 M}{\partial \tau^2} = \frac{2\sum_{k=1}^{K}\frac{b_k}{a_k}}{\tau^3} \tag{4.44}$$

$$\frac{\partial^2 M}{\partial r_k^2} = \frac{2\sum_{k=1}^{K} d_k}{\tau\sum_{k=1}^{K} r_k^3 a_k} \tag{4.45}$$

$$\frac{\partial^2 M}{\partial r_k \partial \tau} = \frac{\sum_{k=1}^{K} d_k}{\tau^2\sum_{k=1}^{K} r_k^2 a_k} \tag{4.46}$$

$$\frac{\partial^2 N}{\partial \tau^2} = \frac{1}{Q^2}\left[\frac{2(Q')^2}{Q} - Q''\right] \tag{4.47}$$

(57, 58, 60-62) are defined $\forall\ k \in \mathcal{K}$. Since all the variables and constants in the right-hand side expressions are all positive, it can be easily shown that $\frac{\partial^2 M}{\partial \tau^2}$, $\frac{\partial^2 M}{\partial r_k \partial \tau}$, $\frac{\partial^2 M}{\partial r_k^2}$ are all positive, whereas $\frac{\partial M}{\partial \tau}$ and $\frac{\partial M}{\partial r_k}$are negative. Thus, we need to show that $\frac{\partial N}{\partial \tau}$ is negative and $\frac{\partial^2 N}{\partial \tau^2}$ is positive. It can be shown with the denominator of (59) being Q^2, it's sufficient to prove that $Q' > 0$ to show that $\frac{\partial N}{\partial \tau} > 0$.

$$Q' = \frac{B}{\tau^2}\Big[C^{\tau}\big[1 - (lnC)\tau\big] - 1\Big] \tag{4.48}$$

It is clear that $\frac{B}{\tau^2} > 0$ and therefore, to ensure $Q' > 0$, we need to satisfy $C^{\tau}[1-(\ln C)\tau] > 1$. From the Bernoulli inequality, we know that $C^{\tau} \geq (C-1)\tau + 1$. Assuming the worst case where the equality holds, the expression can be written as $[(C-1)\tau + 1][1 - (\ln C)\tau] > 1$. By expanding the expression we get:

$$\tau[(\ln C)\tau - C(\ln C) - 1 + C] > 0 \tag{4.49}$$

We know that a feasible τ^* is always greater than 0. We thus examine the term enclosed in the square brackets. By re-arranging (65) to express τ as an inequality in C, we get:

$$\tau > \frac{C(\ln C) + 1 - C}{\ln C} \tag{4.50}$$

Recall that $C = \eta\beta + 1$ where η is chosen such that $\eta\beta \leq 1$, and $\eta, \beta > 0$. Hence, it follows that $1 < \eta\beta + 1 \leq 2$. If we plot $f(C) = \frac{C(\ln C)+1-C}{\ln C}$ against the domain of C, it can be easily shown that when $\tau > 0$, $\frac{\partial N}{\partial \tau}$ is strictly negative. As we can see in Fig. 4.1, $0 < f(C) < 1$, and because a feasible $\tau \in \{1, 2, 3, \ldots\}$, this inequality will always hold when a feasible τ exists.

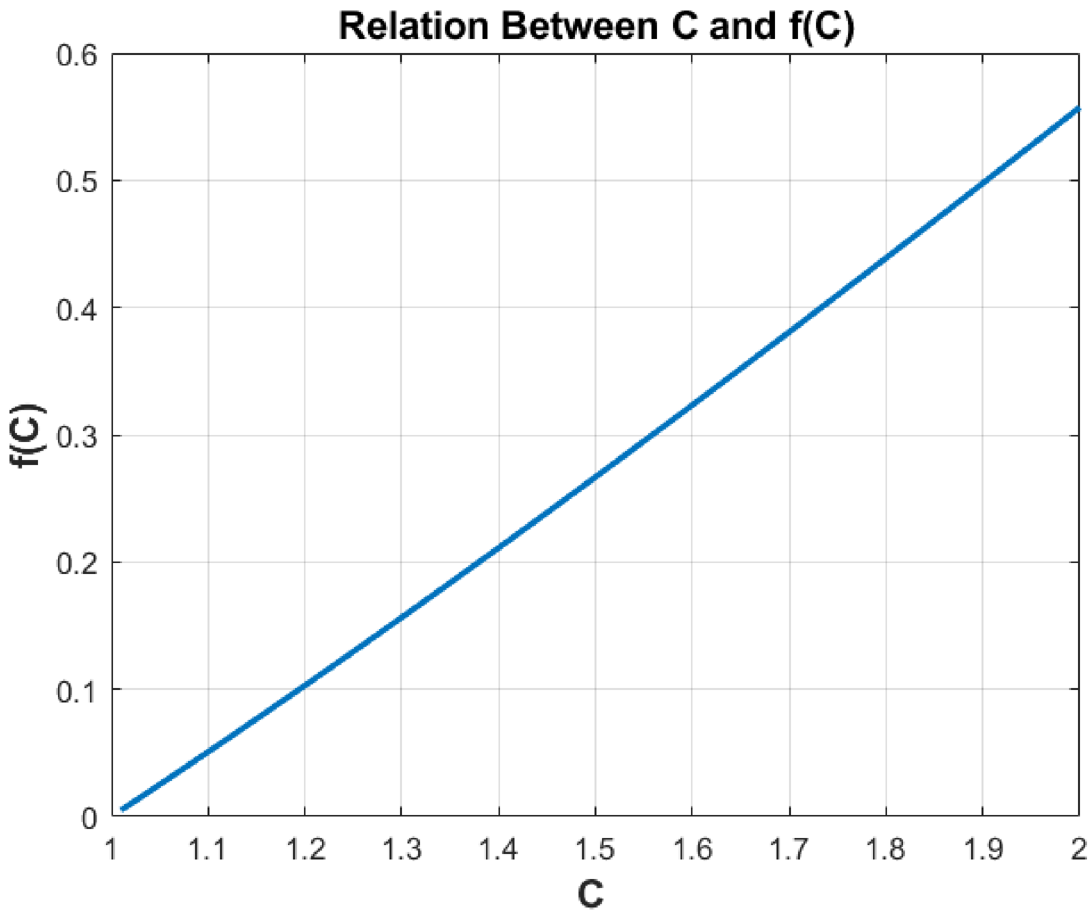

Figure 4.1: Relation Between C and f(C).

let us define $S = C^{\tau}[1-(lnC)\tau] - 1$, $\frac{\partial S}{\partial \tau} = S'$ and $Q' = B\frac{S}{\tau^2}$. This means that Q'' can be defined as follows:

$$Q'' = \frac{B}{\tau^4}\left[\tau^2 S' - 2\tau S\right] \tag{4.51}$$

It is clear that $\frac{B}{\tau^4} > 0$ and $2\tau S$ is a negative term, which means that it will be sufficient to prove that $S' < 0$ to prove that $Q'' < 0$. Thus we calculate S' as:

$$S' = -(lnC)^2 \tau C^\tau \tag{4.52}$$

As shown before, all the terms in (68) are positive, where $\tau \in \{1, 2, 3, \ldots\}$ and C being always positive guarantees that the terms are positive and greater than zero. thus, (68) will always be negative. This satisfies $Q'' < 0$ and concludes our proof that all the elements of **H** are positive elements.

Defining x as:

$$x = \begin{bmatrix} \tau \\ r_1 \\ r_2 \\ \vdots \\ r_k \\ \vdots \\ r_K \end{bmatrix}, \tag{4.53}$$

the Hessian matrix **H** is considered strictly convex if $x^T H x > 0$ for all elements in x. Since our work is in the domain of $\tau > 0$ and $0 < r_k < r_{max}$, this condition is always true for the calculated Hessian matrix **H**. After obtaining the optimal values of τ and r_k's, L can be calculated using (4.29) and then all the d_k's can be calculated using (4.28).

4.3.4 Rate Re-adjustment Algorithm

Solving the problems in (4.25) and (4.34) will provide a set of rates $r_k^* \ \forall\ k$ between the orchestrator and learners, which could be different than the set of initial rates $r_k^i \ \forall\ k$ computed in (4.23). The final step of our proposed three-step solution is to minimize the rate gaps defined by:

$$e_k = r_k^* - r_k^i \quad \forall\ k \tag{4.54}$$

while respecting the maximum power constraint of the orchestrator. In the considered OFDMA-based system, this can be achieved by subcarrier reallocation and bit reloading between the different learners while making sure the maximum power constraint is not exceeded. This can done by formulating a new problem that minimizes the above gaps given the maximum power constraint. Yet, for the sake of simplicity, we will employ a simple greedy subcarrier reallocation and bit reloading algorithm, such as the ones used in several OFDMA resource allocation works (e.g,. [61]).

This greedy algorithm simply sorts the rate gaps in ascending order and moving the least loaded subcarrier from the learner at the bottom to list to the one the top of the list, and load bits for the latter learner on this new reallocated subcarrier as long as the total power constraint is still maintained. Next, the new rates and rate gaps for these two learners are recomputed, the rate gaps are re-sorted, and the above step is repeated. This continues until no more subcarriers can move from one learner to the other without violating the maximum power constraint. To ensure convergence, the subcarrier reallocated from one learner to another in a step cannot be reallocated in subsequent steps, thus setting an $O(N)$ bound on the total number of subcarrier reallocations.

4.3.5 Proposed Algorithms

In this section, we propose two algorithms, namely the static and dynamic algorithms, to implement our full three-step solution to solve the problem in (4.19).

Algorithm 1 Static Algorithm

```
Require: $T$, $D$, $K$, $N$, $\mathcal{Q}$, $P_{max}$, $\rho$, $\epsilon$, $\omega$
Ensure: $w[L]$
    Initialize $l = 0$, $\tau \leftarrow 1$, $d_k \leftarrow \frac{D}{K}$ Set $w[0]$ as a random vector
 1: Solve (4.20)
 2: Calculate $r_k^i$ $\forall$ $k$ and $R_{max}$ from (4.23) and (4.24)
 3: Send $w[l]$ and (for PL) $d_k$ samples to each learner $k$ at a rate $r_k^i$
 4: After one local iteration, collect $w_k[0]$ and estimate $w$, $\beta$ and $\delta$ as in [41]
 5: $T \leftarrow T - \max_k\{t_k\}$
 6: Solve (4.25) or (4.34) for PL or FL, respectively, to find the optimized values of $L$, $\tau$ $d_k$,
    and $r_k$ $\forall$ $k$.
 7: Re-adjust rates as described in Section 4.3.4
 8: while $T > 0$ do
 9:    Send $w[l]$ and (for PL) $d_k$ samples to each learner $k$ with adjusted rates $r_k$ $\forall$ $k$
10:    Set $T \leftarrow T - \max_k\{t_k\}$
11:    if $T < 0$ then
12:       Reduce $\tau$ to maximum value $\geq 0$ such that $T \geq 0$
13:       Set $T \leftarrow 0$
14:    end if
15:    $l \leftarrow l + \tau$
16:    Each learner $k$ trains its model for $\tau$ local iteration using $d_k$ data samples
17:    Orchestrator collects $w_k[l]$ from all learners and estimate $w[l]$, $\beta$ and $\delta$ as in [?]
18: end while
19: return $w[L]$
```

Static Algorithm: The static algorithm, detailed in Algorithm 1, solves the optimization problem only once at the beginning of the learning process. As shown, the algorithm starts by Step 1 of the algorithm in Lines 1-2. It then runs a dummy local iteration to determine initial learning parameters (Lines 3-4), and removes the consumed time in this step from the total time constraint (Line 5). It then executes Steps 2 and 3 in Lines 6 and 7, respectively. For PL, a numerical solver, namely OPTI [53], is used in Step 2 to calculated the optimized values of the problem variables. For FL, since the problem in (4.34) is shown to be convex, any convex solver can be used to calculate the optimized values of the problem variables. Finally, the algorithm executes the actual PL or FL algorithm in Lines 8-17, until the final model parameters $w[L]$ is obtained at the orchestrator. Note that Lines 11-14 in Algorithm 1 handle the case of having less time in the very last global cycle to execute τ

local iterations. This situation may arise due the disturbances in the relation of L and T as a result of the simplifications and relaxations done in Step 2 of our proposed three-step solution, as well as the remaining deviations of the employed rates from the optimized rates after rate adjustments in Step 3. The algorithm thus utilizes the remaining time to execute this last global cycle with less number of local iterations than τ to consume the remaining time.

Algorithm 2 Dynamic Algorithm

```
Require: $T$, $D$, $K$, $N$, $\mathcal{Q}$, $P_{max}$, $\rho$, $\epsilon$, $\omega$
Ensure: $w[L]$
    Initialize $l = 0$, $\tau \leftarrow 1$, $d_k \leftarrow \frac{D}{K}$ Set $w[0]$ as a random vector
 1: Execute Steps 1-5 of Algorithm 1
 2: while $T \neq 0$ do
 3:     Solve (4.25) or (4.34) for PL or FL, respectively, to find the optimized values of $L$, $\tau$
        $d_k$, and $r_k$ $\forall$ $k$.
 4:     Re-adjust rates as described in Section 4.3.4
 5:     Execute Steps 9-17 of Algorithm 1
 6: end while
 7: return $w[L]$
```

Dynamic Algorithm: Unlike the static algorithm, the dynamic algorithm re-optimizes all the problem parameters after every global cycle. The motivation behind this algorithm is that the learning parameters, such as δ, β, are not static, and their most updated values can be extracted from the global learning model at the orchestrator. In addition, the channel conditions between the orchestrator and the learners can typically change during each global cycle, which is not accounted for when our three-step optimization is computed once in the very beginning. Consequently, these updated learning and physical values may change the optimal number of remaining local and total iterations. Such updated values can be obtained by running our three-step optimization with the new values of the aforementioned parameters to produce a more optimized solution.

Algorithm 2 explains the steps of the dynamic algorithm in details. It can be easily shown that the only difference of this algorithm compared to Algorithm 1 is that the optimization process is repeated after each τ local iterations (i.e after each global cycle) using updated

Table 4.1: Simulation parameters

Parameter	Value
Number of Subcarriers N	64
Target BER	10^{-4}
Symbol Time T_s	122μs
Device Proximity(DP)	500 m
Node Power	23 dBm
Noise Power Density N_o	-174 dBm/Hz
Attenuation Model	128+37.1log(DP)dB
Computation Capability f_k	2.4 GHz and 1.2 GHz
MNIST dataset size D	54,000 images
MNIST dataset Features F	784 features

channel conditions and learning parameters extracted from the global learning model.

4.4 Simulation Results

In this section, we illustrate the simulation results for the joint task and resource allocation problem of interest in heterogeneous mobile edge environments. We also compare its performance to that of our two baseline schemes TP and EDA.

4.4.1 Physical Simulation Environment

From the physical perspective, the edge learners were divided into two groups, simulating both the computational capabilities of a portable computing devices and a commercial microcontroller. In addition, random distances (with a maximum distance of 500 m) and fading conditions were generated for each of the learners with respect to the orchestrator. In all simulations, we chose to emulate 802.11 links and channels between the learners and the orchestrator. Yet, our proposed solution can be implemented over any multicarrier communication (e.g., LTE or 5G using either side links or the base station playing the role of the orchestrator).

4.4.2 Dataset and Learning Settings

The MNIST [54] dataset was chosen to test our proposed scenarios. This dataset consists of $54,000$ images for training and $6,000$ images for validation, where each of these images includes 784 features. The employed neural network consists of one input, three hidden, and one output layers with $[784, 300, 124, 60, 10]$ learning elements in each layer. For this network, the model size is calculated to be $8,974,080$ bits, and the required floating-points operations are $1,123,736$. Table 2 summarizes the simulation parameters for both the physical resources/setup and the employed dataset.

Different numbers of learners are used in our simulations to train the above neural network using the MNIST dataset for different total training times $T = [60, 80, 100, 120, 140]$. The learning rate η is set 0.9 as it was shown to provide the optimum performance for the aforementioned neural network.

4.4.3 Simulation Results for PL

In this section, we present our simulation results for the PL setting and show the performance of the proposed algorithms in comparison to the aforementioned TP and EDA schemes.

Fig. 4.2 and 4.3 illustrate the total number of iterations L achieved by both the static and dynamic algorithms when implementing the joint, TP, and EDA schemes. Fig. 4.2 plots the results against the number of learners for a total training time of 60 seconds, whereas Fig. 4.3 plots the results against the total training time for 20 learners.

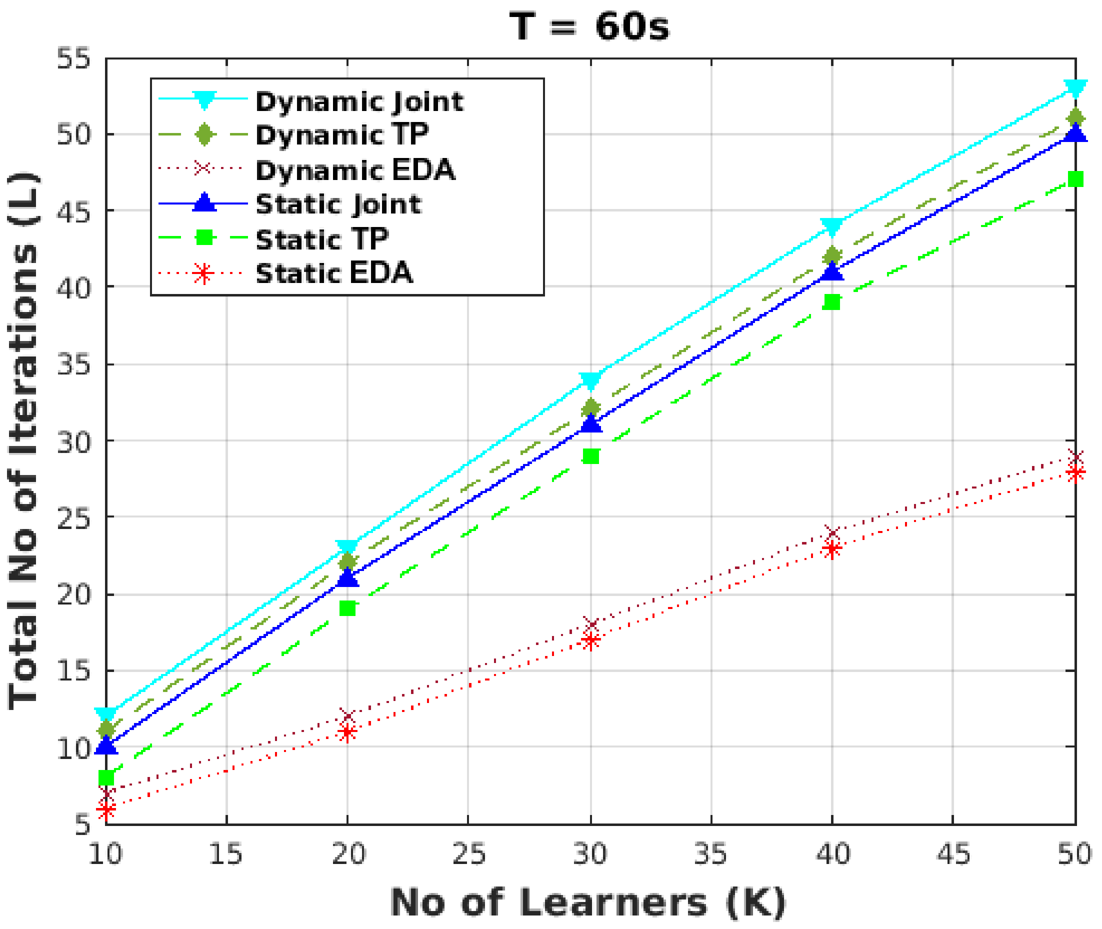

Figure 4.2: Total number of iterations achieved by different numbers of learners for $T = 60$ s.

The first observation from both figures is that our joint task planning and resource allocation approach achieves a higher number of iterations than both the TP and EDA schemes, for both the static and dynamic algorithms. We can also see that the TP scheme achieves a much better performance than the EDA scheme. For a total training time constraint of 60 seconds and 20 learners, the dynamic EDA, TP, and joint schemes achieve 12, 22, and 23 total iterations, which results in 91.67% and 4.54% more training iterations for the joint scheme over the EDA and TP schemes, respectively. The performed number of iterations between the joint dynamic and static algorithms at the same setting differs by only two iterations, a gain of 7.69%. As the total training time increases to 140 seconds, the static and dynamic joint algorithms can complete 55 and 61 learning iterations, respectively, thus achieving a gain of 10.9% for the latter over the former. A final interesting observation from these two figures is that the dynamic TP scheme outperforms the static joint scheme, which exhibits the merits of the dynamic algorithm even for a less optimized scheme.

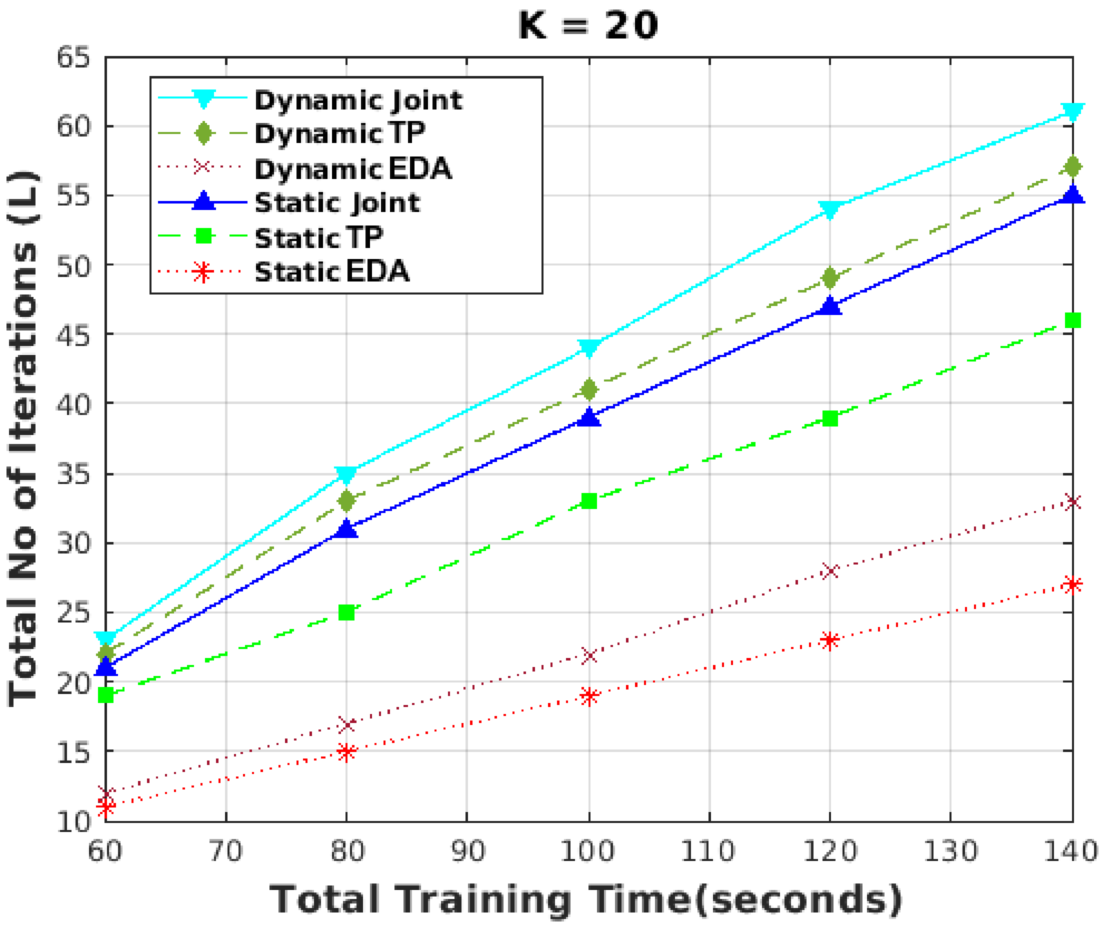

Figure 4.3: Total number of iterations achieved at different global time constraints for $K = 20$.

Zooming out from the iterations to the global cycles level, Fig. 4.4 depicts the timings and number of global cycles performed by the static and dynamic joint, TP, and EDA algorithms as training time progress to a maximum of 60 seconds and for 20 learners. We can first clearly observe that the dynamic algorithms always finish the learning global cycles before their same-scheme static algorithms. This helps the dynamic algorithms to run for more total learning iterations and, when the training time increases, the dynamic algorithms are able to complete more global cycles than their static counterparts. In addition, we can observe from the figure that the joint, TP, and EDA schemes complete five, four, and two global cycles, a 25% and 50% increase in the number of cycles for the joint scheme over the TP and EDA schemes, respectively. This promotes for an improved performance for the joint scheme compared to the two other schemes.

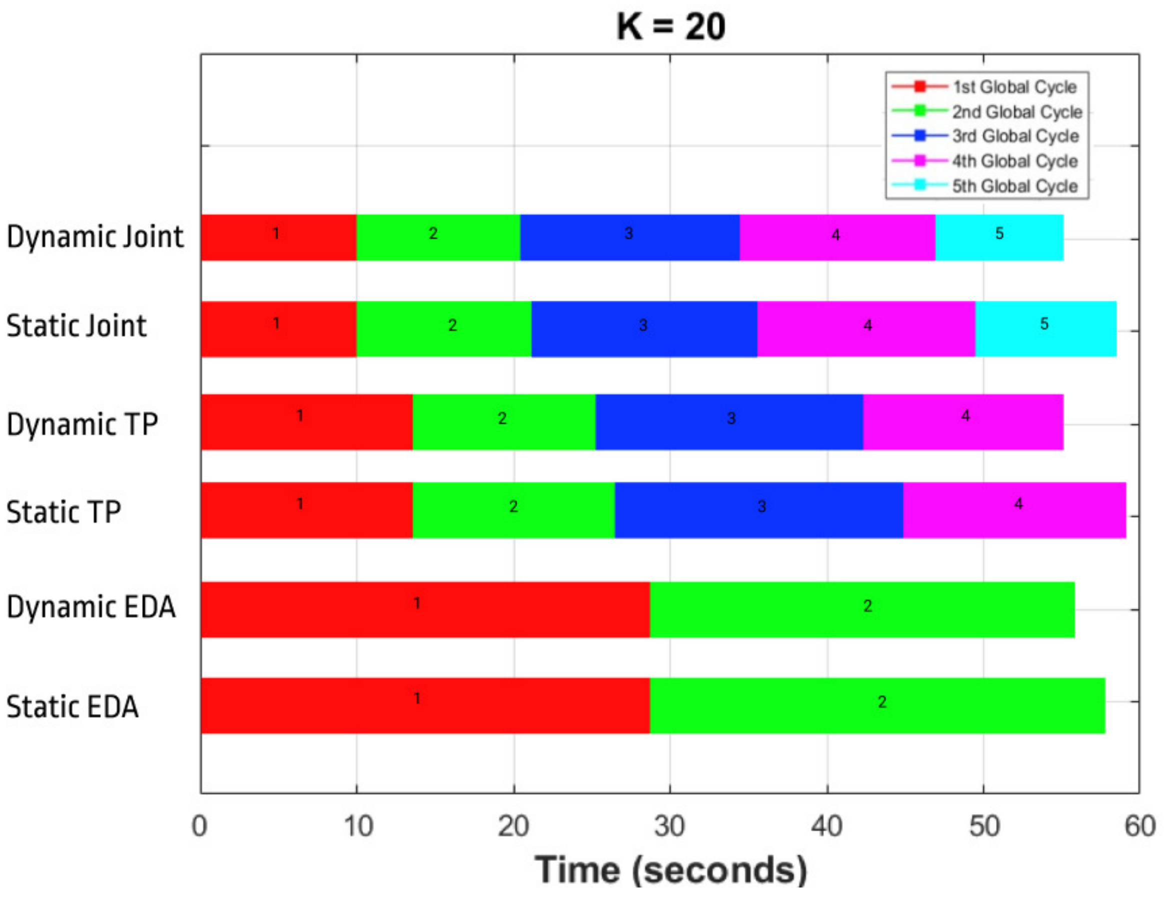

Figure 4.4: Temporal progression of global cycles achieved by the different algorithms for $T = 60$ s and $K = 20$.

Now moving on to the performance of the trained models, Fig. 4.5 illustrates the terminal loss functions of the trained models by 20 learners using the static joint, TP, and EDA algorithms at the end of total training times ranging between 60 and 140 seconds. For the same settings, the figure also depicts the validation accuracy of the trained models using these algorithms. We can see from both sub-figures that the joint scheme outperforms both the TP and the EDA schemes for any given total training time. In addition, we can observe that the joint and TP schemes require a total training time of 100 and 120 seconds, respectively, to achieve an above-96% accuracy.

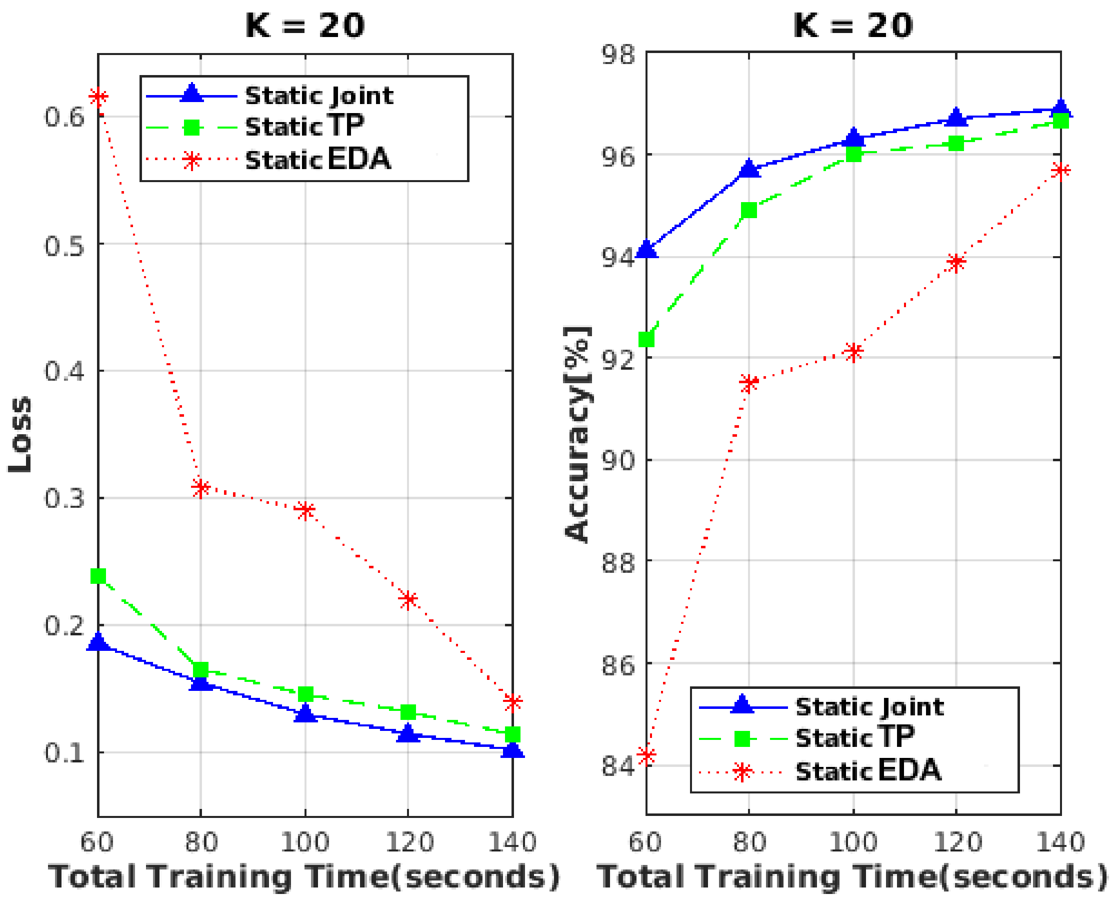

Figure 4.5: Terminal loss function and accuracy performance of the static joint, TP, and EDA algorithms for $K = 20$.

This means that our proposed joint scheme can save up to 16.67% in training time compared to the TP scheme to reach this target accuracy. We can also see that the joint scheme at $T = 60$ seconds achieves a higher accuracy of 94.07% compared to the 93.9% accuracy achieved by the EDA scheme at $T = 120$ seconds. Yet, we can also observe that, as the total training time increases, the gap between the performance of the three schemes decreases.

Fig. 4.6 depicts the same comparison of Fig. 4.5 but for the dynamic algorithms. It can again seen that the joint scheme outperforms the other two schemes at any given total training time. Moreover, an accuracy above 96% is achieved at total training times of 80, 100, and 140 seconds for the joint, TP, and EDA scheme, respectively. This is equivalent to up to 20% and 42.9% savings in the total training time for the joint scheme compared to the TP and EDA schemes, respectively.

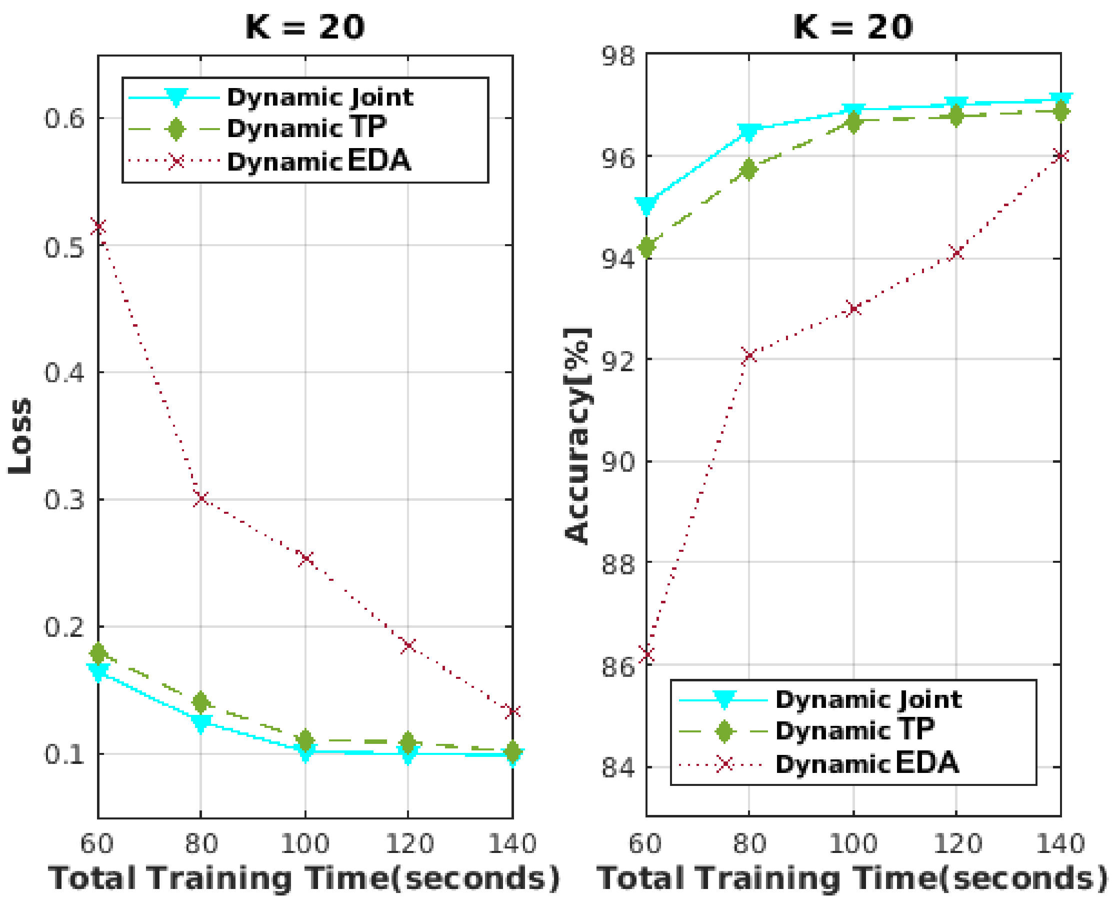

Figure 4.6: Terminal loss function and accuracy performance of the dynamic joint, TP, and EDA algorithms for $K = 20$.

4.4.4 Simulation Results for FL

In this section, we illustrate the performance of the dynamic and static algorithms, while focusing only on our new joint scheme for FL.

In Fig. 4.7, the total number of iterations achieved by both the FL dynamic and static joint algorithms are depicted for total training times ranging between 60 and 140 seconds and for 20 learners . As expected, the FL dynamic algorithm is able to achieve more total iterations than the FL static one, as the dynamic algorithm optimizes the performance of FL after each global cycle. For a total training time of 120 seconds, the FL dynamic and static joint algorithms achieve 63 and 51 total iterations, respectively, which results in 23.53% more training iterations for the former over the latter.

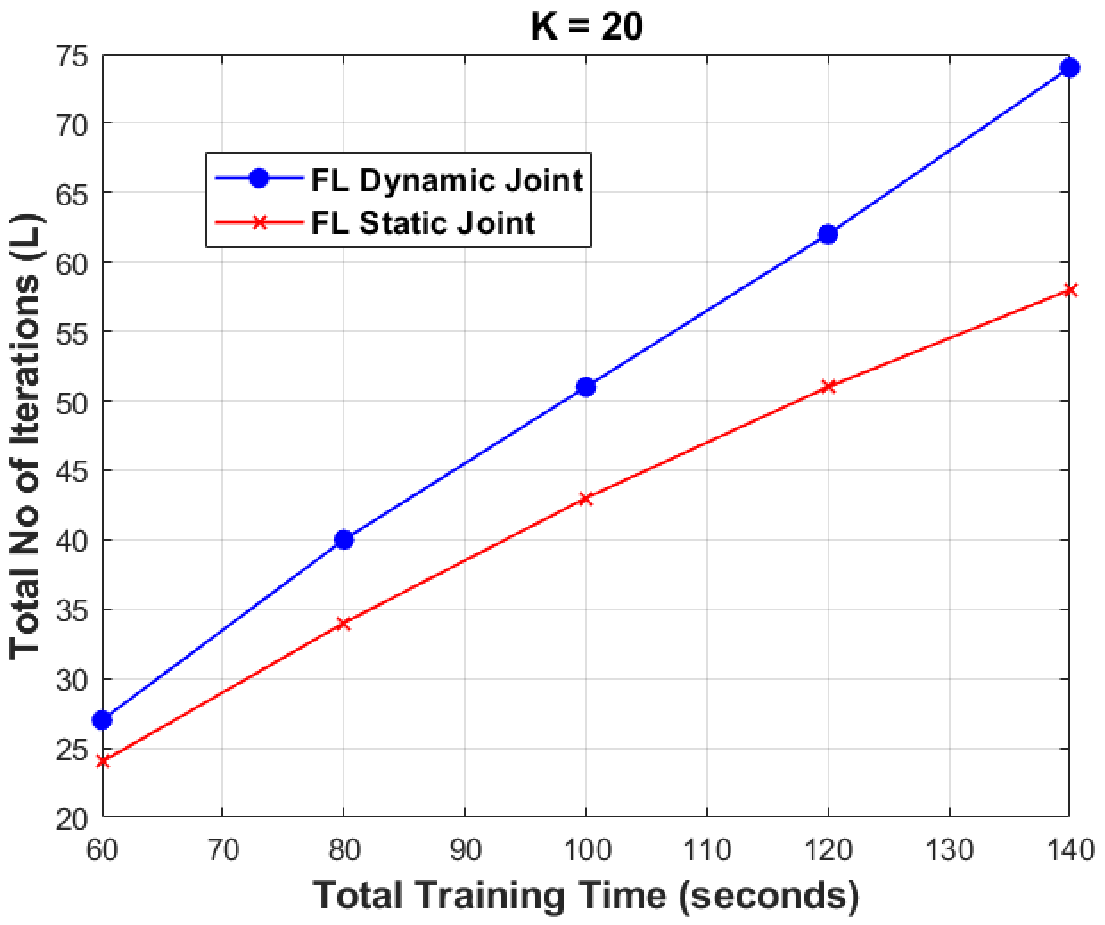

Figure 4.7: Total no. of iterations achieved in FL using the static and dynamic joint algorithms for $K = 20$.

Another interesting observation can be noticed when looking at the number of iterations achieved by these algorithms in PL and FL settings in Fig. 4.3 and Fig. 4.7, respectively. First, we can observe that the FL dynamic and static joint algorithms always run more training iterations their corresponding counterparts in PL settings. This is quite anticipated as, unlike PL, no data is sent along with the model in FL, which saves more time for computations and thus training iterations in the latter. Yet, we can also observe that, at $T = 140s$, the PL dynamic joint algorithm achieves 61 total iterations while the FL static joint algorithm achieves 58 total iterations, proving again the merits of the dynamic algorithm in adapting the task planning within the course of the training period, thus achieving a better performance.

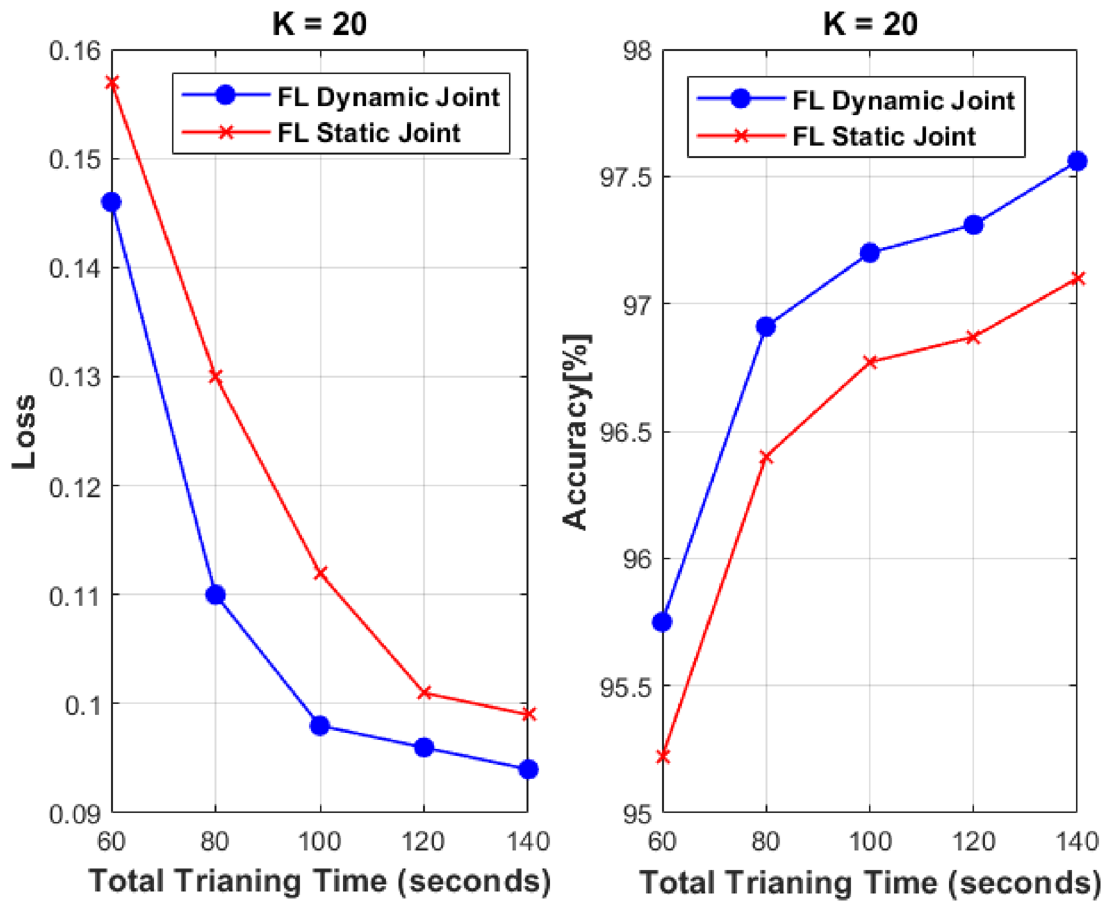

Figure 4.8: Terminal loss function and accuracy achieved in FL using the static and dynamic joint algorithms for $K = 20$.

In Fig. 4.8, the terminal loss function and validation accuracy of the trained models using both the FL dynamic and static joint algorithms is plotted against total training times ranging between 60 and 140 seconds. Again, the FL dynamic algorithm can be observed to be the top-performing algorithm at any given total training time. For instance, this algorithm achieved 97.1% accuracy at $T = 90$ s, while the static algorithm achieved 96.61%.

At $T = 140$ s, the dynamic algorithm was able to cross the 97.5% mark, while the static algorithm being 0.5% below. Comparing the curves in Fig. 4.8 with those in Fig. 4.5 and 4.6 for the dynamic and static joint algorithms, we can again see that they can achieve higher accuracy in FL than in PL for the same total training time, which is again anticipated due to the aforementioned reasons. Yet, we can observe that the dynamic algorithm in PL settings achieves almost the same accuracy as that of static algorithm in FL settings, demonstrating the importance of in-training adaptation of the task planning.

Chapter 5
Conclusions

5.1 Summary and Conclusion

In this work, we focused on finding a solution to the MEL problem of how to improve the overall aggregated model efficiency for both FL and PL, which is the problem of evaluating the planning parameters and the allocation of tasks and resources to achieve a desired accuracy or to minimize the global loss function. This was investigated by jointly optimizing the planning of learning tasks and physical resource allocation with the two objectives in mind.

1. Maximizing the number of local iterations in each time-constrained global cycle
2. Minimizing the terminal loss function of the trained model given a global training time constraint

We began by exploring how the number of local iterations affects the aggregated model in both FL and PL and formulated our problem with the intention of maximizing the number of local iterations within each time-constrained global cycle time.. The general joint problem was formulated as a nonlinear constrained integer-linear problem, which was then simplified by finding the trivial optimal solutions for several variables. Being still complex, we solved the problem numerically and quantified its gains in maximizing the number of local iterations and converging to a given accuracy compared to the TP and EDA schemes. Through extensive experimenting, the joint scheme was proven to outperform both schemes by achieving the same accuracy in less time with a smaller number of learners.

We then moved on to investigating the problem of jointly planning learning tasks and allocating physical resources for MEL, both in PL and FL settings with a global training time constraint, over networks of nodes employing OFDMA for physical communications. Our problem is formulated with the goal of minimizing the terminal loss function of the trained model, by optimizing the task planning and physical resource parameters. Being a complicated problem to solve, we put forward a three-step approach to simplify and solve it. We were also able to prove that the second and most critical step of our proposed solution for FL settings involves solving a strictly convex problem, which makes it easier to solve. We then developed a static and dynamic algorithm to implement the three-steps solution for both PL and FL. Our simulation results showed obvious dominance in performance for our proposed algorithms when compared to the TP and EDA schemes, especially for the shorter total training times that are typically available in mobile and ad-hoc edge learning environments. Lastly, our results consistently confirmed the merits of the proposed dynamic algorithm compared to the static algorithm.

5.2 Future Work and Recommendations

In this section, we discuss our recommendations based on the various challenges we faced throughout our work. We also propose future research initiatives that would improve the overall performance of the MEL system.

- As MEL is an emerging field, now is the optimum point in time to standardize the nature of the developments made in this area ensuring they are grounded in real-world settings, assumptions, and datasets. It is critical for new research to use and expand on the existing implementations and benchmarking tools, such as LEAF [62] rather than starting back from zero.

- It is important to note that the work discussed thus far has been developed with the task of supervised learning in mind. Although this work is applicable to other learning models, in practice we would need to perform some exploratory data analysis, determine aggregate statistics, or run a more complex task such as reinforcement learning. Tackling problems beyond supervised learning in MEL networks will likely require addressing similar challenges of scalability, heterogeneity, and privacy

- We plan to extend our studies to scenarios with multiple simultaneous PL and FL tasks. In these scenarios, we will seek to optimize the associations of learners to different learning tasks alongside with the planning of these tasks and the allocation of physical resources to each involved learner.

- In practical scenarios the set of learners does not necessarily need to be benevolent, which is an assumption that we made before starting our learning process. Finding ways to not only identify malicious users but also eliminate them from being part of the set of learners for the system is an extremely important direction that will bring this research even closer to real and practical scenarios.

References

[1] "Cisco annual internet report," vol. 2018-2023, White Paper, Cisco, San Jose, CA, USA 2020. Available from: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

[2] M. Chiang and T. Zhang, "Fog and iot: An overview of research opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, 2016.

[3] D. A. Gupta and A. Dhami, "Measuring the impact of security, trust and privacy in information sharing: A study on social networking sites," *Direct*, vol. 17, 2015.

[4] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. M. Mao, S. Sen, and O. Spatscheck, "An in-depth study of lte: Effect of network protocol and application behavior on performance," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, p. 363–374, 2013.

[5] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, H. B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," 2019.

[6] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song, "The secret sharer: Evaluating and testing unintended memorization in neural networks," *28th Security Symposium*, pp. 267–284, 2019.

[7] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Privacy aware learning," *Journal of the ACM (JACM)*, vol. 61, no. 6, pp. 1–57, 2014.

[8] S. Russell and P. Norvig, "Artificial intelligence: a modern approach," 2002.

[9] M. I. Jordan and D. E. Rumelhart, "Internal world models and supervised learning," *Machine Learning Proceedings*, pp. 70–74, 1991.

[10] Z.-H. Zhou, "A brief introduction to weakly supervised learning," *National science review*, vol. 5, no. 1, pp. 44–53, 2018.

[11] E. Alpaydin, *Introduction to machine learning.* MIT press, 2010.

[12] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, no. 1, pp. 3–24, 2007.

[13] F. Osisanwo, J. Akinsola, O. Awodele, J. Hinmikaiye, O. Olakanmi, and J. Akinjobi, "Supervised machine learning algorithms: classification and comparison," *International Journal of Computer Trends and Technology (IJCTT)*, vol. 48, no. 3, pp. 128–138, 2017.

[14] K. A. BOLLEN and R. W. JACKMAN, "Regression diagnostics: An expository treatment of outliers and influential cases," *Sociological Methods & Research*, vol. 13, no. 4, pp. 510–542, 1985.

[15] A. Dasgupta, Y. V. Sun, I. R. König, J. E. Bailey-Wilson, and J. D. Malley, "Brief review of regression-based and machine learning methods in genetic epidemiology: the genetic analysis workshop 17 experience," *Genetic epidemiology*, vol. 35, no. S1, pp. S5–S11, 2011.

[16] Z.-H. Zhou and M. Li, "Semi-supervised regression with co-training." *International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 5, pp. 908–913, 2005.

[17] E. Oja, "Finding clusters and components by unsupervised learning," *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pp. 1–15, 2004.

[18] M. Usama, J. Qadir, A. Raza, H. Arif, K.-L. A. Yau, Y. Elkhatib, A. Hussain, and A. Al-Fuqaha, "Unsupervised machine learning for networking: Techniques, applications and research challenges," *IEEE Access*, vol. 7, pp. 65 579–65 615, 2019.

[19] R. Gentleman and V. J. Carey, "Unsupervised machine learning," *Bioconductor case studies*, pp. 137–157, 2008.

[20] M. Z. Rodriguez, C. H. Comin, D. Casanova, O. M. Bruno, D. R. Amancio, L. d. F. Costa, and F. A. Rodrigues, "Clustering algorithms: A comparative approach," *PLOS ONE*, vol. 14, no. 1, pp. 1–34, 2019.

[21] J. Cui, Z. Ding, P. Fan, and N. Al-Dhahir, "Unsupervised machine learning-based user clustering in millimeter-wave-noma systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 11, pp. 7425–7440, 2018.

[22] A. Kassambara, "Practical guide to cluster analysis in r: Unsupervised machine learning," vol. 1, 2017.

[23] K. Thangavel and A. Pethalakshmi, "Dimensionality reduction based on rough set theory: A review," *Applied Soft Computing*, vol. 9, no. 1, pp. 1–12, 2009.

[24] O. Kramer, *Dimensionality reduction with unsupervised nearest neighbors.* Springer, 2013.

[25] M. Dash, H. Liu, and J. Yao, "Dimensionality reduction of unsupervised data," *Proceedings ninth ieee international conference on tools with artificial intelligence*, pp. 532–539, 1997.

[26] P. Dayan and Y. Niv, "Reinforcement learning: the good, the bad and the ugly," *Current opinion in neurobiology*, vol. 18, no. 2, pp. 185–196, 2008.

[27] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018.

[28] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[29] S. Wang, T. Tuor, T. Salonidis, 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. PP, pp. 1–1, 2019.

[30] M. M. Shahabadi and M. Uplane, “Synchronous and asynchronous e-learning styles and academic performance of e-learners,” *Procedia - Social and Behavioral Sciences*, vol. 176, pp. 129–138, 2015.

[31] M. Ashouri, F. Lorig, P. Davidsson, R. Spalazzese, and S. Svorobej, “Analyzing distributed deep neural network deployment on edge and cloud nodes in iot systems,” pp. 59–66, 2020.

[32] U. Mohammad, S. Sorour, and M. Hefeida, “Task allocation for asynchronous mobile edge learning with delay and energy constraints,” 2020.

[33] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. a. Ranzato, A. Senior, P. Tucker, K. Yang, Q. Le, and A. Ng, “Large scale distributed deep networks,” *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[34] W. Zhang, S. Gupta, X. Lian, and J. Liu, “Staleness-aware async-sgd for distributed deep learning,” 2016.

[35] T. Li, A. K. Sahu, A. S. Talwalkar, and V. Smith, “Federated learning: Challenges, methods, and future directions,” *IEEE Signal Processing Magazine*, vol. 37, pp. 50–60, 2020.

[36] M. Li, L. Zhou, Z. Yang, A. Li, F. Xia, D. G. Andersen, and A. Smola, “Parameter server for distributed machine learning,” *Big Learning NIPS Workshop*, vol. 6, p. 2, 2013.

[37] M. M. Wadu, S. Samarakoon, and M. Bennis, "Federated learning under channel uncertainty: Joint client scheduling and resource allocation," *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, 2020.

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

[39] O. Habachi, M.-A. Adjif, and J.-P. Cances, "Fast uplink grant for noma: a federated learning based approach," 2019.

[40] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu, "Gaia: Geo-distributed machine learning approaching lan speeds," *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation*, p. 629–647, 2017.

[41] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," *IEEE Conference on Computer Communications*, pp. 63–71, 2018.

[42] 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, 2021.

[43] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 269–283, 2021.

[44] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Convergence time optimization for federated learning over wireless networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 4, pp. 2457–2471, 2021.

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

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

[47] J. Konecny, J. Liu, P. Richtarik, and M. Takac, “Mini-batch semi-stochastic gradient descent in the proximal setting,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 2, p. 242–255, 2016.

[48] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, “Deep learning for iot big data and streaming analytics: A survey,” *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 2923–2960, 2018.

[49] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. a. Ranzato, A. Senior, P. Tucker, K. Yang, Q. Le, and A. Ng, “Large scale distributed deep networks,” *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[50] T. Tuor, S. Wang, T. Salonidis, B. J. Ko, and K. K. Leung, “Demo abstract: Distributed machine learning at resource-limited edge nodes,” *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 1–2, 2018.

[51] A. Del Pia, S. Dey, and M. Molinaro, “Mixed-integer quadratic programming is in np,” *Mathematical Programming*, vol. 162, 2014.

[52] 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.

[53] J. Currie and D. Wilson, "Opti: Lowering the barrier between open source optimizers and the industrial matlab user," 2012.

[54] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[55] L. Bottou, "Large-scale machine learning with stochastic gradient descent," *Proceedings of COMPSTAT*, pp. 177–186, 2010.

[56] J. Ji, X. Chen, Q. Wang, L. Yu, and P. Li, "Learning to learn gradient aggregation by gradient descent," *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 2614–2620, 2019.

[57] I. Czarnowski, "Prototype selection algorithms for distributed learning," *Pattern Recognition*, vol. 43, pp. 2292–2300, 2010.

[58] I. Kim, H. L. Lee, B. Kim, and Y. Lee, "On the use of linear programming for dynamic subchannel and bit allocation in multiuser ofdm," *IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 6, pp. 3648–3652 vol.6, 2001.

[59] Y. J. Zhang and K. Letaief, "Multiuser adaptive subcarrier-and-bit allocation with adaptive cell selection for ofdm systems," *IEEE Transactions on Wireless Communications*, vol. 3, no. 5, pp. 1566–1575, 2004.

[60] G. Zhang, "Subcarrier and bit allocation for real-time services in multiuser ofdm systems," *IEEE International Conference on Communications*, vol. 5, pp. 2985–2989 Vol.5, 2004.

[61] C. Y. Wong, C. Tsui, R. Cheng, and K. Letaief, "A real-time sub-carrier allocation scheme for multiple access downlink ofdm transmission," *IEEE VTS 50th Vehicular Technology Conference*, vol. 2, pp. 1124–1128 vol.2, 1999.

[62] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečný, H. B. McMahan, V. Smith, and A. Talwalkar, “Leaf: A benchmark for federated settings,” 2019.