

On Task Allocation in Mobile Edge Computing with a Focus on Machine Learning
Applications

A Dissertation

Presented in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

with a

Major in Electrical Engineering

in the

College of Graduate Studies

University of Idaho

by

Umair Mohammad

Major Professor: Mohamed Hefeida, Ph.D.

Committee Members: Sameh Sorour, Ph.D.; Brian K. Johnson, Ph.D., P.E.;

Yacine Chakhchoukh, Ph.D.; Ahmed Ibrahim, Ph.D., P.E.

Department Administrator: Joseph D. Law, Ph.D.

August 2020

Authorization to Submit Dissertation

This dissertation of Umair Mohammad, submitted for the degree of Doctor of Philosophy with a major in Electrical Engineering and titled “On Task Allocation in Mobile Edge Computing with a Focus on Machine Learning Applications,” has been reviewed in final form. Permission, as indicated by the signatures and dates given below, is now granted to submit final copies to the College of Graduate Studies for approval.

Major Professor: _____ Date: _____

Mohamed Hefeida, Ph.D.

Committee Members: _____ Date: _____

Sameh Sorour, Ph.D.

Brian K. Johnson, Ph.D., P.E.

Yacine Chakhchoukh, Ph.D.

Ahmed Ibrahim, Ph.D., P.E.

Department
Administrator: _____ Date: _____
Joseph D. Law, Ph.D.

Abstract

Mobile Edge Computing (MEC) is proving to be a very successful alternative to cloud computing (CC) for executing computationally intensive tasks that cannot be handled by end devices such as laptops and smart phones. The MEC paradigm facilitates task computation at the edge of the network rather than the cloud. It has been shown that MEC decreases delay and energy consumption while simultaneously reducing the load on backbone networks and the clouds. The hierarchical-MEC (H-MEC) paradigm adds on to MEC by involving idle end devices in the task computation process. H-MEC can offer the advantages of MEC, while further reducing the load on edge networks and edge servers. A large proportion of edge computing tasks comprise some form of data analytics where there is an increasing trend to use machine learning (ML) techniques. Therefore, in order to share the burden of complex ML algorithms and to preserve data privacy, employing the ML training in a distributed manner on end devices or learners, is becoming more common. This is especially important with the advent of the deployment of 5th generation (5G) networks which may offer the feature of fast device-to-device (D2D) communication.

This dissertation proposes a method for optimal task offloading in H-MEC while jointly minimizing delay and energy consumption. Analysis on H-MEC task allocation with the joint time and energy minimization showed that the problem is NP-hard and hence, a heuristic solution was proposed. Results indicate that allowing one idle user to act as a server can provide up-to a 13% reduction in completion time and up-to 17% reduction in energy consumption. The focus of the dissertation then shifts to the major component of this work, which is enabling and optimizing the execution of machine learning tasks in a distributed manner, referred to as distributed Learning (DL), on H-MEC systems.

Consequently, we design the novel paradigm of Mobile Edge Learning (MEL), where the goal is to allocate tasks optimally such that the ML model accuracy is maximized while taking into consideration the heterogeneous communication and computational capabilities of

individual learners in a wireless MEC system. More specifically, our approach is heterogeneity aware (HA) compared to previous works which were heterogeneity unaware (HU). This part of the research investigates MEL optimal task allocation for synchronous (HA-Sync) and asynchronous (HA-Asyn) settings, with limits on the global completion time and local energy consumption. In the last part of the work, we provide recommendations on best scheme selection and how to apply the MEL in context of H-MEC.

The problem of optimal task allocation in MEL is divided into four sub-problems consisting of HA-Sync and HA-Asyn with only time constraints and the HA-Sync/Asyn with dual time and energy constraints. All sub-problems are shown to be non-polynomial (NP) hard and hence, solutions based on relaxations are proposed. For the HA-Sync with time constraints, analytical upper bounds are proposed and shown to perform similar to the numerical approaches. For the HA-Asyn with time constraints and the HA-Sync/Asyn with dual time and energy constraints, solutions based on the suggest-and-improve (SAI) framework are proposed.

Simulation results on MEL show that our proposed HA approaches achieve a superior validation accuracy and provide significant reductions in time for reaching a certain accuracy threshold compared to HU schemes when there is a limit on the global completion time. For example, the HA-Sync schemes reduce training time by up-to 25% compared to HU, whereas the HA-Asyn can provide further gains of up-to 10% in some settings. When there are joint global time and local energy consumption constraints, the HA-Sync/Asyn approaches can provide gains of up-to 25% compared to the HU schemes. Furthermore, because of different settings, where the HA-Asyn and HA-Sync outperform each other, this dissertation concludes by providing recommendations on how to select the appropriate scheme with the correct parameters, describing different application scenarios and identifying areas for future research.

Acknowledgements

First of all, I would like to thank God and send peace and blessings to the last Prophet. Then, I would like to start by expressing my gratitude to my advisor Dr. Mohamed Hefeida and committee member Dr. Sameh Sorour who for all intents and purposes, was a co-advisor. In fact, I would like to thank him for giving me this opportunity to advance my research career at the University of Idaho (UI) as a research assistant (RA) and ensuring that I would be able to continue my work even after he moved for a new position at another institution. I would then like to thank Dr. Mohamed Hefeida for agreeing to become my advisor and actively participating in our research by providing valuable feedback, advice, and career counseling among other things. His support throughout my research and other endeavors has been invaluable.

Besides my advisors, I would like to thank the rest of my dissertation committee: Professors Brian Johnson, Yacine Chakhchoukh, and Ahmed Ibrahim for their insightful comments and encouragement. One of my first classes at UI was with Dr. Johnson; his vast knowledge and willingness to explore new areas never ceases to amaze me. I would like to thank Dr. Yacine with whom I had the pleasure to take a class and also do some research. I am also grateful to Dr. Ahmed Ibrahim for providing his insights as an external member and for stepping up to support the Muslim Student Association (MSA) at UI by taking on the role of faculty advisor.

I would like to give a special mention to the Moscow community in general and the Islamic Center/Community of Moscow (ICM) in particular. I would like to mention two of my closest friends and lab neighbors Mohammad Allehyani and Ahmed Momen, whom I got to know very well starting from my very first days at UI. I would also like to point out my previous roommates Ammar Tarar and Bassam Al-Sharif, especially Ammar for putting up with my tough schedule. For being my support system, I am grateful to the Pakistani community at Moscow including: Ammar, Aamir, Farjad, Faizan, Kevin, and Subhan.

Finally, I would like to thank my family starting with my parents, who provided strong moral support and words of encouragement during my early days at UI. From high school to Bachelor's degree all the way up-to graduate school and Ph.D., nothing would have been possible without their sacrifices. I would also like to thank my brothers and sisters especially for making my vacations memorable which always provided me a boost for when I returned to work.

Last but not least, I would like to express my sincere appreciation and gratitude to my wife, Afraa, who has been with me on this journey for the last year and a half; spending the last year with me here at Moscow. A special mention to her for taking a break in her career and supporting me in my endeavors through the many lows and few highs. This final push would not have been possible without her support.

Dedication

To my mom Kausar and my dad Yaqub

To my wife Afraa

To my brothers and sisters

Table of Contents

Authorization to Submit Dissertation	ii
Abstract.....	iii
Acknowledgements	v
Dedication	vii
Table of Contents	viii
List of Tables	xii
List of Figures	xiii
List of Abbreviations	xvi
1 Introduction	1
1.1 Motivation.....	5
1.2 Applications	7
1.3 Dissertation Organization	9
2 Literature Review	12
2.1 Contributions	17
3 Hierarchical Mobile Edge Computing (H-MEC).....	20
3.1 Introduction to MEC	20
3.2 Transition to H-MEC.....	21
3.3 Formulation with Joint Delay and Energy Efficiency	24
3.4 Proposed Solution	28
3.4.1 Peer Offloading Priority Based Heuristic Algorithm	31

3.5	Results and Discussion.....	33
4	Mobile Edge Learning (MEL) System Model.....	36
4.1	Machine Learning.....	36
4.2	Distributed Learning.....	38
4.3	Transition to MEL.....	39
4.3.1	Local Completion Time.....	44
4.3.2	Local Energy Consumption.....	47
4.4	MEL Model Summary.....	48
5	MEL with Time Constraints.....	51
5.1	Synchronous Task Allocation.....	51
5.1.1	Problem Formulation.....	51
5.1.2	Proposed Solution.....	53
5.1.3	Simulation Environment for Testing the HA-Sync.....	55
5.1.4	Simulation Results for Parallelized Learning (PL/TP).....	57
5.1.5	Simulation Results for Federated Learning (FL/DD).....	61
5.1.6	Complexity Analysis.....	69
5.2	Asynchronous Task Allocation.....	72
5.2.1	Formulation.....	73
5.2.2	Proposed Solution.....	75
5.2.3	Results and Discussion.....	77
5.3	Chapter Summary.....	81
6	MEL with Dual Time and Energy Constraints.....	82
6.1	Synchronous Task Allocation with Time and Energy Constraints.....	82
6.1.1	Formulation.....	82
6.1.2	Proposed Solution.....	84
6.1.3	Results and Discussion.....	86

6.2	Asynchronous Task Allocation with Time and Energy Constraints.....	91
6.2.1	Problem Formulation.....	92
6.2.2	Proposed Solution.....	94
6.2.3	Test Setup for HA-Asyn with Dual Time and Energy Constraints	96
6.2.4	Convergence Proof Results	98
6.2.5	Validation Accuracy Results.....	100
6.2.6	Discussions	103
7	Optimal Task Allocation.....	104
7.1	Problem Formulation	104
7.1.1	Convergence Bounds.....	106
7.2	Proposed Solution	109
7.2.1	Relating Bounds to the Number of Local Updates (τ).....	110
7.3	Results and Discussion.....	111
8	Recommendations, Conclusions, and Future Outlook.....	114
8.1	Recommendations	114
8.2	Conclusions	115
8.3	Future Work.....	116
	Bibliography	117
	Appendix A Proof of Theorem 1	134
	Appendix B Proof of Theorem 2	137
	B.1 Obtaining \mathbf{u} and \mathbf{u}'	137
	Appendix C Proof of Theorem 3	139
	Appendix D Proof of Lemma 1	141

Appendix E Proof of Theorem 4	143
Appendix F List of Published, Accepted, or Submitted Papers.....	146
F.1 Journal Papers	146
F.2 Conference Papers.....	146
F.3 Other Journal Publications.....	147
Appendix G IEEE Formal Reuse Licenses	148

List of Tables

5.1	Simulation parameters for testing the HA-Sync with only time constraints . . .	56
5.2	Simulation parameters for testing the HA-Asyn with only time constraints . . .	78
6.1	Simulation parameters for HA-Sync with dual time and energy constraints. . . .	87
6.2	Simulation parameters for HA-Asyn with dual time and energy constraints . . .	97
7.1	Simulation environment for optimal task allocation	112

List of Figures

1.1	An illustration of several edge networks connected to a central cloud, each containing multiple nodes.	2
1.2	IoT market-share forecast of various industries from 2017-2022. Data was taken from [1].	6
2.1	Review chart and research gap for H-MEC	17
2.2	Review chart and research gap for MEL	18
3.1	Illustration of multiple edge networks with a zoomed in view of one network to demonstrate the concept of MEC.	21
3.2	Logical grouping of end devices in an H-MEC. Note that the devices are grouped together for classification/presentation purposes. Practically, the servers and offloaders will be co-located such that they are randomly placed within the radius served by the edge server.	22
3.3	Illustration of the proposed peer offloading priority based heuristic algorithm.	32
3.4	Plot of system utility versus number of users	34
3.5	Plot number of users offloading to the edge versus number of users	35
4.1	Illustration of the DL process with DP	40
4.2	Comparison of DD/FL and TP/PL paradigms.	41
4.3	System model of a MEL setting	43
5.1	Achievable local iterations τ for the MNSIT dataset. (a) Performance comparison of all schemes for $T = 30$ and 60 s vs K (b) Performance comparison of all schemes for $K = 10$ and 20 vs T	58
5.2	Achievable local iterations τ for the Pedestrian dataset. (a) Performance comparison of all schemes for $T = 15$ and 5 s vs K (b) Performance comparison of all schemes for $K = 10$ and 20 vs T	59

5.3	Validation accuracy progression for the MNIST Dataset after global cycles updates for (a) $K = 20$ and $T = 12$ s (b) $K = 10$ and $T = 30$ s	61
5.4	Validation accuracy progression for the Pedestrian Dataset after global cycles updates for (a) $K = 20$ and $T = 5$ s (b) $K = 20$ and $T = 3$ s	62
5.5	Achievable local iterations τ for the MNSIT dataset. (a) Performance comparison of all schemes for $T = 30$ and 60 s vs K (b) Performance comparison of all schemes for $K = 10$ and 20 vs T	63
5.6	Achievable local iterations τ for the Pedestrian dataset. (a) Performance comparison of all schemes for $T = 15$ and 5 s vs K (b) Performance comparison of all schemes for $K = 10$ and 20 vs T	64
5.7	Achievable local iterations τ for the MNSIT dataset. (a) Performance comparison of all schemes for $T = 30$ and 60 s vs K (b) Performance comparison of all schemes for $K = 10$ and 20 vs T	65
5.8	Achievable local iterations τ for the Pedestrian dataset. (a) Performance comparison of all schemes for $T = 15$ and 5 s vs K (b) Performance comparison of all schemes for $K = 10$ and 20 vs T	66
5.9	MNIST Learning Results. Learning accuracy comparison for $T = 60$ s and $K = 10$ using both, the dynamic and ETA approaches, for both frameworks: distributed datasets and task-parallelization.	67
5.10	Pedestrian learning results. Learning accuracy comparison for $T = 5$ s and $K = 10$ using both, the dynamic and ETA approaches, for both frameworks: distributed datasets and task-parallelization.	69
5.11	MNIST results (a) Performance comparison of all schemes for $T = 30$ and 60 s vs K (b) Performance comparison of all schemes for $K = 10$ and 20 vs T	70

5.12	Complexity comparisons for the MNIST dataset for the TP scheme for $T = 30$ and 60 s vs K illustrating: (a) Execution times of all methods (HA analytical and numerical against the HU) (b) Execution times of the HA analytical versus HU scheme.	72
5.13	Maximum and Average Staleness vs K for $T = 7.5$ s and $T = 15$ s.	79
5.14	Validation accuracy progression after global update cycles for $K = 10, 15$ and 20 for $T = 15$ s	80
6.1	Achievable number of local update cycles for average energies of 50 to 250 J in steps of 50 J by all schemes (a) vs K for $T = 20$ s (b) vs T for $K = 20$	88
6.2	Learning accuracy progression after global cycles updates for $K = 20$	90
6.3	Learning accuracy progression after global cycles updates for $K = 20$ learners	91
6.4	Final validation accuracy achieved after a total of 12 global epochs with various training times for an average device energy consumption of 10 J.	98
6.5	Accuracy progression for the cases when $T = 10$ s and 20 s and for average energy values of $E = \{10, 20, 30\}J$	99
6.6	Final validation accuracy values after 6 and 12 global updates, respectively, for various training times for average energy consumption limits $E = 10$ J and $E = 20$ J, respectively.	100
6.7	Number of global updates needed to reach a validation accuracy of 95.5% for: (a) $E = 10$ J and (b) $E = 20$ J and the number of updates needed to reach an accuracy of 97.3% for: (c) $E = 10$ J and (d) $E = 20$ J.	102
6.8	Results for an MEL system with average energy $E = 30$ J. Final validation after (a) 6 global updates and (b) 12 global updates, and number of global updates to reach an accuracy threshold of (c) 95.5% and (d) 97.3%	102
7.1	Training loss versus total training time for $K = 20$ learners	112
7.2	Validation accuracy versus total training time for $K = 20$ learner	113

List of Abbreviations

5G	...	5th Generation
6G	...	6th Generation
AI	...	Artificial Intelligence
CC	...	Cloud Computing
CNN	...	Convolutional Neural Network
COVID-19	...	Coronavirus Disease 2019
CSI	...	Channel State Information
CV	...	Connected Vehicles
D2D	...	device-to-device
DD	...	Distributed Datasets
DL	...	Distributed Learning
DNN	...	Deep Neural Network
EC	...	Edge Computing
EI	...	Edge Intelligence
ETA	...	Equal Task Allocation
FL	...	Federated Learning
GPU	...	Graphical Processing Unit
HA	...	Heterogeneity Aware
H-MEC	...	Hierarchical Mobile Edge Computing
HPC	...	High Performance Computing
HU	...	Heterogeneity Unaware
ILP	...	Integer Linear Program
IoE	...	Internet of Everything
IoT	...	Internet of Things
KKT	...	Karush-Kuhn-Tucker
MEC	...	Mobile Edge Computing
MEL	...	Mobile Edge Learning
ML	...	Machine Learning
NN	...	Neural Network
OBU	...	On-board Unit
PL	...	Parallelized Learning
QCILP	...	Quadratically-Constrained Integer Linear Program
QCIQP	...	Quadratically-Constrained Integer Quadratic Program
QCQP	...	Quadratically-Constrained Quadratic Program
RL	...	Reinforcement Learning
RSU	...	Roadside Unit
SAI	...	Suggest-And-Improve
SVM	...	Support Vector Machine
TP	...	Task Parallelization
UE	...	User Equipment

Chapter 1: Introduction

Resource intensive applications including facial recognition, virtual reality, and advanced image enhancement/filtering have become a vital component of modern-day mobile devices and networks. Furthermore, rapid advancements of smart infrastructure such as smart grids, cities, and cars is mandating the deployment of a large number of Internet-of-Things (IoT) devices. This equipment generates exponentially increasing amounts of data at the edge of the network, which needs to be communicated to the cloud for processing. Although cloud computing (CC) technologies have bridged the gap between the required and available resources effectively [2], it is expected that the rate and nature of this generated data will prohibit their centralized processing and analytics at cloud servers. The expected exponential rise in the amount of end devices served by base stations and edge routers coupled with the deployment of 5th generation (5G) networks that consist of macro-cells and the possibilities for D2D communication in the future, have encouraged researchers to investigate alternate computing paradigms. Mobile edge computing (MEC) has emerged as a promising paradigm that provides an effective alternative to CC [3].

The cloud-edge-user model is a three layer architecture, where several end users are connected to a wireless network via an edge server, and multiple edge servers connect several such 'edge networks' to a central server called the cloud. The edge layer is also known as the fog layer. The terms 'edge' and 'fog' have been used interchangeably in the literature, but we will strictly use the term 'edge' in this dissertation. The end users are also known as end devices, user equipment (UE), nodes, edge nodes, fog nodes, and edge devices among other names. In this dissertation, we will use some of the mentioned terms interchangeably. The end users are usually connected to each other via a wireless network which is controlled by an edge server. Hence, one such network of devices is called the wireless edge network or the 'wireless edge'/'edge' for short.

The edge may be a mobile base station, Wi-Fi router, a road side unit (RSU) in a connected vehicle (CV) network, etc. An end device may be a laptop, smart phone, a

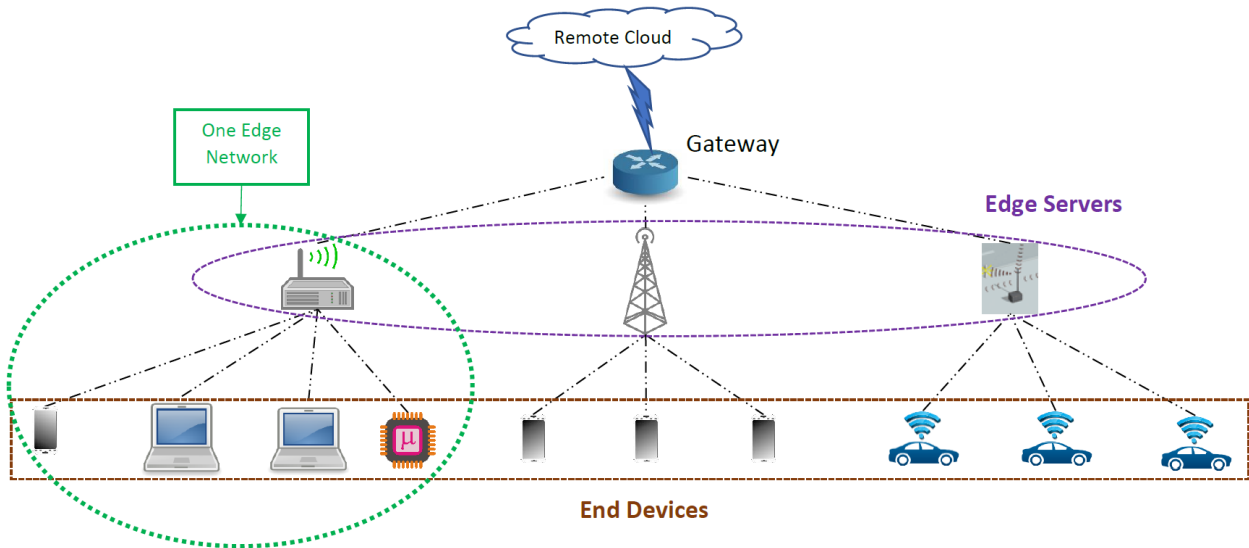


Figure 1.1: An illustration of several edge networks connected to a central cloud, each containing multiple nodes.

processor attached to the on-board unit (OBU) on a smart car, a smart meter, a micro-processor attached to a microcontroller such as the Raspberry Pi or Arduino, etc. One edge network can comprise the same or multiple types of such devices as illustrated in Figure 1.1.

Because several of the edge nodes may be mobile, such as a smart phone on a person or an OBU on a car, these types of networks are called mobile edge networks or the 'mobile/wireless edge' for short. The wireless edge is already resource-constrained. In addition to limited energy constraints, a wireless edge suffers from a condensed spectrum, fading effects, and noisy channels. The aspect of mobility adds an additional challenge because the channel will now suffer from fast fading, multipath effects, and interference. Furthermore, end users typically require tasks to be completed as quickly as possible while consuming the least energy because many are battery-powered. Therefore, it is vital to allocate tasks such that resource utilization is optimized.

The concept behind MEC is to allow users the option to locally execute tasks or offload the more computationally intensive tasks to the edge server. The motivation behind this paradigm is that edge servers are typically equipped with multi-core processors and are

powered by the main supply. Because they are mostly being used as routers, a large amount of processing power is wasted. Therefore, employing this equipment for task computation is a promising solution for reducing the burden on backbone networks.

Research on MEC has included completion time and energy consumption minimization, offloading decision optimization, and proposing of different architectures for MEC. One proposed architecture is to merge the previously explored distributed computing paradigms with MEC to develop a hierarchical structure where end devices may offload tasks either to another end device or to the edge server. This new idea falls under the umbrella of H-MEC, where end devices either compute tasks locally, offload to a neighboring device within their edge network or offload to the edge server.

In our work, we originally propose to optimize the offloading decisions as well as the communication and computational resources such that the sum of completion times and energies is jointly minimized. Results show that our proposed solution offers significant advantages over the MEC with similar resource optimization. Although our model for generic task offloading for H-MEC offered interesting results, a deeper literature review revealed that one of the biggest open problems was resource optimization for machine learning (ML) over MEC/H-MEC systems.

Indeed, UEs and other devices nowadays come equipped with various types of sensors including cameras that can collect a large amount of data in a short time. To serve a useful purpose, this data needs to be analyzed. Therefore, increasingly, most task computation comprises data analytics such as forecasting, predictive modeling, object recognition, etc. ML algorithms such as regression (linear and logistic), support vector machines (SVM), neural networks (NN), deep NNs (DNN), convolutional NNs (CNN), etc. have shown superior performance in such applications. Most ML algorithms including the ones mentioned above rely on iterative approaches to predict an output based on an input which comprises the set of features belonging to each data sample of a given dataset.

Because ML models are large and these iterative approaches can be very computationally

complex and time-consuming, we focus on distributing the ML training process, also known as distributed learning (DL), over high performance computing (HPC) systems and graphical processing units (GPUs). For deploying ML models distributedly on edge networks, the classical approach so far has been to collect the data from multiple nodes and transmit it to the cloud for centralized analysis. Transferring these monumental amounts of data generated on devices mostly connected via wireless edge networks spread across vast geographical regions to cloud servers for analysis via multiple backhaul links is time-consuming, costly, and raises security and privacy concerns [4]. Therefore, it is expected that 90% of data analytics will be done on either edge processors using MEC or on the end devices themselves using H-MEC [5]. Thus, training ML algorithms and models over the resource-constrained wireless edge has become a significant area of interest for researchers.

Due to the above reasons and an existing research gap in this area, we narrowed our focus to solving the special case of optimal task allocation for training DL algorithms at the wireless edge. To this end, we designed the novel Mobile Edge Learning or “MEL” paradigm. In the literature, some of this work may fall under the keywords of “Edge Artificial Intelligence” (Edge AI) or “Federated Learning (FL) at the resource-constrained edge”. DL over the wireless edge can take two forms: Federated Learning (FL) where all learners own their own dataset or Parallelized Learning (PL) where the a central server distributes subsets of the data to all learners for collaborative ML training. Please note that FL may also be referred to as Distributed Datasets (DD) and PL as Task-Parallelization (TP).

The novelty of the work presented in this dissertation stems from incorporating: a) heterogeneity awareness (HA), b) Parallelized Learning (PL), and c) dataset size allocations, all of which were partially or completely ignored in previous works. Before we present the related work, we will motivate the subjects of this dissertation with some statistical forecasts on the impact of Edge AI and some applications.

1.1 Motivation

The concept of IoT started a couple of decades ago and was upgraded in the last few years to the internet of everything (IoE). Starting with the example of a refrigerator that orders your milk for you, to fully functional smart homes that come equipped with devices such as the Amazon Echo with its own AI interface Alexa, IoT/IoE has come a long way. Furthermore, in addition to smart homes, the world is embracing smart infrastructure including smart cities, smart grids, smart homes, etc., at an exponential pace.

It is expected that 41 billion IoT devices will be connected to the internet by 2022 [1]. Examples of such end user devices include smart phones, traffic cameras, autonomous connected vehicles, unmanned aerial vehicles (UAVs), etc. Furthermore, Cisco estimates that up-to 800 zettabytes ($\sim 10^{21}$) of data will be present on this equipment [1]. Forecasts by Statista show that the industrial IoT market which was worth \$285 billion USD in 2017 will almost double in five years to \$540 billion USD. Figure 1.2 charts this exponential growth and provides statistics on how various industries such as healthcare and transportation are expected grow in terms of IoT.

According to [6], though edge analytics may not replace the cloud, it will clearly be a game changer. One of the major benefits to service providers for enabling IoT analytics at the edge will be the reduction in CC costs in the form of less communication and storage requirements because data not needed for later use can be analyzed and discarded. Other benefits include enhanced customer experience and more robustness and security.

For the consumers, the promotion of edge analytics is very important because it provides several benefits. Although the general population enjoy advertising their life on social media and many have made careers out of it, they have become more aware of their rights to privacy. Therefore, it is a paradox where people want their services to be delivered as soon as possible in this fast-paced world without compromising their data privacy rights. Edge analytics will remove some of the latencies due to CC and the analysis of data at the edge

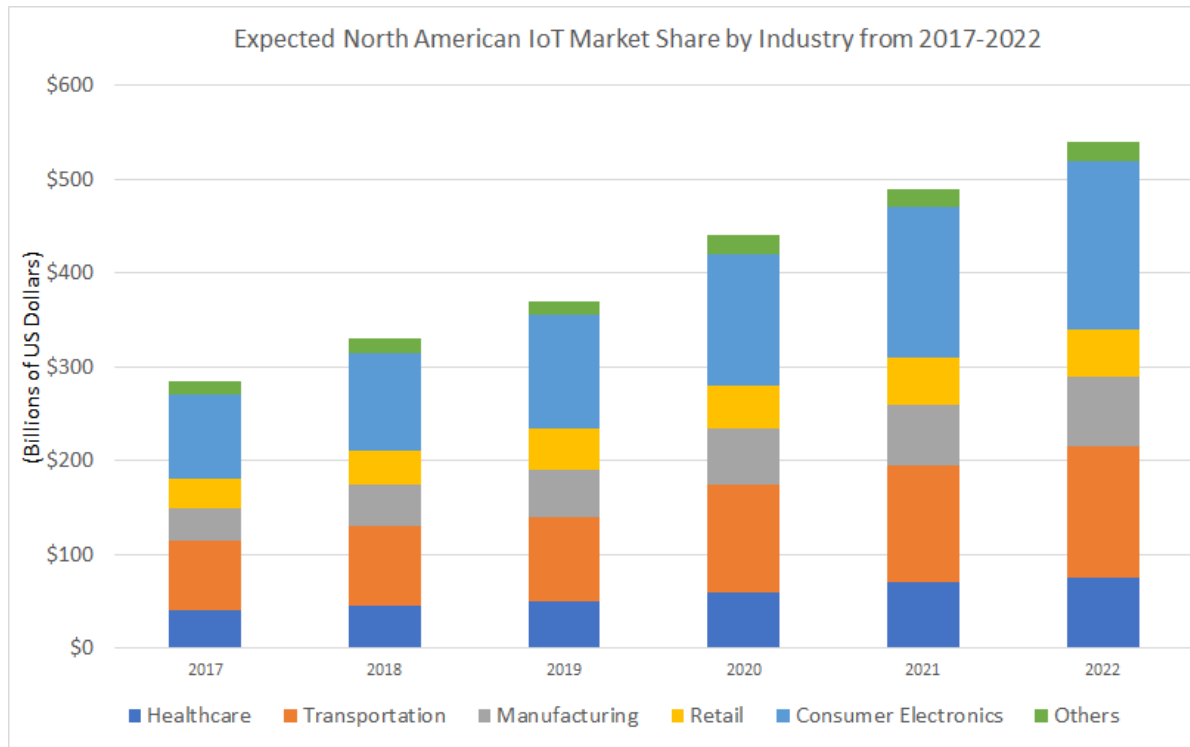


Figure 1.2: IoT market-share forecast of various industries from 2017-2022. Data was taken from [1].

will prevent its collection by central servers which may be owned by third-parties.

As described before, with the success of ML algorithms such as deep learning and reinforcement learning (RL) in analytics and their expected usage in Edge AI or edge intelligence (EI) for short, most of the focus nowadays is enabling DL over the wireless edge [7]. In fact, edge DL is expected to be a key part of technologies driving 6th generation networks (6G) [8]. Indeed, some of the biggest players are actively involved in the R&D of EI deployment. For example, Google has dedicated a research group for FL specifically [9]. Furthermore, Intel is now teaming up with the online education provider Udacity to train developers for edge ML and EI [10]. The aim of this initiative is to supply the talent pool required for deploying real-time analytics platforms at the edge. Most importantly, the IEEE is already working on a standard for deploying DL over the edge [11].

Because of this push from the industry towards EI, many researchers have worked on applying DL concepts to the edge for different applications. For example, related to the

health industry, the work of [12] explores the role of federated learning (FL) in biomedical applications whereas related to the transportation industry, the works of [13] and [14] explore the applications of FL for reliable communications in vehicular networks and for urban sensing, respectively. Overall, we can conclude that edge computing in general and DL over the wireless edge in particular will play a huge role in near-future IoT analytics.

1.2 Applications

We have described two scenarios for DL over the edge, FL and PL. Though we will discuss them in details later, let us go over a couple of quick examples for each. We will begin by the FL scenario first.

Currently, most of the world is dealing with the corona virus 2019 (COVID-19) crisis with an expectation that a second wave of infections and epidemics may pop up in different cities and localities. As it has become clear from this crisis, the number of medical professionals and testing kits are not available in abundance especially in densely populated areas; some of which are located in the developing world. On the other hand, basic equipment such as laptop computers and wireless connections are accessible. Moreover, typically there are strict regulations that are enforced to protect patient privacy.

Let us imagine a large drive-through COVID-19 testing facility such as the ones that were developed in Europe or a stadium that has been converted into a COVID-19 isolation area with medical professionals conducting tests and quarantining positive individuals. Many people who feel even slight symptoms may naturally drive up to get tested. However, a lot of negative results can lead to wasted test kits and consume the time of medical professionals which could have been better served elsewhere.

In such a scenario, if a simple system could be developed where all of the equipment such as tablets or laptops that may be connected via Wi-Fi or Cellular networks, could collect patient data locally and with a program that accepts symptoms and level of severity. Based on initial data of symptoms and test result (positive or negative), a simple ML model

such as logistic regression or SVM can be developed to correlate multiple symptoms to test results. In this case, medical professionals can be replaced by Kiosks where patients enter their symptoms (may be even based on scales of 1 to 10) and this model can be used to adaptively train and predict the probability of a person having COVID-19. A threshold can be set such that the people with the highest probable combination of symptoms end up getting tested or referred to a medical professional.

There are several benefits to such a system. Firstly, patient data will be analyzed at the terminal and there is no need for centralized transfer to the cloud which would require additional steps to make the data anonymous in real-time. For later use or record-keeping, this may be done offline. Medical professionals can focus their attention on serving the most vulnerable patients especially if online Kiosks will be used. Either way, AI has shown promise in several areas such as providing more accurate diagnoses for cancer detection using tumor images compared to human doctors. Hence, there is a potential for less false positives which would be inconvenient to the individual or even false negatives which can be catastrophic. Therefore, a junior medical professional may collect the history/symptom data and enter into the system which in turn would provide a suggestion on next steps. The experts can then utilize their time better by helping the most vulnerable patients.

For the PL scenario, let us go back to that refrigerator example of the early 2000's. Imagine a smart fridge equipped with a small processor and memory that stores data related to the items bought. Let us say it was to train an ML model to forecast the expected demand of different items to create an automated grocery list. The refrigerator has the data but not the computational capability to run such an analysis. In this case, it may collaborate with devices already located at the household such as other smart phones and laptops which are all connected via a Wi-Fi router. In this way, the refrigerator may leverage the computing power of some of the idle devices and a good network by distributing partial data to each learner. Notice that the data still remains private within the confines of the household in this example.

With the deployment of 5G and rapid manufacturing of smart medical devices and the likes, such applications will very soon become a reality. For example, the COVID-19 example can be extended for thermal tracking at airports and at large scale events such as sporting competitions once normal life resumes. Despite the promise and many advantages offered by edge computing (EC) in general and EI in particular, challenges remain.

For example, there is the social challenge where people need to be convinced that these paradigms will preserve their security and privacy. Furthermore, there are remaining challenges where applications have to be deployed with the help of equipment and resources owned by multiple vendors and hence, standards need to be designed and implemented to promote interoperability. At the fundamental level though, the biggest challenge is that the edge is resource-constrained. For example, the available resources in the wireless channel are limited and battery power is at a premium on end devices. Moreover, some of these applications may be time-critical.

In addition to designing applications, standards, and proper marketing, researchers need to focus on coming up with solutions that would facilitate MEC and EI while taking into account these widely heterogeneous capabilities and low resource budgets. Therefore, in this dissertation, the focus is developing task allocation schemes that will optimize the implementation of technologies for edge analytics.

1.3 Dissertation Organization

Using statistical data and recent R&D trends for edge computing and DL, Chapter 1 has shown the value and importance of further exploring these areas. The rest of this dissertation is organized as follows:

- Chapter 2 **Literature Review** presents the work related to MEC/H-MEC in general with a particular focus on the transition to DL over the wireless edge. We present some of the work published, identify the research gaps, and highlight the contributions

of this dissertation towards fulfilling these gaps.

- Chapter 3 **Hierarchical Mobile Edge Computing (H-MEC)** completely covers the work on H-MEC including the system model, problem formulation, proposed solution and the results.
- Chapter 4 **Mobile Edge Learning (MEL) System Model** presents the overall complete system model with the associated parameters for the proposed novel “MEL” paradigm. We begin by introducing the basic concepts of machine learning (ML); specifically gradient-based ML. Then, the chapter transitions to the generic DL model and extends that to the MEL paradigm. Finally, the system parameters of MEL are described. The chapter also discusses how these variables relate directly to the physical resources of time and energy. In other words, this chapter fully describes the proposed heterogeneity aware (HA) model.
- Chapter 5 **MEL Task Allocation with Time Constraints** solves the first two sub-problems where solutions are proposed for MEL with only time constraints for both settings, the HA synchronous (HA-Sync) and asynchronous (HA-Asyn) task allocation approaches. The results are evaluated for both settings in terms of multiple metrics. Because an analytical solution is derived for the HA-Sync case, a complexity analysis is also performed to support the proposed solutions.
- Chapter 6 **MEL with Dual Time and Energy Constraints** provides solutions for HA task allocation for both HA-Sync and HA-Asyn, with joint dual completion time and energy consumption constraints. Based on the results, we first prove that the HA approaches are better than the heterogeneity unaware (HU) approaches. Finally, we provide recommendations on how to select the appropriate scheme with the correct choice of parameters.
- Chapter 7 **Synchronous MEL Extension** presents the work done on HA-Sync with only time constraints with optimal task allocation instead of maximal task allocation. Because our models did not directly relate the ML or DL convergence properties to

the resource consumption linked variables in that work, this part of the dissertation explored the performance gains of HA-Sync versus HU-Sync with only time constraints when the HA model loss function convergence is linked directly to the HA model parameters.

- Chapter 8 **Recommendations, Conclusions, and Future Work** summarizes the work by providing recommendation on how to select the best scheme for MEL, giving brief conclusions on the work done and setting some of the future research directions in the area of H-MEC in general and MEL in particular.

Chapter 2: Literature Review

The booming need for edge processing and analytics is supported by recent advancements in mobile edge computing (MEC) [3, 15–17] and hierarchical MEC (H-MEC) [18–20]. While MEC enables edge nodes to either compute their tasks locally or offload them to the edge servers, H-MEC gives an end device the additional option to offload to a peer. These offloading decisions are usually made such that metrics such as completion time, energy efficiency, and bandwidth usage are optimized.

For MEC, the two key performance metrics are the delay and the energy consumed to fulfill a particular task. Previously, researchers have proposed techniques to minimize task latency [21] or maximize energy efficiency [15]. Some of the most important works focus on minimizing the task latency and energy consumption in a joint manner [4, 22]. However, for hierarchical edge computing, the idea of optimizing delay and energy efficiency jointly has not been investigated before.

Partial offloading and cooperative communication is investigated in [18] where it is assumed only two end devices, one user and the other acting as a helper, jointly achieve task offloading and computation for the user. The transmission/computation time is divided into four slots. In the first period, the local user transmits to the local helper. In the second slot, both the user and helper nodes send the partial task to the access point (AP). In this scenario, there is only cooperative communication. The AP computes the task and transmits the results back to the user in the 3rd and 4th slots, respectively. The problem is constrained by task completion time (no energy constraints) and solved using the Lyapunov optimization techniques.

The model in [19] consists of a set of edges with devices and access points connected to the cloud. An external controller stores task fingerprints and makes the offloading decisions. There are four possibilities: execute locally, offload to the cloud, offload to peers who have recently done a similar task (sharing), or ask for resources (co-operative) to share the load. The edges (AP's) are just only used for communication and hence, this differs model only

offers edge communication, no computing. The network-wide average energy consumption is optimized using Lyapunov optimization.

The authors of [20] consider a set of sensor nodes (edges) connected to a cloud; but they focus on only the edge layer, i.e. cooperative computing between sensors. The edge servers cooperatively share computational resources to minimize the total energy consumption of the system by accepting partial tasks from users. An exhaustive search and a heuristic algorithm is proposed to optimize the partial offloading parameters.

The concept of exploiting D2D communication as an alternative to MEC is explored in [17, 23, 24]. The authors of [17] propose optimal and periodic access schemes for users to identify and utilize resources on peer devices. The work of [23] focuses on using heterogeneous networks to enhance capacity and minimize offloading probability. Furthermore, they show that exploiting content similarity among user tasks can reduce offloading probability.

So far, there is very little work in the literature which talks about H-MEC where users have the choice to offload to their peers or the edge server. To the best of our knowledge, no such work exists which talks about dual-objective joint resource optimization in such multi-level MEC. Therefore, our first goal is to do joint task latency and energy consumption minimization for 2-level MEC, where users can offload tasks to their peers or the edge.

While most of the previous works on MEC/H-MEC were limited to independent computing tasks, one of the most important forthcoming applications of MEC/H-MEC is implementing distributed learning (DL) on edge nodes. Using DL, multiple edge nodes can collectively analyze and learn from their local possessed data with no or limited dependence on cloud/edge servers. As the topic of H-MEC was explored further, it was discovered that researchers are devoting most of their attention to machine learning over H-MEC. Some works have focused on optimizing the process of task allocation using ML techniques [25]. Other researchers have worked on caching results of ML based applications at the edge server so that other users running similar apps may have quick access to results [26]. Lately, the focus has shifted to deploying ML models for training in a distributed manner over multiple

end devices on the edge [27].

Although distinct from each other, optimal task allocation for H-MEC or for DL over the wireless edge are two sides of the same coin. In the former, end devices compete with each other to offload computationally intensive tasks to the edge server or neighboring nodes based on the heterogeneous communication channels and the computation capacities of each node. In the latter, one orchestrator facilitates the learning process at edge nodes (or learners as we will refer to devices in the context of DL over the wireless edge). This is done while simultaneously trying to maximize ML accuracy and satisfying the resource constraints such as delay and energy consumption limits. Therefore, one can see how optimal task allocation for DL at the wireless edge is an important extension of the H-MEC paradigm.

Distributed learning (DL) in general, has attracted much attention in the ML research community because of the complexities of the ML models. Until recently, most works on DL were studied over wired and/or non-heterogeneous distributed computing and data transfer environments such as high performance computing (HPC) systems and graphical processing units (GPUs) [28,29]. However, applying DL in such environments is very different from the wireless edge which requires dealing with the eccentricities of wireless communication and the heterogeneous capacities of end devices in terms of processing power.

Extending these DL studies to resource-constrained edge servers and nodes was not explored until very recently [30,31]. The work was started by a research group at Google [27] where they proposed the general concept of federated learning for on-device intelligence at the edge [32]. The focus of these early works was deploying FL at the edge rather while evaluating the communication and latency issues. The heterogeneous capabilities of learners or resource consumption issues were not tackled specifically in light of the physical layer parameters.

From then on, researchers have focused on developing DL algorithms for the mobile edge computing paradigm without fully considering resource allocation issues and usually for specific ML techniques [33,34]. The work of [33] focuses on the cloud rather than the edge

and optimizes learning by automatically selecting a particular model via ensemble learning and auto-tuning of parameters. The authors of [34] tackle the SVM technique by reducing the communication overhead in a generic manner which may make their algorithm attractive for implementation on the edge. However, a general method for all ML algorithms was not explored. A few works have focused on the optimization algorithms themselves. For example, the authors of [35] investigate the benefits and drawbacks of performing local ML training at the learners as opposed to a global training. On the other hand, the work of [36] proposes an optimal global optimization method based on annealing for DL at the edge.

A significant amount of research has been done on the implementation of federated learning for specific applications such as vehicular networks [37–41], multi-robot systems [42], human activity recognition [43], keyword spotting [44], and general mobile object recognition using deep learning [45]. Edge DL has also been applied to smart infrastructure including smart cities [46], smart homes [47], smart health care systems for e-health [48, 49] and for smart cyber-physical systems such as AI-based control of multiple inverted pendulums [50]. Because privacy and security are of concern in edge DL, blockchain-based networks have been proposed to support edge DL [51, 52]. In fact, hybrid blockchain-based edge DL solutions have been proposed for a few applications such as connected vehicular networks [53–55] and smart cyber-physical systems [56, 56].

Because devices that participate in edge DL are expected to be resource-constrained, some research has also been carried out on ways to incentivize FL for learners at the edge [57–62], either by associating a cost for initiating a learning process or a prize to motivate end users for participating in a round of FL. As one can observe, work has been carried out to facilitate edge DL by proposing distributed models for specific ML algorithms, designing distributed optimization methods, and by implementing edge DL for specific applications such as for autonomous connected vehicles. However, very little work has been done on how the physical constraints in MEC's and H-MEC's directly impact the edge DL process.

Limited works have proposed solutions to tackle some of the physical layer challenges.

For example, one area researchers have worked on is analog aggregation or over-the-air computation, where the objective is to speed up the aggregation of ML models resulting from multiple learners by utilizing the analog communication over antennas [63–68]. The work of [69] tackles the issue when perfect channel state information (CSI) is not available at the edge server whereas the works of [70, 71] propose scheduling policies to streamline data exchange between learners and servers in wireless media with limited resources.

While not relating physical layer parameters directly to the resources consumed during the DL process, the works of [72–75] have proposed communication-efficient approaches. The work of [72] designed a feature fusion approach to reduce the size of the data exchanged, compression strategies for ML model parameters were designed in [73], using dual streamed models to reduce communication rounds was proposed in [75], and the authors of [74] designed a mechanism to exclude learners that were not performing well which also resulted in less communication rounds.

A couple of conclusions can be made after examining the most recent work on DL over the wireless edge. First of all, FL has been extensively studied in literature [76–83] whereas PL or MEL which comprises both, has been sparsely studied. Secondly, most works have proposed distributed algorithms for the MEL paradigm without focusing on the resource allocation issues [33, 34], and the researchers that consider resource consumption only do so in generic terms [76–79] without considering the heterogeneities, i.e. they are heterogeneity unaware (HU). Although the works of [80–82] are heterogeneity aware (HA), they ignore the aspect of batch allocation and the PL scenario.

The works of [76–79] have aimed to jointly optimize the number of local learning and global update cycles in resource-constrained edge environments, to maximize learning accuracy. However, these approaches are HU; they do not consider the inherent heterogeneities in the computing and communication capacities of different edge learners and links, respectively. The implications of such heterogeneities on optimizing the task allocation to different learners, selecting learning models, improving learning accuracy, minimizing local

and global cycle times, and/or minimizing energy consumption, are clearly game-changing, yet have rarely been investigated. Recently, the works of [81,82] have investigated the impact of energy and communication time, but only in the context of federated learning and only for assigning the number of local updates without any task size allocation.

To the best of our knowledge, this dissertation is the first attempt to synergize the novel trends of DL and H-MEC, in order to establish an optimization framework for efficiently executing distributed learning tasks on a neighboring set of heterogeneous wireless edge nodes or learners while taking into account these heterogeneities in order to ensure that resource consumption limits such as completion time and energy usage, are not violated.

2.1 Contributions

In the area of H-MEC, most works that proposed optimal task offloading and resource allocation while satisfying dual time and energy constraints did so for MEC. For H-MEC, either the designed systems treated the peers as relays or if any optimization was done, it was either only for delay minimization or energy efficiency. The existing research gap, as highlighted in Figure 2.1, is to allocate tasks and resources such that delay and energy consumption are jointly minimized.

As briefly described before, ML or DL algorithms rely on iterative approaches where

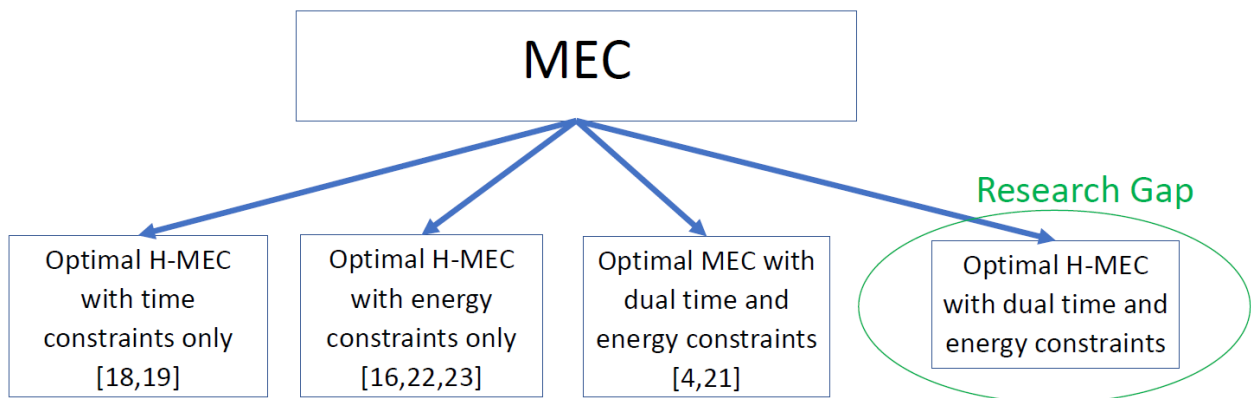


Figure 2.1: Review chart and research gap for H-MEC

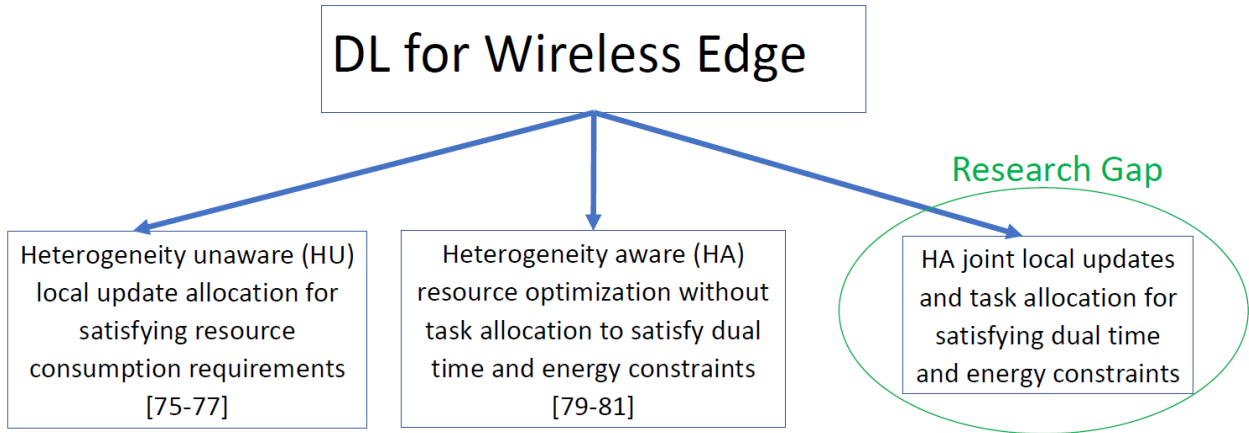


Figure 2.2: Review chart and research gap for MEL

individual learners will have to make several updates. The current literature tackled the problem of either only optimizing the number of local updates for HU resource consumption or optimizing resource allocation without task allocation, specifically batch-size allocation, to optimize the performance of the ML model. Therefore, a research gap exists in the area of HA optimal task allocation, both, the number of local updates and task size for maximizing the accuracy while respecting the time and energy constraints. Our proposed MEL design fills this gap as illustrated in Figure 2.2.

To this end, this dissertation has the following contributions:

1. In contrast to the previous works, this dissertation proposes to jointly minimize delay and energy consumption via optimal task offloading and resource allocation for H-MEC systems.
2. The design of a novel new paradigm for DL over the wireless edge called mobile edge learning (MEL), which incorporates both edge DL scenarios, parallelized learning (PL) and federated learning (FL).
3. Ensures that the designed system is HA, in contrast to most previous works which were HU, such that one or both of completion time and energy consumption constraints are satisfied as they relate to the physical channel parameters and computational capabilities of learners.

4. Proposes solutions for the MEL system with HA synchronous (HA-Sync) task allocation and asynchronous (HA-Asyn) task allocation. Further, this dissertation examines regions where one scheme may outperform the other in terms of the available resources, i.e. time and energy.
5. Illustrates the performance gains of the proposed HA solutions as compared to the HU through extensive simulations using more than one model and dataset under different conditions in terms of ML model accuracy. For some schemes, we also study the complexity of the optimization algorithms to study the trade-off between complexity and performance gains.
6. Provides recommendations for selecting the best scheme out of the HA-Sync and HA-Asyn.
7. Additional work is also done on optimizing task allocation in an HA manner while relating these parameters directly to the performance of the ML algorithm. This was done for the synchronous case with only time constraints to compare its performance to the previous works.

Chapter 3: Hierarchical Mobile Edge Computing (H-MEC)

This chapter starts off the first contribution of this dissertation by presenting our foray into the world of mobile edge computing (MEC). In this work, we propose a new framework for MEC called Hierarchical-MEC (H-MEC). Edge devices could offload their task either to the server or to another edge node instead. The assumptions are that the any edge node is either available to serve its peers or had its own tasks that need to be computed. Furthermore, one device is allowed to offload to one server only; either the edge server or a neighbor node. Also, one edge node is allowed to serve one other node only whereas the edge server is allowed to accept tasks from multiple end devices. In the following sections, we introduce the mobile edge computing (MEC) concept, then transition to the proposed H-MEC system model, presenting the problem followed by our proposed solution, and then discuss the achieved results. This chapter comprises material from a paper that has already been published¹.

3.1 Introduction to MEC

Consider a set $\mathcal{K} = \{1, 2, \dots, k, \dots, K\}$ of end user devices connected via an edge server O such as a router or a base station. At any given instant, an end device $k \in \mathcal{K}$ has a task that needs to be computed. In CC, user k would either attempt to do local execution or transmit its task to the edge which would forward it to the cloud for processing. This approach would be slow because the central server may be serving a queue of tasks and may also be located very far away. In contrast, the MEC paradigm allows the user to either compute the task locally or offload to the edge instead. This concept is illustrated in Figure 3.1.

¹Part of the material in this chapter was published as “Multi-Objective Resource Optimization for Hierarchical Mobile Edge Computing” in proceedings of the 2018 IEEE Global Communications Conference: Mobile and Wireless Networks (Globecom2018 MWN) [84].

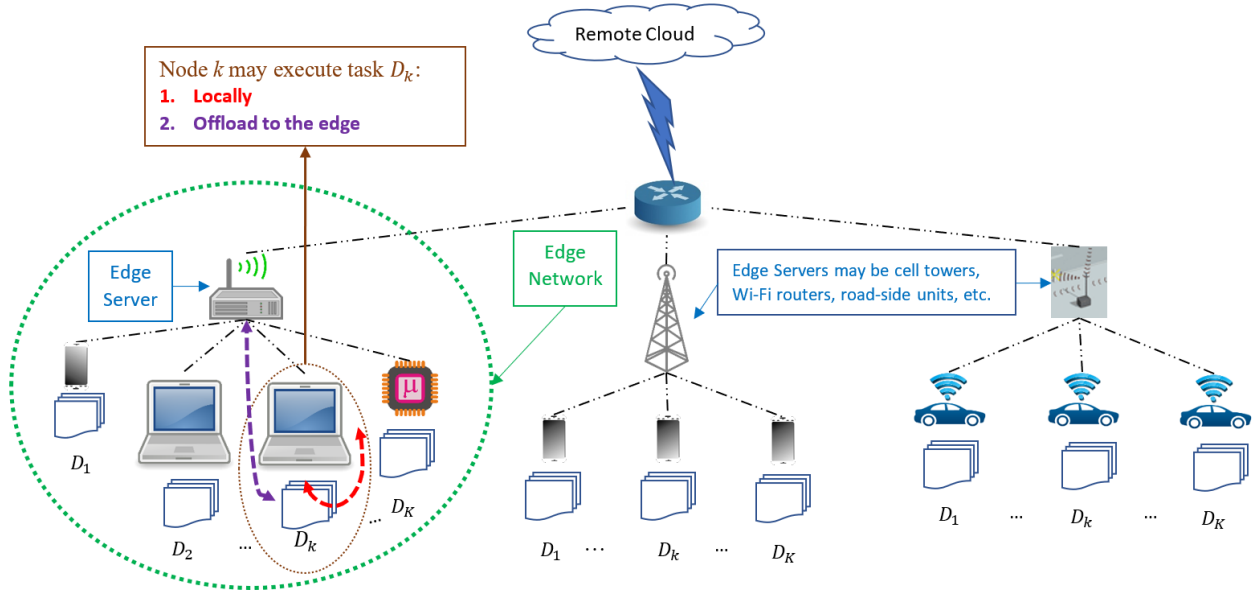


Figure 3.1: Illustration of multiple edge networks with a zoomed in view of one network to demonstrate the concept of MEC.

3.2 Transition to H-MEC

Consider a system that consists of K user devices connected via an edge server O such as a router or base station. At any given instant, an end device $k \in \mathcal{K}$ where $\mathcal{K} = \{1, 2, \dots, k, \dots, K\}$, has a task that needs to be computed. Now, assume that the total number of devices connected to the edge server in the MEC are given by K' , where the set of these devices is denoted by $\mathcal{K}' = \{1, \dots, k', \dots, K'\}$. The remaining $M = K' - K$ nodes are idle and each such node $m \in \{1, \dots, M\}$ may advertise its services for the set of neighboring nodes \mathcal{K} . In such a system, each node $k \in \mathcal{K}$ can take one of the following actions:

1. Perform the task locally.
2. Offload the task to the edge.
3. Offload the task to a neighbor $m \in \mathcal{M}$.

The additional option of being able to offload the task to another user makes this an H-MEC and distinguishes it from the classic MEC paradigm. Figure 3.2 illustrates the H-MEC

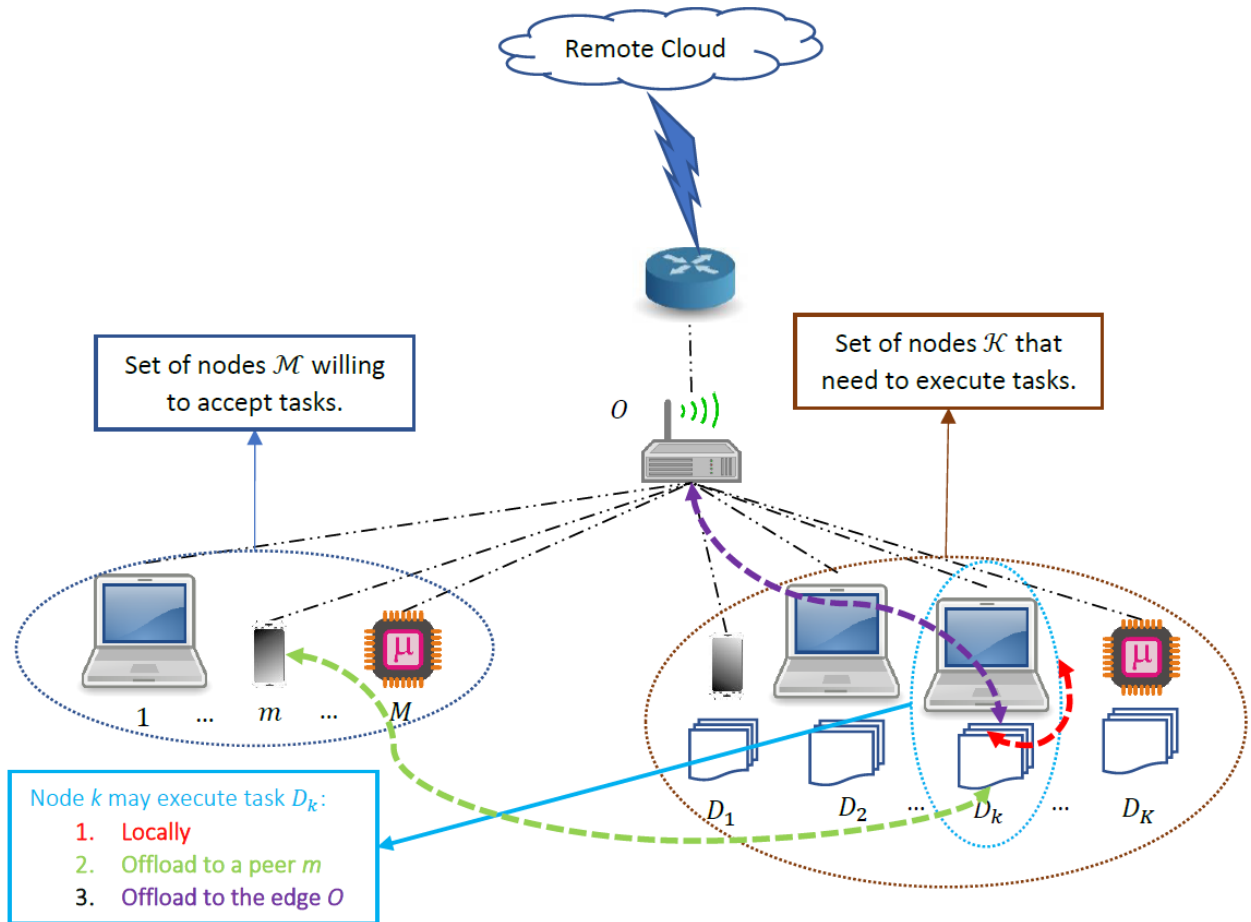


Figure 3.2: Logical grouping of end devices in an H-MEC. Note that the devices are grouped together for classification/presentation purposes. Practically, the servers and offloaders will be co-located such that they are randomly placed within the radius served by the edge server.

concept. We can now define the complete set of servers as $\mathcal{M} = \{1, \dots, m, \dots, M, M+1\}$ where servers $1, \dots, M$ represent the peers accepting tasks and server $M+1$ represents the edge server O .

The task associated with user $k \in \mathcal{K}$ can be described by its task size in bits D_k and the CPU clock cycles X_k required to complete this task. Assume that the edge server has knowledge of the channel (via channel estimation) and that before each computation, the nodes share their task-related information along with their computational/communication capabilities. Whether the task is executed locally or offloaded either to the edge or a peer, the task size and the associated clock cycles required will directly influence the time taken and energy consumed.

Let T_k^L represent the execution time needed to compute the task locally on the local processor. Then T_k^L can be given by:

$$T_k^L = \frac{X_k}{f_k^L}, \quad k \in \mathcal{K} \quad (3.1)$$

The processor's capability is given by f_k^L in Hz. The local energy E_k^L dissipated during task execution is given by:

$$E_k^L = \mu X_k f_k^{L\zeta-1}, \quad k \in \mathcal{K} \quad (3.2)$$

The on-board CPU capacitance is given by μ and $\zeta \geq 2$. [3].

The time taken to complete a task in case it is offloaded, T_k^O , comprises the following: the time taken to transmit the task from user $k \in \mathcal{K}$ to either, the edge O or user $m \in \mathcal{M}$ denoted by T_{km}^{TO} , task execution time at the server T_{km}^{EO} , and the time needed to transfer the results back T_{km}^{RO} . However, due to the results' size being significantly less than the task size in bits, we ignore the last quantity. Therefore, the total time T_{km}^O is given by:

$$T_{km}^O = T_{km}^{TO} + T_{km}^{EO} \quad (3.3)$$

Assuming that the channels are perfectly orthogonal and do not suffer from interference, the transmission time to a server is given by:

$$T_{km}^{TO} = \frac{D_k}{R_{km}}, \quad (3.4)$$

where R_{km} is the data rate between user k and the server as given below:

$$R_{km} = W \log_2(1 + a_{km}P_{km}) \quad (3.5)$$

The transmission power from the user to an adjacent device or the edge server is P_{km} . The coefficient a_{km} is $\frac{H_{km}}{N_0}$ where H_{km} represents the channel gain from user k to server m for

$k \in \mathcal{K}, m \in \mathcal{M}$. The available system bandwidth for one user k is denoted by W . The time taken to complete the task by a server is given by:

$$T_{km}^{EO} = \frac{X_k}{f_{km}}, \quad (3.6)$$

where f_{km} is the processing power reserved by the serving unit for user $k \in \mathcal{K}$. The energy consumption will cover task transmission from user $k \in \mathcal{K}$ to $m \in \mathcal{M}$. However, in case the task is offloaded to the edge, execution energy is ignored. It is only taken into account if the task is offloaded to a peer, i.e. for $m \in \{1, \dots, M\}$. The energy is related to T_{km}^{TO} and the power P_{km} available to user k for communication as given in (3.7).

$$E_{km}^O = \begin{cases} T_{km}^{TO} P_{km} + \mu X_k f_{km}^{\zeta-1}, & \text{if } m \in \{1, \dots, M\} \\ T_{km}^{TO} P_{km}, & \text{if } m = M + 1 \end{cases} \quad (3.7)$$

3.3 Formulation with Joint Delay and Energy Efficiency

Each end device $k \in \mathcal{K}$ can compute the task locally with time T_k^L and an energy consumption E_k^L . It may also offload its task to a neighbor node $m \in \{1, \dots, M\}$ or the edge server O denoted by server $M + 1$. The probability D_{km} of user $k \in \mathcal{K}$ to offload to any server $m \in \mathcal{M}$ is 1 if the user offloads and 0 for local execution as shown below:

$$D_{km} = \begin{cases} 1 & \text{if user } k \in \mathcal{K} \text{ offloads to server } m \in \mathcal{M} \\ 0 & \text{if user } k \in \mathcal{K} \text{ executes the task locally} \end{cases} \quad (3.8)$$

So, the decision probability D_{km} will form an offloading matrix as follows:

$$\mathbf{D} = \begin{bmatrix} D_{11} & D_{12} & \dots & D_{1m} & \dots & D_{1,M+1} \\ D_{21} & D_{22} & \dots & D_{2m} & \dots & D_{2,M+1} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ D_{k1} & D_{k2} & \dots & D_{km} & \dots & D_{k,M+1} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ D_{K1} & D_{K2} & \dots & D_{Km} & \dots & D_{K,M+1} \end{bmatrix} \quad (3.9)$$

The matrix \mathbf{D} is $K \times M + 1$, where the rows represent the users and the columns represent servers. The first M columns are the neighboring devices that have available resources and the last column represents the edge server O . For example, for a system with $K = 2$ and $M = 1$ implying two servers, one of them a peer and the other being the edge O , if $D_{11} = 0$, $D_{12} = 1$, $D_{21} = 1$, and $D_{22} = 0$, user # 1 offloads to the peer and user # 2 offloads to the edge.

The completion times and energy consumed for offloading are given by T_{km}^O and E_{km}^O , respectively, whereas the T_k^L and E_k^L denote the local execution time and energy, respectively. Given the description of the decision offloading probability in (3.8), the total time to finish the task for user $k \in \mathcal{K}$ is given in (3.10) and the total energy consumption in (3.11).

$$T_k = \sum_{m=1}^{M+1} d_{km}^O \times T_{km}^O + (1 - d_{km}^O) \times T_k^L \quad (3.10)$$

$$E_k = \sum_{m=1}^{M+1} d_{km}^O \times E_{km}^O + (1 - d_{km}^O) \times E_k^L \quad (3.11)$$

We propose that in our H-MEC model, each user k can be allowed to offload the complete task to one server only and that the peer devices acting as servers may serve one user only. The edge server can compute tasks for several users as limited by the total system bandwidth. The objective is to jointly allocate resources such that we minimize the two physical quantities of time and energy. Because, both quantities have different units, a

useful metric is needed; one that will represent the time and energy saved due to offloading as compared to local execution. To this end, the utility for each node $k \in \mathcal{K}$ is defined as follows:

$$U_k(\mathbf{D}_k, \mathbf{F}_k, \mathbf{P}_k) = \sum_{m=1}^{M+1} (D_{km}^O) \times \left[\eta_k^T \frac{T_k^L - T_{km}^O}{T_k^L} + \eta_k^E \frac{E_k^L - E_{km}^O}{E_k^L} \right] \quad (3.12)$$

The $M + 1$ row vectors $F_k, P_k \forall k \in \mathcal{K}$ represent the computational resource in Hz allotted to any user k by server $m \in \mathcal{M}$ and the transmission power resource available to user k for send data to $m \in \mathcal{M}$, respectively. The parameters η_k^T and $\eta_k^E \forall k \in \mathcal{K}$ are scaling factors that each represent the preference of user k for either saving time or energy, respectively. They can be set such that $0 < \eta_k^T < 1$, $0 < \eta_k^E < 1$, and $\eta_k^E + \eta_k^T = 1$. We can re-write the overall system utility as follows:

$$U(\mathbf{D}, \mathbf{F}, \mathbf{P}) = \sum_{k=1}^K U_k(\mathbf{D}_k, \mathbf{F}_k, \mathbf{P}_k) \quad (3.13)$$

The matrices F and P represent the computational resource in Hz allotted to user $k \in \mathcal{K}$ by server $m \in \mathcal{M}$ and the transmission power resource available to user $k \in \mathcal{K}$ to send data to $m \in \mathcal{M}$, respectively. Therefore, each of them is also $K \times (M + 1)$ similar to \mathbf{D} .

Given that a user can offload to one server only, the joint latency and energy minimization

problem can be formulated as:

$$\begin{aligned}
& \arg \max_{\mathcal{O}, \mathbf{D}, \mathbf{F}, \mathbf{P}} U(\mathbf{D}, \mathbf{F}, \mathbf{P}) & (3.14) \\
\text{s.t. } & C1 : D_{km} \in \{0, 1\} \\
& C2 : \|D_k\|_1 \leq 1 \\
& C3 : \|D_m^T\|_1 \leq 1, \quad m = 1, \dots, M \\
& C4 : 0 < \|D_{M+1}^T\|_1 \leq N \\
& C5 : \mathcal{O} = \{k | D_{km} = 1\}, \quad \text{for } m = 1, \dots, M + 1 \\
& C6 : 0 < P_{km} \leq P_{k,max} \\
& C7 : f_{km} > 0, \forall k \in \mathcal{O} \\
& C8 : \sum_{k \in \mathcal{O}} f_{k,M+1} = f_{M+1}^{max}
\end{aligned}$$

Although, the utility is a function of the computational resource \mathbf{F} , the communication resource \mathbf{P} , and the offloading decisions \mathbf{D} , the optimization is done over these three variables and an additional variable \mathcal{O} which represents the set of users that are offloading, whether to the edge or a peer. Adding the set of offloading users facilitates the analysis and the solution later.

The constraint C1 simply ensures that the elements of the offloading matrix are either 0 for local execution or 1 for offloading. C2 and C3 ensure that a user k offloads to one server only and that a peer acting as a server only accepts task from one end device only. The transpose operation is represented by the operator \mathbf{A}^T , where \mathbf{A} is a two-dimensional (2D) matrix, and the subscript i in A_i represents the i^{th} row of the 2D matrix \mathbf{A} . Similarly, A_i^T implies the i^{th} column of the matrix \mathbf{A} or the i^{th} row of the transposed of \mathbf{A} . C4 ensures that for edge offloading, the number of users are restricted by the communication resources. C5 defines the set of offloading users \mathcal{O} in terms of the offloading matrix. C6 sets the power constraint such that any node $k \in \mathcal{K}$ does not exceed its maximum transmission power given

by $P_{k,max} \forall k \in \mathcal{K}$. C7 ensures that any server $m \in \mathcal{M}$ assigns a non-zero computational resource whereas C8 guarantees that the edge (server $M + 1$), does not exceed its maximum computational capacity denoted by f_{M+1}^{max} .

3.4 Proposed Solution

The problem can be decomposed into maximizing the communication and computational resources over an offloading set described by C5 and eliminating it as shown in (3.15). The constraints C1–C4 are not related to the resource allocation, only to the offloading decisions.

$$\begin{aligned} & \max_{\mathcal{O}, \mathbf{D}} \max_{\mathbf{F}, \mathbf{P}} U(\mathbf{D}, \mathbf{F}, \mathbf{P}) \\ \text{s.t. } & C1 - C4 \text{ and } C6 - C8 \end{aligned} \quad (3.15)$$

So, the joint computation and communication resource allocation problem can be modeled as a maximization problem over a set of offloading decisions as demonstrated in (3.16).

$$\begin{aligned} & \max_{\mathcal{O}} U(\mathcal{O}) \\ \text{s.t. } & C6 - C8 \end{aligned} \quad (3.16)$$

Thus, the resource allocation problem can be re-written as:

$$\begin{aligned} U(\mathcal{O}) &= \sum_{k \in \mathcal{O}} \max_{\mathbf{F}, \mathbf{P}} U_k(\mathbf{D}_k, \mathbf{F}_k, \mathbf{P}_k) \\ &= \max_{\mathbf{F}, \mathbf{P}} \sum_{k \in \mathcal{O}} \sum_{m=1}^{M+1} \left[\eta_k^T \frac{T_k^L - T_{km}^O}{T_k^L} + \eta_k^E \frac{E_k^L - E_{km}^O}{E_k^L} \right] \\ & \text{s.t. } C6 - C8 \end{aligned} \quad (3.17)$$

We can form this problem as a minimization problem as described by **P1**.

$$\mathbf{P1} : \min_{\mathbf{F}, \mathbf{P}} \sum_{k \in \mathcal{O}} \sum_{m=1}^{M+1} \left[\eta_k^T \frac{T_{km}^O}{T_k^L} + \eta_k^E \frac{E_{km}^O}{E_k^L} \right] \quad (3.18)$$

s.t: $C6 - C8$

We can substitute the expressions for the local and remote execution times and energy consumption in (3.1), (3.6), (3.2), and (3.7), respectively, in **P1** and write the problem as shown in **P2**.

$$\mathbf{P2} : \min_{\mathbf{F}, \mathbf{P}} \sum_{k \in \mathcal{O}} \sum_{m=1}^{M+1} \left[\frac{C_k^1 + C_k^2 P_{km}}{\log_2(1 + a_{km} P_{km})} + \eta_k^T \frac{f_k^L}{f_{km}} \right] \quad (3.19)$$

s.t: $C6 - C8$

The two constants can be described by $C_k^1 = \frac{\eta_k^T D_k}{W T_k^L}$ and $C_k^2 = \frac{\eta_k^E D_k}{W E_k^L}$. Interestingly, the two constants are still only dependent on user k . Furthermore, the minimization problem consists of two terms. The term on the left-hand side is only constrained by $C6$, the communication or transmission power constraint. On the other hand, the term on the right hand side is constrained by the computation resources as described by $C7 - C8$.

Therefore, similar to [2], the problem can be separated into optimal power transmission allocation and computational power assignment calculation from each server to the user offloading to that particular server. The optimal transmission power can be calculated by solving the problem in **P3**.

$$\mathbf{P3} : \min_{\mathbf{P}} \sum_{k \in \mathcal{O}} \sum_{m=1}^{M+1} \left[\frac{C_k^1 + C_k^2 P_{km}}{\log_2(1 + a_{km} P_{km})} \right] \quad (3.20)$$

s.t: $C6$

We can exchange the summations in **P3** and re-write the problem as shown in **P4**. The reason for this is that in our current models, we assume that the channels are independent of each other and there is no interference. The cases of interfering channels and contention

schemes will be studied later. It can be noted that the optimal power transmission from each user to each potential server (whether the edge or a peer) can be calculated based on channel parameters regardless of the offloading decision.

$$\mathbf{P4} : \min_{\mathbf{P}} \sum_{m=1}^{M+1} \sum_{k \in \mathcal{O}} \frac{C_k^1 + C_k^2 P_{km}}{\log_2(1 + a_{km} P_{km})} \quad (3.21)$$

s.t: $C6$

The optimization program in (3.21) is shown to be quasi-convex in [2] and can be solved efficiently using the bisection method. This process is repeated for each channel and thus the, optimal transmission power matrix \mathbf{P}^* can be obtained.

The computational resource assignment problem can be modeled by **P5**.

$$\mathbf{P5} : \min_{\mathbf{F}} \sum_{k \in \mathcal{O}} \sum_{m=1}^{M+1} \eta_k^T \frac{f_k^L}{f_{km}} \quad (3.22)$$

s.t: $C7 - C8$

Notice that constraint $C7$ only affects the cases where the user is offloading to a peer device whereas constraint $C8$ only considers the users that are offloading to the edge. This problem can be separated into two parts as shown in **P6**. Furthermore, for our system we consider the scenario where a 'free' user will advertise its services to the edge along with the corresponding CPU clock cycles available.

$$\mathbf{P6} : \min_{\mathbf{F}} \sum_{k \in \mathcal{O}} \sum_{m=1}^M \eta_k^T \frac{f_k^L}{f_{km}} + \sum_{k \in \mathcal{O}} \eta_k^T \frac{f_k^L}{f_{k,M+1}} \quad (3.23)$$

s.t: $C7 - C8$

The problem on the left hand side may consider parameters for partial offloading which will be discussed later. The problem on the right hand side can be solved using the Lagrangian dual method and the optimal frequency assigned to user $k \in \mathcal{O}$ by the edge can be calculated

using [2]:

$$f_{k,M+1}^* = \frac{\sqrt{\eta_k^T f_k^L}}{\sum_{k \in \mathcal{O}} \sqrt{\eta_k^T f_k^L}} \quad (3.24)$$

Once we know how to get the optimal transmission powers and the optimal computational resource allocation by each server to its clients, we can search over the different offloading decision set possibilities. In a system where users only offload to one edge server, there are 2^K possible offloading decisions. A brute force solution becomes impractical for large number of users. In a system where there are K users and M peer devices acting as servers in addition to the edge, the complexity can be given by $\sum_{m=0}^M \binom{M}{m} \binom{K}{m} 2^{K-m}$.

3.4.1 Peer Offloading Priority Based Heuristic Algorithm

Due to the high complexity of the brute force approach, it is necessary to design our own algorithm that can reduce the complexity and provide nearly optimal performance. Before doing so, let us appreciate that although D2D communication is assumed, the system is still centralized. In other words, the edge server will decide on the final offloading decisions, and broadcast them to all devices who will offload their tasks accordingly. Consequently, the goal of our heuristic algorithm is to minimize the tasks offloaded to the edge server.

Let us define some new terms before discussing our algorithm. The expression $U(\mathbf{S})$ represents the utility of a given offloading set of users. We can define $\Delta_k U(\mathbf{S})$ as the increase in utility due to the adding of user k in the set \mathbf{S} . This term comprises a part that is constant and another part $\Delta_m(k)$ that changes for each user if it offloads to server m as shown below:

$$\Delta_m(k) = \frac{C_k^1 + C_k^2 P_{km}}{\log_2(1 + a_{km} P_{km})} + \eta_k^T \frac{f_k^L}{f_{km}} \quad (3.25)$$

The task will be done locally if the $\eta_k^T + \eta_k^E - \sum_{m=1}^{M+1} \Delta_m(k) < 0$. For all the users that do not satisfy this condition, they will send an offloading request to the server. At the same time, the user devices advertising their services will send their available resources to the server. The server will sort the peers according to the maximum available resources.

After that, the server will select the device such that $U_{km} \geq 0$ and $k = \arg \max \Delta_m(k)$ for $m = 1, \dots, M$.

At each time a decision is made, the corresponding user k is excluded from the search set and device m will only serve 1 device. This condition will satisfy constraints C2 and C3 of the original problem. The last check is to make sure that the utility of user k is not higher while offloading to the edge server. If that is the case, we will move onto the next device.

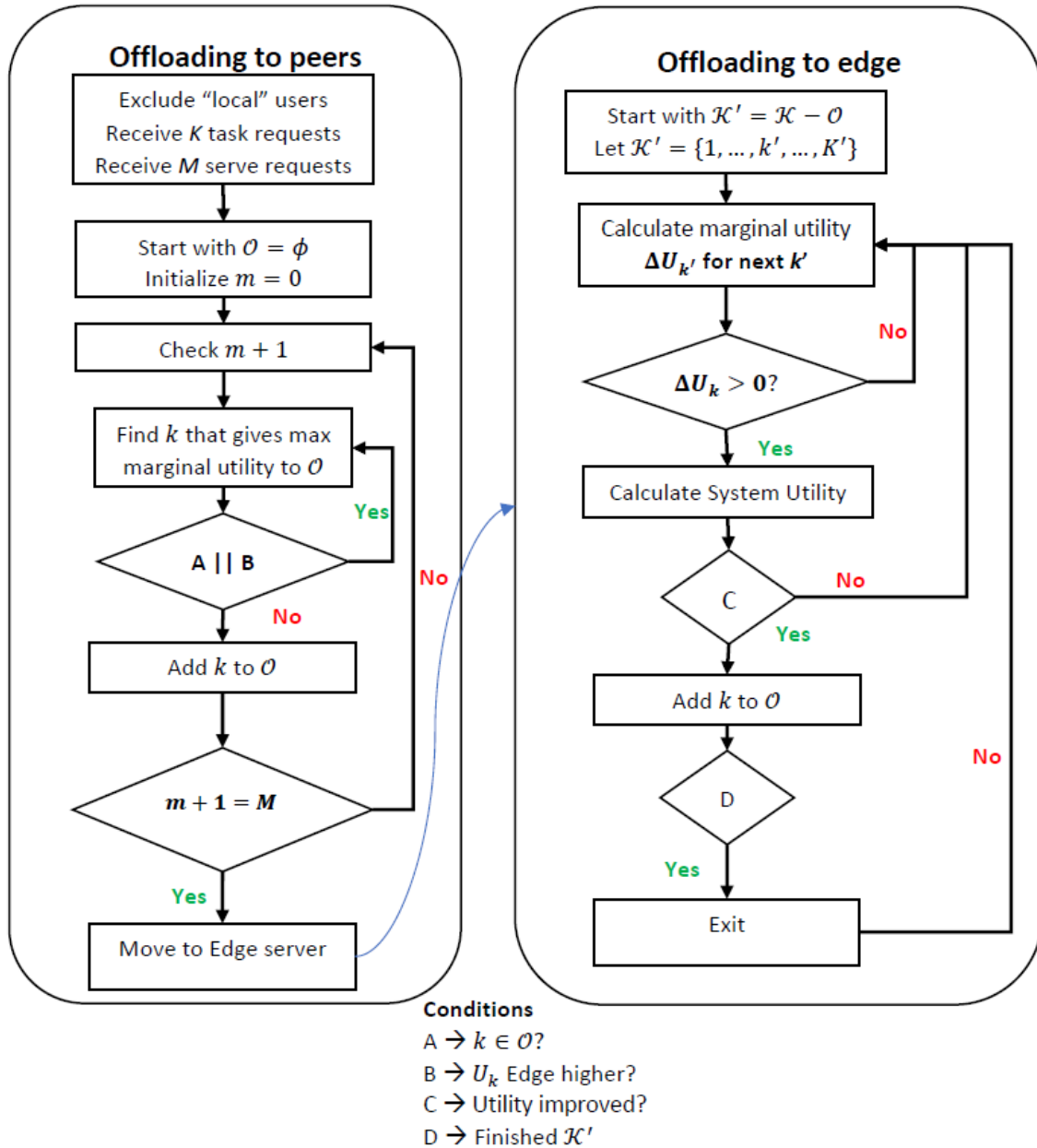


Figure 3.3: Illustration of the proposed peer offloading priority based heuristic algorithm.

Once the offloading decisions have been determined by offloading to the peers according to the maximum utility, the remaining set of users can be offloaded to the edge based on the HODA algorithm of [2]. The complete algorithm is given illustrated in Figure 3.3.

The complexity of the HODA algorithm itself is $O(K^3)$. The complexity of making offloading decisions to peers depends requires $O(K)$ operations for each peer. In total, in the worst case $M \times O(K)$ operations are required in case no user offloads to an adjacent device. In the scenario with $K \gg M$, the complexity is nearly $O(K)$. On the other hand, when M is comparable to K , the complexity will be $O(K^2)$. The last step of ensuring that U_{km} is greater than the utility of offloading to the edge has a negligible cost and in the worst case scenario, the complexity will be $O(K^3)$. Overall, the expected complexity of the system is in the range of $O(K^5) \sim O(K^6)$.

3.5 Results and Discussion

The comparisons are made for the case of offloading to the edge compared to our model where users can offload to the neighboring devices. Therefore, similar parameters are used to the system tested in [2]. The edge base station is assumed to serve users located within a 500 m radius with a path-loss model following the lognormal shadowing. A total bandwidth of 20 MHz is available with each offloading user assigned a 1 MHz. The users are all assumed to be located in a 20 m radius. The application studied is facial recognition with a task size of 420 kB requiring 1 billion cycles. Local devices are assumed to have processor speeds of between 1-2 GHz and the server's processor speed is 20 GHz.

Figure 3.4 shows that as the number of users increase, offloading to a peer improves system utility. To elaborate, the blue crossed curve shows that for the case of users only offloading to the edge in an optimal way, the utility will saturate faster. In contrast, the performance of our system continually improves as users compete for a peer with free resources. The results show that configuration is beneficial where there are a large number of users competing for resources. In modern day networks with large number of devices, this configuration has the

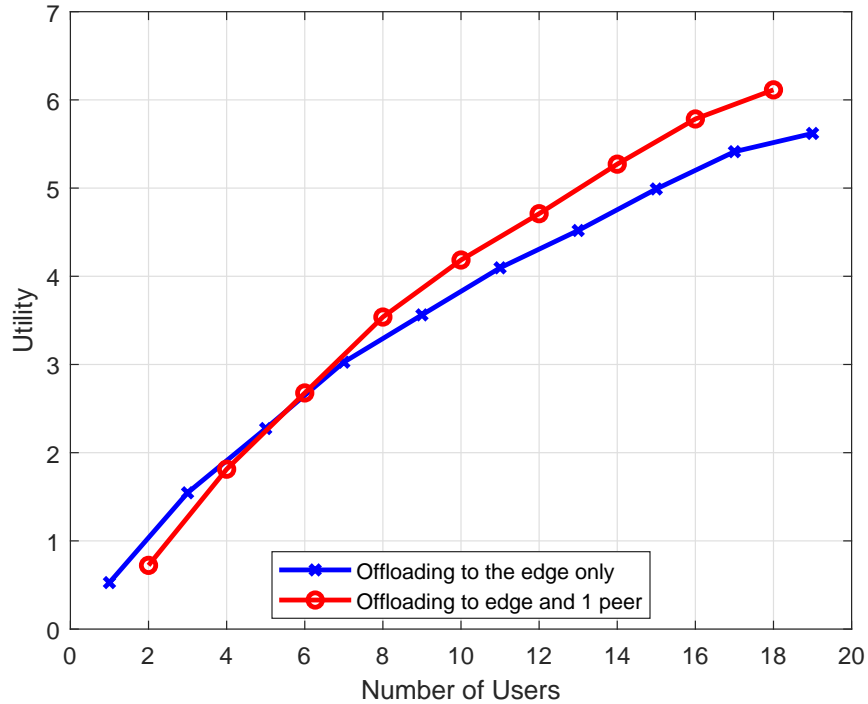


Figure 3.4: Plot of system utility versus number of users

potential to significantly lighten the load on edge servers.

Figure 3.5 shows the average number of offloaded users as we increase the total number of devices in the scenario where one peer is accepting tasks. The blue crossed-line shows the case where devices only offload to the edge and the performance of our model is illustrated by the red-dotted line. As one can observe, offloading tasks to neighboring devices reduces the need to offload to the edge. However, as the number of users increase, more devices will offload tasks to the edge and hence, the performance will approach the MEC case. The performance in terms of average offloading of users can be improved in systems where more than one user device acts as a server.

The best performance in terms of both the highest utility and the least average offloading users can be achieved in MEC systems serving a large number of devices within a limit. For example, in the case of one user accepting tasks, the best jointly optimal performance can be achieved by an MEC cloud consisting of between 7 – 15 users. Testing using the heuristic

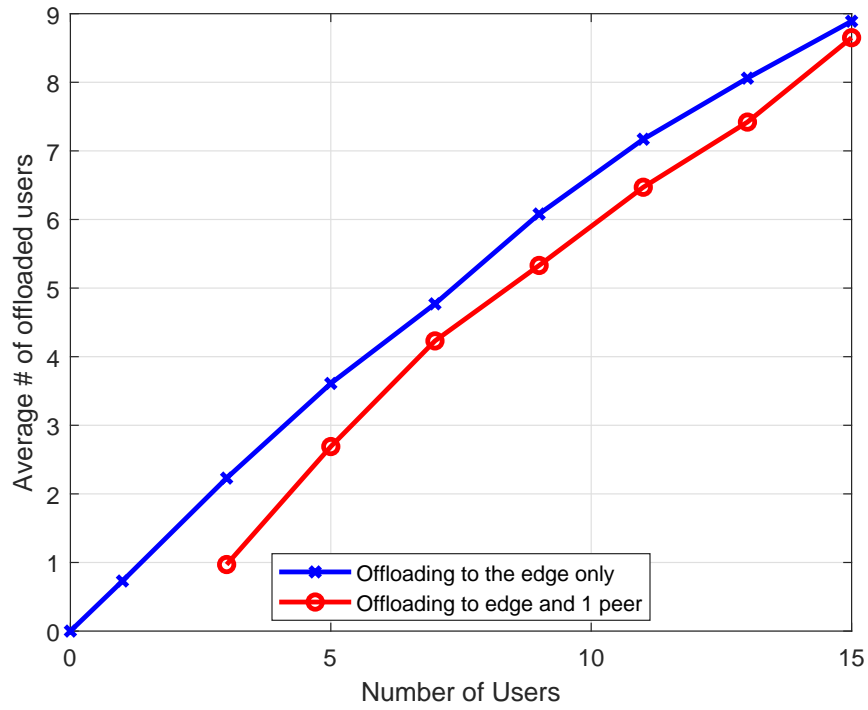


Figure 3.5: Plot number of users offloading to the edge versus number of users

algorithm with larger configurations (e.g. more users or offloading to more than one peer) may show a similar trend but with even lesser users offloading to the edge.

Chapter 4: Mobile Edge Learning (MEL) System Model

In this section, we present the model for the proposed paradigm of “MEL”. We first briefly introduce the concept of ML and extend the discussion to DL. Finally, we elaborate on how DL is applied over multiple learners connected via the wireless edge.

4.1 Machine Learning

Machine learning (ML) algorithms are designed to solve a problem automatically from available data. There are several types of problems that ML can solve. Broadly, these can be grouped into two categories: supervised and unsupervised learning. In supervised learning, the data is labeled, i.e. the output or target is known apriori and the goal of the algorithm is to predict the output with minimum error using the given data as input. In unsupervised learning, the labels are not known and the aim is to cluster or group similar data together with minimum error based on the input data.

We will limit the discussion to supervised learning, though the work in this thesis also applies to unsupervised learning. In supervised learning, the ML process consists of two stages: training and testing. In contrast, unsupervised learning only comprises training which may involve fitting or clustering data into groups. In the training phase, for most methods, the ML model parameters are updated using an iterative procedure until a loss function, which is pre-defined, is minimized. The algorithm can see the output during the training phase and keeps adjusting the model parameters until the error reaches below a certain threshold.

The output of the ML algorithm may be a real number, a binary digit or a set of integers depending upon the application. In binary classification, the output is 0 or 1, 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 that represent the probability with which a data sample belongs 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.

In the testing phase, the learned parameters are simply applied to the test dataset in one go and the output of the model is compared to the actual targets in order to gauge accuracy. Often, there is an additional stage which is called validation. In validation, during the training process, part of the training dataset is kept hidden from the model in order to see how the algorithm will generalize to real data. Usually, the validation accuracy is a good indicator of the performance during training because it expresses how the ML model will perform when tested on previously unseen data.

Typically, an ML algorithm will be trained on a dataset \mathcal{D} comprising a total of d samples. Each data sample has a set of features that serve as the inputs to the ML algorithm. Therefore, each data sample \mathcal{D}_n for $n = 1, \dots, d$ has \mathcal{F} features that can be denoted by x_j where $j = 1, \dots, \mathcal{F}$. The set of features belonging to data sample number n can be denoted by $\mathbf{x}_n = \{x_1, \dots, x_j, \dots, x_{\mathcal{F}}\}$.

In ML, the objective is to find the relationship between \mathbf{x}_n and y_n using a set of parameters \mathbf{w} such that a loss function, $F(\mathbf{x}_n, \mathbf{y}_n, \mathbf{w})$, or $F_n(\mathbf{w})$ for short because \mathbf{x}_n and y_n are known, is minimized. The total loss over the complete dataset can be given by:

$$F(\mathbf{w}) = \frac{1}{d} \sum_{n=1}^d F_n(\mathbf{w}) \quad (4.1)$$

Because it is generally difficult to find an analytical solution, typically an iterative gradient descent approach is used to optimize the set of model parameters such that the model parameter set at any discrete time-step l , for $l = 1, \dots, L$, is related to the model at the previous time-step and the gradient of the loss in the following way:

$$\mathbf{w}[l] = \mathbf{w}[l-1] - \eta \nabla F(\mathbf{w}[l-1]) \quad (4.2)$$

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 go over each data sample one-by-one until it passes over the whole dataset; a complete pass is known as one epoch. This approach is known as the deterministic gradient descent (DGD). In recent times, more computationally efficient approaches have been proposed such as the mini-batch gradient descent where the algorithm goes over batches of data by matrix operations. One of the best performing approaches in terms of computational efficiency and error performance is the stochastic gradient descent (SGD) with mini-batch training [85] where, before the start of each new epoch, the dataset is re-shuffled randomly.

4.2 Distributed Learning

Many ML techniques, including regression, support vector machine (SVM) and neural networks (NN) are built on iterative gradient-based learning. Because such iterative approaches can be exhaustive for a single device, distributed learning (DL) has been proposed to train ML algorithms over multiple learners. There are two possible scenarios: training an ML model on a large dataset in a distributed manner where subsets of the data are located across multiple learners (data parallelism), or training very large models distributedly on one dataset co-located at each learner (model parallelism). Although both options apply to the wireless edge, most of the discussion focuses on the DP scenario. However, the proposed “MEL” paradigm can support MP and this will be mentioned where appropriate.

Consider the case where there exists one centralized controller or orchestrator that trains an ML model to solve a specific problem (classification, prediction, image segmentation, etc.) on a set of $\mathcal{K} = \{1, \dots, k, \dots, K\}$ learners. In DL with DP, a batch of the data \mathcal{D}_k of size d_k is present at each individual learner k which may be locally owned or supplied by the orchestrator. The orchestrator initiates the learning process by sending a global model \mathbf{w} and possibly d_k samples to each learner k . Each learner k applies the gradient descent approach to the local model \mathbf{w}_k as shown in (4.3) multiple (τ_k) times in parallel, and sends back the local models to the orchestrator for global aggregation. One such cycle can be

called the global update cycle.

$$\mathbf{w}_k[l] = \mathbf{w}_k[l-1] - \eta \nabla F_k(\mathbf{w}_k[l-1]) \quad (4.3)$$

The local model parameter set at learner k is given by \mathbf{w}_k , the local loss is given by F_k , and η is the learning rate. At time-step l , the local model $\mathbf{w}_k[l]$ depends on the model $\mathbf{w}_k[l]$ at previous step l and the gradient of the local loss $\nabla F_k(\mathbf{w})$. The local loss $F_k \forall k \in \mathcal{K}$ can be calculated using the local dataset \mathcal{D}_k of size d_k in the following way [76]:

$$F_k(\mathbf{w}_k) = \frac{1}{d_k} \sum_{n=1}^{d_k} F_n(\mathbf{w}_k) \quad (4.4)$$

The global optimal model parameter set \mathbf{w} will only be visible to the learners after a global aggregation which may occur at any arbitrary time-step l for $l = 1, \dots, L$. For that particular time-step, $\mathbf{w}_k = \mathbf{w} \forall k \in \mathcal{K}$. In the synchronous case, at all learners k , a global aggregation occurs after τ time-steps; whereas in the asynchronous case, the global aggregations will occur after potentially different τ_k updates for each learner $k \forall k \in \mathcal{K}$. For both scenarios, the globally optimal model parameter set can be obtained by applying the following aggregation mechanism [76]:

$$\mathbf{w} = \frac{1}{d} \sum_{k=1}^K d_k \mathbf{w}_k \quad (4.5)$$

The orchestrator may perform multiple global cycles until a stopping criteria is reached such as when the model reaches a pre-set accuracy threshold or the resources such as multi-core processors are no longer available. This process is summarized in Figure 4.1.

4.3 Transition to MEL

In this section, we introduce the system model for MEL by transitioning the aforementioned DL setting to the heterogeneous edge nodes setting. This will be performed by

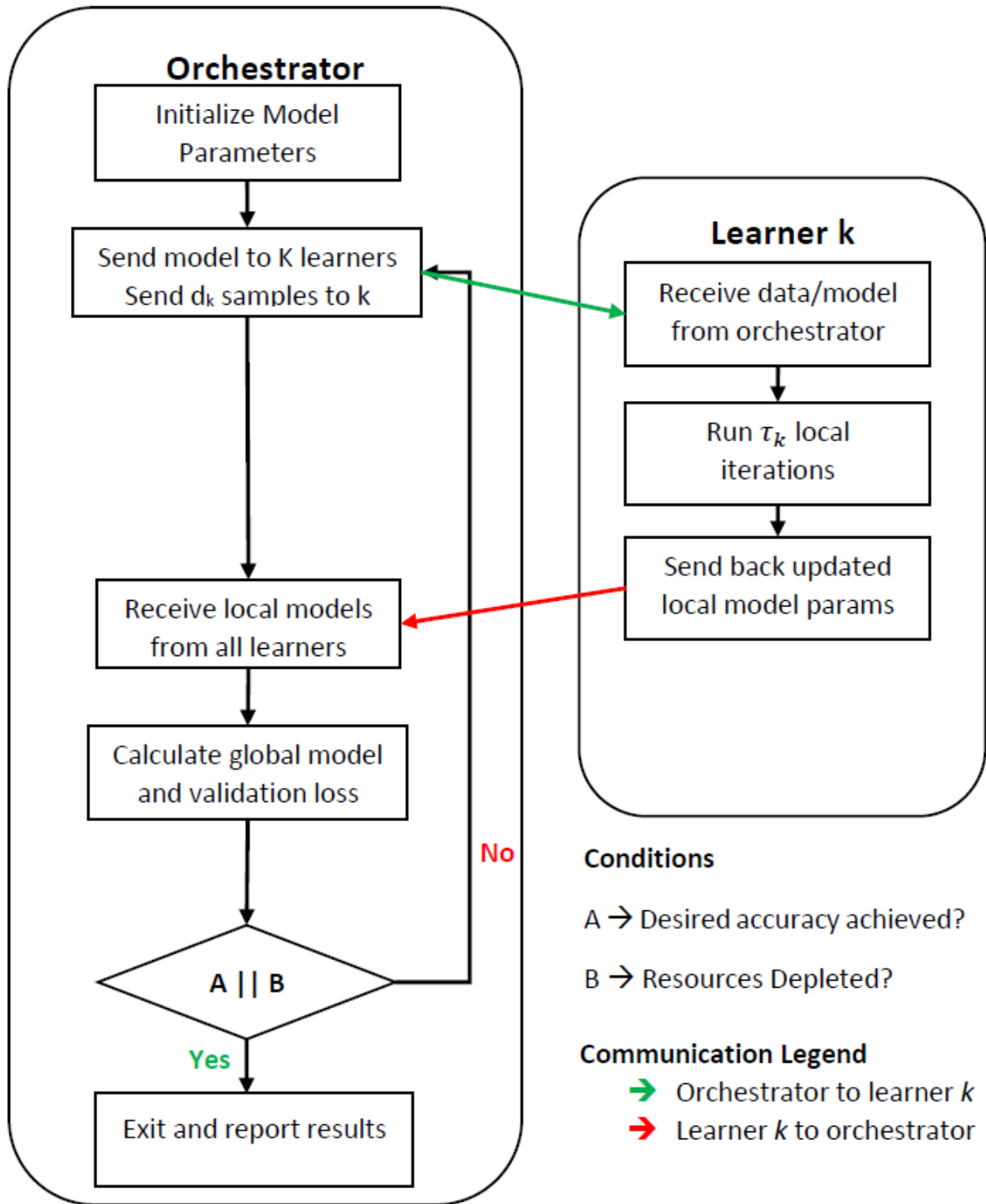


Figure 4.1: Illustration of the DL process with DP

defining the parameters that relate 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 have been defined in Chapter 1: namely Federated Learning (FL) which is also referred to as distributed datasets (DD), and parallelized learning (PL) which may also be referred to as task-parallelization (TP).

In FL, data is generated and collected by multiple nodes, but cannot be transferred to a central hub for analytics due to some constraints (e.g., bandwidth, privacy) [86]. In this case, the learning process cycles between these nodes performing local training/learning on their individual datasets, and a central orchestrator collecting the locally derived parameters, performing global processing, and returning globally updated parameters to the learners. On the other hand, PL scenario usually involves a main node, which maybe the edge server or one of the end devices. This edge server/end device parallelizes the learning process over its local dataset on multiple cores/nodes due to one or more reasons (e.g., limited main node resources, faster processing, lower energy consumption) [28]. Thus, the orchestrator must distribute subsets of the dataset to other multiple learners for local learning followed by global aggregations until a stopping criteria is reached. Figure 4.2 illustrates the differences between both approaches.

Performing learning in mobile, edge, and IoT environments is the clear manifestation of both of the above scenarios. Indeed, edge nodes generate/collect data that they cannot share to edge/cloud servers due to bandwidth limitations. Many of them are also computationally-limited, and can thus use H-MEC to parallelize the learning process over multiple neighboring

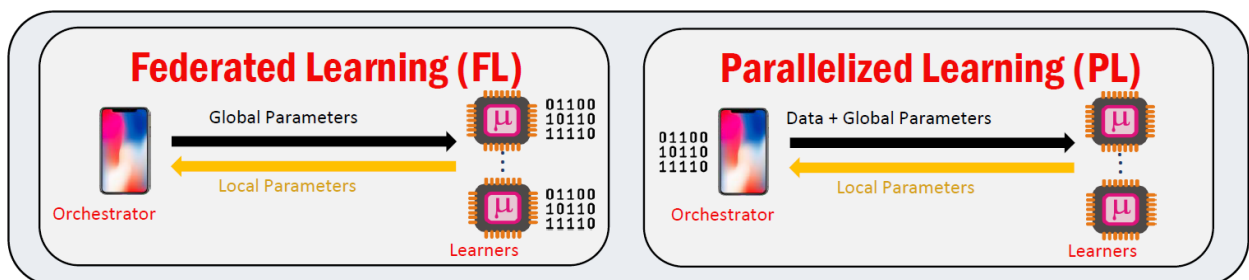


Figure 4.2: Comparison of DD/FL and TP/PL paradigms.

nodes for faster processing and/or lower energy consumption. Clearly, PL fully encompasses the FL, but only adds to it the batch transfer component from the orchestrator to the learners. We will therefore mainly consider the latter scenario in our discussions but will show the variations in the model when the former scenario is considered and how the parameters may change. In fact, the PL scenario separates the work in this dissertation from other works that mainly focus on FL.

Consider an MEL system comprising the set of learners $\mathcal{K} = \{1, \dots, k, \dots, K\}$ where learner k trains its local learning model on a batch size of d_k data samples. The goal is to minimize the local loss function [77]. The total size of all batches is denoted by $d = \sum_{k=1}^K d_k$, which is usually preset by the orchestrator O given its computational capabilities, the desired accuracy, and the time/energy constraints of the training/learning process. The number of local iterations or local updates run by learners on their allocated batch is denoted by τ_k . For the synchronous case, $\tau_k = \tau \forall k \in \mathcal{K}$; whereas in asynchronous task allocation, each learner can perform a different number of τ_k local iterations. Figure 4.3 illustrates the considered MEL setting.

Traditionally, most works explore the synchronous approach, though both methods have their merits. In this dissertation, we will explore both options. In addition to the two approaches, two key aspects of the MEL process are the expected task completion time and the energy consumption per learner k . In this dissertation, we will consider both time and energy constraints in our model. There are many variables that can impact the time and energy consumption in MEL and some of these may also impact the accuracy.

For example, the number of local updates will directly impact the execution time and the dataset size will impact both the execution time and the transmission time. If MP is applied, dataset size may impact the completion time in both FL and PL, whereas with DP it will only impact in PL. A smaller batch-size may allow for more local updates which may improve accuracy because typically, in SGD, the loss decreases as the number of iterations are increased. However, if the dataset size is too small, that may also adversely affect the ac-

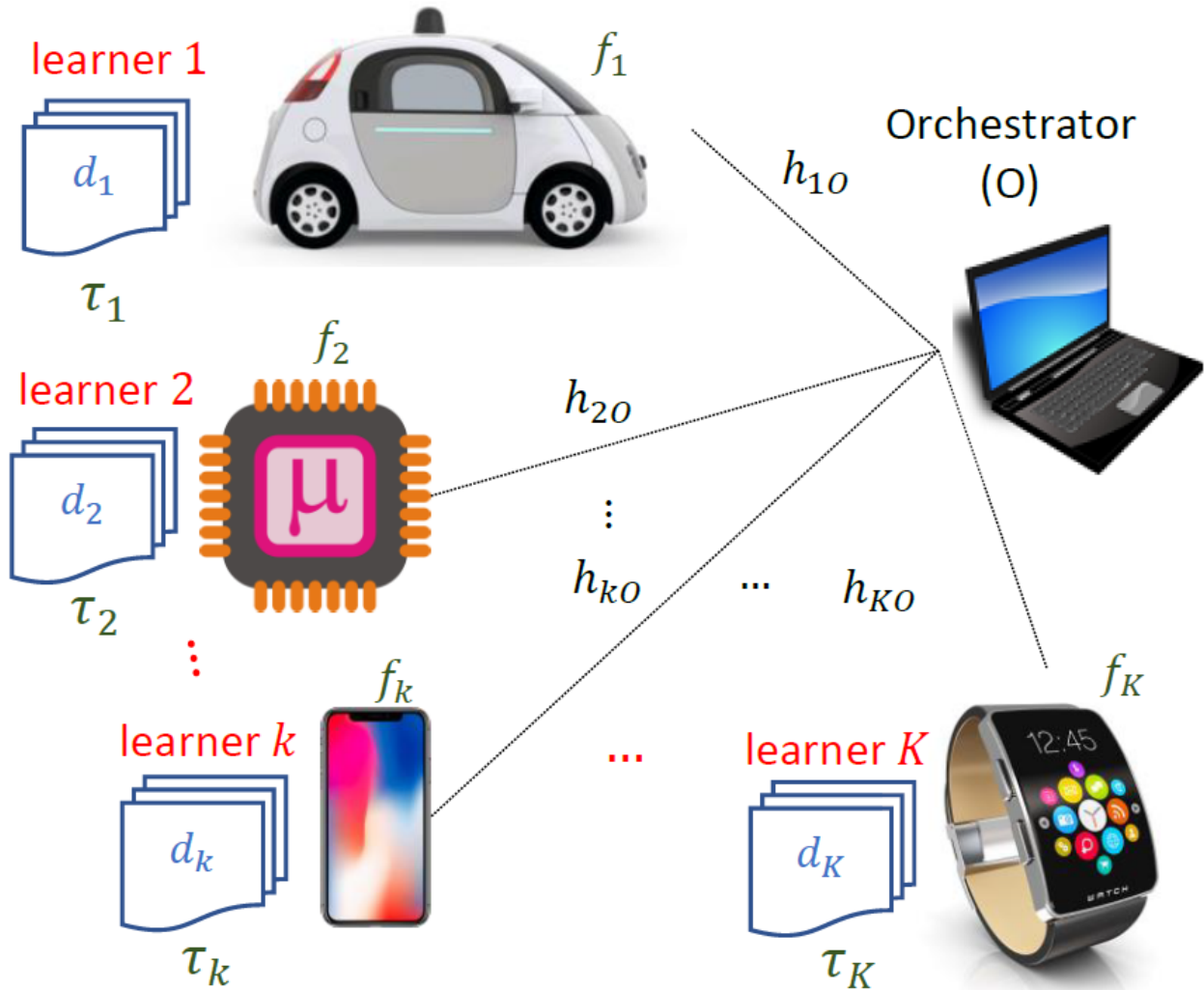


Figure 4.3: System model of a MEL setting

curacy. Other variables that may impact the local completion time and energy consumption include the transmission power, local computational power and the complexity of the ML model. Whereas these components may not be of significant influence when DL is executed over controlled wired and infrastructural servers, their high heterogeneity can tremendously impact the performance of DL when applied in wireless and mobile edge environments. This is where the MEL paradigm comes into play.

In the following paragraphs, we will relate these parameters for user k to both, its local time and energy consumption for one global cycle. The orchestrator performs the aggregation of the parameters only once after all learners send back their result within that particular

global update cycle after doing $\tau_k \forall k$ local updates. 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 phases:

1. transmission of the global parameter matrix w to each learners $k \in \mathcal{K}$
2. computation of τ_k local update cycles at each learner k
3. return of the local parameter matrices $\mathbf{w}_k \forall k \in \mathcal{K}$ from each learner to the orchestrator
4. global aggregation at the orchestrator as defined in (4.5)

The orchestrator will typically demand the results within pre-set duration within which all of these four steps should be completed. In previous works, it has been assumed that only devices that are charging or fully charged will take part in H-MEC in general. However, the devices or learners in MEL, may not be fully charged or on direct power and hence, they may have a limit on the amount of battery power they are willing to drain. To this end, in the following two sub-sections, we define the time taken and the energy consumed by one learner $k \in \mathcal{K}$, respectively, to complete the MEL process.

4.3.1 Local Completion Time

Let us define B_k^{data} as the number of bits of the batch allocated to learner k , which can be expressed as follows:

$$B_k^{data} = d_k \mathcal{F} \mathcal{P}_d \quad (4.6)$$

Recall that \mathcal{F} is the feature vector size and \mathcal{P}_d defines the storage strategy of the data which may simply be the precision in bits or an advanced quantization/compression scheme. The size in bits of learner k 's local parameter matrix $w_k \forall k$ is denoted by B_k^{model} and can be expressed as:

$$B_k^{model} = \mathcal{P}_m (d_k S_d + S_m) \quad (4.7)$$

where \mathcal{P}_m is the model bit precision or compression ratio. As shown in the above equation, the local parameter matrix size consists of two parts, one depending on the batch size (represented by the term $d_k S_d$, where S_d is the number of model coefficients related to each sample of the batch), and the other related to the constant size of the employed ML model (denoted by S_m). The first represents the support for MP in our scheme. Please note that the aggregation mechanism described in (4.5) cannot be employed with MP.

At the start of each global cycle, the orchestrator sends the optimal global model parameter matrix of size B_k^{model} and in the case of PL, also a batch of size B_k^{data} in bits, to each learner k in parallel. Assuming that enough orthogonal channels are available without interference, the orchestrator will send this information with power P_{k_o} over a wireless channel of bandwidth W having a gain equal to h_{k_o} and a noise spectral density N_0 . The orchestrator sends the bit concatenation of the data batch and initial global parameter matrix to learner k with power P_{k_o} over a wireless channel, having a bandwidth W and a channel power gain h_{k_o} . Given the above description, the time taken for the first step denoted by t_k^S can be expressed as¹:

$$t_k^S = \frac{d_k \mathcal{F} \mathcal{P}_d + \mathcal{P}_m (d_k \mathcal{S}_d + \mathcal{S}_m)}{W \log_2 \left(1 + \frac{P_{k_o} h_{k_o}}{N_0} \right)} \quad (4.8)$$

Once the global parameter matrix \mathbf{w} is received by each learner k , it sets its local parameter matrix $\mathbf{w}_k \forall k$ to the initial matrix \mathbf{w} provided by the orchestrator. For PL, it may have received d_k data samples whereas in FL, it will randomly select d_k data samples from its complete dataset. Each then performs τ_k local update steps using (4.3) on the local parameter matrix $\mathbf{w}_k \forall k$ using its allocated batch. In the local model, the iterative gradient procedure will be applied sequentially to each data sample once per local iteration. Consequently, the number of computations required per iteration X_k is equal to:

$$X_k = d_k C_m \quad (4.9)$$

¹For FL, the only difference in the model is that the first term of the numerator ($d_k \mathcal{F} \mathcal{P}_d$) will not exist.

which clearly depends on the number of data samples d_k assigned to each learner and the computational complexity C_m of the model. The second time consists of τ_k times the duration t_k^C needed by learner k to perform one local update cycle. Defining f_k as learner k 's local processor frequency dedicated to the DL task, t_k^C can be expressed as:

$$t_k^C = \frac{X_k}{f_k} = \frac{d_k C_m}{f_k} \quad (4.10)$$

For simplicity, we will assume that the bits of the computed local parameter matrix $\mathbf{w}_k \forall k \in \mathcal{K}$ are transmitted back to the orchestrator over a reciprocal channel with symmetric transmission power P_{ko} . Then, the third time t_k^R needed by learner k to send its updated local parameter matrix to the orchestrator can be described as:

$$t_k^R = \frac{\mathcal{P}_m(d_k \mathcal{S}_d + \mathcal{S}_m)}{W \log_2 \left(1 + \frac{P_{ko} h_{ko}}{N_0} \right)} \quad (4.11)$$

The orchestrator will then re-compute the global parameter matrix \mathbf{w} as described in (4.5). Once computed, it send this matrix back with a new random batch of samples from the dataset to each learner², and the process repeats. Because the global aggregation is a lot less computationally complex than the iterative GD, and also because the communication time will be significantly higher than simple weighted summing, the time for the last stage can be ignored.

Thus, the total time $t_k \forall k \in \mathcal{K}$ taken by learner k to complete the first three processes of MEL is equal to:

$$\begin{aligned} t_k &= t_k^S + \tau_k t_k^C + t_k^R \\ &= \frac{d_k \mathcal{F} \mathcal{P}_d + 2 \mathcal{P}_m(d_k \mathcal{S}_d + \mathcal{S}_m)}{W \log_2 \left(1 + \frac{P_{ko} h_{ko}}{N_0} \right)} + \tau_k \frac{d_k C_m}{f_k} \end{aligned} \quad (4.12)$$

²In the FL/DD setting, each learner selects a new random set of d_k samples for the new cycle.

4.3.2 Local Energy Consumption

Given the MEL model described, learner k consumes e_k^C (J) to perform one learning iteration over its allocated batch-size. It then consumes a further energy e_k^R (J) to send back the updated local model parameters. The energy consumed by the orchestrator for transmitting the global model to each learner or for aggregation can be ignored because it does not affect learner k . (This is expected to be true in practical scenarios, especially if the orchestrator is the edge server or if it is on charging mode.) Therefore, for each learner, the energy consumed in one global cycle can be given by:

$$e_k = \tau_k e_k^C + e_k^R \quad (4.13)$$

Given that learner k 's processing capability depends upon the processor speed denoted by f_k in GHz, the energy consumed by performing a learning iteration on a sample size of d_k is given by [84]:

$$e_k^C = \mu X_k f_k^{\nu-1} = \mu d_k C_m f_k^{\zeta-1}, \quad k \in \mathcal{K} \quad (4.14)$$

where μ is the on-board chip capacitance (typically $10^{-9} \sim 10^{-12}$ F) and $\nu = 2$ [84]. The energy consumed by learner k to transmit the latest local model parameters is given by:

$$e_k^R = \frac{P_{k0} B_k^{model}}{R_k} = \frac{\mathcal{P}_m (d_k \mathcal{S}_d + \mathcal{S}_m)}{W \log_2 \left(1 + \frac{P_{k0} h_{k\alpha}}{N_0} \right)}, \quad k \in \mathcal{K} \quad (4.15)$$

The total energy consumed by learner $k \in \mathcal{K}$ in one global update cycle can be given by:

$$\begin{aligned} e_k &= \tau_k e_k^C + e_k^R \\ &= \frac{P_{k0} \mathcal{P}_m (d_k \mathcal{S}_d + \mathcal{S}_m)}{W \log_2 \left(1 + \frac{P_{k0} h_{k\alpha}}{N_0} \right)} + \tau_k d_k \mu C_m f_k^{\zeta-1} \end{aligned} \quad (4.16)$$

4.4 MEL Model Summary

As one can see, time taken and energy consumed depend on several physical parameters including the number of local updates $\tau_k \forall k$, the local dataset size $d_k \forall k$, transmission power P_{ko} , model size S_m , and model computational complexity C_m . The model sizes and complexities will depend on the DL model usually pre-set by the orchestrator. On the other hand, $\tau_k, d_k, P_{ko} \forall k \in \mathcal{K}$ can be optimized or controlled for best use of the resources.

As discussed earlier, previous works have focused on optimizing τ_k 's [76] without studying the impact of d_k 's with respect to how it will impact the possible local updates given the limited and heterogeneous nature of the resources in a wireless environment. Other more recent works include investigating the impact of power allocation P_{ko} as well as the number of channels allocated [80–82].

However, the impact of batch allocation $d_k \forall k \in \mathcal{K}$ has never been studied, which means the PL scenario has been completely overlooked. To this end, we will study the joint impact of τ_k and d_k on resource consumption in the form of time and energy and try to optimize them such that this may enhance the performance of the ML model. It can be assumed that each learner $k \in \mathcal{K}$ has only one channel available and has a limit on the available transmission power. Hence, the optimization variables will be τ_k and $d_k \forall k \in \mathcal{K}$.

To make things clearer, we can re-write the expressions of $t_k \forall k \in \mathcal{K}$ in (4.12) as a function of τ_k and d_k as follows:

$$t_k = C_k^2 \tau_k d_k + C_k^1 d_k + C_k^0 \quad (4.17)$$

where C_k^2 , C_k^1 , and C_k^0 represent the quadratic, linear, and constant coefficients of learner k

in terms of τ_k and d_k , expressed as:

$$C_k^2 = \frac{C_m}{f_k} \quad (4.18)$$

$$C_k^1 = \frac{\mathcal{F}\mathcal{P}_d + 2\mathcal{P}_m\mathcal{S}_d}{W \log_2 \left(1 + \frac{P_{k0}h_{k0}}{N_0} \right)} \quad (4.19)$$

$$C_k^0 = \frac{2\mathcal{P}_m\mathcal{S}_m}{W \log_2 \left(1 + \frac{P_{k0}h_{k0}}{N_0} \right)} \quad (4.20)$$

We can also re-write the expression for $e_k \forall k \in \mathcal{K}$ as follows:

$$e_k = G_k^2 \tau_k d_k + G_k^1 d_k + G_k^0 \quad (4.21)$$

The quadratic, linear, and constant coefficients of learner $k \in \mathcal{K}$ denoted by G_k^2 , G_k^1 , and G_k^0 , respectively, can be expressed as follows:

$$G_k^2 = \mu C_m f_k^{\zeta-1} \quad (4.22)$$

$$G_k^1 = \frac{P_{k0}\mathcal{P}_m\mathcal{S}_d}{W \log_2 \left(1 + \frac{P_{k0}h_{k0}}{N_0} \right)} \quad (4.23)$$

$$G_k^0 = \frac{P_{k0}\mathcal{P}_m\mathcal{S}_m}{W \log_2 \left(1 + \frac{P_{k0}h_{k0}}{N_0} \right)} \quad (4.24)$$

It is clear that the expressions of the task completion time t_k and the energy consumed e_k are quadratic in terms of τ_k and d_k for all learners $k \in \mathcal{K}$. The significance of this will be discussed in later chapters. For the time-being, note that we have described the complete heterogeneity aware (HA) MEL system model including the time and energy consumption per learner per global cycle. This model can now be sub-divided into two cases. The first case comprises the scenario when there is a limit only on the completion time for every global cycle. The second case consists of the scenario when there is an additional constraint on the energy consumption per global cycle for each learner $k \in \mathcal{K}$. Furthermore, these two

scenarios can be sub-divided into two categories: the synchronous case (HA-Sync) where the number of local updates $\tau_k = \tau \forall k \in \mathcal{K}$, and the asynchronous case where the number of updates can be variable among any two learners. Therefore, there are four scenarios possible:

1. HA-Sync with time constraints only
2. HA-Asyn with time constraints only
3. HA-Sync with dual time and energy constraints
4. HA-Asyn with dual time and energy constraints

We will discuss each scenario as a sub-problem where the two sub-problems with only time-constraints are tackled in Chapter 5 and the two sub-problems with dual time and energy constraints are presented in Chapter 6.

Chapter 5: MEL with Time Constraints

This chapter presents the results achieved in the area of mobile edge learning “MEL” with synchronous and asynchronous task allocation when there is a global completion time constraint on the global update cycle. We begin by presenting our proposed heterogeneity aware (HA) synchronous (HA-Sync) task allocation scheme in Section 5.1 followed by the presentation of the HA asynchronous (HA-Asyn) scheme in Section 5.2.

5.1 Synchronous Task Allocation

This section discusses optimal task allocation for MEL when there is a time constraint on the amount of local ML iterations that can be performed within every global aggregation. In the discussed setting, the number of iterations are synchronized among all learners and hence, we call it synchronous task allocation. This work has already been published¹.

5.1.1 Problem Formulation

In Section 4.3.1, we introduced the MEL system model parameters and how they relate to the time consumed by each learner $k \in \mathcal{K}$ for one global cycle. Assuming that the orchestrator limits the collection of the model parameters in every global cycle to time T , then $t_k \leq T$ must hold $\forall k$ for the orchestrator to acquire all the needed information for performing its global cycle update process. Moreover, in the sub-problem in this section, we assume that the all learners perform a synchronized number of local updates such that $\tau_k = \tau \forall k \in \mathcal{K}$ and that the constraint is only on the global completion time. This may happen when the orchestrator has the luxury to select learners that are charging or being powered by batteries that are sufficiently charged.

¹This section is part of a paper titled “Adaptive Task Allocation for Mobile Edge Learning” which was presented at the 2019 2nd Workshop on Intelligent Computing and Caching at the Network Edge. This event was part of the 2019 IEEE Wireless Communications and Networking Conference (IEEE WCNC 2019). The paper was published in proceedings of the IEEE WCNCW 2019 and is available on the IEEE Xplore library [87]. An extended part of this work titled “Dynamic Task Allocation for Mobile Edge Learning” has also been submitted to the IEEE Transactions on Mobile Computing.

It is well established in the literature that the loss function in general GD/SGD-based ML are minimized (and thus the learning accuracy is maximized) by increasing the number of learning iterations [28]. For synchronous DL, this is equivalent to maximizing the number of local iterations τ in each global cycle [88]. Thus, maximizing the MEL accuracy is achieved by maximizing τ . Given the model in Section 4.3 and the above facts, the objective can be re-worded as optimizing the assigned batch sizes d_k to each of the learners so as to maximize the number of local iterations τ per global updated cycle, while bounding $t_k \forall k$ by the preset global cycle clock T . As mentioned in Section 4.4, the optimization variables are τ and d_k , and $t_k \forall k \in \mathcal{K}$ can be expressed as:

$$t_k = C_k^2 \tau d_k + C_k^1 d_k + C_k^0 \quad (5.1)$$

where C_k^2 , C_k^1 , and C_k^0 represent the quadratic, linear, and constant coefficients of learner k in terms of the optimization variables τ and d_k as defined in (4.18)-(4.20)².

Given the above expressions and facts, the problem of interest in this paper can be formulated as an integer linear program with quadratic and linear constraints as follows:³

$$\max_{\tau, d_k \forall k} \tau \quad (5.2)$$

$$\text{s.t.} \quad C_k^2 \tau d_k + C_k^1 d_k + C_k^0 \leq T, \quad k = 1, \dots, K \quad (5.2a)$$

$$\sum_{k=1}^K d_k = d \quad (5.2b)$$

$$\tau \in \mathcal{Z}_+ \quad (5.2c)$$

$$d_k \in \mathcal{Z}_+, \quad k = 1, \dots, K \quad (5.2d)$$

²Note that, for the FL scenario, the only difference in the model is that the first term of the numerator in (4.19) will not exist.

³Note that, for the FL scenario, the only difference in the formulation is the simpler expression of C_k^1 . Thus, the problem type and solution remain the same with different C_k^1 expressions for the two scenarios.

Constraint (5.2a) guarantees that $t_k \leq T \forall k$. Constraint (5.2b) ensures that the sum of batch sizes assigned to all learners is equal to the total dataset size that the orchestrator needs to analyze. Constraints (5.2c) and (5.2d) are simply non-negativity and integer constraints for the optimization variables. Note that the solutions of (5.2) having τ and/or all d_k 's being zero represent settings where MEL is not feasible.

Clearly the relationship between the optimization variables d_k and τ is quadratic in $t_k \forall k \in \mathcal{K}$ in (5.1). Furthermore, the optimization variables τ and $d_k \forall k$ are all non-negative integers. Thus, the above problem is a quadratically-constrained integer linear program (QCILP), which is well-known to be NP-hard [89]. We will thus propose a simpler solution to it through relaxation of the integer constraint in the next section.

5.1.2 Proposed Solution

As clarified in the previous section, the considered problem is NP-hard due to its integer decision variables. Therefore, we propose to simplify the problem by relaxing the integer constraints in (5.2c) and (5.2d), solving the relaxed problem, then rounding the obtained real results back into integers. The relaxed problem can therefore be given by:

$$\max_{\tau, d_k \forall k} \tau \quad (5.3)$$

$$\text{s.t.} \quad C_k^2 \tau d_k + C_k^1 d_k + C_k^0 \leq T, \quad k = 1, \dots, K \quad (5.3a)$$

$$\sum_{k=1}^K d_k = d \quad (5.3b)$$

$$\tau \geq 0 \quad (5.3c)$$

$$d_k \geq 0, \quad k = 1, \dots, K \quad (5.3d)$$

where the cases for τ and/or all d_k 's being zero represent scenarios where MEL will not be feasible, and hence, the orchestrator must send the learning tasks to the edge or cloud

server. The above resulting program becomes a linear program with quadratic constraints. This problem can be solved by using interior-point or alternating direction of multiplier (ADMM) methods, and there are efficient solvers (such as OPTI) that implement these approaches [90].

Although the associated matrices for each of the quadratic constraints in the relaxed problems can be written in a symmetric form, they will have two non-zero values that are positive and equal to each other. The eigenvalues will then sum to zero, which means these matrices are not positive semi-definite, and hence the relaxed problem is non-convex. Consequently, we cannot derive the optimal solution of this problem analytically. Yet, we can still derive upper bounds on the optimal variables and optimal solution using Lagrangian relaxation and the Karush-Kuhn-Tucker (KKT) conditions.

Thus, the philosophy of our proposed solution is to calculate these upper bounds values on the optimal variables, then implement suggest-and-improve steps until a feasible real solution is reached. The integer values for the optimization variables can be obtained by flooring these real solutions. Using the well-known KKT conditions, the following theorem introduces upper bounds on the optimal variables of the relaxed problem.

Theorem 1 *The optimal values d_k^* of the allocated batch sizes to different users in the relaxed problem satisfy the following bound:*

$$d_k^* \leq \frac{T - C_k^0}{\tau^* C_k^2 + C_k^1} \quad \forall k \in \mathcal{K} \quad (5.4)$$

Moreover, the analytical upper bound on the optimization variable τ belongs to the solution set of the polynomial given by:

$$d \prod_{k=1}^K (\tau^* + b_k) - \sum_{k=1}^K a_k \prod_{\substack{l=1 \\ l \neq k}}^K (\tau^* + b_l) = 0 \quad (5.5)$$

where $r_k^0 = C_k^0 - T$, $a_k = -\frac{r_k^0}{C_k^2}$ and $b_k = \frac{C_k^1}{C_k^2}$.

Algorithm 1 Process at the Orchestrator

Input: T, d, K **Output:** \mathbf{w}

Initialize \mathbf{w} and set the flag $STOP \leftarrow \mathbf{FALSE}$

- 1: **while not** $STOP$ **do**
- 2: *In Parallel:* Send \mathbf{w} to each learner $k \in \mathcal{K}$
- 3: *In Parallel:* Receive P_{ko}, h_{ko} , and f_k from learner $k \in \mathcal{K}$
- 4: Solve (5.5) to obtain τ and approximate $d_k \forall K \in \mathcal{K}$ using (5.4)
- 5: Ensure feasibility using (A.14) and (A.15)
- 6: *In Parallel:* Send $\lfloor \tau \rfloor$ and $\lfloor d_k \rfloor$ samples to each learner $k \in \mathcal{K}$ ⁴
- 7: *WAIT* for each learner $k \in \mathcal{K}$ to do τ local updates
- 8: *In Parallel:* Receive $w_k \forall k \in \mathcal{K}$
- 9: Obtain \mathbf{w} using (4.5)
- 10: **if** *STOPPING CRITERIA REACHED* **then**
- 11: Set $STOP \leftarrow \mathbf{TRUE}$
- 12: **end if**
- 13: **end while**
- 14: **return** \mathbf{w}

Proof: Please find the proof in Appendix A. ■

Although it was suspected that the above bounds would be loose, by simulations we show that the approximate solutions closely match the numerical solutions from the solver. The complete steps followed by the orchestrator over multiple global cycles to update τ and $d_k \forall K \in \mathcal{K}$ are summarized in Algorithm 1.

5.1.3 Simulation Environment for Testing the HA-Sync

In this section, we describe the realistic learning and edge node environments used to test our proposed adaptive task allocation solutions for HA synchronous (HA-Sync) MEL with only time constraints. More specifically, for both schemes FL (also referred to as DD) and PL (also referred to as TP), we compare the numerical solutions from OPTI of the relaxed version of the QCILP formulation of the proposed HA (HA-TP/DD-Num) scheme against the analytical results (HA-TP/DD-Ana) from (5.4) and (5.5) solutions. We show that the upper bounds are tight and match the numerical solution. We also show the merits of these two

⁴In FL, the orchestrator only sends the value of d_k and the learner chooses $\min(d_k, d_k^{max})$ data points from its privately owned dataset randomly where d_k^{max} is learner k 's dataset size.

Table 5.1: Simulation parameters for testing the HA-Sync with only time constraints

Parameter	Value
Wi-Fi Attenuation Model	$7 + 2.1 \log(R)$ dB [91]
Cell Attenuation Model	$128 + 37.1 \log(R)$ dB [84]
Node Bandwidth (W)	5 MHz
Device proximity indoor (R)	50m
Device proximity outdoor	500m
Transmission Power (P_k)	23 dBm
Noise Power Density (N_0)	-174 dBm/Hz
Computation Capability (f_k)	2.4 GHz and 700 MHz
Pedestrian Dataset size (d)	9,000 images
Pedestrian Dataset Features (\mathcal{F})	648 (18×36) pixels
MNIST Dataset size (d)	60,000 images
MNIST Dataset Features (\mathcal{F})	784 (28×28) pixels

solutions compared to the heterogeneity unaware (HU) equal task allocation (HU-TP/DD) scheme employed in [77, 78]. We will first introduce the simulation environment, and then present the testing results.

A typical MEC will consist of a cloudlet of heterogeneous devices, channel and computing devices. In our simulation, the edge nodes are assumed to be located in an area of 50m of radius for an 802.11 environment and a 500m radius for a cellular type environment. Half of the considered nodes emulate the capacity of a typical fixed/portable computing device (e.g., laptops, tablets, road-side units. etc.) and the other half emulates the capacity of commercial microcontrollers (e.g., Raspberry Pi) that can be attached to different indoor or outdoor systems (e.g., smart meters, traffic cameras). The setting thus emulates an edge environment that can be located either indoor or outdoor. The employed channel model between these devices is summarized in Table 5.1, which emulates 802.11/Cellular type links between the edge nodes.

Two datasets are considered in our simulation: namely the MNIST [92] dataset and the pedestrian [93]. The forward and backward passes will require 781,208 floating point operations [94]. The MNIST dataset consists of 60,000 images 28x28 images (784 features). The employed ML model for this data is a 3-layer neural network with the following configuration [784, 300, 124, 60, 10]. (The details about the resulting model sizes and complexities can be

found in [87].) On the other hand, the pedestrian dataset has 8,000 training images consisting of 684 features (18 x 36 pixels). The ML model used for this dataset is a single-layer neural network with 300 neurons in the hidden layer. For this model, the set of weights $\mathbf{w} = [w_1, w_2]$ is the concatenation of two sub-matrices, where w_1 is 300×648 and w_2 is 300×2 , neither of which depending on the batch size ($S_d = 0$). Thus, the size of the model is 6,240,000 bits, which is fixed for all edge nodes.

To evaluate the performance of MEL, two metrics are used: achievable local updates and validation accuracy progression per global cycle for both datasets. The MNIST dataset is a classic dataset typically used by ML researchers to demonstrate the superiority of their proposed approaches. On the other hand, the pedestrian dataset emulates a real-life scenario where a traffic camera periodically captures the image of a side-walk and the objective is to detect whether a pedestrian is present. In such an application, the detection accuracy and speed of detection are both equally important. Thus, datasets and model should be sufficiently large but small enough to ensure a higher number of local updates and thus, a faster accuracy progression.

5.1.4 Simulation Results for Parallelized Learning (PL/TP)

Figure 5.1 shows the results for training using the MNIST dataset with the aforementioned deep neural network model. Figures 5.1a and 5.1b depict the number of local iterations τ achieved by all tested approaches versus the number of edge nodes (for $T = 30$ s and $T = 60$ s) and against the global cycle time (for $K = 10$ and $K = 20$), respectively. We can first notice τ increases as the number of edge nodes increases. This is trivial as a smaller batch size will be allocated to each of the nodes as the number of learners is increased. We can also see that the performance of the OPTI-based numerical (HA-TP-Num) and UB-Analytical (HA-TP-Ana) solutions are identical for all simulated number of learners and global update cycle times.

We can finally observe from both sub-figures that the dynamically optimized heterogene-

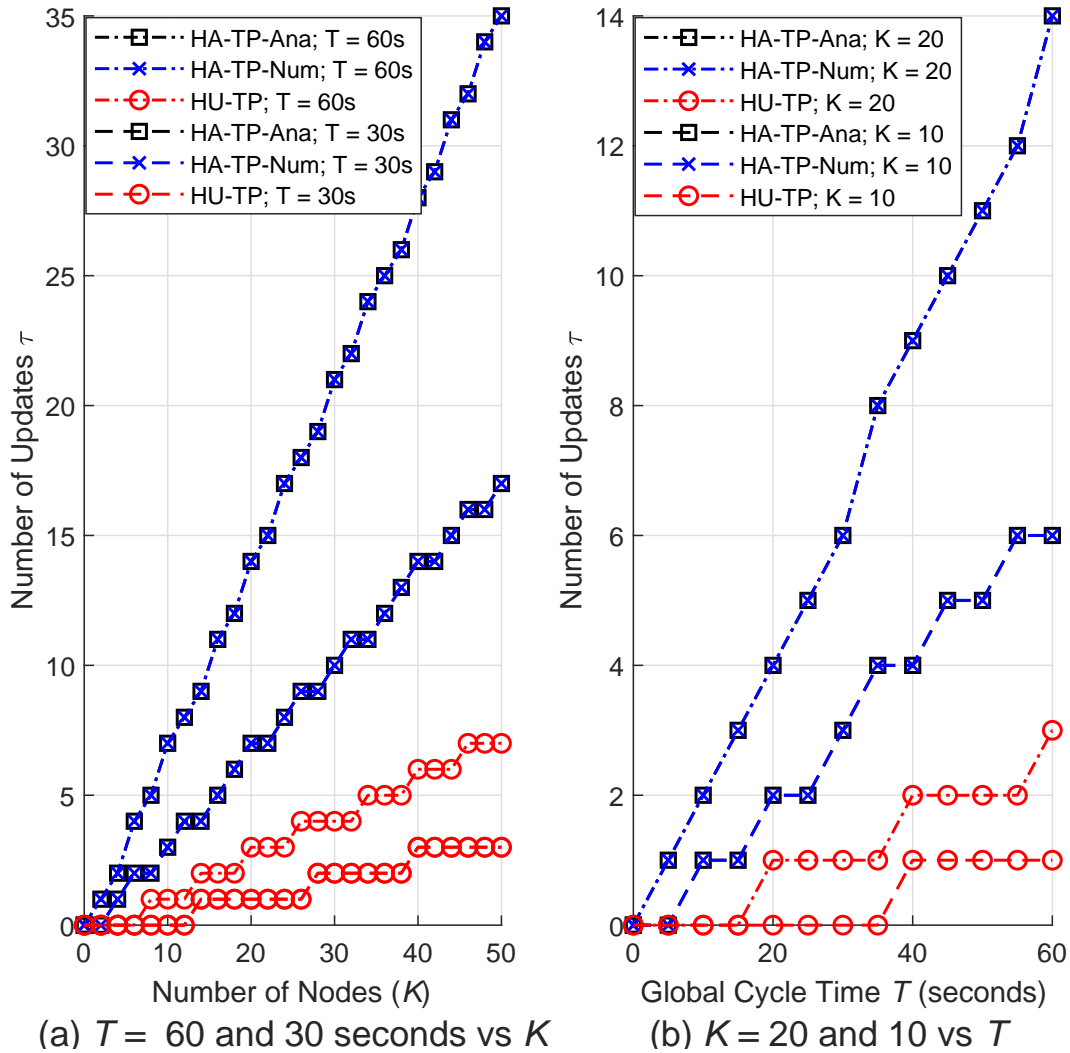


Figure 5.1: Achievable local iterations τ for the MNSIT dataset. (a) Performance comparison of all schemes for $T = 30$ and 60 s vs K (b) Performance comparison of all schemes for $K = 10$ and 20 vs T

ity aware (HA) scheme achieves a significantly larger number of local updates compared to the heterogeneity unaware (HU) scheme. For instance, the HA scheme makes it possible to perform 6 updates (as opposed to 1) for 20 learners each with a global cycle time of 30 s, a gain of 600%. When $K = 10$, at $T = 60$ s the HA approach for adaptive batch allocation gives $\tau = 7$ updates whereas only 3 updates are possible with equal allocation, a gain of 233%. Another interesting result is that the performance of the HU scheme for $T = 60$ s or $K = 20$ is actually much lower than the performance of our proposed solutions for $T = 30$

s or $K = 10$, respectively. In other words, our scheme can achieve a better level of accuracy as the HU scheme in half the time or with half the number of learners.

Figure 5.2a shows the number of local iterations τ achieved by all tested approaches versus the number of edge nodes, for $T = 15$ and $T = 5$ s. Similar to the results of the MNIST dataset, the numerical solution (HA-TP-Num) matches the analytical upper bound solution (HA-TP-Ana) for all simulated scenarios. For both cases, when $T = 15$ s and 15

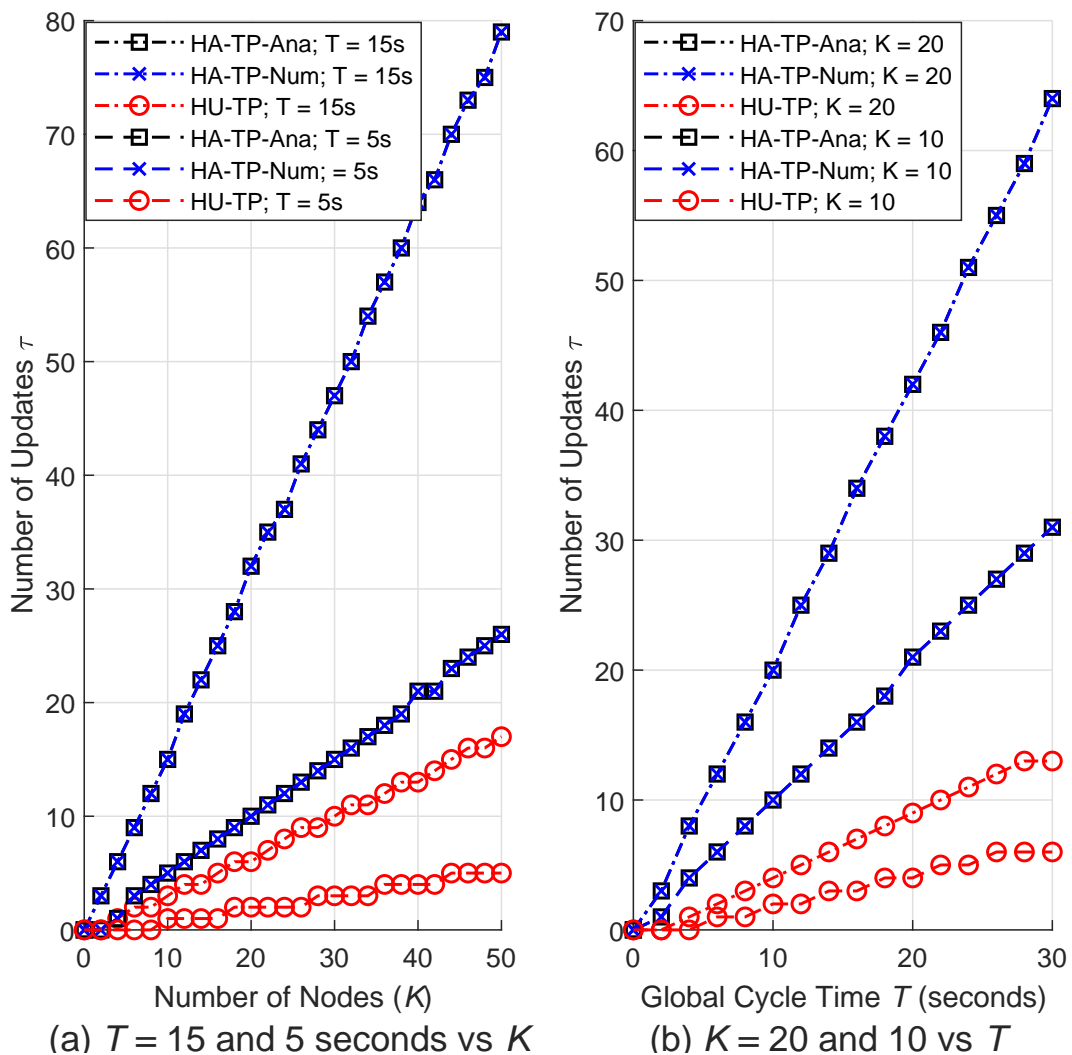


Figure 5.2: Achievable local iterations τ for the Pedestrian dataset. (a) Performance comparison of all schemes for $T = 15$ and 5 s vs K (b) Performance comparison of all schemes for $K = 10$ and 20 vs T

s, the figures show that both HA approaches result in a significantly higher number of local iterations than the HU approach. For example, for $T = 15$ s, 50 learners can perform only 26 iterations each, whereas 78 can be achieved with our proposed solutions, a gain of 300%. Another interesting result is that the performance of ETA scheme for $T = 15$ s is actually worse than the performance of our proposed solutions for $T = 5$ s. In other words, our scheme can achieve a better level of accuracy as the ETA scheme in half the time.

Figure 5.2b illustrates the number of local iterations τ versus the global cycle time T , for $K = 10$ and 20 learners. Once again, the solutions of HA-TP-Num match HA-TP-Ana. Comparing this performance to the HU approach, the figures shows that, when $T = 10$ s, our solutions can facilitate 20 local iterations on each of the 20 edge learners, versus only 4 possible iterations with equal batch allocation, a gain of 500%. For $T = 30$ s, the HA scheme can reach up-to 64 iterations whereas the HU scheme achieves only 12 updates, less than the number of local iterations made possible by our HA scheme for a system of 10 learners.

Validation Accuracy

The left and right sub-figures in Figure 5.3 depict the progression of learning accuracy achieved by both, our dynamic HA scheme (Figure 5.3a) and HU (5.3b) after global update cycles of $T = 12$ s each for $K = 20$ and $T = 30$ s each for $K = 10$. The figure shows a significant improvement achieved by our proposed scheme over the HU method in reaching a high accuracy in fewer global cycles. For $K = 20$ and $T = 12$ s, to reach an accuracy of 96.67%, the HA scheme require 4 cycles as opposed to 7 for the HU, a reduction of about 43% (i.e., 48 s). Furthermore, an accuracy of 97% can be achieved with our proposed scheme which is not possible with the HU scheme. For the case of $K = 10$ and $T = 30$ s, HA can cross the 97% mark for accuracy in 5 updates whereas HU requires 9 updates which represents a reduction in time of 2 minutes or about 56%. This clearly exhibits the merits of the adaptive scheme in reaching learning goals in much less time and number of cycles, which also saves significantly on nodes' energy.

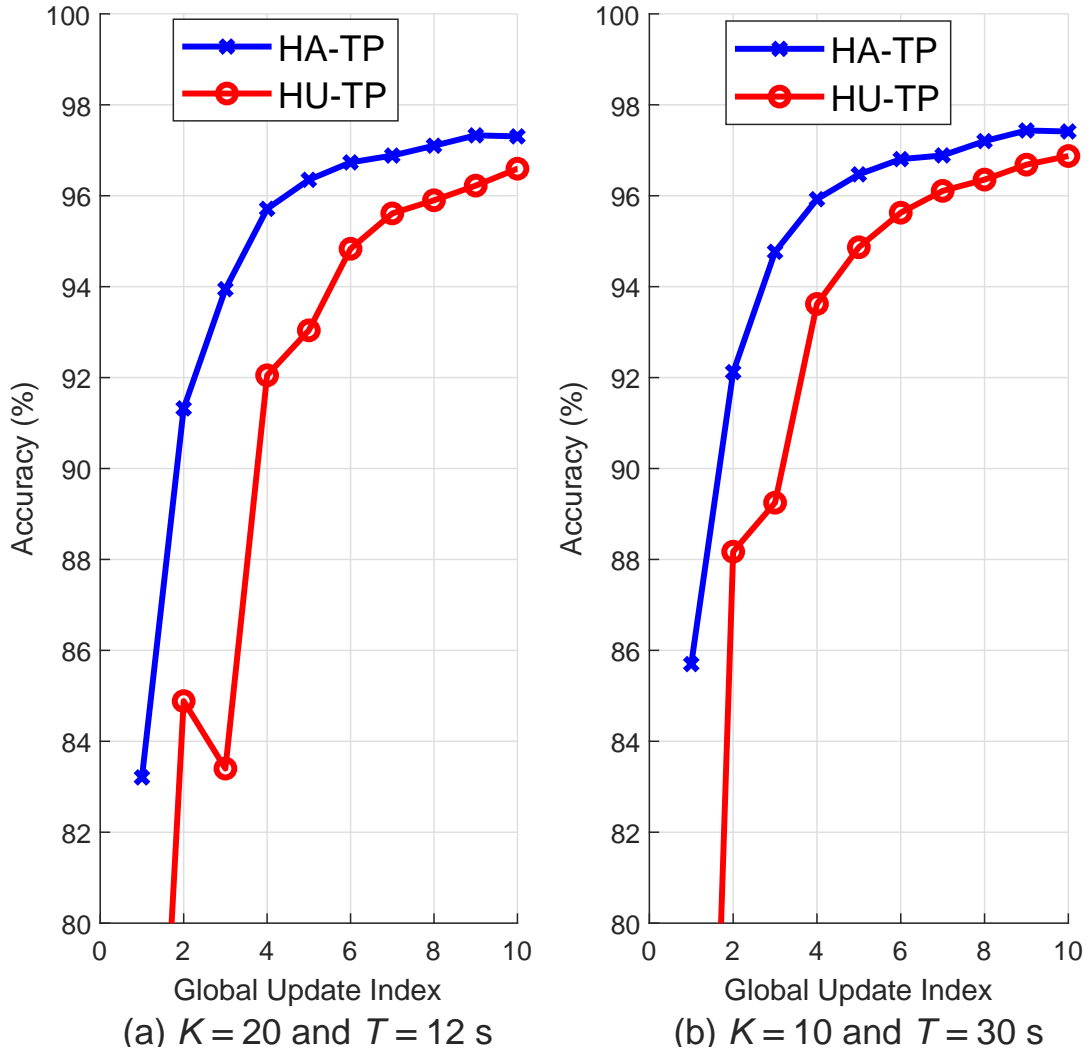


Figure 5.3: Validation accuracy progression for the MNIST Dataset after global cycles updates for (a) $K = 20$ and $T = 12$ s (b) $K = 10$ and $T = 30$ s

5.1.5 Simulation Results for Federated Learning (FL/DD)

In the next phase of our simulations, we examine the performance of all schemes in an outdoor setting with cellular-type communication links. The parameters for this setting have also been given in Table 5.1. There are four possibilities in this scenario. We can have a system employing the distributed datasets (DD) approach or the task-parallelization (TP) with our optimized task allocation or heterogeneity aware (HA) scheme. Both scenarios

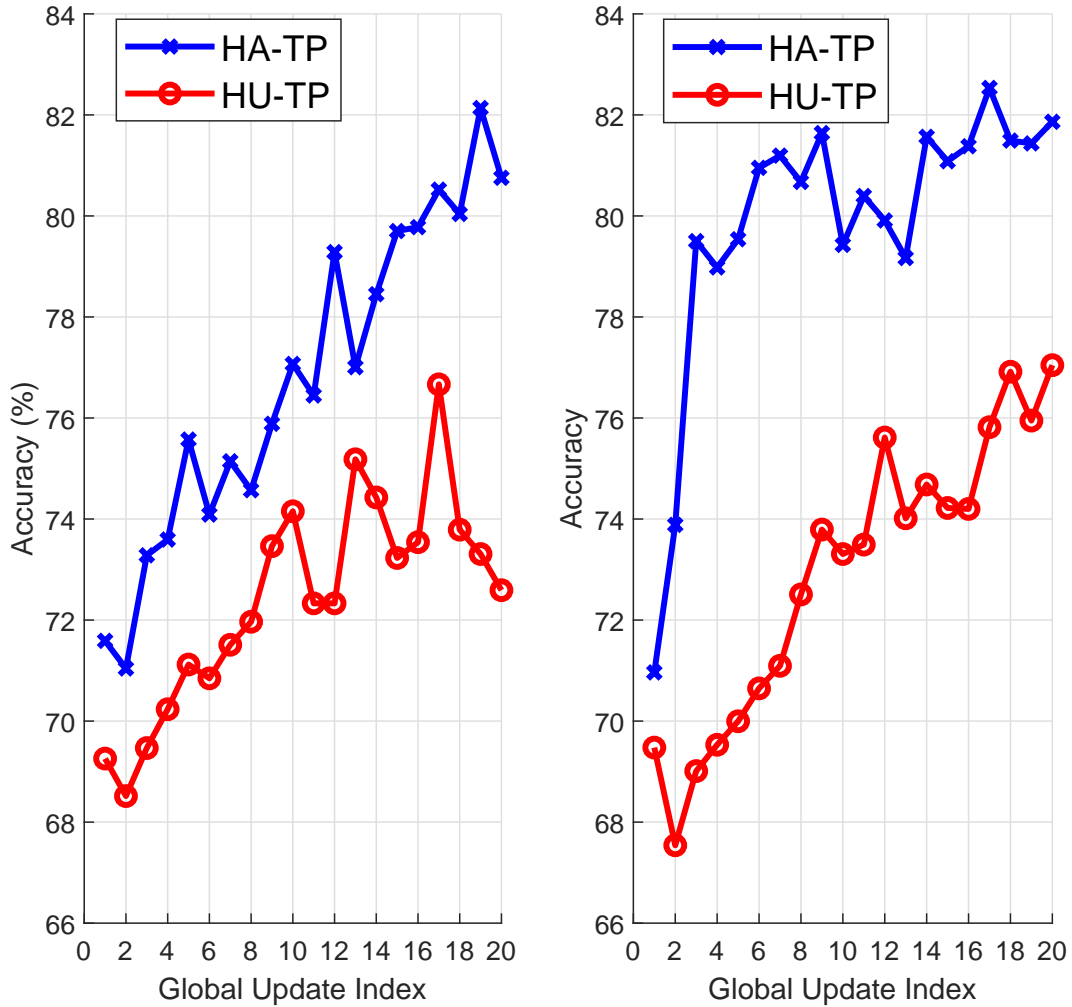


Figure 5.4: Validation accuracy progression for the Pedestrian Dataset after global cycles updates for (a) $K = 20$ and $T = 5$ s (b) $K = 20$ and $T = 3$ s

can also occur where the system simply follows the ETA or heterogeneity unaware (HU) scheme. For our proposed method, we show that the analytical approximations (HA-TP-Ana and HA-DD-Ana) match the OPTI-based solutions (HA-TP-Num and HA-DD-Num). Furthermore, we compare the performances of the dynamic task allocation in both scenarios against the performance of the HU scheme (HU-TP and HU-DD).

Figure 5.5 and Figure 5.6 show the number of local iterations achieved by all schemes for the MNIST and Pedestrian datasets, respectively, in both scenarios against the number

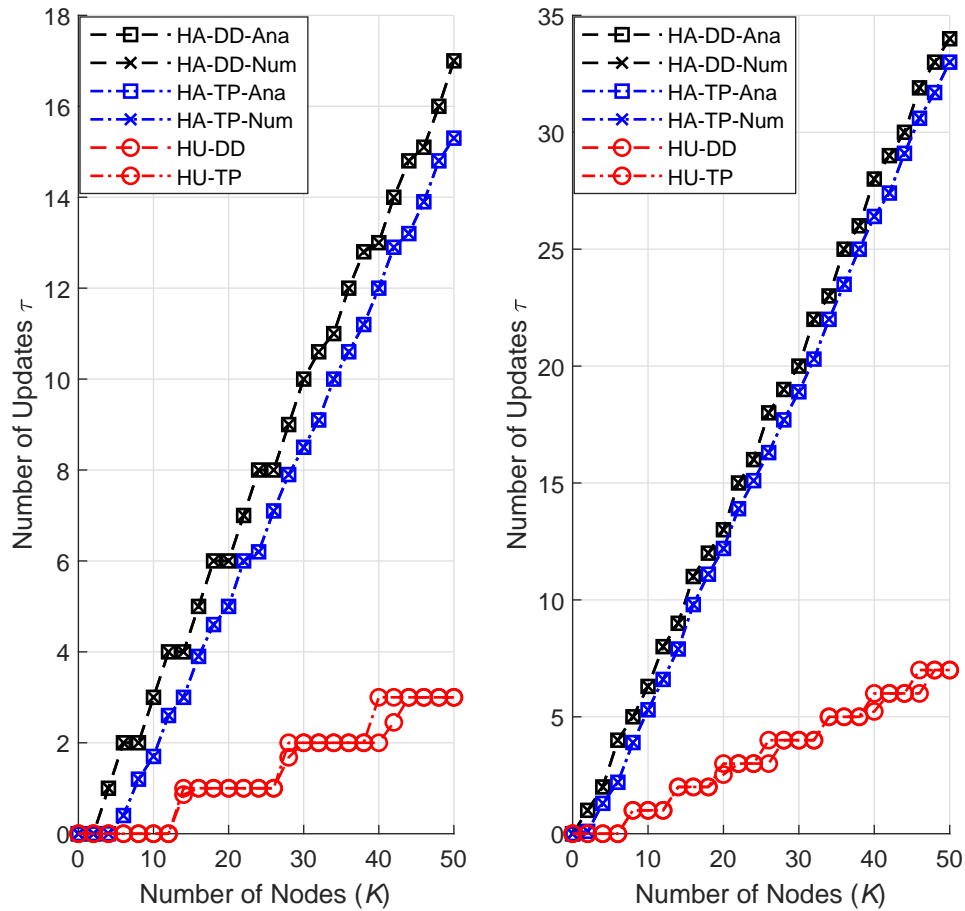


Figure 5.5: Achievable local iterations τ for the MNSIT dataset. (a) Performance comparison of all schemes for $T = 30$ and 60 s vs K (b) Performance comparison of all schemes for $K = 10$ and 20 vs T

of edge nodes for fixed times (30 s and 60 s for MNIST and 1 s and 2 s for pedestrian, respectively). The MNIST dataset is larger than the Pedestrian dataset and we employ a more complex and larger ML model for MNIST classification. Therefore, the distributed datasets approach gives a smaller gain compared to the Pedestrian dataset. For example, while employing 20 learners, for a global cycle time of 30 s, 5 updates can be achieved with task-parallelization whereas 6 updates can be achieved in the distributed datasets approach, a gain of 20%. When T is increased to 60 s, both approaches give similar performance because the size of the model parameters' set is much larger than the dataset itself.

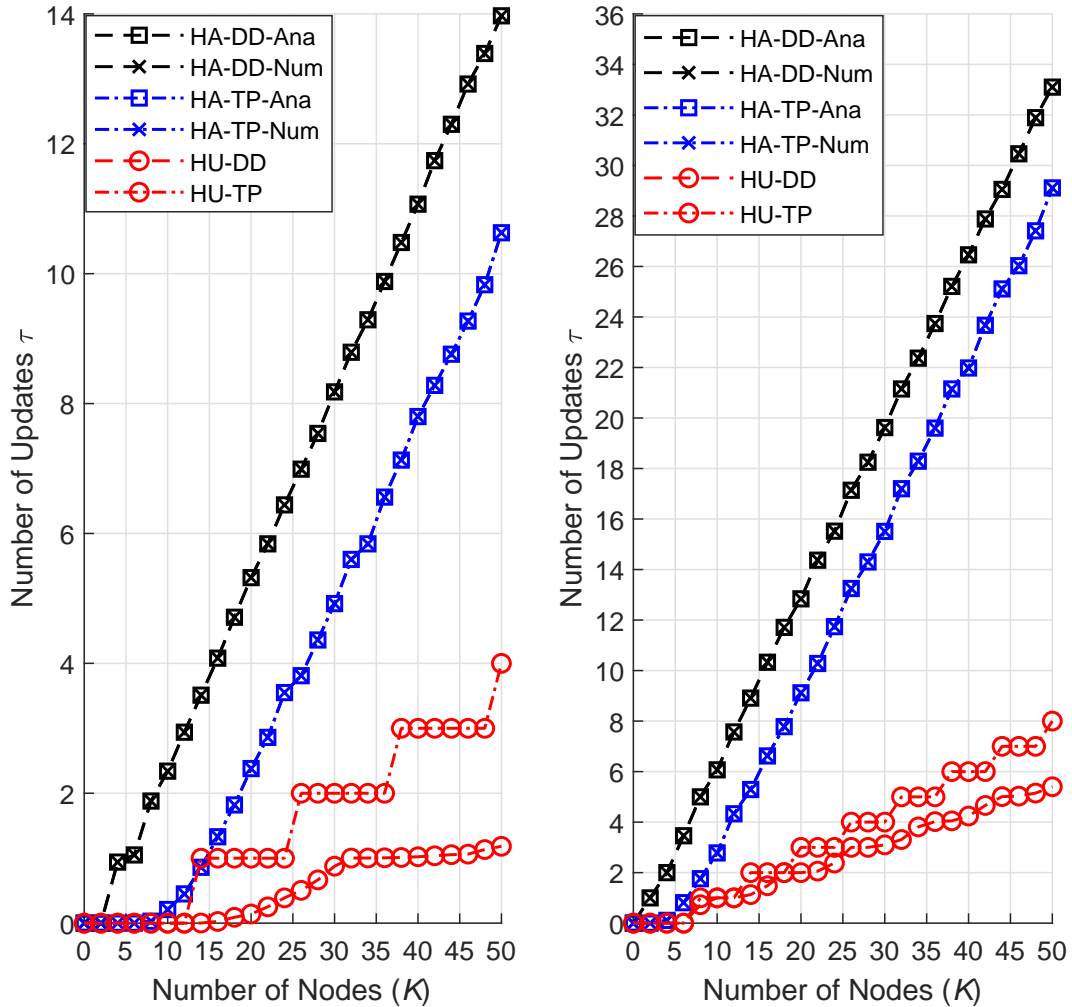


Figure 5.6: Achievable local iterations τ for the Pedestrian dataset. (a) Performance comparison of all schemes for $T = 15$ and 5 s vs K (b) Performance comparison of all schemes for $K = 10$ and 20 vs T

On the other hand, in the scenario where T is restricted to 1 s for the Pedestrian dataset, a system of 20 learners allows for 2 local iterations with the task-parallelization approach whereas 5 iterations can be achieved with the distributed datasets approach, a gain of 150%. In contrast, when T is increased to 2 s, the task-parallelization and distributed datasets approaches can achieve 9 and 13 updates, respectively, which offers a gain of 44.44% only. In general, it is noticeable that the gain of the distributed datasets approach is significant over the task-parallelization approach when the size of the model is comparable to the subsets of

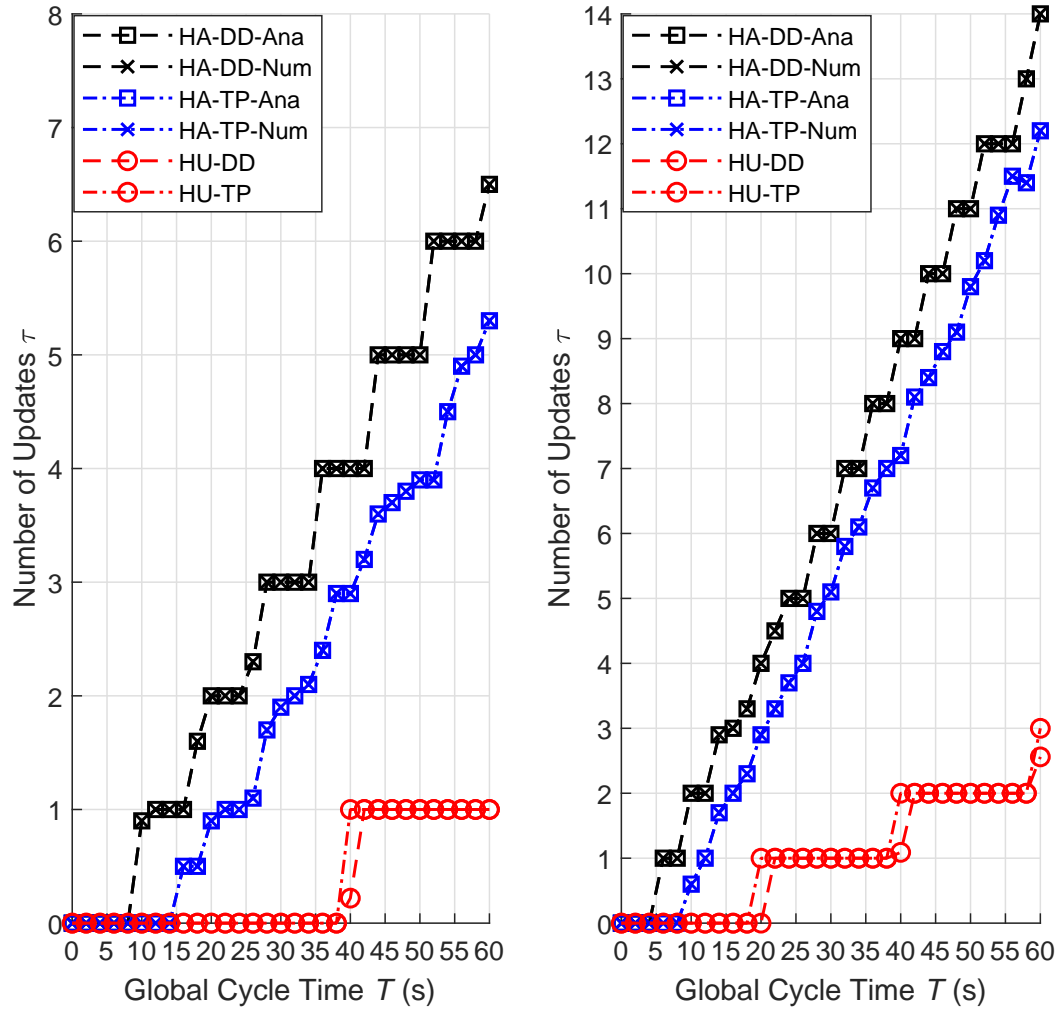


Figure 5.7: Achievable local iterations τ for the MNSIT dataset. (a) Performance comparison of all schemes for $T = 30$ and 60 s vs K (b) Performance comparison of all schemes for $K = 10$ and 20 vs T

the data. Furthermore, as larger global update cycles are allowed, which is typically not the case in edge computing, the performance of both approaches is similar.

Figures 5.7 and 5.8 show the number of local iterations achieved by all schemes for the MNIST and Pedestrian dataset in both scenarios against time for a fixed number of learners ($K = 10$ and 20 , respectively). Similar to the previous cases, as the number of learners are increased, the amount of gain achieved in terms of the number of local iterations that can be performed is reduced. A similar trend is also noticed where the gains achieved in terms

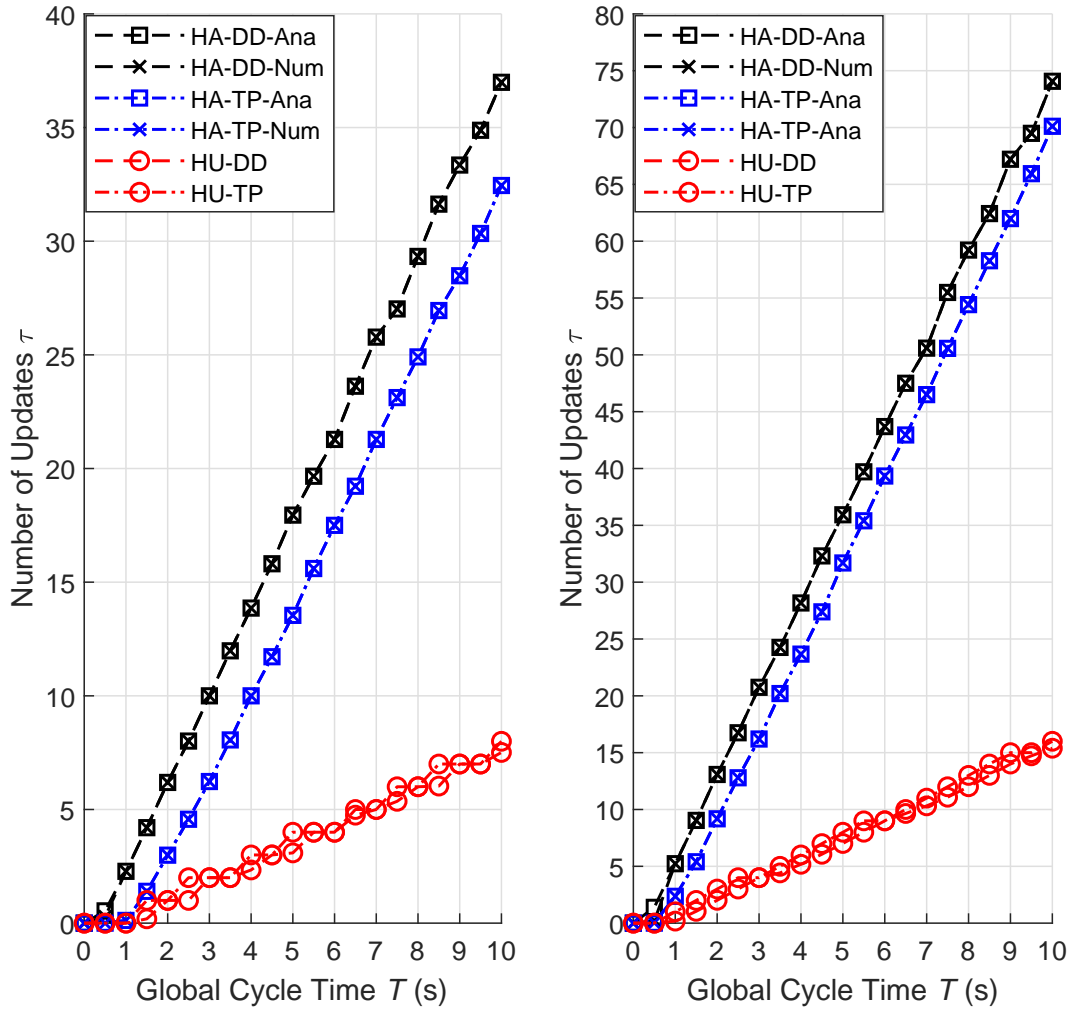


Figure 5.8: Achievable local iterations τ for the Pedestrian dataset. (a) Performance comparison of all schemes for $T = 15$ and 5 s vs K (b) Performance comparison of all schemes for $K = 10$ and 20 vs T

of the number of possible updates for the smaller dataset employing the smaller ML model are higher than for the MNIST dataset.

For example, for the MNIST dataset with the larger ML model, a system employing DD in a setting with $K = 20$ for $T = 30$ s can allow for 7 local updates whereas TP gives 6 local iterations; in other words, a gain of 16.67%. In contrast, employing the smaller dataset with the smaller model and setting $T = 2$ s for $K = 20$, TP allows 9 updates whereas the other approach allows 13 updates leading to a gain of 44.44%. For the same scenario, if we

reduce the global cycle time to 1 s, then in that case, TP gives 2 updates whereas DD gives 5 updates or a gain of 150%.

Validation Accuracy

This section presents the learning accuracy performance comparisons for both dynamic schemes (HA and HU), DD and TP, in an outdoor mobile channel environment. Figure 5.9 compares the learning accuracy of all schemes (HA-TP and HA-DD versus HU-TP and HU-DD) for the MNIST dataset when the global cycle time is set to 60 s and the system consists of 10 learners. In general, the dynamic HA schemes are far superior to the HU

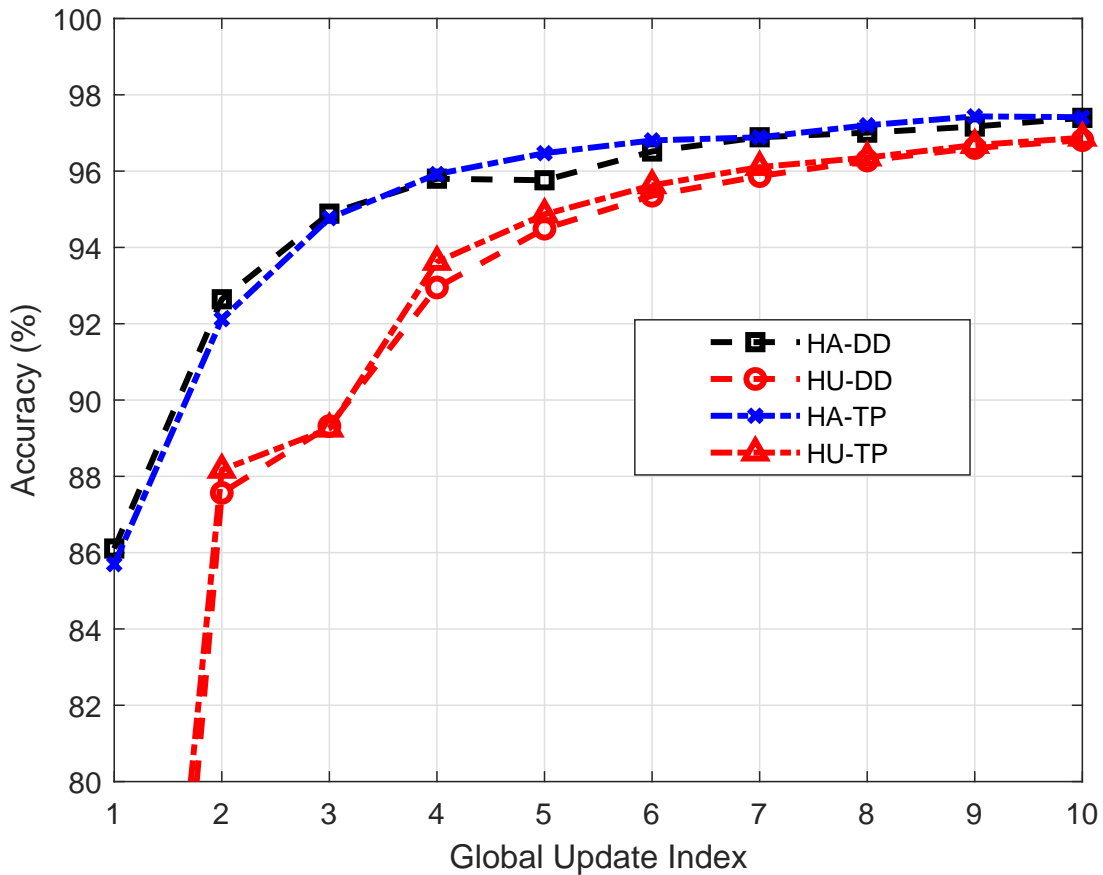


Figure 5.9: MNIST Learning Results. Learning accuracy comparison for $T = 60$ s and $K = 10$ using both, the dynamic and ETA approaches, for both frameworks: distributed datasets and task-parallelization.

approach. For example, HA reaches an accuracy of 96% in 4 global cycles whereas the HU approaches require 7 cycles, a reduction in time of about 43%. We also notice that TP seems to work slightly better than the DD approach despite more iterations (larger τ) are possible with the latter scenario. The reason for that is that the former approach resembles SGD more accurately because each node performs learning on a different random subset of data after each global update. On the other hand, devices learn on all or part of the same subset of data at each global cycle in the DD scenario.

Figure 5.10 compares the learning accuracy of all schemes for the Pedestrian dataset when the global cycle time is set to 5 s and the system consists of 10 learners. Once again, the dynamic HA schemes are far superior to the HU approach. For example, HA reaches an accuracy of 76% in at most 10 global cycles whereas the HU approaches require at least 10 cycles, a reduction in processing-time requirement by 40%. We also notice that the DD approach seems to work better than the TP approach. One possible interpretation is that the difference in the average number of iterations per node (more than 5) makes up for the deterministic nature of the DD scenario. For example, the DD approach with dynamic task allocation crosses the 77% accuracy mark in the 8th global update cycle whereas the TP approach requires 12 updates, in other words, it takes 50% more time. On the other hand, the TP approach seems to converge more smoothly to an accuracy of above 80% which maybe due to the more “stochastic” nature of the algorithm as compared to the DD scenario.

Figure 5.11 compares the learning accuracy of only the dynamic schemes for the Pedestrian dataset when the global cycle time is reduced to 2 s and the system consists of 10 and 20 learners, respectively. With such low global cycle time, the HU schemes fail. Interestingly, after the first few iterations, the TP approach clearly works better when there are only 10 learners. For example, this approach requires only 8 updates to cross the 79% accuracy benchmark whereas the DD approach requires 16 updates or an increase in time by 100%. On the other hand, when we have 20 learners, i.e. each device has a smaller subset of data

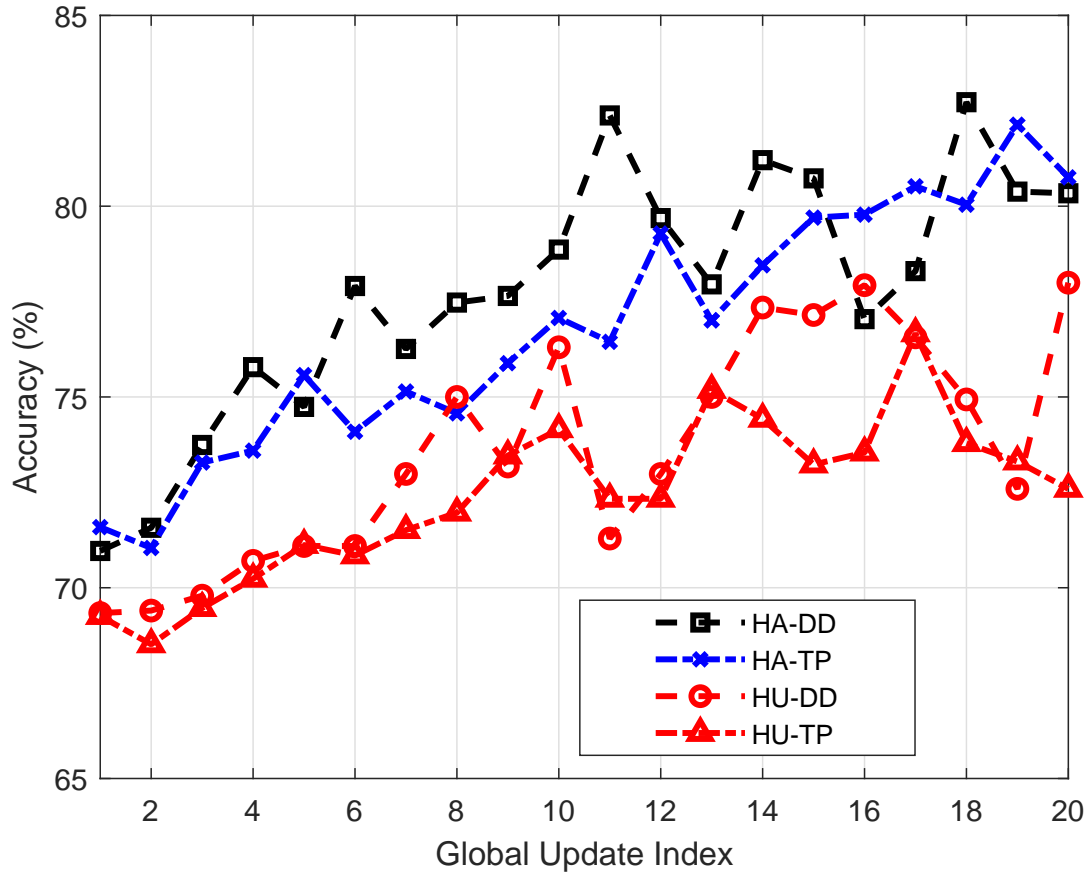


Figure 5.10: Pedestrian learning results. Learning accuracy comparison for $T = 5$ s and $K = 10$ using both, the dynamic and ETA approaches, for both frameworks: distributed datasets and task-parallelization.

to learn from, having a larger number of iterations has an impact because the performance of both schemes is similar. For example, the distributed approach reaches 80% in 15 updates compared to 17 updates for the parallelization approach; a reduction in time of about 12%.

5.1.6 Complexity Analysis

As described in previous sections, the problem of interest is a non-convex QCLP after relaxation, which can still be written in the form of a QCQP. The solvers of the type we used typically employ the interior-point (IP) methods mixed with heuristics. For the case of a convex QCQP, IP methods can achieve a solution in polynomial time but are generally NP-

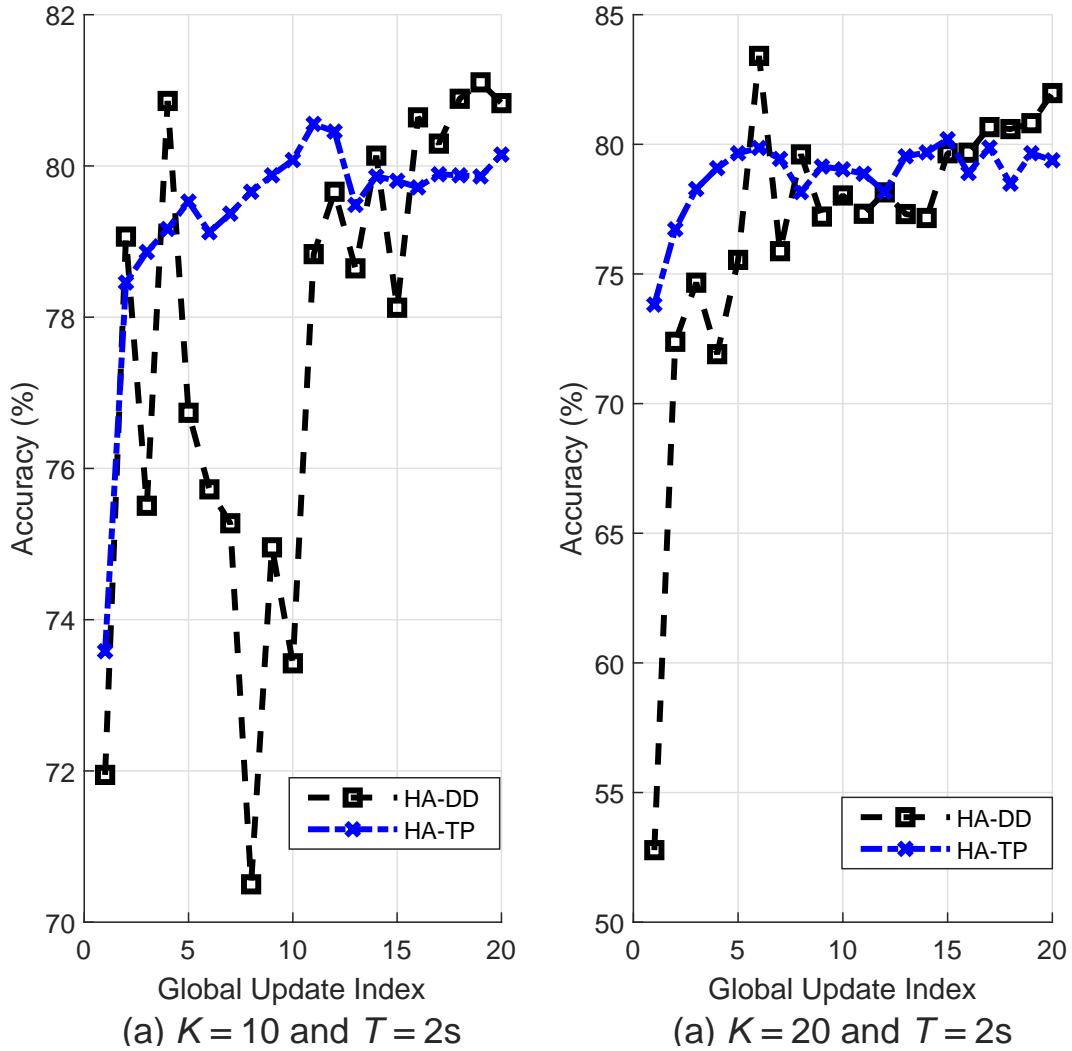


Figure 5.11: MNIST results (a) Performance comparison of all schemes for $T = 30$ and 60 s vs K (b) Performance comparison of all schemes for $K = 10$ and 20 vs T

hard for non-convex problems. On the other hand, our approximation of the upper bounds is based on the solving of a K^{th} degree polynomial. Most solvers such as MATLAB and LAPACK find the roots using the Eigenvalues of the companion matrix which can typically be achieved in polynomial time.

The complexity of the polynomial solvers arises mainly from the QR factorization needed for diagonalization, and with the fast DQR algorithms, can be achieved in $\mathbf{O}(4/3n^3 + n^2)$ [95]. For our solution, the complexity for the K^{th} degree-polynomial can be given by:

$\mathbf{O}(4/3K^3 + K^2)$. The complexity of an IP depends on the size of the quadratic matrix and the number of quadratic constraints in the problem. It is given by $\mathbf{O}(n^{1/2} [m + n] n^2)$ [96]. For our problem, the complexity will be $\mathbf{O}([K + 1]^{1/2} [2K + 1] [K + 1]^2)$. The final expression can be expanded to the following form: $\mathbf{O}([K + 1]^{1/2} [2K^3 + 5K^2 + 4K + 1])$; this is for convex problems. Since our problem is non-convex, the complexity will actually be non-polynomial.

Figure 5.12 better illustrates the complexity of the solution in terms of the execution times for each method. The results are presented for the HA-TP schemes because DD is a subset of TP. The simulations were ran on a laptop with an 8-core Intel i7 processor and the OPTI toolbox [90] was used for the numerical optimization. For each K , the execution time was reported for an average of 100 runs with difference communication and computation capacities across each run (same for all schemes in a given run). Since the optimization in the HU scheme is linear, it can be done in constant time whereas our proposed analytical solution has a complexity on the order of $\mathbf{O}(K^3)$. On the other hand, the complexity is much higher for the case when a numerical solution is pursued. Figure 5.12a demonstrates that the complexity of our proposed scheme is comparable to the HU scheme. Furthermore, it seems that increasing the global cycle time does not have any impact on the execution time requirements of the HU and HA-TP-Ana schemes. In contrast, a larger T reduces the execution time of the HA-TP-Num method because it allows the optimizer to reach the solution faster.

Figure 5.12b represents a zoomed-in view of the behavior of the HU and HA-TP-Ana schemes. Interestingly, our proposed solution provides a lower execution time compared to the equal task allocation approach up-to certain values of K ($K = 40$) while at the same time, providing gains on accuracy and reductions in convergence times. Therefore, our proposed method improves the accuracy with a lower execution time for smaller systems. Furthermore, in the region where the execution time of our proposed HA method exceeds the HU scheme, the increase in processing time is much less compared to the reductions in

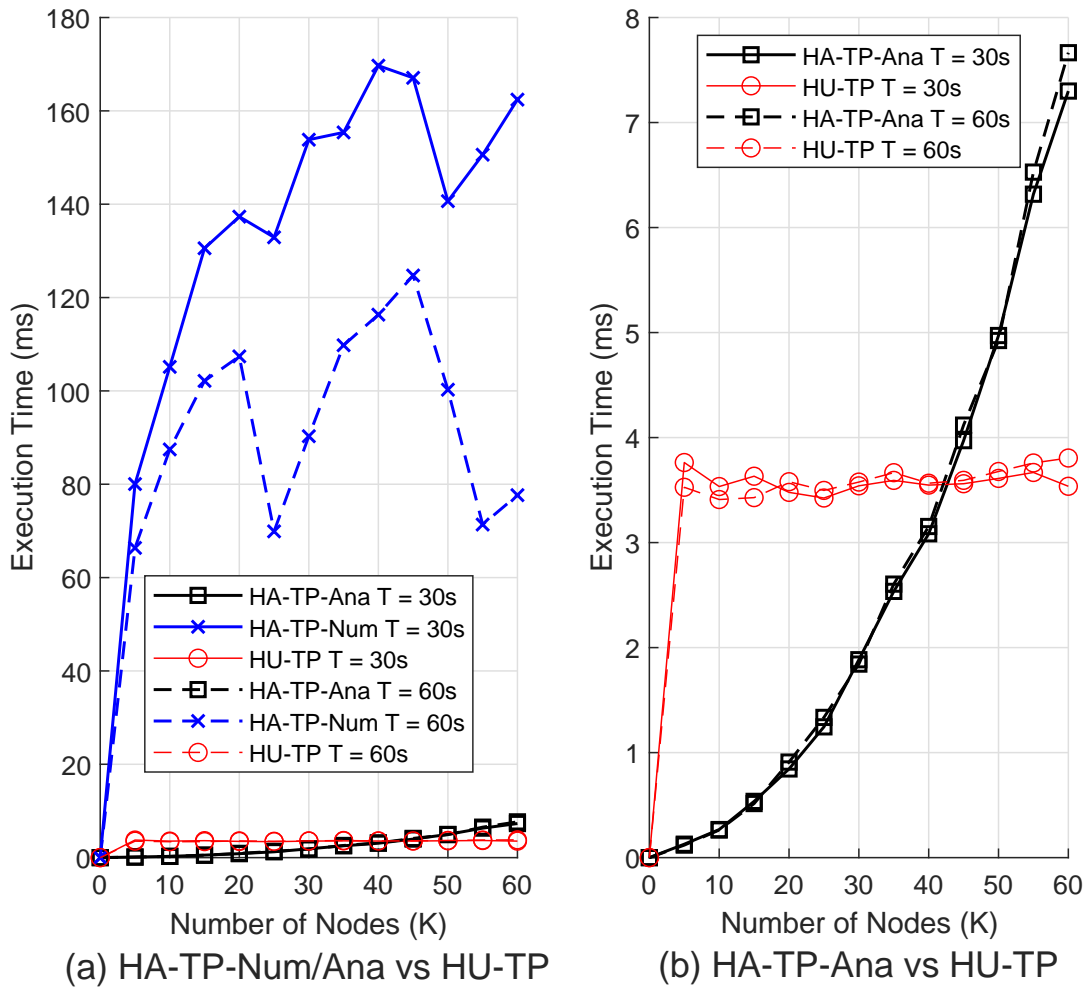


Figure 5.12: Complexity comparisons for the MNIST dataset for the TP scheme for $T = 30$ and 60 s vs K illustrating: (a) Execution times of all methods (HA analytical and numerical against the HU) (b) Execution times of the HA analytical versus HU scheme.

time taken to converge to a certain validation accuracy. The execution time increases on the order of milliseconds whereas the convergence time saved is on the order of s.

5.2 Asynchronous Task Allocation

This section presents results for MEL in the scenario where learners are allowed to perform an asynchronous number of updates while satisfying a given time constraint. Hence, this approach is called the HA asynchronous task allocation (HA-Asyn). The material presented

in this section is part of an article available as a pre-print on Arxiv⁵.

5.2.1 Formulation

In contrast to the synchronous case presented in Section 5.1, any learner $k \in \mathcal{K}$ will now be allowed to perform τ_k local updates. Hence, the number of local iterations done by any learner $k \in \mathcal{K}$ may now differ from the number of local updates done by learner $l \in \{\mathcal{K} \mid l \neq k\}$. To this end, we can define the staleness between the models of any two learners as:

$$s_{k,l} = |\tau_k - \tau_l|, \forall k \in \mathcal{K} \ \& \ l \in \{\mathcal{K} \mid l \neq k \ \forall k\} \quad (5.6)$$

So, the staleness between any two learners is the difference between the number of local learning cycles each has performed. It has been shown in the literature for asynchronous SGD [88] and FL [98] that the loss function of SGD-based ML is minimized (and thus the learning accuracy is maximized) by minimizing the staleness between the gradients.

Overall, the maximum staleness has to be minimized while satisfying other constraints including the global cycle time constraint where $t_k \leq T \ \forall k \in \mathcal{K}$. The expression for t_k can now be given as:

$$t_k = C_k^2 \tau d_k + C_k^1 d_k + C_k^0 \quad k \in \mathcal{K} \quad (5.7)$$

where C_k^2 , C_k^1 , and C_k^0 represent the quadratic, linear, and constant coefficients of learner k in terms of the optimization variables τ_k and d_k as defined in (4.18)-(4.20)⁶.

Therefore, the problem can be formulated as an ILP with quadratic and linear constraints

⁵This work titled ‘‘Adaptive Task Allocation for Asynchronous Federated Mobile Edge Learning’’ has been uploaded as a pre-print article on Arxiv [97] and has been submitted to the IEEE Transactions on Vehicular Technology (TVT) as a correspondence paper.

⁶Note that, for the FL scenario, the only difference in the model is that the first term of the numerator in (4.19) will not exist.

as follows⁷:

$$\min_{\tau_k, d_k \forall k} \max_{s_{k,l}} \quad k \in \mathcal{K} \ \& \ l \in \{\mathcal{K} \mid l \neq k \ \forall k\} \quad (5.8)$$

$$\text{s.t.} \quad C_k^2 \tau_k d_k + C_k^1 d_k + C_k^0 = T, \quad k \in \mathcal{K} \quad (5.8a)$$

$$\sum_{k=1}^K d_k = d \quad (5.8b)$$

$$\tau_k \in \mathbb{Z}_+, \quad \forall k \in \mathcal{K} \quad (5.8c)$$

$$d_k \in \mathbb{Z}_+, \quad \forall k \in \mathcal{K} \quad (5.8d)$$

$$d_l \leq d_k \leq d_u, \quad \forall k \in \mathcal{K} \quad (5.8e)$$

Constraint (5.8a) guarantees that $t_k = T \ \forall k$, which means that all devices work for the full allotted time though they may perform different number of epochs. Constraint (5.8b) ensures that the sum of batch sizes assigned to all learners is equal to the total dataset size that the orchestrator needs to analyze. Constraints (5.8c) and (5.8d) are simply non-negativity and integer constraints for the optimization variables. Please note that the solutions of (5.8) having any τ_k and/or d_k being zero represent conditions where MEL is not feasible for learner $k \in \mathcal{K}$.

Constraint (5.8e) bounds the number of data points considered by each learner in order to ensure that the task allocation algorithm does not assign extremely small batches to certain learners and very large batches to others. In PL, the orchestrator controls the dispatching of the data, which implies that it will allocate and transmit d_k samples such that $d_l \leq d_k \leq d_u$. On the other hand, in FL, the orchestrator can ensure that all selected learners during the initiation process have d_l samples. Then, in every global update cycle, learner k will learn on a subset of the data of size equal to $\min(d_k, \beta_k)$, where β_k is the total size of data at learner k . Since $d_k \leq d_u$ in either case, the upper bound will be satisfied.

⁷Note that, for the FL scenario, the only difference in the formulation is the simpler expression of C_k^1 .

Please note that jointly, the bounds on d_k and the equality constraint on $t_k, \forall k \in \mathcal{K}$ will help ensure that minimizing the staleness does not lead to solutions with the minimum possible updates with zero staleness, which will eliminate the benefits of HA-Asyn.

5.2.2 Proposed Solution

Clearly, the relationship between t_k and the optimization variables d_k and $\tau_k \forall k \in \mathcal{K}$ is quadratic. Furthermore, the optimization variables τ_k and $d_k \forall k$ are all non-negative integers. Consequently, the program in (5.9) is a QCILP, which is well-known to be NP-hard [89]. The problem is simplified by relaxing the integer constraints in (5.9d) and (5.9e), solving the relaxed problem, then flooring the obtained real results back into integers. We begin by applying a min-max transformation and relaxing the integer constraints as follows:

$$\min_{\tau_k, d_k \forall k} z \quad (5.9)$$

$$\text{s.t.} \quad |\tau_k - \tau_l| \leq z, \quad k \in \mathcal{K} \ \& \ l \in \{\mathcal{K} \mid l > k \ \forall k\} \quad (5.9a)$$

$$C_k^2 \tau_k d_k + C_k^1 d_k + C_k^0 = T, \quad k \in \mathcal{K} \quad (5.9b)$$

$$\sum_{k=1}^K d_k = d \quad (5.9c)$$

$$\tau_k \geq 0, \quad \forall k \in \mathcal{K} \quad (5.9d)$$

$$d_l \leq d_k \leq d_u, \quad \forall k \in \mathcal{K} \quad (5.9e)$$

The slack variable z is introduced in (5.9a) and an additional constraint is defined in (5.9a) to ensure that the staleness is less than z , which in turn guarantees that the maximum staleness is minimized. Please note that constraint (5.8d) has been eliminated due to the lower bound on d_k .

The resulting program in (5.9) becomes a linear program with quadratic constraints. This problem can be solved by using interior-point or ADMM methods implemented in

commercial solvers. From the analytical viewpoint, the associated matrices to each of the quadratic constraints in (5.9b) can be written in a symmetric form. However, these matrices will have two non-zero values that are positive and equal. The eigenvalues will thus sum to zero, which means these matrices are not positive semi-definite, and hence, the relaxed problem is non-convex. Consequently, the optimal solution cannot be obtained analytically. Therefore, upper bounds will be derived using Lagrangian analysis followed by improve steps to reach a feasible solution.

Let $\tau = \{\tau_1, \dots, \tau_k, \dots, \tau_K\}$ and $\mathbf{d} = \{d_1, \dots, d_k, \dots, d_K\}$. The Lagrangian of the relaxed problem is given by:

$$L(z, \tau, \mathbf{d}, \lambda, \alpha, \omega, \nu, \nu', \mu, \mu') = z + \sum_{k=1}^K \lambda_k (C_k^2 \tau_k d_k + C_k^1 d_k + C_k^0 - T) + \alpha_k \tau_k + \omega \left(\sum_{k=1}^K d_k - d \right) + \sum_{k=1}^K \nu_k (-d_k + d_l) + \sum_{k=1}^K \nu'_k (d_k - d_u) + \sum_{n=1}^N \mu_n (-z + \tau_{c_{n,1}} - \tau_{c_{n,2}}) + \sum_{n=1}^N \mu'_n (-z - \tau_{c_{n,1}} + \tau_{c_{n,2}}) \quad (5.10)$$

where the λ_k 's $k \in \mathcal{K}$, ω , and ν_k/ν'_k $k \in \mathcal{K}$, are the Lagrangian multipliers associated with the time constraints of the K learners in (5.9b), the total batch size constraint in (5.9c), the non-negative constraints of the number of epochs at each node τ_k in (5.9d) and the lower and upper bounds in (5.9e), respectively. The multipliers μ_n and μ'_n $n \in \{1, \dots, N\}$ are associated with (5.9a). Note that the absolute value constraint in (5.9a) can be decoupled as $\tau_k - \tau_l \leq z$ and $\tau_l - \tau_k \leq z$, $k \in \mathcal{K}$ & $l \in \{\mathcal{K} \mid l > k \forall k\}$.

The matrix $\mathbf{c} \in \mathbb{R}^{N \times 2}$ where N is the number of possibilities of mutual staleness for K set of users, i.e. $N = \binom{K}{2}$. For example, for a set of 4 users, $N = 6$ and the matrix of possibilities will be:

$$\mathbf{c} = \begin{bmatrix} 1 & 1 & 1 & 2 & 2 & 3 \\ 2 & 3 & 4 & 3 & 4 & 4 \end{bmatrix}^T \quad (5.11)$$

Using the KKT conditions $\nabla L_{\mathbf{x}} = 0$, the following theorem gives a way to find the optimal values of τ_k and d_k using the Lagrange multipliers.

Theorem 2 *The optimal number of updates each user node can perform τ_k can be given by:*

$$\tau_k^* = -\frac{\lambda_k C_k^1 + \nu_k + \nu'_k + \omega}{\lambda_k C_k^2} \quad \forall k \quad (5.12)$$

Moreover, the optimal value of d_k can be given by the following equation:

$$d_k^* = -\frac{u_k + u'_k + \alpha_k}{\lambda_k C_k^2} \quad \forall k \quad (5.13)$$

Each element of the vectors \mathbf{u} and \mathbf{u}' is a function of the Lagrange multipliers μ_n and μ'_n . Please refer to the proof below.

Proof: The proof of this theorem can be found in Appendix B. ■

As suspected, due to the relaxed problem being non-convex with quadratic constraints, in some situations, the approach described above resulted in infeasible solutions. In that case, we performed constraint checks and then used the initial solution to carry out suggest-and-improve (SAI) steps to reach a feasible solution.

5.2.3 Results and Discussion

This section presents the results of the proposed scheme by testing in MEL scenarios emulating realistic edge node environments and learning. We show the merits of the proposed solution compared to performing asynchronous learning with the heterogeneity unaware (HU) equal task allocation (ETA) in terms of staleness and learning. For the staleness, one of the the metrics will be maximum staleness as described in (5.6). In addition, we would like to introduce average staleness as shown in (5.14) which will give a measure of the mutual staleness between every two learners for all learners. The metric for evaluating the learning

performance is validation accuracy.

$$s_{avg} = \frac{1}{N} \sum_{n=1}^N |\tau_{c_{n,1}} - \tau_{c_{n,2}}| \quad (5.14)$$

Simulation Environment, Dataset, and Learning Model

The simulation environment considered is an indoor environment which emulates 802.11-type links between the edge nodes that are located within a radius of 50m. We assume that that approximately half of the nodes have the processing capabilities of typical computing devices such as desktops/laptops and the other half consists of industrial micro-controller types such as a Raspberry Pi. The employed channel model is summarized in Table 5.2.

As a benchmark, the MNIST dataset [92] is used to evaluate the proposed scheme. The training data comprises 60,000 28x28 pixel images contributing 784 features each. The ML algorithm tested is a deep neural network with the following configuration [784, 300, 124, 60, 10]. The input layer has 784 nodes for each feature and the output represents the number of classes (10 for each digit). The size of the resulting model is 8,974,080 bits, which is fixed for all edge nodes, and the forward and backward passes will require 1,123,736 floating point operations [87].

Table 5.2: Simulation parameters for testing the HA-Asyn with only time constraints

Parameter	Value
Attenuation Model	$7 + 2.1 \log(R)$ dB [91]
System Bandwidth B	100 MHz
Node Bandwidth W	5 MHz
Device proximity R	50m
Transmission Power P_k	23 dBm
Noise Power Density N_0	-174 dBm/Hz
Computation Capability f_k	2.4 GHz and 700 MHz
MNIST Dataset size d	60,000 images
MNIST Dataset Features \mathcal{F}	784 (28×28) pixels

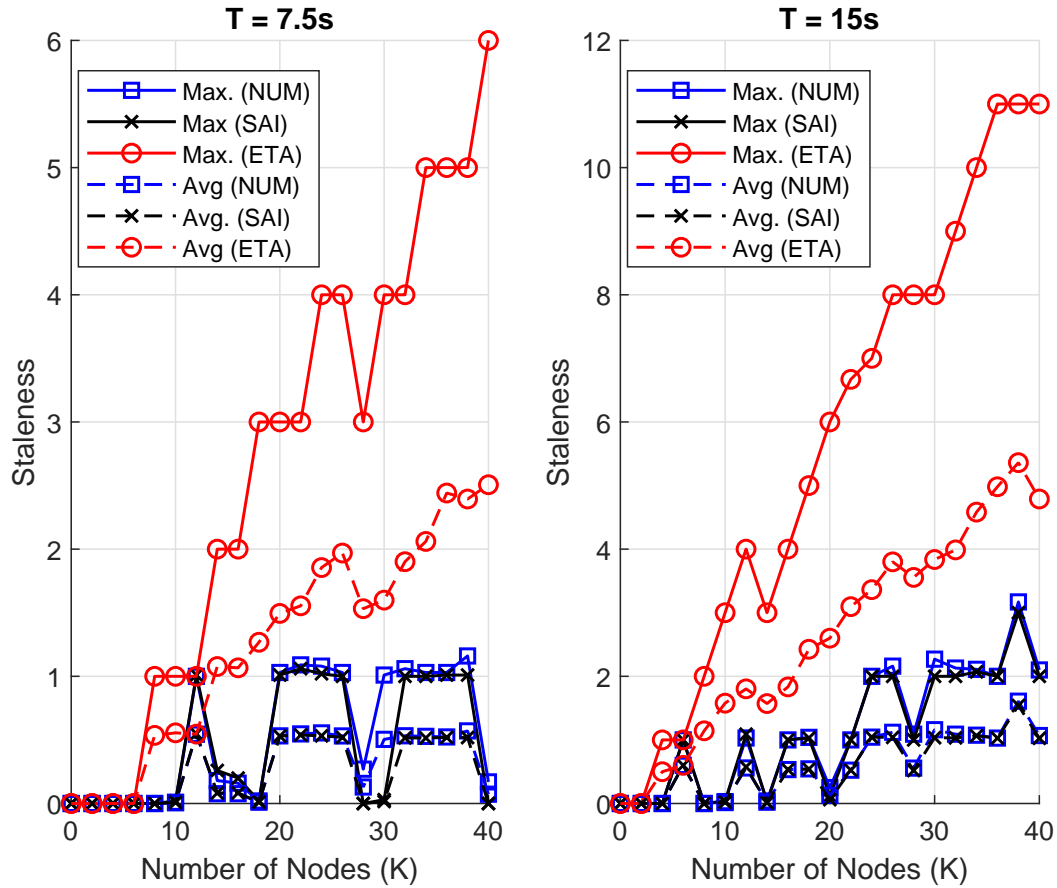


Figure 5.13: Maximum and Average Staleness vs K for $T = 7.5$ s and $T = 15$ s.

Staleness Analysis

Fig 5.13 shows the maximum and average staleness versus the number of learners K for global cycle times T of 7.5 s and 15 s. Comparisons are made among the solutions to our optimal batch allocation using both methods, optimizer-based/numerical (NUM) and SAI methods, and the ETA scheme as well. In general, the SAI based approach gives similar staleness to the numerical solution from the optimizer. The general trend is that as the number of updates τ_k increases, the staleness tends to increase with the exception when $T = 7.5$ s, the average maximum staleness of our proposed scheme does not exceed 1.2 as K increases. For example, our scheme with 20 users at $T = 7.5$ s gives a maximum staleness of 1 compared to 4 for ETA which is 400% higher and the average staleness is 1.5 compared

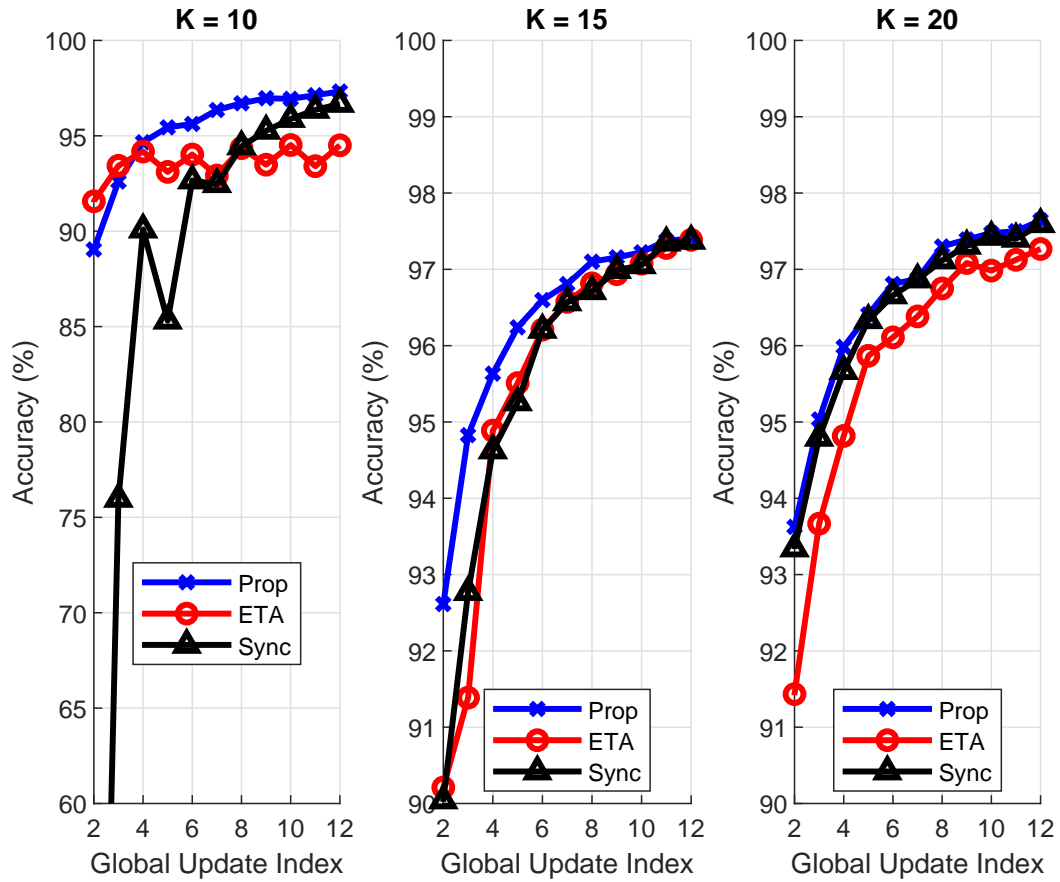


Figure 5.14: Validation accuracy progression after global update cycles for $K = 10, 15$ and 20 for $T = 15$ s

to 0.5 which is 300% higher. One curious aspect to note is that for certain specific number of learners or K , the asynchronous scheme is able to find an optimal solution where the staleness is zero. One such example is $K = 14$ for $T = 15$ s and $K = 18$ for $T = 7.5$ s.

Learning Accuracy

Figure 5.14 compares the validation accuracy progression of the proposed asynchronous scheme (Prop), ETA and the synchronous scheme in [87] (Sync). We test systems with 10, 15 and 20 learners, respectively, for $T = 15$ s. When $K = 10$, the proposed scheme achieves a validation accuracy of 95% within 4 updates or 1 minute of learning as compared to the synchronous scheme which requires 8 updates. In other words, we obtain a gain of 50%.

In contrast, ETA fails to achieve a 95% accuracy. An accuracy of 95% is achieved by our scheme within 3 updates with 15 users whereas the other schemes require 4 updates; which gives represents a gain of 25%. Moreover, our scheme achieves an accuracy of 97% within 8 updates whereas the other 2 methods require 10 global cycles leading to a gain of 25%.

A similar gain is achieved for $K = 20$ learners for the 95% accuracy mark. For the case of 97% validation accuracy, our scheme requires 7 updates whereas the ETA needs 11 cycles, representing a gain of of about 64%. On the other hand, the synchronous scheme requires 8 updates which translates to a gain of 12.5%. The gain appears marginal compared to the synchronous scheme because as the number of users increase, each learner has to process less data which means a larger number of synchronized updates can be done even in heterogeneous conditions. In contrast, the gain is significant compared to the ETA scheme because the staleness for ETA increases significantly versus K for a fixed global cycle T .

5.3 Chapter Summary

This chapter presented the results for the proposed HA schemes with only time constraints for the synchronous case (HA-Sync) and the asynchronous setting (HA-Asyn). It was shown that both problems are NP-hard QCILP. An analytical upper bound was derived for HA-Sync and an upper bound based on the SAI approach was derived for HA-Asyn. For both schemes, it was shown that the proposed solutions matched the results of the numerical solvers. In terms of performance, it was shown that the HA-Sync can provide up-to a 45% reduction in convergence time compared to the HU-sync. Furthermore, it was shown that the HA-Asyn performs better than the HU-asyn and in some cases, the HA-Asyn may provide a reduction in convergence time compared to HA-Sync as well. In the next chapter, we will propose solutions for both HA-Sync and HA-Asyn with dual time and energy constraints.

Chapter 6: MEL with Dual Time and Energy Constraints

In the previous chapter, we have proposed solutions for optimal task allocation in MEL such that the accuracy is maximized when there is a limit only on the global completion time constraint. However, most of the devices in edge networks are expected to be battery-operated. Though many papers make the assumption that only learners that are charging or with full battery will participate, this is not feasible in many settings. Therefore, learners will have limited energy, and thus may put a limit on the amount consumed. To this end, we propose solutions to maximize local updates while optimizing batch size allocation. We begin with the case when the number of updates are synchronized (HA-Sync) in Section 6.1 and follow-up with the asynchronous version (HA-Asyn) in Section 6.2.

6.1 Synchronous Task Allocation with Time and Energy Constraints

This section presents results for the HA-Sync scheme with dual global cycle time and local energy consumption constraint and has already been presented at the IEEE ICC 2020 Conference¹.

6.1.1 Formulation

Recall the MEL model described in Section 4.3, where the time taken t_k and the energy consumed e_k for one global update cycle is given by (6.1) and (6.2), respectively. Please note that once again, for the synchronous case, we set $\tau_k = \tau$, and the optimization variables are $\tau, d_k \forall k \in \mathcal{K}$.

$$t_k = C_k^2 d_k \tau + C_k^1 d_k + C_k^0 \quad \forall k \in \mathcal{K} \quad (6.1)$$

$$e_k = G_k^2 d_k \tau + G_k^1 d_k + G_k^0 \quad \forall k \in \mathcal{K} \quad (6.2)$$

¹The paper was presented at the 2020 IEEE International Communications Conference (IEEE ICC 2020) Workshop on Edge Machine Learning for 5G (EML5G) and will appear in Proceedings of the IEEE ICC 2020 [99].

The coefficients C_k^2 , C_k^1 , and C_k^0 , related to the completion time t_k , and the coefficients G_k^2 , G_k^1 , and G_k^0 , related to the energy consumption $e_k \forall k \in \mathcal{K}$, have been described in Sections 4.3.1 and 4.3.2, respectively.

It has been previously shown that maximizing the local iterations per global cycle can lead to a faster progression of the learning process [87]. Therefore, the objective is to allocate batches d_k such that we maximize τ . This has to be achieved while maintaining a global cycle time constraint T on each learner and not letting learner k consume more than its limit of E_k^0 J of energy per global cycle. As it can be observed, there exists a quadratic relationship among the optimization variables τ and d_k in t_k and e_k . Furthermore, due to τ and $d_k \forall k$ being non-negative integers, the resulting optimization problem is a QCILP as shown below:

$$\max_{\tau, d_k \forall k} \tau \quad (6.3)$$

$$\text{s.t.} \quad C_k^2 \tau d_k + C_k^1 d_k + C_k^0 \leq T, \quad \forall k \in \mathcal{K} \quad (6.3a)$$

$$G_k^2 d_k \tau + G_k^1 d_k + G_k^0 \leq E_k^0, \quad \forall k \in \mathcal{K} \quad (6.3b)$$

$$\sum_{k=1}^K d_k = d \quad (6.3c)$$

$$\tau \in \mathcal{Z}_+ \quad (6.3d)$$

$$d_k \in \mathcal{Z}_+, \quad \forall k \in \mathcal{K} \quad (6.3e)$$

$$d_k > d_l \quad \forall k \in \mathcal{K} \quad (6.3f)$$

Constraints (6.3a) and (6.3b) guarantee that the time and energy constraints are not violated for each learner, respectively. Constraint (6.3c) ensures that the orchestrator facilitates the learning of the complete dataset consisting of d data samples. Constraints (6.3d) and (6.3e) imply that τ and d_k 's are non-negative integers $\forall k$. Furthermore, there is an additional lower bound constraint on d_k 's $\forall k$ in (6.3f) to ensure that some learners are not completely

eliminated which may also effect accuracy. Note that the solutions of (6.3) having τ and/or all d_k 's being zero represent settings where MEL is not feasible. Thus, the program in (6.3) is a QCILP which is well-known to be NP-hard [89]. This problem can be solved numerically using interior point methods and many commercially available solvers do the job. However, in the next section, we propose an analytical-numerical solution based on a relaxation approach.

6.1.2 Proposed Solution

We simplify the NP-hard problem by relaxing the integer constraints in (6.3d) and (6.3e), and obtaining the final τ and d_k 's by taking the floor of the original solution. Hence, the relaxed problem can be represented as:

$$\max_{\tau, d_k \forall k} \tau \quad (6.4)$$

$$\text{s.t.} \quad C_k^2 d_k \tau + C_k^1 d_k + C_k^0 \leq T, \quad \forall k \in \mathcal{K} \quad (6.4a)$$

$$G_k^2 d_k \tau + G_k^1 d_k + G_k^0 \leq E_k^0, \quad \forall k \in \mathcal{K} \quad (6.4b)$$

$$\sum_{k=1}^K d_k = d \quad (6.4c)$$

$$\tau \geq 0 \quad (6.4d)$$

$$d_k \geq d_l, \quad \forall k \in \mathcal{K} \quad (6.4e)$$

Constraint (6.3e) has been eliminated after relaxation because $d_l \geq 0$ in (6.4e). Analytically, the matrices associated with the quadratic constraints in (6.4a) and (6.4b) are symmetric sparse matrices with 2 non-negative values each. However, they will have one positive and one negative eigenvalue which means the matrices are not positive semi-definite leading to a non-convex program. Consequently, we need to pursue methods that can provide solutions for a non-convex QCQP. Therefore, we will use the suggest-and-improve (SAI) approach [100]

by deriving upper bounds on the optimal variables and solution using Lagrangian analysis and then use a local optimizer to reach the optimal solution.

The equality constraint in (6.4c) can be written as the following two inequality constraints: $\sum_{k=1}^K d_k - d \leq 0$ and $-\sum_{k=1}^K d_k + d \leq 0$. In that case, the Lagrangian function of the relaxed problem is given by:

$$L(\mathbf{x}, \lambda, \gamma, \alpha, \bar{\alpha}, \omega, \nu) = -\tau + \sum_{k=1}^K \lambda_k (C_k^2 \tau d_k + C_k^1 d_k + C_k^0 - T) + \sum_{k=1}^K \gamma_k (G_k^2 \tau d_k + G_k^1 d_k + G_k^0 - E_k^0) + \alpha \left(\sum_{k=1}^K d_k - d \right) - \bar{\alpha} \left(\sum_{k=1}^K d_k - d \right) - \omega \tau - \sum_{k=1}^K \nu_k d_k \quad (6.5)$$

The Lagrange multipliers associated with the global cycle time and local energy constraints are given by λ_k and γ_k , respectively, $\forall k \in \mathcal{K}$. The Lagrange multipliers related to the two total task size constraint inequalities are given by $\alpha/\bar{\alpha}$, and ω/ν_k $k \in \mathcal{K}$ are the Lagrangian multipliers associated with the non-negative constraints of both sets of optimization variables τ and d_k , respectively.

Let us denote the set of optimization variables by $\mathbf{x} = [\tau \ d_1 \ d_2 \ \dots \ d_k \ \dots \ d_K]^T$ and the set of Lagrange multipliers by $\mathbf{\Gamma} = [\lambda, \gamma, \alpha, \bar{\alpha}, \omega, \nu]^T$, where $\lambda = [\lambda_1 \ \dots \ \lambda_k \ \dots \ \lambda_K]^T$, $\gamma = [\gamma_1 \ \dots \ \gamma_k \ \dots \ \gamma_K]^T$, and $\nu = [\nu_1 \ \dots \ \nu_k \ \dots \ \nu_K]^T$. :

Theorem 3 *The set of optimal Lagrange multipliers $\mathbf{\Gamma}^*$ can be obtained by solving the dual problem in the following semi-definite program (SDP):*

$$\begin{aligned} & \max_{\mathbf{\Gamma}} \quad \zeta & (6.6) \\ \text{s.t.} \quad & \begin{bmatrix} \mathbf{F}^2(\mathbf{\Gamma}) & \frac{1}{2}\mathbf{f}^1(\mathbf{\Gamma}) \\ \frac{1}{2}\mathbf{f}^1(\mathbf{\Gamma}) & f_0(\mathbf{\Gamma}) - \zeta \end{bmatrix} \succcurlyeq 0 \\ & \mathbf{\Gamma} \succcurlyeq 0 \end{aligned}$$

The functions of the Lagrange multipliers $\mathbf{F}^2(\boldsymbol{\Gamma})$, $\mathbf{f}^1(\boldsymbol{\Gamma})$ and $f_0(\boldsymbol{\Gamma})$, are defined in the proof.

Proof: Please refer to Appendix C for the proof. ■

A candidate solution is given by:

$$\hat{\mathbf{x}} = -\frac{1}{4}\mathbf{F}^2(\boldsymbol{\Gamma})^{-1}\mathbf{f}^1(\boldsymbol{\Gamma}) \quad (6.7)$$

In the case of a convex QCQP, the resulting solution will be optimal with zero duality gap, i.e. $\hat{\mathbf{x}} = \mathbf{x}$. In our case, because of the problem being non-convex, we have to use a simple local optimizer called the coordinate descent method to improve the candidate solution where $\mathbf{x}^* = \text{coordinate-descent}(\hat{\mathbf{x}})$ [100].

6.1.3 Results and Discussion

In this section, we show that our proposed SAI approach gives similar results to the numerical solvers. We also compare our HA scheme against the HU scheme in terms of the average number of achievable updates per global cycle and the progression of the validation accuracy after each global update index. Lastly, we study the impact of different amounts of energy caps on various devices.

Simulation Environment, Dataset, and Learning Model

The edge nodes performing ML are assumed to be in a cellular type environment and located within a distance of 500m. The devices are evenly distributed to emulate multi-core laptops, simple processors (in cell phones, road side units, smart meters, etc.), advanced micro-processors such as Raspberry Pi's and some of the less powerful microcontrollers such as the Arduino, respectively. Typically, such micro-controllers are attached to various IoT devices these days. The channel parameters and other specs are listed in Table 6.1. To test our proposed MEL paradigm, the well-known commonly used MNIST [92] is employed. The ML model trained is a deep neural network with 3 hidden layers consisting of 300, 124

Table 6.1: Simulation parameters for HA-Sync with dual time and energy constraints.

Parameter	Value
Cell Attenuation Model	$128 + 37.1 \log(R)$ dB [84]
Node Bandwidth (W)	5 MHz
Device proximity (R)	500m
Transmission Power (P_k)	23 dBm
Noise Power Density (N_0)	-174 dBm/Hz
Computation Capabilities (f_k)	$\sim \{6.0, 2.4, 1.4, 0.7\}$ GHz
MNIST Dataset size (d)	54,000 images
MNIST Dataset Features (\mathcal{F})	784 (28×28) pixels

and 60 neurons, respectively. The details of the resulting model sizes and complexities are discussed in [87].

The major contribution of this paper is to facilitate learning while satisfying the various energy constraints imposed by learners locally. If a device k consumes a total of E_k^0 joules (J) or in other words, E_k^0 watt-seconds in one global cycle, it translates to an energy consumption of $\frac{E_k^0}{3.6}$ milliwatt-hours (mWh). Since battery capacities are determined by their amperage (mAh) and voltage (V), it is easy to measure the battery drainage in mWh. For example, a device that uses up 200 J per global update cycle for 12 cycles will consume a total of 666.67 mWh. Now, let us consider a device that has a 12.5V battery rated at 4800mAh. The corresponding wattage is 60,000 mWH. In that scenario, after 12 cycles, $\frac{666.67}{60,000} * 100 = 1.11\%$ of the battery would have been drained by the complete learning process.

In our simulations, all learners have a total energy constraint such that the average energy constraint per learner per global cycle is E J, where the constraint for each learner can be in the interval $[E - 25, E + 25]$ J. For example, when the average energy is constrained by $E = 50$ J, it means that for any learner $k \forall k \in \mathcal{K}$, the total energy consumed per global cycle $E_k^0 \in [25, 75]$ J. Our experiments are performed for the set of values of $E = \{50, 100, 150, 200, 250\}$ J which would, for a 60WH battery, correspond to a drainage of $E = \{0.28, 0.56, 0.83, 1.11, 1.39\}\%$, respectively.

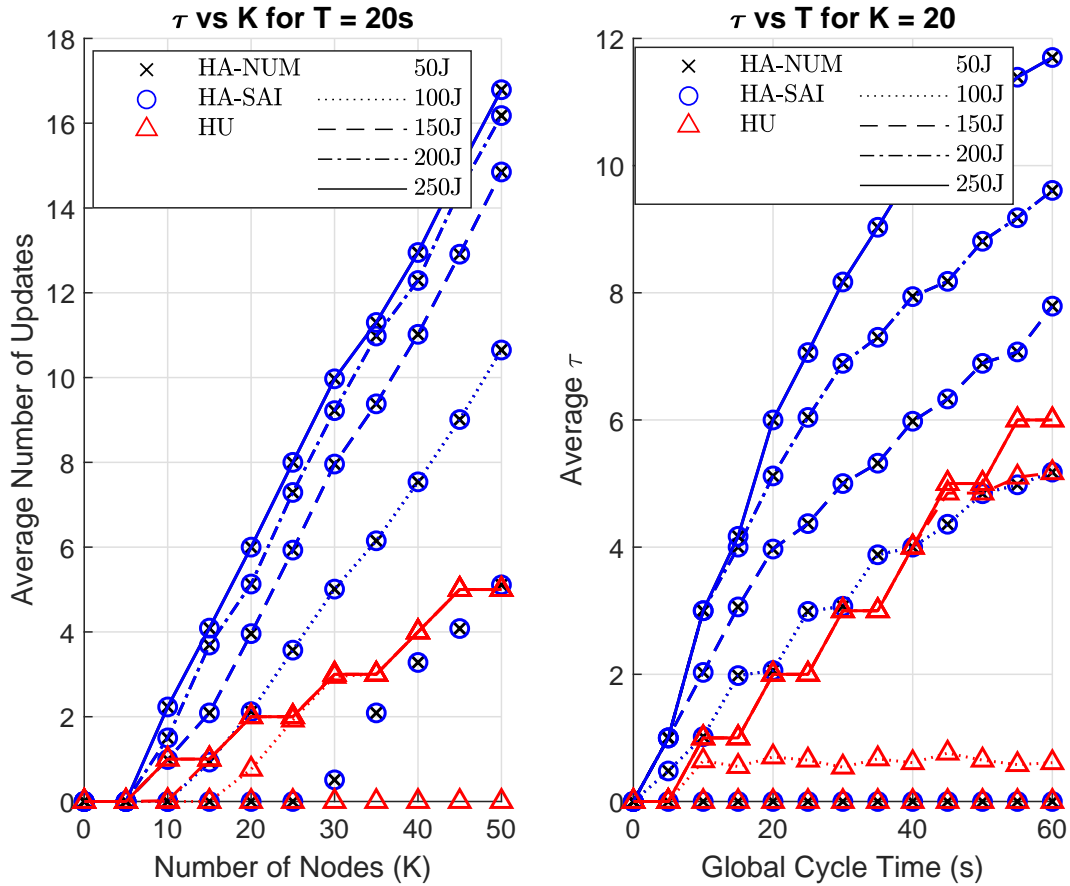


Figure 6.1: Achievable number of local update cycles for average energies of 50 to 250 J in steps of 50 J by all schemes (a) vs K for $T = 20$ s (b) vs T for $K = 20$.

Number of Local Updates

Figure 6.1 compares the achievable number of local updates τ per global cycle for our HA scheme versus the HU scheme. Figure 6.1a shows the impact of increasing the number of nodes while fixing $T = 20$ s for different energy levels. In contrast, Figure 6.1b shows the impact of increasing T for different energy levels for a system of $K = 20$ learners. It can be observed that the performance of the HA scheme in terms of the number of updates is much higher compared to the HU scheme. For example, we can see from both figures that with $K = 20$ for a global cycle time of $T = 20$ s, for $E = 150$ J, 200 J and 250 J, the possible number of updates for the HA scheme are 4, 5 and 6, respectively. On the other hand, the

HU scheme can only achieve a total of 2 updates for all levels of energy. This represents gains of 200%, 250% and 300%, respectively.

Interestingly, we notice that the performance of the HU scheme is capped for such stringent constraints and time and energy. We can observe from the left sub-figure that for the HA scheme, as the constraints on energy are relaxed, more updates are possible up to a certain level. The limiting factor is T . From the right sub-figure, we notice that for a fixed number of learners, as T is relaxed to higher values, allowing larger energy consumption can provide significant gains. For example, when $T = 40$ s, average E 's of 150 J, 200 J and 250 J will allow for 6, 8 and 10, local iterations, respectively. For the HU scheme, only 4 updates are possible which represents gains of 50%, 100% and 150%, respectively. The HA scheme gives an interesting trade-off between time and energy.

Learning Accuracy

Figure 6.2 depicts the progression of learning accuracy achieved by both, our HA scheme compared to the HU, right after each global update cycle for 12 cycles each of $T = 20$ s with $K = 20$. We examine the learning for average energy constraints E of 100 J, 150 J, 200 J and 250 J per learner per global cycle. For example, at $E = 250$ J, the HA technique achieves a 97% accuracy in 6 global cycles compared to 11 cycles needed for the HU scheme which represents a 45% reduction in time (about 100 s). For energy levels of 150 and 200 J, a reduction 37% in time is achieved. The performance of the HU scheme across varying energies is the same because only 2 updates are possible.

Figure 6.3 studies the impact of increasing the energy on the validation accuracy progression of the HA scheme. There is a huge jump in performance when the average energy per cycle is increased from 100 J to 150 J. Overall, we observe that increasing the energy with restricted time will allow for fine-tuning of the accuracy. In general, $E = 250$ J provides a 0.1% higher accuracy compared to 200 and 150 J except at global update cycle 11, which is an outlier for all cases. For example, $E = 250$ J will reach an accuracy of 97% in 6 global

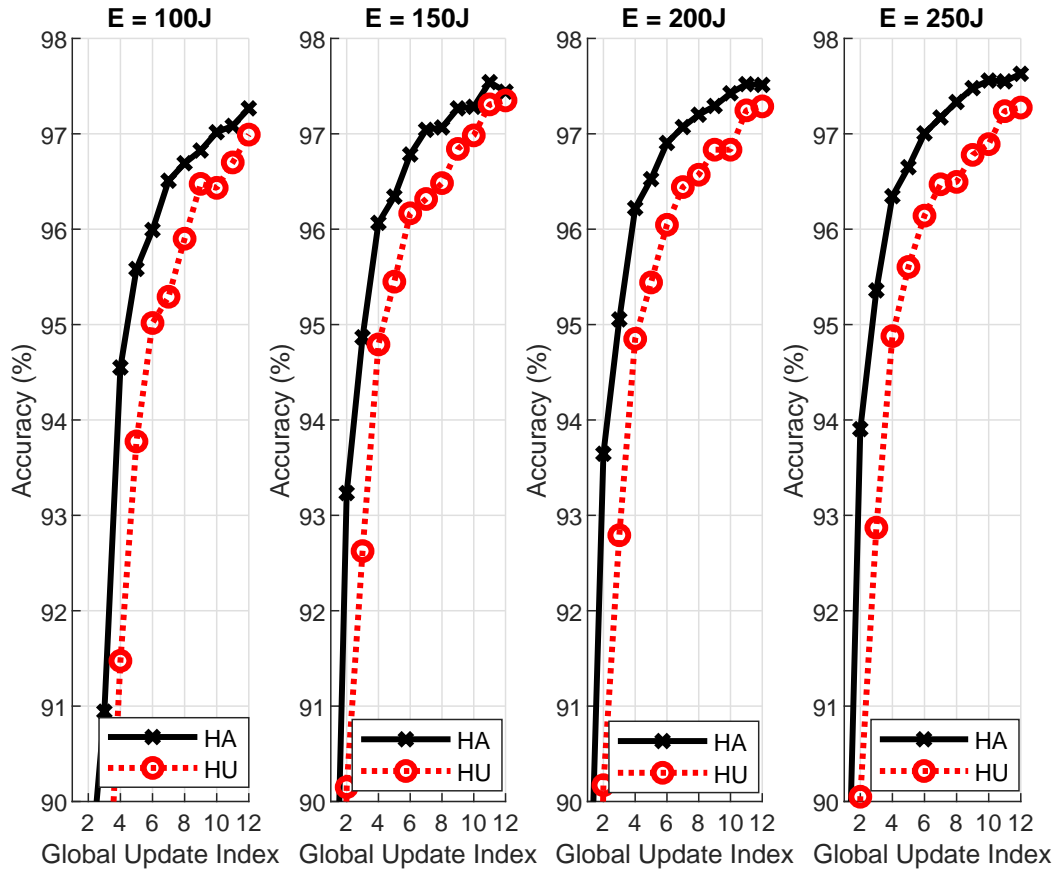


Figure 6.2: Learning accuracy progression after global cycles updates for $K = 20$

updates compared to 7 updates needed for $E = 150$ J and 200 J which reduces learning time by about 14%. For an accuracy of 97.25%, 250 J, 200 J, and 150 J require 7, 8 and, 9 cycles, respectively. This represents gains of 12.5% and 22.2%, respectively, for $E = 250$ J compared to 200 J and 150 J.

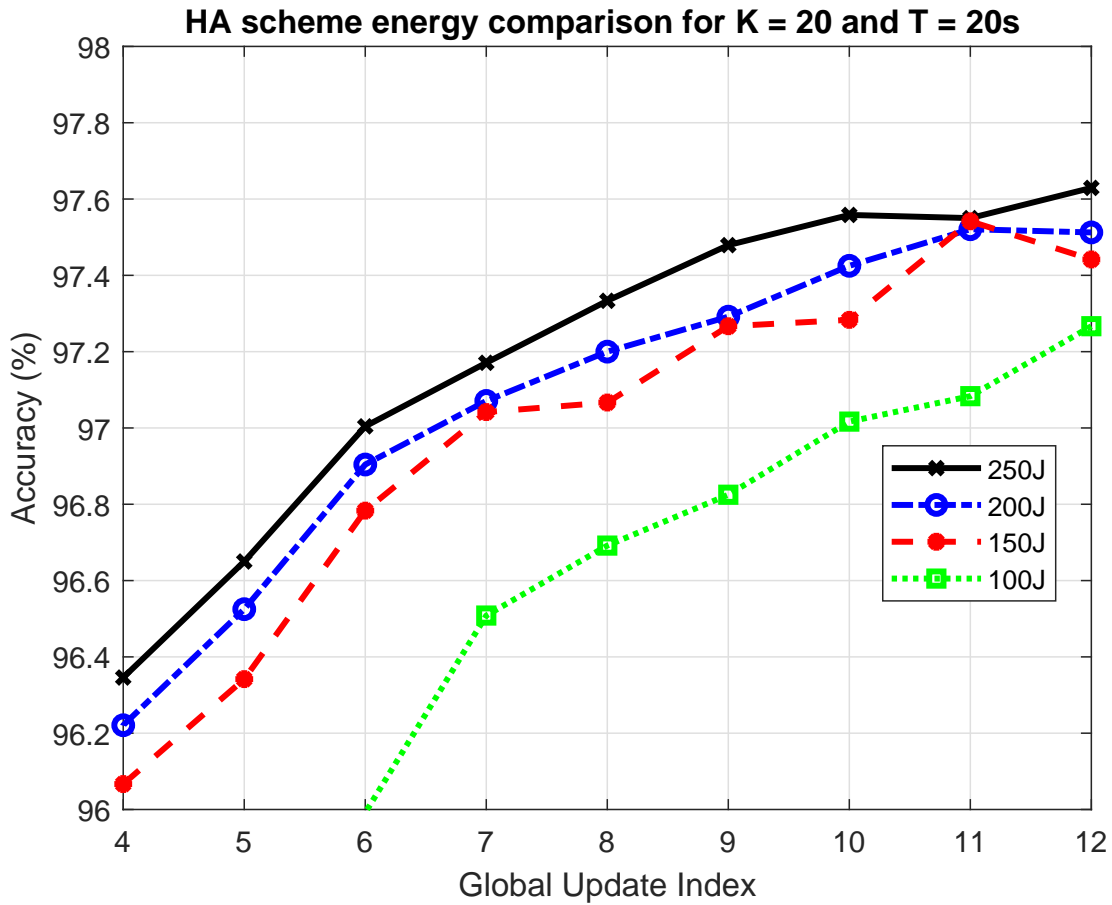


Figure 6.3: Learning accuracy progression after global cycles updates for $K = 20$ learners

6.2 Asynchronous Task Allocation with Time and Energy Constraints

This section presents results for the scenario when each learner k can perform τ_k local updates $\forall k \in \mathcal{K}$ and there are dual time and energy constraints on the global completion time and local energy consumption, respectively. The following sub-sections provide more details on the system model and problem formulation, discusses the proposed solution and presents the results. Part of this work has already been submitted to a journal².

²The material in this section forms part of a paper that has been submitted to a special issue on parallel and distributed ML/AI of the IEEE Transactions on Parallel and Distributed Systems.

6.2.1 Problem Formulation

In MEL, one approach is to have all K learners do $\tau_k = \tau$ local updates of the model parameters $\mathbf{w}_k \forall k \in \mathcal{K}$; this is the synchronous approach presented in Section 6.1 and [99]. The other approach is to optimize τ_k for each learner while controlling the difference among the number of local updates done by different learners. part of this work with only time constraints has been presented in Section 5.2. the term staleness $s_{k,l}$ has been defined as the difference between the number of updates done by two arbitrary learners k and l in the following way:

$$s_{k,l} = \tau_k - \tau_l \quad \text{if} \quad \begin{cases} k \ \& \ l \in \mathcal{K} \mid l \neq k \\ \tau_k \geq \tau_l \end{cases} \quad (6.8)$$

It has been previously shown that maximizing the local iterations per global cycle can lead to a faster progression of the learning process [87,99]. On the other hand, for asynchronous approaches, it has been shown that accuracy can be optimized by controlling the staleness [88,97,98]. Although our model is different to the models in [88,98], Lemma 1 shows the way to maximize accuracy for our proposed HA-Asyn with dual time and energy constraints.

Lemma 1 *Jointly controlling the staleness $s_{k,l} \forall k \in \mathcal{K} \ \& \ l \in \{\mathcal{K} \mid l \neq k\}$ while maximizing the minimum number of updates $\min(\tau_k)_{k \in \mathcal{K}}$ will minimize the global loss of the proposed HA-Asyn MEL.*

Proof: Please refer to Appendix D for the proof. ■

Therefore, the objective is to allocate batches d_k such that we maximize $\min(\tau_k)$ for $k \in \mathcal{K}$ while minimizing $s_{k,l} \forall k$. However, this problem will be non-tractable and difficult to solve. A more tractable way to achieve these objectives is to maximize the average of the local updates τ_k while controlling the staleness $s_{k,l} \forall k$. In this way, we can increase the number of local updates by the worst performing learner while controlling the model staleness. Based on this, the optimization variables are τ_k and d_k , and we can study the impact of different values of staleness by an additional constraint $s_{k,l} \leq c$.

In addition to this, the optimization needs to be done such that the global cycle is completed before time T and does not violate the energy consumption limit E_k^0 (J) per global cycle of any learner k . It is observable that the relationship between the optimization variables in the global cycle time and local energy consumption constraints will be quadratic in τ_k and d_k . Moreover, due to τ_k and $d_k \forall k$ being non-negative integers, the resulting problem is a QCILP as shown:

$$\max_{\tau_k, d_k \forall k} \frac{1}{K} \sum_{k=1}^K \tau_k \quad (6.9)$$

$$\text{s.t.} \quad C_k^2 \tau d_k + C_k^1 d_k + C_k^0 \leq T, \quad \forall k \in \mathcal{K} \quad (6.9a)$$

$$G_k^2 d_k \tau + G_k^1 d_k + G_k^0 \leq E_k^0, \quad \forall k \in \mathcal{K} \quad (6.9b)$$

$$|\tau_k - \tau_l| \leq c \quad \forall k, l \in \mathcal{K} \mid l \neq k \quad (6.9c)$$

$$\sum_{k=1}^K d_k = d \quad (6.9d)$$

$$\tau_k \in \mathcal{Z}_+, \quad \forall k \in \mathcal{K} \quad (6.9e)$$

$$d_k \in \mathcal{Z}_+, \quad \forall k \in \mathcal{K} \quad (6.9f)$$

$$d_k > d_l, \quad \forall k \in \mathcal{K} \quad (6.9g)$$

Constraint (6.9a) and (6.9b) guarantee that all learners k satisfy the global cycle time constraint T and their energy consumption limit constraint $E_k^0 \forall k \in \mathcal{K}$, respectively. Constraint (6.9c) ensures that the staleness does not exceed a desired amount c . We will test for multiple values of the staleness and report the results later in section 6.2.4-6.2.5. The assurance that the orchestrator will learn on the complete dataset \mathcal{D} is given by (6.9d). Lastly, constraints (6.9e) and (6.9f) ensure that both optimization variables are non-negative integers whereas constraint (6.9g) is a lower bound d_k to ensure all learners are allocated a reasonable batch size. Note that the a lower bound of zero represents the case where d_k can

take any positive value. Solutions of (6.9) where τ_k or d_k is zero for any k represents a setting where learner k cannot participate in the learning process and MEL may be sub-optimal.

In contrast to the problem in (5.8) in Section 5.2 representing the HA-Asyn approach with only time-constraints, the formulation in (6.9) is different in that the objective of maximizing the average number of updates across K learners is distinct. At the same time, instead of minimizing the staleness, we control it by putting caps on the maximum value of $s_{k,l} \forall k \in \mathcal{K} \ \& \ l \in \{\mathcal{K} \mid l > k \ \forall k\}$. The advantage of this approach is that it eliminates the need for an upper bound on $d_k \forall k \in \mathcal{K}$ and more importantly, the need for an equality constraint which makes finding feasible solutions a lot more difficult. Similar to the previous formulations, the program in 6.9 is an NP-hard QCILP. Therefore, in the next section, we propose a two-step solution based on relaxations and the SAI framework.

6.2.2 Proposed Solution

Instead of applying the SAI technique directly on the asynchronous problem in (6.9), we first propose to solve the problem by getting candidate solutions for d_k and τ_k from the synchronous problem in [99] by setting $\tau_k = \tau \forall k \in \mathcal{K}$. In the next step, we obtain the solution to (6.9) by applying the improve step using the candidate solutions as the initial values. The reason for doing this is because a system of K learners would produce at least $\binom{K}{2}$ constraints. For example, an MEL system of 100 learners will result in 4950 mutual staleness bounds. It was found that applying SAI directly to (6.9) does not work but applying the suggest step to the synchronous problem in [99] and the improve step to our problem provides solutions that converge. In the last step, the real values of the obtained τ_k 's and d_k 's are floored to get the integer values.

The problem can be simplified by replacing τ_k 's with the optimization variable τ and

relaxing the integer constraints in (6.9e and (6.9f) as follows:

$$\max_{\tau, d_k \forall k} \tau \quad (6.10a)$$

$$\text{s.t.} \quad C_k^2 d_k \tau + C_k^1 d_k + C_k^0 \leq T, \quad \forall k \in \mathcal{K} \quad (6.10b)$$

$$G_k^2 d_k \tau + G_k^1 d_k + G_k^0 \leq E_k^0, \quad \forall k \in \mathcal{K} \quad (6.10c)$$

$$\sum_{k=1}^K d_k = d \quad (6.10d)$$

$$\tau \geq 0 \quad (6.10e)$$

$$d_k \geq d_l, \quad \forall k \in \mathcal{K} \quad (6.10f)$$

Note that $\frac{1}{K} \sum_{k=1}^K \tau_k = \tau$ when $\tau_k = \tau \forall k \in \mathcal{K}$. The non-negative integer constraint on d_k 's in (6.9f) has been relaxed and can be covered by (6.10f). Constraint (6.9f) has been eliminated after relaxation because $d_l \geq 0$ in (6.10f). Therefore, the problem is now in the form of the QCQP of the synchronous problem in (6.10) in Section 6.1.2. If the steps described in Section 6.1.2 are followed, the solution obtained will be for the synchronous problem. Consequently, we propose a novel two-step procedure to obtain the solution to (6.9).

We can follow the steps of Lagrangian relaxation described in Appendix C to obtain the candidate solution $\hat{\mathbf{x}}$ as shown in (6.7). Then, for the proposed asynchronous approach, the optimal solution can be obtained by applying the local optimizer coordinate descent (CD) to the problem in (6.9 with the relaxed constraints in (6.10e) and (6.10f) and after replacing τ with $\tau_k \forall k \in \mathcal{K}$ in (6.10e) and the additional constraint in (6.9c). The complete steps followed by the orchestrator over multiple global cycles are summarized in Algorithm 2.

Algorithm 2 Process at the Orchestrator

Input: T, d, d_l, K **Output:** \mathbf{w}

Initialize \mathbf{w} and set the flag $STOP \leftarrow \mathbf{FALSE}$

- 1: **while not** $STOP$ **do**
- 2: *In Parallel:* Receive P_{ko}, h_{ko}, f_k , and e_k^0 from $k \in \mathcal{K}$
- 3: Solve (6.7) to obtain $\hat{\tau}, \hat{d}_k$
- 4: Transform (7.9) by setting $\tau_k > 0 \forall k \in \mathcal{K}$ in (6.10e) and removing (6.10f)
- 5: Get τ_k and d_k by applying CD to (7.9) using $\hat{\tau}, \hat{d}_k$
- 6: *In Parallel:* Send $\lfloor \tau_k \rfloor, \lfloor d_k \rfloor$ to each learner $k \in \mathcal{K}$ ³
- 7: *In Parallel:* Send \mathbf{w} to each learner $k \in \mathcal{K}$
- 8: *In Parallel:* After τ_k local updates, receive $w_k \forall k \in \mathcal{K}$
- 9: Obtain \mathbf{w} using (4.5)
- 10: **if** *STOPPING CRITERIA REACHED* **then**
- 11: Set $STOP \leftarrow \mathbf{TRUE}$
- 12: **end if**
- 13: **end while**
- 14: **return** \mathbf{w}

6.2.3 Test Setup for HA-Asyn with Dual Time and Energy Constraints

It is assumed that the learners comprise a combination of the following: laptops with multi-core processors, smart phones simple processors, advanced micro-controllers such as the Raspberry Pi, and very simple micro-controllers such as the Arduino. The learners are co-located within 500 meters in a cellular type environment and may be mobile. The channel parameters and other specs are listed in Table 6.2 To test our proposed MEL paradigm, a fully-connected deep neural network consisting of 300, 124, and 60 neurons is used to train on the MNIST dataset [92]. For detailed descriptions on how to obtain the model size and computational complexity, the readers are referred to [87].

The two major additional contributions of this paper are the study of different levels of caps on the energy consumed per global cycle per learner k and the impact of different values of staleness capped by c . We plot the validation accuracy related metrics for values

³In PL, the orchestrator sends d_k samples after randomly shuffling its dataset whereas it sends d_k in FL and the learner chooses $\min(d_k, d_k^{max})$ data points where d_k^{max} is learner k 's dataset size.

Table 6.2: Simulation parameters for HA-Asyn with dual time and energy constraints

Parameter	Value
Wi-Fi Attenuation Model	$7 + 2.1 \log(R)$ dB [91]
Cell Attenuation Model	$128 + 37.1 \log(R)$ dB [84]
Channel Bandwidth (W)	5 MHz
Device proximity (R)	500m
Transmission Power (P_k)	23 dBm
Noise Power Density (N_0)	-174 dBm/Hz
Computation Capability (f_k)	$\sim \{6.0, 2.4, 1.4, 0.7\}$ GHz
MNIST Dataset size (d)	60,000 images
MNIST Dataset Features (\mathcal{F})	784 (28×28) pixels

of staleness of up-to $c = 5$, because it was found that having a higher $c > 5$ does offer any improvements.

As for the constraint on the local energy consumption $E_k^0 \forall k \in K$, it is possible that devices will have wildly varying consumption limits. However, to quantify the impact of these limits, we define an average energy consumption per global cycle across all learners E (J) and E_k^0 (J) in any global cycle varies by σ_k^0 (J) such that $E_k^0 = E \pm \sigma_k U \forall k \in \mathcal{K}$ where $U \sim \mathcal{U}(0, 1)$ (J). To put these numbers into perspective, modern batteries are rated in terms of voltage (V) and milliampere-hours (mAH). An average consumption of 20 J per global cycle for 10 cycles would imply a total consumption of 200 J. For a battery rated at 5V, this represents a consumption of 11.1 mAH which for a 2000 mAH battery, represents 0.36% of the maximum load.

In the following sub-sections, we present results for an MEL system comprising $K = 20$ learners tested for global cycle times of $T = 5$ s to $T = 40$ s in steps of 5 s and for average energy values of $E = \{10, 20, 30\}$ J with $\sigma_k^0 = 2.5$ J. The results are discussed for our proposed HA asynchronous (HA-Asyn) scheme with PL and compared to the HA synchronous (HA-Sync) scheme in [87] and the performance that would have been achieved if the HU random equal task allocation approach was used (HU-Sync/Asyn) such as the one described in [76].

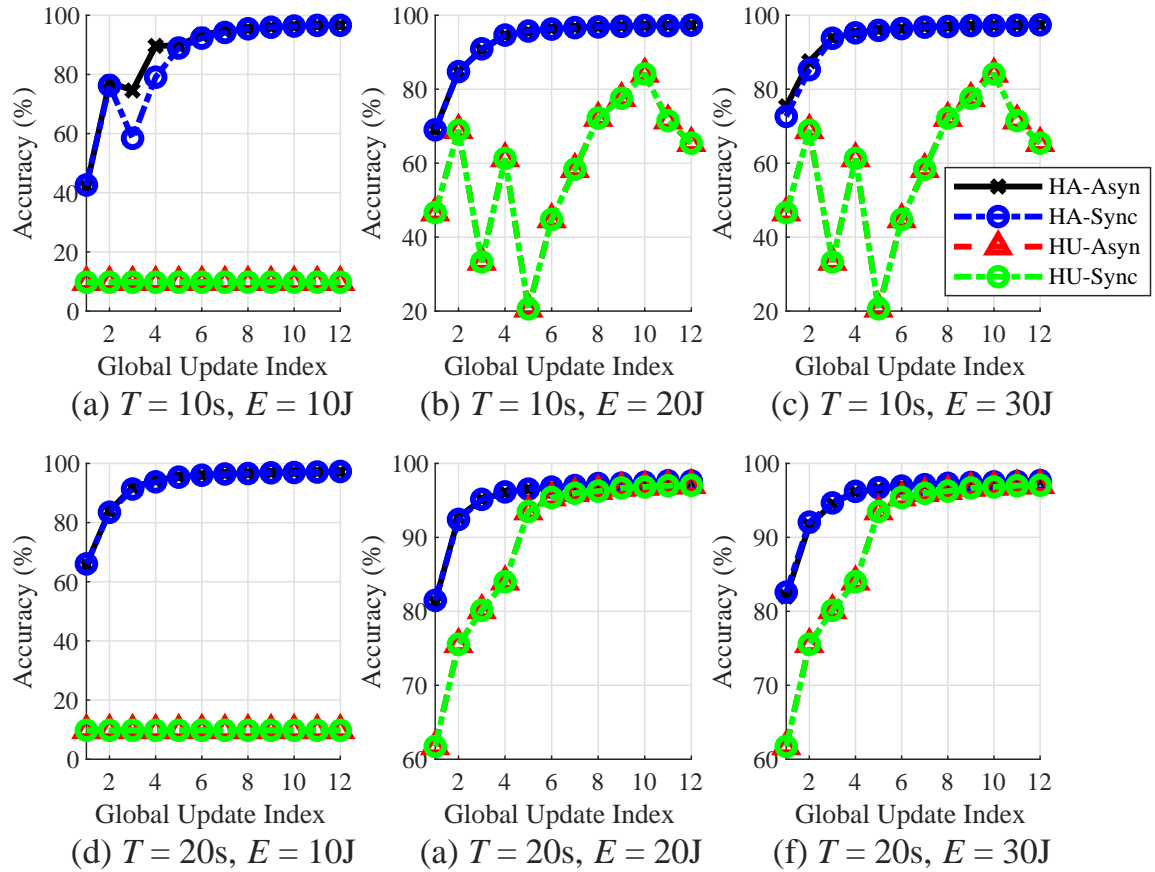


Figure 6.4: Final validation accuracy achieved after a total of 12 global epochs with various training times for an average device energy consumption of 10 J.

6.2.4 Convergence Proof Results

Figure 6.4 shows the plots for learning accuracy for the synchronous case only where HA-Sync represents solution obtained using the approach in [99] and HA-Asyn represents solutions to problem (6.10) with c set to zero. The HU-Sync/Asyn plots represent the HU solution for both approaches. As observable, we have confirmed that our solution in this paper converges to the synchronous case. Moreover, for extremely low values of energy and time (10 J and 10 s, respectively), the proposed approach works better than the approach to solve the synchronous problem. Last but not least, for the synchronous case ($c = 0$), the HA scheme always works better than the HU scheme where the HU scheme fails to converge on several

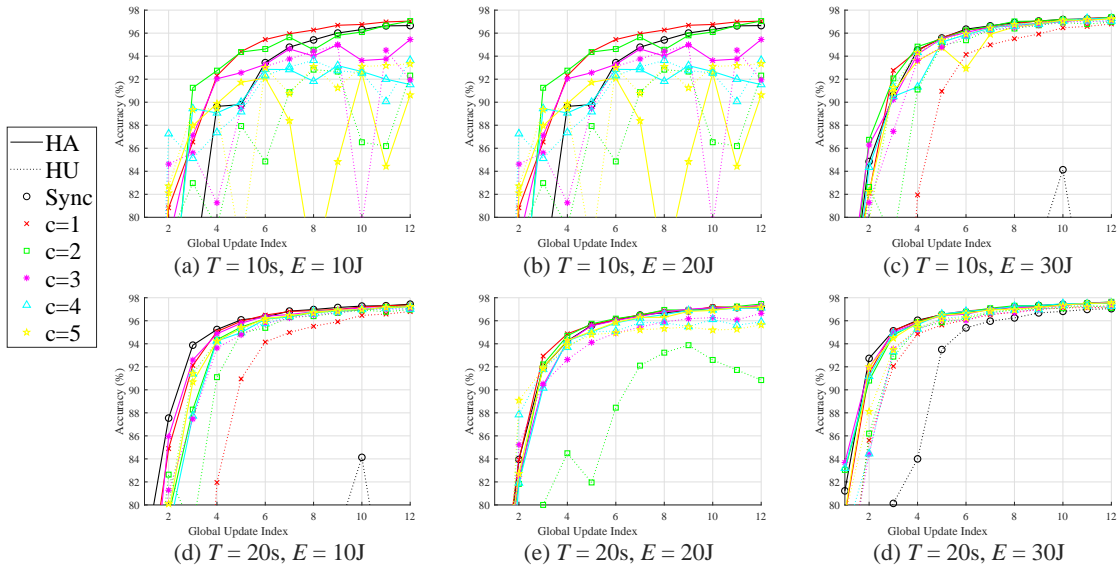


Figure 6.5: Accuracy progression for the cases when $T = 10$ s and 20 s and for average energy values of $E = \{10, 20, 30\}J$.

occasions.

Figure 6.5 plots the learning accuracy progression for different values of staleness (including the synchronous case with $c = 0$) with varying values of T and E for both, the HA and HU schemes. The purpose of these figures is to demonstrate the importance of utilizing HA schemes, especially when the resources are limited. Observe that the HA schemes generally converge faster and reach a higher level of final validation accuracy. The plots also demonstrate the usefulness of having a staleness aware asynchronous scheme.

For example, as we can see from Figure 6.4a that when $T = 10$ s and $E = 10$ J, the HA-Asyn with $c = 1$ and $c = 2$ requires 5 global updates to reach a 94% accuracy whereas the HA-Sync requires 7 updates, a reduction in time of 29%. Similarly, 6 updates are required to achieve a 95% accuracy with $c = 1$ whereas the HA-Sync needs 8 updates representing a reduction of 25%. Because this is not clear from Figure 6.4 in general, the next subsection demonstrates these gains more clearly using bar charts.

6.2.5 Validation Accuracy Results

Low and Medium Energy Regions

In this part, we focus on some specific metrics such as final validation accuracy after a set number of global updates or the number of updates required to reach an accuracy threshold. These results are presented for all schemes for the case where the devices have a low average energy consumption of about 10 J and a higher consumption of 20 J per global cycle for each learner for all global cycle times. This study is important because fewer updates for a given global cycle time constraint results in lower total training time for a given time constraint.

Figures 6.6a and 6.6b present the final validation accuracy achieved after a total of 6 global updates with different total training times for the case when the average energy consumption limit per cycle per learner $E = 10$ J and $E = 20$ J, respectively. The final validation accuracies for those two settings after 12 global cycles is given in Figure 6.6c and Figure 6.6d, respectively. The case when $c = 0$ implies the HA/HU-Sync scheme and cases

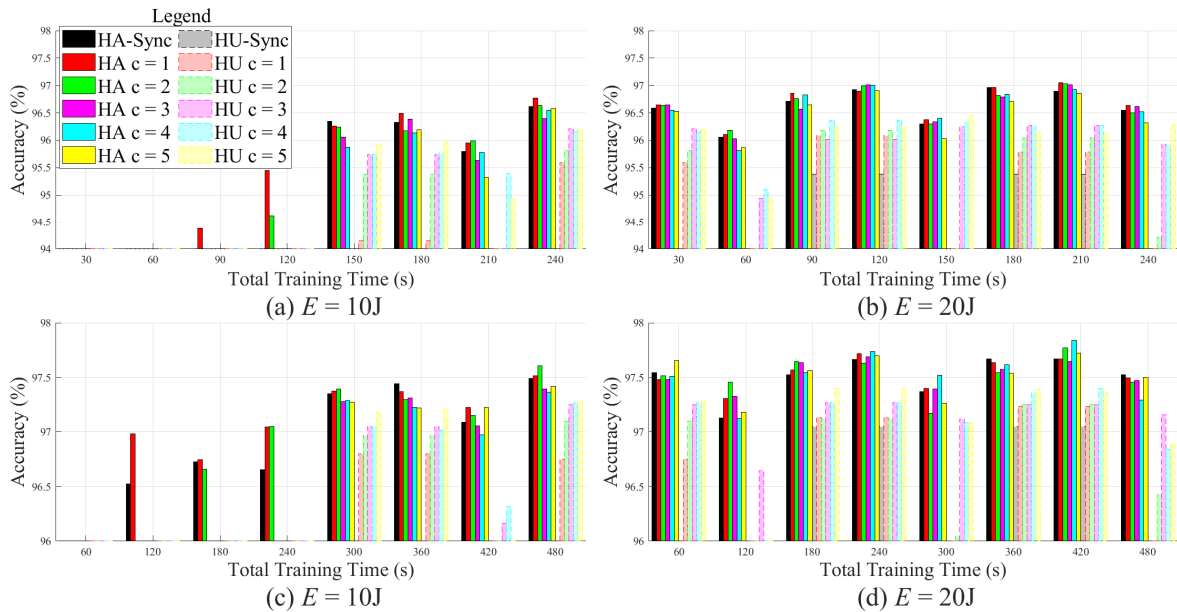


Figure 6.6: Final validation accuracy values after 6 and 12 global updates, respectively, for various training times for average energy consumption limits $E = 10$ J and $E = 20$ J, respectively.

when $c > 0$ represent the HA/HU-Asyn schemes.

Overall, it can be seen that the HA schemes provide a better performance compared to the best performance of the HU schemes. In most regions, the best performance is provided by the HA-Asyn schemes. For example, in the low resource region with $T = 15$ and 20 s and $E = 10$ J, after 6 global updates the HA-Asyn with $c = 1$ provides a best accuracy of 94.4% and 95.4%, respectively, whereas the HA-Sync fails to reach 94%. The HA-Asyn with $c = 2$ crosses 94.5% for $T = 20$ s. For a higher energy of 20 J, the HA-Asyn scheme with $c = 1-3$ is able to provide an accuracy of 97% when $T = 20$ s and for c values of 2-4 for $T = 35$ s. After 12 global cycles, for $E = 10$ J and $T = 10$ s and 20 s, the HA-Asyn schemes with $c = 1$ and $c = \{1, 2\}$, respectively, are able to provide a significantly higher final accuracy with a difference of more than 0.4%. Similarly when $E = 20$ J, for global cycle times of $T = \{5, 10, 15, 20, 25\}$ s, the HA-Asyn scheme provides the best validation accuracy with corresponding staleness $c = \{5, 2, 2, 4, 4\}$. with the difference ranging from 0.1-0.3%.

Figure 6.7 displays these differences clearly by showing the number of updates required to reach a 95.5% accuracy for various global cycle times T for $E = 10$ J (Figure 6.7a) and $E = 20$ J (Figure 6.7b). For the same settings, the number of updates required to achieve an accuracy of 97.3% are plotted in Figures 6.7c and Fig 6.7d, respectively. For example, when $E = 10$ J and $T = 10$ s, a final accuracy of 95% can be achieved in 7 global updates with HA-Asyn as opposed to 9 updates with the HA-Sync, representing a reduction of 22.2%. This means that the MEL system needs to train for 20 s less to reach the same accuracy. Similarly, after 12 global update cycles, an MEL system with $E = 10$ J can reach an accuracy of 97.3% with HA-Asyn ($c = 1, 2, 3$) whereas the HA-Sync cannot reach that level of accuracy. Moreover, for $T = 25$ s, the HA-Asyn ($c = 3, 4$) can 97.3% in 10 updates as compared to 11, representing a reduction of 9.1% or 25 s.

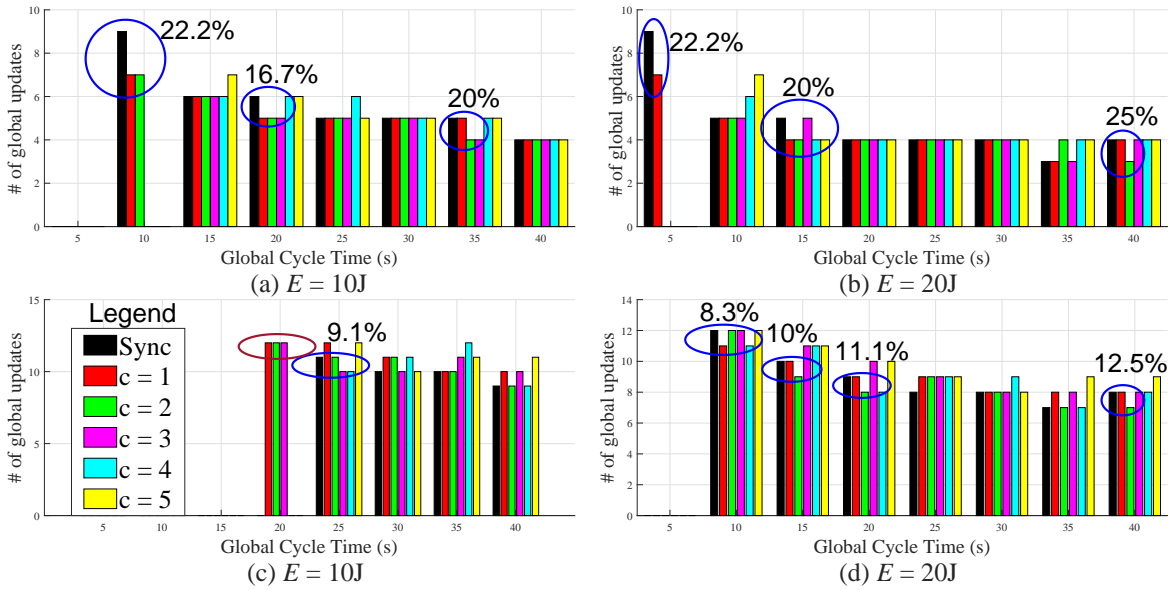


Figure 6.7: Number of global updates needed to reach a validation accuracy of 95.5% for: (a) $E = 10 J$ and (b) $E = 20 J$ and the number of updates needed to reach an accuracy of 97.3% for: (c) $E = 10 J$ and (d) $E = 20 J$.

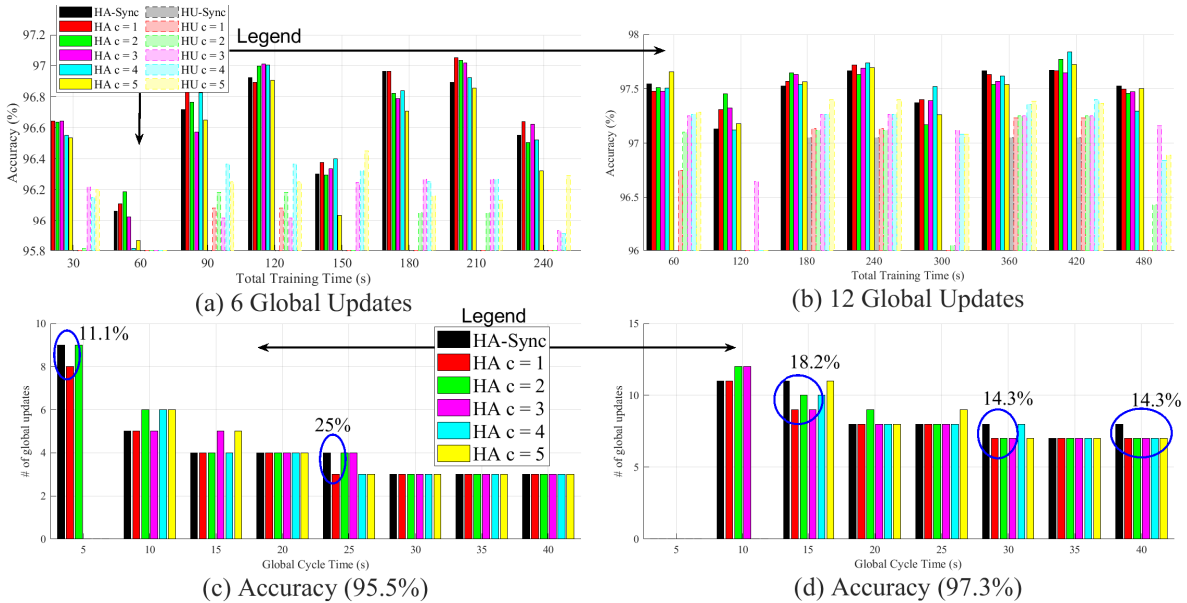


Figure 6.8: Results for an MEL system with average energy $E = 30 J$. Final validation after (a) 6 global updates and (b) 12 global updates, and number of global updates to reach an accuracy threshold of (c) 95.5% and (d) 97.3%.

High Energy Region

Figure 6.8 presents the results for the high energy region of $E = 30 J$. Figures 6.8a and 6.8b, display the final validation accuracy achieved after 6 and 12 global updates, respectively.

From Figures 6.8c and 6.8d, we can see that it takes 11.1% less global cycles to reach an accuracy of 95.5% for $T = 5$ s and a reduction of 18.2% is achieved to reach an accuracy of 97.3% when $T = 15$ s. In the case when both energy and time are in abundance, the HA-Asyn reduces the number of global cycles by 14.3% though the HA-Sync reaches a slightly higher final accuracy as after 12 updates as observable from Figure 6.8c. In most cases, the HA-Asyn reaches a higher final validation accuracy with an improvement of up-to 0.2%.

6.2.6 Discussions

Despite these gains, it can be seen that in some situations, the HA-Sync approach provides the best results. For example, when $T = 25$ s and $E = 20$ J, the least time to reach an accuracy of 97.2% is by the HA-Sync scheme. It also reaches an accuracy of 95.5% and 97% with the same number of updates for values T in the range 25 – 35 s for $E = 10$ J and 20 J. Moreover, the best final validation accuracy after 12 global cycles is provided by HA-Sync for $E = 10$ J when $T = 30$ s and also for $E = 20$ J when $T = 30$ s and $T = 40$ s.

The reason for this is that when the resources are quite low, a straggler or a very bad performing learner brings down the whole system for the HA-Sync which results in low convergence rates and final accuracy. On the other hand, the HA-Asyn scheme provides the flexibility for better performing learners to improve accuracy, but only when the staleness is low. When one of the resources is higher, then increasing the staleness further improves the performance because the minimum number of local updates are still close to the HA-Sync.

However, when both resources are in the medium range, the HA-Sync can allow for enough local updates per global cycle such that it overtakes the adverse impacts of a high staleness among gradients. Lastly, when both time and energy are plenty, the HA-Asyn starts to provide better results because the average number of updates done by the HA-Sync are capped whereas the HA-Asyn can allow for a higher average number of local updates which masks the adverse impacts of staleness. Based on these observations, recommendations will be provided in Section 8.1 on how to choose the best scheme with the correct parameters.

Chapter 7: Optimal Task Allocation

The work in this chapter discusses optimal task allocation for MEL when there is a time constraint and when the number of iterations are synchronized among all learner. Hence, it is an extension of the work done on synchronous task allocation presented in Section 5.1 where we simply maximized the number of local updates per global cycle. This in turn led to the maximum possible total ML model updates, which generally reduces the loss and improves the accuracy.

In contrast, the aim of the work done in this chapter is to jointly optimize the number of total local updates by adaptively optimizing the number of local updates for every global aggregation step until a pre-set total training time is exhausted. This work is important because recent works have related edge DL models directly to the progression of the ML model global loss. Our MEL model, though HA, has not related the physical parameters directly to the loss/accuracy so far, but was built on a solid hypothesis that facilitating more total updates given a constraint on global completion time would increase accuracy [85].

Therefore, in this part, we propose an HA model for MEL with the optimization variables linked directly to the bounds on the global ML model loss; this work has already been submitted to a conference¹.

7.1 Problem Formulation

In this work, the objective is to optimize the task allocation, i.e. the distributed batch sizes d_k for each learner k and the associated τ updates to be performed locally per global cycle. Over a total of L updates, the goal is to simultaneously minimize the global DL loss and thus, maximize the accuracy. To this end, the problem is formulated as a loss-function minimization problem over the optimization variables L , τ and d_k .

Consider that after every τ local iterations, a global aggregation will be performed and

¹This section is part of a paper titled “Optimal Task Allocation for Mobile Edge Learning” which was submitted to the 2020 IEEE Global Communications Conference (IEEE GLOBECOM 2020). It is also available as a pre-print online [101].

a total of G global aggregations are performed. In any global update cycle, between any learner $k \in \mathcal{K}$, and the orchestrator O , there will be one communication round each and τ local updates. For now, to facilitate the analysis, let us assume that L is an integer multiple of τ such that $L = G\tau$ and that the communication and computation related parameters remain unchanged over the complete training process. In that case, each learner needs time $t_k^C \forall k \in \mathcal{K}$ for one local update and $t_k^S + t_k^R$ for the g^{th} global aggregation for $g = 1, \dots, G$. Overall, L local updates and $G = L/\tau$ global updates will be performed. Then, the total time consumed by learner k denoted by $t_k \forall k \in \mathcal{K}$ can be expressed as:

$$t_k = L \left(t_k^C + \frac{t_k^S + t_k^R}{\tau} \right) \quad (7.1)$$

Later on, we will show how the values of τ and d_k for each set of τ local ML iterations in one global cycle g will be re-calculated according to the latest channel parameters and computational capabilities. The total training time within which the process should be completed is given bounded by T . Because the τ iterations occur in parallel over the K learners, we need the time for the most time-consuming learner to be less than T such that $\max(t_k) \leq T$. Alternatively, it is sufficient for this condition to hold that $t_k \leq T \forall k \in \mathcal{K}$.

This point differentiates our work from that of [77] where we actually capture the time consumed by parallel local update processes rather than a generic resource consumption model. Therefore, the optimization problem can be written as:

$$\begin{aligned} & \min_{L, \tau, d_k \forall k} F(\mathbf{w}[L]) & (7.2) \\ \text{s.t.} & \quad L \left(C_k^2 d_k + \frac{C_k^1 d_k + C_k^0}{\tau} \right) \leq T, \quad k \in \mathcal{K} \end{aligned}$$

The constants C_k^2, C_k^1 , and C_k^0 can be defined as:

$$C_k^2 = \frac{\mathcal{C}_m}{f_k} \quad (7.3a)$$

$$C_k^1 = \frac{\mathcal{F}\mathcal{P}_d + 2\mathcal{P}_m\mathcal{S}_d}{W \log_2 \left(1 + \frac{P_{k0}h_{k0}}{N_0} \right)} \quad (7.3b)$$

$$C_k^0 = \frac{2\mathcal{P}_m\mathcal{S}_m}{W \log_2 \left(1 + \frac{P_{k0}h_{k0}}{N_0} \right)} \quad (7.3c)$$

It is generally impossible to find an exact expression relating the optimization variables to the objective for most ML models. Therefore, the objective will be re-formulated as a function of the convergence bounds on the DL process over the edge. For more details on these bounds, the readers are referred to [77]. We will use the results and extend the discussion to our formulation, and then propose a strategy to jointly find the optimal τ , d_k , and L .

7.1.1 Convergence Bounds

The convergence bounds have been derived and well-discussed in [77]. For completeness, we will present some of the important results here in order to support our analysis. Let us continue with the assumption that L is an integer multiple of τ . Then, the global aggregation will only occur at every τ updates. therefore, the local updates occur at every iteration $l = 1, \dots, L$ and a global update will occur whenever $l = g\tau$ for $g = 1, \dots, G$. For any interval $[g]$ defined over $[(g-1)\tau, g\tau]$, define an auxiliary global model denoted by \mathbf{v} , which would have been calculated if a global update occurred, as follows:

$$\mathbf{v}_{[g]}[l] = \mathbf{v}_{[g]}[l-1] - \eta \nabla F(\mathbf{v}_{[g]}[l-1]) \quad (7.4)$$

Let the local model parameter set of learner k be denoted by \mathbf{w}_k and the local loss by $F_k(\mathbf{w}_k)$. Then, the optimal model at iteration l can be obtained by:

$$\mathbf{w}[l] = \frac{1}{d} \sum_{k=1}^K d_k \mathbf{w}_k[l] \quad (7.5)$$

The optimal $\mathbf{w}[l]$ will only be visible when $l = g\tau$ and for that iteration, the global loss can be defined by:

$$F(\mathbf{w}) = \frac{1}{d} \sum_{k=1}^K d_k F_k(\mathbf{w}) \quad (7.6)$$

The following assumptions are made about the loss function $F_k(\mathbf{w})$ at learner k : $F_k(\mathbf{w})$ is convex, $\|F_k(\mathbf{w}) - F_k(\bar{\mathbf{w}})\| \leq \rho|\mathbf{w} - \bar{\mathbf{w}}|$, and $\|\nabla F_k(\mathbf{w}) - \nabla F_k(\bar{\mathbf{w}})\| \leq \beta|\mathbf{w} - \bar{\mathbf{w}}|$ for any $\mathbf{w}, \bar{\mathbf{w}}$. These assumptions will hold for ML models with convex loss function such as linear regression and SVM. By simulations, we will show that the proposed solutions work for non-convex models such as the neural networks with ReLU activation.

Let us also assume that the local loss function at $F_k(\mathbf{w})$ does not diverge by more than δ_k such that $|F_k(\mathbf{w}) - F(\mathbf{w})| \leq \delta_k$ and we can apply the following estimation:

$$\delta = \frac{\sum_k d_k \delta_k}{d} \quad (7.7)$$

Furthermore, $|\mathbf{w} - \mathbf{v}_{[g]}[l-1]| \leq h(l - (g-1)\tau)$. For any τ , $h(\tau) = \frac{\delta}{\beta} [(\eta\beta + 1)^\tau - 1] - \eta\delta\tau$. Recall that η is the learning rate and β can be estimated by $\beta = \frac{\sum_k d_k \beta_k}{d}$ where:

$$\beta_k = \frac{\|\nabla F_k(\mathbf{w}_k[1]) - \nabla F_k(\mathbf{w}[1])\|}{|\mathbf{w}_k[1] - \mathbf{w}[1]|} \quad (7.8)$$

Based on this, the objective can be written as a function of the difference of the global loss after iteration L and the optimal global loss. Given the above assumptions about the loss function, and the constraints on the optimization variables and the time taken by learner

$k \in \mathcal{K}$, the optimization problem can be written as:

$$\min_{L, \tau, d_k \forall k} F(\mathbf{w}[L]) - F(\mathbf{w}^*) \quad (7.9)$$

$$\text{s.t.} \quad L \left(C_k^2 d_k + \frac{C_k^1 d_k + C_k^0}{\tau} \right) \leq T, \quad k \in \mathcal{K} \quad (7.9a)$$

$$\sum_{k=1}^K d_k = d \quad (7.9b)$$

$$\tau \in \mathcal{Z}_+ \quad (7.9c)$$

$$L \in \mathcal{Z}_+ \quad (7.9d)$$

$$d_k \in \mathcal{Z}_+, \quad k = 1, \dots, K \quad (7.9e)$$

$$\eta \left(1 - \frac{\beta \eta}{2} \right) - \frac{\rho}{\omega \epsilon^2} \frac{h(\tau)}{\tau} \geq 0 \quad (7.9f)$$

$$\eta \beta \leq 1 \quad (7.9g)$$

$$F(\mathbf{v}_{[g]}[l]) - F(\mathbf{w}^*) \geq \epsilon \quad (7.9h)$$

$$F(\mathbf{w}(L)) - F(\mathbf{w}^*) \geq \epsilon \quad (7.9i)$$

Constraint (7.9a) guarantees that the time consumed by a total of L updates does not exceed the total training time available given by T s. Constraint (7.9b) ensures that the total dataset comprising d samples is utilized. Constraints (7.9c) - (7.9e) are simply non-negativity and integer constraints for the optimization variables where L , τ and/or all d_k 's being zero represent cases where DL is not possible in the MEL environment. Constraints (7.9g) and (7.9f) represent a bound on the learning rate, meaning it should be small enough such that it guarantees convergence. When (7.9g) holds, (7.9f) will always hold. Constraints (7.9h) and (7.9i) define a lower bound on the gap between the optimal loss and the auxiliary loss at interval $[g]$ and the global loss, respectively, where $\epsilon > 0$. The parameter $\omega \triangleq \min_g \|\mathbf{v}_{[g]}[g-1]\tau - \mathbf{w}^*\|^{-2}$ represents the interval that minimizes the difference between the auxiliary loss and the global loss. The variables ρ , ω , and ϵ appear in a single term $\frac{\rho}{\omega \epsilon^2}$ in

(7.9f) which represents a control parameter. Later on, this term will be represented by B_0 but for now, we will continue with the original terms to make the analysis relatable to the original variables.

It is assumed that $\eta > 0$ (typically $0 < \eta < 1$) and $\beta > 0$. Furthermore η and ϵ can be set to small enough values such that $\eta\beta \leq 1$, and the constraints in (7.9f)-(7.9i) are satisfied. For a β -smooth function, Bernoulli's inequality will hold implying that $(\eta\beta + 1)^\tau \geq \eta\beta\tau + 1$. Furthermore, once all the assumptions about the loss function constraints are satisfied, it can be shown that $F(\mathbf{w}[L]) - F(\mathbf{w}^*) \leq \frac{1}{\eta(1-\frac{\beta\eta}{2}) - \frac{\rho}{\omega\epsilon^2} \frac{h(\tau)}{\tau}}$. Thus, the problem in (7.9) can be re-formulated as:

$$\min_{L, \tau, d_k \forall k} \frac{1}{L \left[\eta \left(1 - \frac{\beta\eta}{2} \right) - \frac{\rho}{\omega\epsilon^2} \frac{h(\tau)}{\tau} \right]} \quad (7.10)$$

$$\text{s.t.} \quad L \leq \frac{T\tau}{C_k^2 \tau d_k + C_k^1 d_k + C_k^0}, \quad k = 1, \dots, K \quad (7.10a)$$

$$\sum_{k=1}^K d_k = d \quad (7.10b)$$

$$\tau \in \mathcal{Z}_+ \quad (7.10c)$$

$$L \geq 0 \quad (7.10d)$$

$$d_k \geq 0, \quad k = 1, \dots, K \quad (7.10e)$$

Note that the integer constraints on d_k and L have been relaxed in (7.10e) and (7.10d) which will help in proposing a solution.

7.2 Proposed Solution

The idea of the proposed solution is to re-write the objective as a function of τ by using the constraints on the total time consumption and the fact that the system must train the model on at least d training samples.

7.2.1 Relating Bounds to the Number of Local Updates (τ)

The orchestrator can ensure that the bounds in constraints (7.9g)-(7.9i) are satisfied by choosing small enough values for η and ϵ . In that case, if constraint (7.9f) holds, the denominator of the objective function will be positive. Furthermore, if we relax the integer constraint on L , the optimal value for the total learning iterations can be given by:

$$L = \frac{T\tau}{C_k^2\tau d_k + C_k^1 d_k + C_k^0}, \quad k \in \mathcal{K} \quad (7.11)$$

By using the equality constraint in (7.10b), and re-arranging (7.11) to make d_k the subject, and defining two new variables $a_k = \frac{C_k^1}{C_k^2}$ and $b_k = \frac{C_k^0}{C_k^2}$, we can write L as a function of τ .

$$L(\tau) = \frac{KT \sum_{k=1}^K \tau \prod_{\substack{l=1 \\ l \neq k}}^K (\tau + b_l)}{d \prod_{k=1}^K (\tau + b_k) + \sum_{k=1}^K a_k \prod_{\substack{l=1 \\ l \neq k}}^K (\tau + b_l)} \quad (7.12)$$

The objective function denoted by O can be re-written as a function of τ in the following manner:

$$O(\tau) = \frac{1}{L(\tau)} \frac{1}{\left[\eta \left(1 - \frac{\beta\eta}{2} \right) - \frac{\rho}{\omega\epsilon^2} \frac{h(\tau)}{\tau} \right]} = \frac{P(\tau)}{L(\tau)} \quad (7.13)$$

Theorem 4 $O(\tau)$ is strictly convex on the domain $\tau \geq 0$.

Proof: The proof can be found in Appendix E ■

Because $\frac{\partial O}{\partial \tau} = 0$ does not have a closed form solution, the optimal τ^* can be obtained by solving the following problem:

$$\tau^* = \arg \min_{\tau} O(\tau) \quad (7.14)$$

The value of τ^* can be difficult to obtain because τ is unbounded. However, we can limit the search space by τ_{max} and then use a brute force approach to find the optimal τ^* . In fact, a binary search procedure has been proposed in [77] which has a complexity of $\mathcal{O}(\log \tau_{max})$. Once τ^* has been determined, L^* can be obtained using (7.12) and re-set to this new value.

Algorithm 3 MEL Process at the Orchestrator

Input: T, B_0, τ_{max}, d, K
Output: $\mathbf{w}[L]$

```

  INITIALIZE  $\tau \leftarrow 1$  and  $d_k \leftarrow \frac{d}{K}$ 
  SET  $\mathbf{w} \leftarrow \mathbf{w}[0]$  as a random vector
1: while  $\tau > 0$  do
2:   for each learner  $k \in \mathcal{K}$  in PARALLEL do
3:     SEND  $\mathbf{w}, \tau,$  and  $d_k$ 
4:     WAIT for completion of  $\tau$  local updates
5:     RECEIVE  $\mathbf{w}_k, \beta_k,$  and  $\nabla F_k(\mathbf{w}_k)$ 
6:     RECEIVE  $P_{k0}, h_{k0}$  and  $f_k$  from each learner  $k \forall k \in \mathcal{K}$ 
7:   end for
8:   ESTIMATE  $t_k \forall k \in \mathcal{K}$  using (7.1)
9:   SET  $\hat{T} \leftarrow \hat{T} + \max(t_k)/L$  and  $T \leftarrow T - \hat{T}$ 
10:  ESTIMATE  $\mathbf{w}, \nabla F(\mathbf{w}), \beta,$  and  $\delta$ 
11:  Find the optimal  $\tau$  by applying the bisection method to (7.13) on  $[0, \tau_{max}]$ 
12:  Calculate  $L$  using (7.12)
13:  Use new value of  $L$  to obtain  $d_k \forall k \in \mathcal{K}$  using (7.11)
14:  SET  $\hat{T} \leftarrow \hat{T} + \max(t_k)$ 
15:  if  $\hat{T} > T$  then
16:    Reduce  $\tau$  to maximum value  $\geq 0$  such that  $\hat{T} \leq T$ 
17:  end if
18: end while
19: return  $\mathbf{w}$ 

```

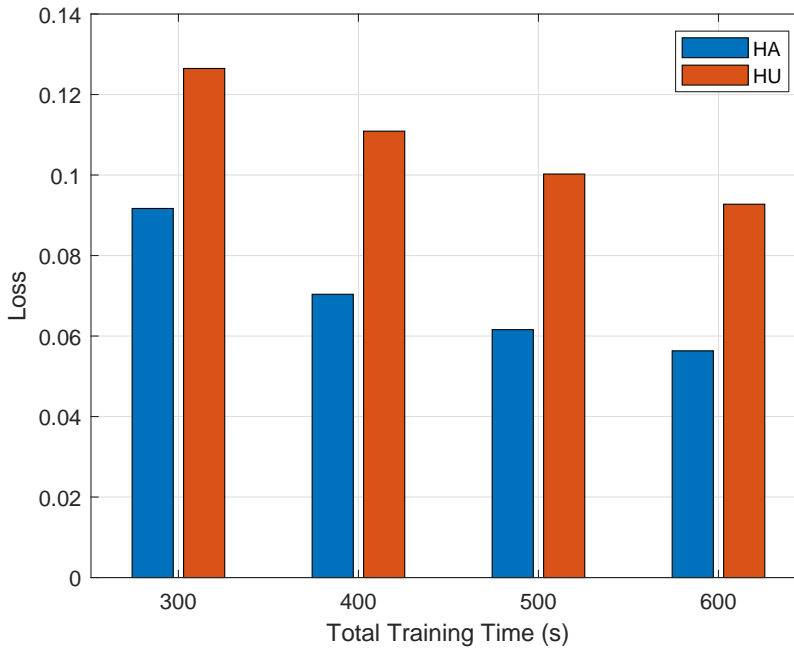
The values of $d_k^* \forall k \in \mathcal{K}$ for the next τ updates can be obtained using (7.11). Because the integer constraint on $d_k \forall k \in \mathcal{K}$ was relaxed in (7.10), they can be set by flooring the actual value. This process is repeated for each global cycle until the total training time is consumed. This process is summarized in algorithm 3.

7.3 Results and Discussion

The learners are assumed to be located in a cellular type environment and are assumed to be a combination of smart phone and Raspberry PI type microcontrollers. The channel parameters and device capabilities are listed in Table 7.1. To test our proposed MEL paradigm, the commonly used MNIST [92] dataset is trained using a DNN with 3 hidden layers consisting of 300, 124 and 60 neurons, respectively. The details of the resulting model

Table 7.1: Simulation environment for optimal task allocation

Parameter	Value
Cell Attenuation Model	$128 + 37.1 \log(R)$ dB [84]
Node Bandwidth (W)	5 MHz
Device proximity (R)	500m
Transmission Power (P_k)	23 dBm
Noise Power Density (N_0)	-174 dBm/Hz
Computation Capabilities (f_k)	$\sim \{ 2.4, 1.2 \}$ GHz
MNIST Dataset size (d)	54,000 images
MNIST Dataset Features (\mathcal{F})	784 (28×28) pixels

Figure 7.1: Training loss versus total training time for $K = 20$ learners

sizes and complexities are discussed in [87].

For the simulation, we consider a set of $K = 20$ learners and test for total training times of $T = \{300, 400, 500, 600\}$ s. It was found that a value of $\eta = 0.01$ for the learning rate works very well and setting B_0 in the range $0.005 - 0.01$ provided solutions that converge. τ_{max} is set to the case where only three global aggregations would be done on $d_k = d/K \forall k \in \mathcal{K}$.

We plot the final loss value after training for time T for the proposed HA approach and the HU approach from [77] in Figure 7.1, and the final accuracy in Figure 7.2. As expected, as the training time is increased, the loss value decreases for all approaches. However, for

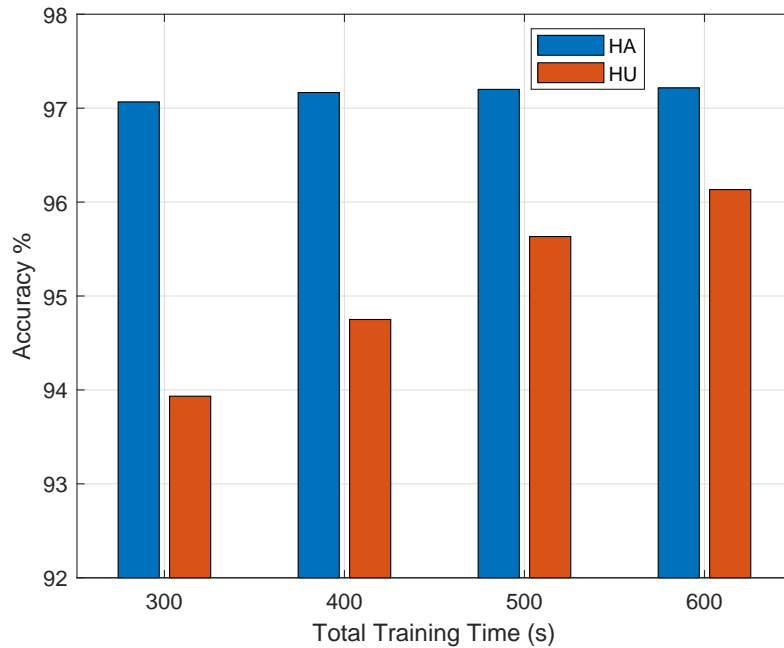


Figure 7.2: Validation accuracy versus total training time for $K = 20$ learner

the HA approach, there is just a slight increase in validation accuracy because it is able to achieve a high level in minimum time. The main conclusion is that optimizing τ and d_k jointly influences the possible number of global aggregations G as well as the total iterations L , which helps in converging to both a lower loss and a higher final accuracy. For example, the loss of the HA approach is lower by 0.03-0.05 which represents gains in the range of 27% - 40%. Furthermore, training for 300 s using the HA approach achieves a 97% accuracy, a value not achieved by the HU approach even in 600 s.

Chapter 8: Recommendations, Conclusions, and Future Outlook

8.1 Recommendations

Although it is difficult to see a concrete trend from the results in Chapter 6 and specifically Sections 6.2.5 and 6.2.5, we may conclude that the HA-Asyn works best when the resources are at their lowest and the synchronous approach may fail. It can also provide gains when one resource is low and the other high, especially, when energy is abundant and time is low. This works for the scenarios where FL has been proposed for devices that are charging and not on battery power. In the medium range of both resources, time and energy, the HA-Sync works best. When both resources are abundant, the HA-Asyn provides faster convergence to certain accuracy thresholds but the HA-Sync may converge to higher final accuracy.

To conclude, we suggest the following procedure to select the best scheme between the HA-Sync and the HA-Asyn approaches:

1. If energy is not a factor at all and the constraint is only on time, choose the HA-Sync approach unless the global cycle time is very low.
2. If both time and energy consumption are a factor, then:
 - if either resource, time or energy is low, choose the HA-Asyn approach
 - if both resources are in the medium range, then go for the HA-Sync
 - if both resources are high and an early exit is desired, choose the HA-Asyn approach; otherwise if time is unlimited, HA-Sync may work best
3. If the HA-Sync is chosen, simply do cross-validation on the ML model and associated hyper-parameters such as model size, learning rate, regularization, etc. On the other hand, when HA-Asyn is used, the parameter c should be added to the cross-validation. In the very low resource region, checking with $c = 1$ $c = 2$ may suffice whereas in case when one of the resources is abundant, a set from the range $c \geq 3$ may be used. For the very high resource region, the search can be done on $0 \leq c \leq 5$.

8.2 Conclusions

This dissertation begins by designing an H-MEC paradigm with optimal resource allocation while jointly minimizing system completion time and energy consumption. The focus then shifts to proposing an optimal task allocation method for DL over the wireless edge while maximizing the ML model accuracy. To this end, a new paradigm called MEL was designed. The aim was to optimize task allocation for MEL systems when there are completion time constraints only and also for the case when there are dual constraints on the completion time and energy consumption. This was done for the synchronous and asynchronous settings.

Results on H-MEC demonstrated that our proposed architecture reduced the number of users offloading to the edge and offered a higher utility (a function of delay and energy usage). Results on task allocation for MEL under time constraints showed that our HA approach (HA-Sync) worked much better than the previously proposed HU (HU-Sync) approaches in literature. From the asynchronous task allocation (HA-Asyn) results, it was shown that under certain conditions, this approach can offer better learning accuracy than even the synchronous (HA-Sync) setting.

Extending these studies to the cases when the HA-Sync/Asyn had dual time and energy constraints, it was shown once again that the HA-Sync performed much better than the HU-Sync. Furthermore, it was shown that both, HA-Asyn and HA-Sync, outperform each other given different scenarios and regions of resource availability. Based on this, the dissertation concluded by provides recommendations on which scheme to select out of the HA-Sync and HA-Asyn with the correct amounts of cap on the staleness.

Lastly, additional work was done to relate our HA parameters directly to the HA MEL model for the HA-Sync with time constraints to compare against previous work. It was found that the HA-Sync performs better then the HU-Asyn in terms of a lower final loss and a higher final validation accuracy.

8.3 Future Work

Overall, as described in the first chapter, with the potential presence of billions of IoT devices on edge networks, edge analytics EI will play an important role in helping preserve backhaul networks, central storage capacities, as well as user privacy which has suddenly become a vital component of the consumer experience. Specifically, DL over the wireless edge is a budding and exciting area especially with the advent of 5G and 6G technologies that are expected to speed up the deployment of Edge AI. Indeed, MEL can be expected to play a huge role for streamlining the implementation of background ML applications that will be required to run for producing the results based on edge analytics and AI.

The following problems are still open in the area of H-MEC and MEL, respectively:

- For H-MEC, adding the component of peer-to-peer offloading raises security and privacy concerns. One extension is to add a security block such as encryption and evaluate the efficacy of the proposed approach with the additional delay.
- For MEL, one challenge is to cluster the best set of learners to each orchestrator in a system where multiple orchestrators are training different ML models. The clustering needs to be done such that the dual objectives of optimal wireless resource allocation and maximum ML accuracy are satisfied.
- Completely distributing the FL process where there is no dependence on a central orchestrator. One scenario where this may apply is when several sets of learners all need to learn different tasks. Based on its capacity, each learner will have the independence to join the most suitable set of learners and any one learner joining in or dropping out of the learning process will not impact the other learners.

Bibliography

- [1] K. Gyarmathy, “Comprehensive Guide to IoT Statistics You Need to Know in 2020,” 2020. [Online]. Available: <https://www.vxchnge.com/blog/iot-statistics>
- [2] X. Lyu, H. Tian, C. Sengul, and P. Zhang, “Multiuser joint task offloading and resource optimization in proximate clouds,” *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3435–3447, 2017. [Online]. Available: <https://ieeexplore.ieee.org/document/7517217>
- [3] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, “A Survey on Mobile Edge Computing: The Communication Perspective,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017. [Online]. Available: <http://arxiv.org/abs/1701.01090>
- [4] 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, Dec 2016. [Online]. Available: <http://ieeexplore.ieee.org/document/7498684/>
- [5] Rhea Kelly, “Internet of Things Data To Top 1.6 Zettabytes by 2020,” 2015. [Online]. Available: <https://campustechnology.com/articles/2015/04/15/internet-of-things-data-to-top-1-6-zettabytes-by-2020.aspx>
- [6] Sciforce, “Can Edge Analytics Become a Game Changer?” Sep 2019. [Online]. Available: <https://medium.com/sciforce/can-edge-analytics-become-a-game-changer-9cc9395d2727>
- [7] W. Yang, B. Lim, N. C. Luong, and D. T. Hoang, “Federated Learning in Mobile Edge Networks: A Comprehensive Survey,” *IEEE Communications*

- Surveys & Tutorials*, no. (Early Access), pp. 1–33, 2020. [Online]. Available: <https://ieeexplore.ieee.org/document/9060868>
- [8] J. F. Monserrat and D. Mart, “Key Technologies for the Advent of the 6G,” in *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. Seoul, Korea (South): IEEE, 2020, pp. 5–10. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/9124725>
- [9] B. McMahan and D. Ramage, “Federated Learning: Collaborative Machine Learning without Centralized Training Data,” Mar 2017. [Online]. Available: <https://ai.googleblog.com/2017/04/federated-learning-collaborative.html>
- [10] J. Wu, “Edge AI Is The Future, Intel And Udacity Are Teaming Up To Train Developers,” Apr 2020. [Online]. Available: <https://www.forbes.com/sites/cognitiveworld/2020/04/16/edge-ai-is-the-future-intel-and-udacity-are-teaming-up-to-train-developers/{\#}5de50f3168f2>
- [11] IEEE, “IEEE Draft Guide for Architectural Framework and Application of Federated Machine Learning,” *IEEE P3652.1/D6, April 2020*, pp. 1–70, Jun 2020.
- [12] J. Jeon, J. Kim, J. Huh, H. Kim, and S. Cho, “Overview of Distributed Federated Learning: Research Issues, Challenges, and Biomedical Applications,” in *2019 International Conference on Information and Communication Technology Convergence (ICTC)*. Jeju Island, South Korea: IEEE, Oct 2019, pp. 1426–1427. [Online]. Available: <https://ieeexplore.ieee.org/document/8939954/>
- [13] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, “Federated Learning for Ultra-Reliable Low-Latency V2V Communications,” in *2018 IEEE Global Communications Conference, GLOBECOM 2018 - Proceedings*. Dubai, UAE: Institute of Electrical and Electronics Engineers Inc., 2018. [Online]. Available: <https://ieeexplore.ieee.org/document/8647927>

- [14] B. Hu, Y. Gao, L. Liu, and H. Ma, “Federated Region-Learning: An Edge Computing Based Framework for Urban Environment Sensing,” in *2018 IEEE Global Communications Conference, GLOBECOM 2018 - Proceedings*. Dubai, UAE: Institute of Electrical and Electronics Engineers Inc., 2018. [Online]. Available: <https://ieeexplore.ieee.org/document/8647649>
- [15] J. Song, Y. Cui, M. Li, J. Qiu, and R. Buyya, “Energy-traffic tradeoff cooperative offloading for mobile cloud computing,” *IEEE International Workshop on Quality of Service, IWQoS*, pp. 284–289, 2014.
- [16] M. Liu and Y. Liu, “Price-Based Distributed Offloading for Mobile-Edge Computing with Computation Capacity Constraints,” *IEEE Wireless Communications Letters*, pp. 1–4, 2017.
- [17] Y. Li, L. Sun, and W. Wang, “Exploring device-to-device communication for mobile cloud computing,” *2014 IEEE International Conference on Communications (ICC)*, pp. 2239 – 44, 2014. [Online]. Available: <http://dx.doi.org/10.1109/ICC.2014.6883656>
- [18] X. Cao, F. Wang, J. Xu, R. Zhang, and S. Cui, “Joint Computation and Communication Cooperation for Mobile Edge Computing,” in *16th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, 2018. [Online]. Available: <http://arxiv.org/abs/1704.06777>
- [19] C. You and K. Huang, “Exploiting Non-Causal CPU-State Information for Energy-Efficient Mobile Cooperative Computing,” *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 4104 – 4117, 2019. [Online]. Available: <https://ieeexplore.ieee.org/document/8330749><https://arxiv.org/abs/1704.04595>
- [20] Z. Sheng, C. Mahapatra, V. Leung, M. Chen, and P. Sahu, “Energy Efficient Cooperative Computing in Mobile Wireless Sensor Networks,” *IEEE Transactions*

- on Cloud Computing*, vol. 7161, no. c, pp. 1–1, 2015. [Online]. Available: <http://ieeexplore.ieee.org/document/7161307/>
- [21] C.-F. Liu, M. Bennis, and H. V. Poor, “Latency and Reliability-Aware Task Offloading and Resource Allocation for Mobile Edge Computing,” in *2017 IEEE Globecom Workshops (GC Wkshps)*, 2017, pp. 1–7. [Online]. Available: <https://ieeexplore.ieee.org/document/8269175>
- [22] X. Zhang, Y. Mao, J. Zhang, and K. B. Letaief, “Multi-Objective Resource Allocation for Mobile Edge Computing Systems,” in *IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), 2017*, vol. 19, no. 4, Montreal, 2017, pp. 1–5. [Online]. Available: <https://ieeexplore.ieee.org/document/8292379>
- [23] M. Jo, T. Maksymyuk, B. Strykhalyuk, and C.-H. Cho, “Device-to-device-based heterogeneous radio access network architecture for mobile cloud computing,” *IEEE Wireless Communications*, vol. 22, no. 3, pp. 50–58, 2015. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7143326>
- [24] K. Doppler, M. Rinne, C. Wijting, C. B. Ribeiro, and K. Hug, “Device-to-device communication as an underlay to LTE-advanced networks,” *IEEE Communications Magazine*, vol. 47, no. 12, pp. 42–49, 2009.
- [25] J. Ren, H. Wang, T. Hou, S. Zheng, and C. Tang, “Federated Learning-Based Computation Offloading Optimization in Edge Computing-Supported Internet of Things,” *IEEE Access*, vol. 7, pp. 69 194–69 201, 2019. [Online]. Available: <https://ieeexplore.ieee.org/document/8728285>
- [26] Z. Yu, J. Hu, G. Min, H. Lu, Z. Zhao, H. Wang, and N. Georgalas, “Federated Learning Based Proactive Content Caching in Edge Computing,” in *2018 IEEE Global Communications Conference, GLOBECOM 2018 - Proceedings*. Dubai, UAE:

- Institute of Electrical and Electronics Engineers Inc., 2018. [Online]. Available: <https://ieeexplore.ieee.org/document/8647616>
- [27] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-Efficient Learning of Deep Networks from Decentralized Data,” 2016. [Online]. Available: <https://arxiv.org/abs/1602.05629>
- [28] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. A. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, “Large Scale Distributed Deep Networks,” in *Advances in Neural Information Processing Systems 25*, 2012, pp. 1223–1231. [Online]. Available: <https://papers.nips.cc/paper/4687-large-scale-distributed-deep-networks>
- [29] M. Langer, A. Hall, Z. He, and W. Rahayu, “MPCA SGD-A Method for Distributed Training of Deep Learning Models on Spark,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 11, pp. 2540–2556, Nov 2018. [Online]. Available: <https://ieeexplore.ieee.org/document/8354695/>
- [30] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D’Oliveira, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao, “Advances and Open Problems in Federated Learning,” *arXiv e-prints*, Dec 2019. [Online]. Available: <http://arxiv.org/abs/1912.04977>

- [31] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, “Federated Learning: Challenges, Methods, and Future Directions,” *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020. [Online]. Available: <https://ieeexplore.ieee.org/document/9084352/>
- [32] J. K. Konečný, H. Brendan, M. Google, D. Ramage Google, and P. Richtárik, “Federated Optimization: Distributed Machine Learning for On-Device Intelligence,” *arXiv e-prints*, 2016.
- [33] R. R. Karn, P. Kudva, and I. A. M. Elfadel, “Dynamic Autoselection and Autotuning of Machine Learning Models for Cloud Network Analytics,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 5, pp. 1052–1064, May 2019. [Online]. Available: <https://ieeexplore.ieee.org/document/8500348/>
- [34] J. Dass, V. Sarin, and R. N. Mahapatra, “Fast and Communication-Efficient Algorithm for Distributed Support Vector Machine Training,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 5, pp. 1065–1076, May 2019. [Online]. Available: <https://ieeexplore.ieee.org/document/8526323/>
- [35] B. Woodworth, K. K. Patel, S. U. Stich, Z. Dai, B. Bullins, H. B. McMahan, O. Shamir, and N. Srebro, “Is Local SGD Better than Minibatch SGD?” *arXiv e-prints*, Feb 2020. [Online]. Available: <http://arxiv.org/abs/2002.07839>
- [36] B. Swenson, S. Kart, H. V. Poor, and J. M. Moura, “Distributed Global Optimization by Annealing,” *2019 IEEE 8th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing, CAMSAP 2019 - Proceedings*, pp. 181–185, 2019. [Online]. Available: <https://ieeexplore.ieee.org/document/9022513>
- [37] Z. Du, C. Wu, S. Member, and T. Y. Member, “Federated Learning for Vehicular Internet of Things : Recent Advances and Open Issues,” *IEEE Open Journal of the Computer Society*, vol. 1, pp. 45–61, 2020. [Online]. Available: <https://ieeexplore.ieee.org/document/9086790>

- [38] S. R. Pokhrel and J. Choi, “Improving TCP Performance over WiFi for Internet of Vehicles: A Federated Learning Approach,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 6798 – 6802, Apr 2020. [Online]. Available: <https://ieeexplore.ieee.org/document/9055134>
- [39] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, “Blockchain Empowered Asynchronous Federated Learning for Secure Data Sharing in Internet of Vehicles,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4298–4311, Apr 2020. [Online]. Available: <https://ieeexplore.ieee.org/document/8998397>
- [40] J. Cao, K. Zhang, F. Wu, and S. Leng, “Learning Cooperation Schemes for Mobile Edge Computing Empowered Internet of Vehicles,” in *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. Seoul, Korea (South): IEEE, 2020, pp. 5–10. [Online]. Available: <https://ieeexplore.ieee.org/document/9120493>
- [41] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, “Federated Learning for Data Privacy Preservation in Vehicular Cyber-Physical Systems,” *IEEE Network*, vol. 34, no. 3, pp. 50–56, 2020. [Online]. Available: <https://ieeexplore.ieee.org/document/9105934>
- [42] W. Zhou, Y. Li, S. Chen, and B. Ding, “Real-Time Data Processing Architecture for Multi-Robots Based on Differential Federated Learning,” in *2018 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCOM/IOP/SCI)*, Oct 2018, pp. 462–471. [Online]. Available: <https://ieeexplore.ieee.org/document/8560084>
- [43] K. Sozinov, V. Vlassov, and S. Girdzijauskas, “Human Activity Recognition Using Federated Learning,” in *2018 IEEE Intl Conf on Parallel Distributed Processing with Applications, Ubiquitous Computing Communications, Big Data Cloud*

- Computing, Social Computing Networking, Sustainable Computing Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, Dec 2018, pp. 1103–1111. [Online]. Available: <https://ieeexplore.ieee.org/document/8672262>
- [44] D. Leroy, A. Coucke, T. Lavril, T. Gisselbrecht, and J. Dureau, “Federated Learning for Keyword Spotting,” in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2019-May. Institute of Electrical and Electronics Engineers Inc., May 2019, pp. 6341–6345. [Online]. Available: <https://ieeexplore.ieee.org/document/8683546>
- [45] D. Li, T. Salonidis, N. V. Desai, and M. C. Chuah, “DeepCham: Collaborative Edge-Mediated Adaptive Deep Learning for Mobile Object Recognition,” in *2016 IEEE/ACM Symposium on Edge Computing (SEC)*. Washington, DC: Institute of Electrical and Electronics Engineers Inc., Dec 2016, pp. 64–76. [Online]. Available: <https://ieeexplore.ieee.org/document/7774674>
- [46] D. R. Mukhametov, “Ubiquitous Computing and Distributed Machine Learning in Smart Cities,” in *2020 Wave Electronics and its Application in Information and Telecommunication Systems (WECONF)*. Saint-Petersburg, Russia: IEEE, 2020, pp. 7–11. [Online]. Available: <https://ieeexplore.ieee.org/document/9131518/>
- [47] T. Yu, T. Li, Y. Sun, S. Nanda, V. Smith, V. Sekar, and S. Seshan, “Learning Context-aware Policies from Multiple Smart Homes via Federated Multi-Task Learning,” in *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*. Sydney, Australia: IEEE, 2020, pp. 104–115. [Online]. Available: <https://ieeexplore.ieee.org/document/9097597/>
- [48] Y. Chen, X. Qin, J. Wang, C. Yu, and W. Gao, “FedHealth : A Federated Transfer Learning Framework for Wearable Healthcare,” *IEEE Intelligent*

- Systems*, no. (Early Access), pp. 1–8, 2020. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/9076082>
- [49] S. Lu, Y. Zhang, and Y. Wang, “Decentralized Federated Learning for Electronic Health Records,” in *2020 54th Annual Conference on Information Sciences and Systems (CISS)*. Princeton, NJ, USA: IEEE, 2020, pp. 1–5. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/9086196/>
- [50] H.-k. Lim, J.-b. Kim, and C.-m. Kim, “Federated Reinforcement Learning for Controlling Multiple Rotary Inverted Pendulums in Edge Computing Environments,” in *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC)*. IEEE, 2020, pp. 463–468. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/9065233/>
- [51] R. Doku and D. B. Rawat, “iFLBC : On the Edge Intelligence Using Federated Learning Blockchain Network,” in *2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*. Baltimore, MD, USA: IEEE, 2020, pp. 5–10. [Online]. Available: <https://ieeexplore.ieee.org/document/9123014>
- [52] Q. Wang, S. Member, Y. Guo, S. Member, X. Wang, and S. Member, “AI at the Edge: Blockchain-Empowered Secure Multiparty Learning with Heterogeneous Models,” *IEEE Internet of Things Journal*, no. (Early Access), pp. 1–11, 2020. [Online]. Available: <https://ieeexplore.ieee.org/document/9066954>
- [53] S. R. Pokhrel and J. Choi, “A Decentralized Federated Learning Approach For Connected Autonomous Vehicles,” in *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, Seoul, Korea (South), 2020, pp. 1–6. [Online]. Available: <https://ieeexplore.ieee.org/document/9124733>

- [54] S. R. Pokhrel, J. Choi, and S. Member, “Federated Learning with Blockchain for Autonomous Vehicles : Analysis and Design Challenges,” *IEEE Transactions on Communications*, no. (Early Access), pp. 1–13, 2020. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/9079513>
- [55] H. Chai, S. Leng, Y. Chen, and K. Zhang, “A Hierarchical Blockchain-Enabled Federated Learning Algorithm for Knowledge Sharing in Internet of Vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, no. (Early Access), pp. 1–12, 2020. [Online]. Available: <https://ieeexplore.ieee.org/document/9127823/>
- [56] Y. Qu, S. R. Pokhrel, and S. Garg, “A Blockchain-enabled Federated Learning Framework for Cognitive Computing in Industry 4.0 Networks,” *IEEE Transactions on Industrial Informatics*, no. (Early Access), pp. 1–10, 2020. [Online]. Available: <https://ieeexplore.ieee.org/document/9134967>
- [57] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, “Incentive Mechanism for Reliable Federated Learning: A Joint Optimization Approach to Combining Reputation and Contract Theory,” *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 700 – 10 714, 2019. [Online]. Available: <https://ieeexplore.ieee.org/document/8832210>
- [58] Y. Zou, S. Feng, D. Niyato, Y. Jiao, S. Gong, and W. Cheng, “Mobile device training strategies in federated learning: An evolutionary game approach,” in *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. Atlanta, GA, USA: IEEE, 2019, pp. 874–879. [Online]. Available: <https://ieeexplore.ieee.org/document/8875353>
- [59] S. Feng, D. Niyato, P. Wang, D. I. Kim, and Y. C. Liang, “Joint service pricing and cooperative relay communication for federated learning,” in *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Commu-*

- nications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCoM) and IEEE Smart Data (SmartData)*. Atlanta, GA, USA: IEEE, 2019, pp. 815–820.
- [60] J. Kang, Z. Xiong, D. Niyato, H. Yu, Y. C. Liang, and D. I. Kim, “Incentive design for efficient federated learning in mobile networks: A contract theory approach,” in *2019 IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS)*. Singapore, Singapore: IEEE, 2019, pp. 1–5.
- [61] T. Huong, T. Le, N. H. Tran, Y. K. Tun, Z. Han, and C. S. Hong, “Auction based Incentive Design for Efficient Federated Learning in Cellular Wireless Networks,” in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*. Seoul, Korea (South): IEEE, 2020, pp. 1–6. [Online]. Available: <https://ieeexplore.ieee.org/document/9120773>
- [62] H. Yu, Z. Liu, Y. Liu, T. Chen, M. Cong, X. Weng, D. Niyato, and Q. Yang, “A Sustainable Incentive Scheme for Federated Learning,” *IEEE Intelligent Systems*, no. (Early Access), pp. 1–9, 2020. [Online]. Available: <https://ieeexplore.ieee.org/document/9069185>
- [63] J.-H. Ahn, O. Simeone, and J. Kang, “Wireless Federated Distillation for Distributed Edge Learning with Heterogeneous Data,” in *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Sep 2019, pp. 1–6. [Online]. Available: <http://arxiv.org/abs/1907.02745><https://ieeexplore.ieee.org/document/8904164>
- [64] K. Yang, T. Jiang, Y. Shi, and Z. Ding, “Federated Learning Based on Over-the-Air Computation,” in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*. Institute of Electrical and Electronics Engineers Inc., May 2019. [Online]. Available: <https://ieeexplore.ieee.org/document/8761429>

- [65] S. Hua, K. Yang, and Y. Shi, “On-device federated learning via second-order optimization with over-the-air computation,” in *IEEE Vehicular Technology Conference*. Honolulu, HI: Institute of Electrical and Electronics Engineers Inc., Sep 2019. [Online]. Available: <https://ieeexplore.ieee.org/document/8891310>
- [66] G. Zhu, Y. Wang, and K. Huang, “Broadband Analog Aggregation for Low-Latency Federated Edge Learning,” *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 491 – 506, 2020. [Online]. Available: <https://ieeexplore.ieee.org/document/8870236>
- [67] M. M. Amiri, S. Member, D. Gündüz, and S. Member, “Machine Learning at the Wireless Edge : Distributed Stochastic Gradient Descent Over-the-Air,” *IEEE Transactions on Signal Processing*, vol. 68, pp. 2155–2169, 2020. [Online]. Available: <https://ieeexplore.ieee.org/document/9042352>
- [68] M. M. Amiri and D. Gunduz, “Federated Learning Over Wireless Fading Channels,” *IEEE Transactions on Wireless Communications*, vol. 19, no. 5, pp. 3546–3557, 2020. [Online]. Available: <https://ieeexplore.ieee.org/document/9014530>
- [69] S. Ha, J. Zhang, O. Simeone, and J. Kang, “Coded Federated Computing in Wireless Networks with Straggling Devices and Imperfect CSI,” in *2019 IEEE International Symposium on Information Theory (ISIT)*. Paris, France: IEEE, 2019, pp. 2649–2653. [Online]. Available: <https://ieeexplore.ieee.org/document/8849507>
- [70] H. H. Yang, Z. Liu, T. Q. S. Quek, and H. V. Poor, “Scheduling Policies for Federated Learning in Wireless Networks,” *IEEE Transactions on Communications*, vol. 68, no. 1, pp. 317–333, Sep 2019. [Online]. Available: <https://ieeexplore.ieee.org/document/8851249>
- [71] M. M. Wadu, S. Samarakoon, and M. Bennis, “Federated Learning under Channel Uncertainty : Joint Client Scheduling and Resource Allocation,” in *2020 IEEE Wireless*

- Communications and Networking Conference (WCNC)*. Seoul, Korea (South): IEEE, 2020, pp. 1–6. [Online]. Available: <https://ieeexplore.ieee.org/document/9120649>
- [72] X. Yao, T. Huang, C. Wu, R. Zhang, and L. Sun, “Towards Faster and Better Federated Learning: A Feature Fusion Approach,” in *2019 IEEE International Conference on Image Processing (ICIP)*. Taipei, Taiwan: IEEE, 2019, pp. 175–179. [Online]. Available: <https://ieeexplore.ieee.org/document/8803001>
- [73] K. Deng, Z. Chen, S. Zhang, C. Gong, and J. Zhu, “Content Compression Coding for Federated Learning,” in *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*. Xi’an, China: IEEE, Oct 2019, pp. 1–6. [Online]. Available: <https://ieeexplore.ieee.org/document/8928018/>
- [74] L. Wang, W. Wang, and B. Li, “CMFL: Mitigating communication overhead for federated learning,” in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. Dallas, TX, USA: Institute of Electrical and Electronics Engineers Inc., Jul 2019, pp. 954–964. [Online]. Available: <https://ieeexplore.ieee.org/document/8885054>
- [75] X. Yao, C. Huang, and L. Sun, “Two-Stream Federated Learning: Reduce the Communication Costs,” in *2018 IEEE Visual Communications and Image Processing (VCIP)*. Taichung, Taiwan: IEEE, 2018, pp. 1–4. [Online]. Available: <https://ieeexplore.ieee.org/document/8698609>
- [76] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, “Adaptive Federated Learning in Resource Constrained Edge Computing Systems,” *IEEE Journal on Selected Areas in Communications*, no. Early Access, pp. 1–1, 2019. [Online]. Available: <https://ieeexplore.ieee.org/document/8664630/>

- [77] —, “When Edge Meets Learning : Adaptive Control for Resource-Constrained Distributed Machine Learning,” in *INFOCOM*, 2018. [Online]. Available: <https://ieeexplore.ieee.org/document/8486403>
- [78] T. Tuor, S. Wang, T. Salonidis, B. J. Ko, and K. K. Leung, “Demo abstract: Distributed machine learning at resource-limited edge nodes,” *INFOCOM 2018 - IEEE Conference on Computer Communications Workshops*, pp. 1–2, 2018.
- [79] D. Conway-Jones, T. Tuor, S. Wang, and K. K. Leung, “Demonstration of Federated Learning in a Resource-Constrained Networked Environment,” in *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, 2019. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/8784064>
- [80] 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,” *arXiv e-prints*, p. arXiv:1909.07972, Sep 2019. [Online]. Available: <https://arxiv.org/abs/1909.07972>
- [81] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, “Energy Efficient Federated Learning Over Wireless Communication Networks,” *arXiv e-prints*, p. arXiv:1911.02417, Nov 2019. [Online]. Available: <https://arxiv.org/abs/1911.02417v1>
- [82] 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,” 2019. [Online]. Available: <https://arxiv.org/abs/1909.07972>
- [83] N. H. Tran, W. Bao, A. Zomaya, M. N. Nguyen, and C. S. Hong, “Federated Learning over Wireless Networks: Optimization Model Design and Analysis,” in *Proceedings - IEEE INFOCOM*. Paris, France: IEEE, 2019, pp. 1387–1395. [Online]. Available: <https://ieeexplore.ieee.org/document/8737464>

- [84] U. Yaquub and S. Sorour, “Multi-Objective Resource Optimization for Hierarchical Mobile Edge Computing,” in *2018 IEEE Global Communications Conference (GLOBECOM)*, Dec 2018, pp. 1–6. [Online]. Available: <https://ieeexplore.ieee.org/document/8648109>
- [85] L. Bottou and O. Bousquet, “The Tradeoffs of Large Scale Learning,” in *Advances in Neural Information Processing Systems*, J. C. Platt, D. Koller, Y. Singer, and S. Roweis, Eds. NIPS Foundation (<http://books.nips.cc>), 2008, vol. 20, pp. 161–168. [Online]. Available: <http://leon.bottou.org/papers/bottou-bousquet-2008>
- [86] S. Teerapittayanon, B. McDanel, and H. T. Kung, “Distributed Deep Neural Networks over the Cloud, the Edge and End Devices,” *Proceedings - International Conference on Distributed Computing Systems*, pp. 328–339, 2017.
- [87] U. Mohammad and S. Sorour, “Adaptive Task Allocation for Mobile Edge Learning,” in *2019 IEEE Wireless Communications and Networking Conference Workshop (WCNCW)*. IEEE, Apr 2019, pp. 1–6. [Online]. Available: <https://ieeexplore.ieee.org/document/8902527/>
- [88] Z. Wei, S. Gupta, X. Lian, and J. Liu, “Staleness-Aware Async-SGD for distributed deep learning,” *IJCAI International Joint Conference on Artificial Intelligence*, pp. 2350–2356, 2016.
- [89] A. D. Pia, S. S. Dey, and M. Molinaro, “Mixed-integer Quadratic Programming is in NP,” *Mathematical Programming*, vol. 162, no. 1, pp. 225–240, 2017.
- [90] J. Currie and D. I. Wilson, “OPTI: Lowering the Barrier Between Open Source Optimizers and the Industrial MATLAB User,” in *Foundations of Computer-Aided Process Operations*, N. Sahinidis and J. Pinto, Eds., Savannah, Georgia, USA, 2012.

- [91] S. Cebula, A. Ahmad, J. M. Graham, C. V. Hinds, L. A. Wahsheh, A. T. Williams, and S. J. DeLoatch, "Empirical channel model for 2.4 GHz ieee 802.11 wlan," *Proceedings of the 2011 International Conference on Wireless Networks*, 2011.
- [92] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based Learning Applied to Document Recognition," *Proceedings of IEEE*, vol. 86, no. 11, pp. 2278 – 2324, 1998. [Online]. Available: <https://ieeexplore.ieee.org/document/726791>
- [93] S. Munder and D. M. Gavrilu, "An experimental study on pedestrian classification." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, pp. 1863–1868, 2006. [Online]. Available: <https://ieeexplore.ieee.org/document/1704841>
- [94] Brendan Shillingford, "What is the time complexity of backpropagation algorithm for training artificial neural networks? - Quora," 2016. [Online]. Available: <https://www.quora.com/What-is-the-time-complexity-of-backpropagation-algorithm-for-training-artificial-neural-networks>
- [95] F. Uhlig, "The DQR algorithm, basic theory, convergence, and conditional stability," *Numerische Mathematik*, vol. 76, no. 4, pp. 515–553, 1997.
- [96] A. Nemirovski, "Interior point polynomial time methods in convex programming," *Lecture Notes*, vol. 42, no. 16, pp. 3215–3224, 2004. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.160.6909{\&}rep=rep1{\&}type=pdf>
- [97] U. Mohammad and S. Sorour, "Adaptive Task Allocation for Asynchronous Federated and Parallelized Mobile Edge Learning," *arXiv e-prints*, May 2019. [Online]. Available: <https://arxiv.org/abs/1905.01656>
- [98] C. Xie, O. Koyejo, and I. Gupta, "Asynchronous Federated Optimization," 2019. [Online]. Available: <https://arxiv.org/abs/1903.03934>

- [99] U. Mohammad, S. Sorour, and M. Hefeida, “Task allocation for mobile federated and offloaded learning with energy and delay constraints,” in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, Jun 2020, pp. 1–6. [Online]. Available: <https://ieeexplore.ieee.org/document/9145450/>
- [100] J. Park and S. Boyd, “General Heuristics for Nonconvex Quadratically Constrained Quadratic Programming,” Mar 2017. [Online]. Available: <http://arxiv.org/abs/1703.07870>
- [101] U. Mohammad, S. Sorour, and M. Hefeida, “Optimal Task Allocation for Mobile Edge Learning with Global Training Time Constraints,” *arXiv e-prints*, pp. 1–6, Jun 2020. [Online]. Available: <http://arxiv.org/abs/2006.07402>

Appendix A: Proof of Theorem 1

The Lagrangian function of the relaxed problem is expressed as:

$$L(\mathbf{x}, \lambda, \nu, \alpha) = -\tau + \sum_{k=1}^K \lambda_k (C_k^2 \tau d_k + C_k^1 d_k + C_k^0 - T) + \nu \left(\sum_{k=1}^K d_k - d \right) - \alpha_0 \tau - \sum_{k=1}^K \alpha_k d_k \quad (\text{A.1})$$

where the λ_k 's $k \in \mathcal{K}$, ν , and α_0/α_k $k \in \mathcal{K}$, are the Lagrangian multipliers associated with the time constraints of the K learners in (5.3a), the total batch size constraint in (5.3b), and the non-negative constraints of all the optimization variables in (5.3c) and (5.3d), respectively.

Then, from the KKT optimality conditions, we have the following relations:

$$C_k^2 \tau^* d_k^* + C_k^1 d_k^* + C_k^0 - T \leq 0, \quad k = 1, \dots, K \quad (\text{A.2})$$

$$\alpha_0^*, \alpha_k^*, \text{ and } \lambda_k^* \geq 0 \quad k = 1, \dots, K \quad (\text{A.3})$$

$$\lambda_k^* (C_k^2 \tau^* d_k^* + C_k^1 d_k^* + C_k^0 - T) = 0, \quad k = 1, \dots, K \quad (\text{A.4})$$

$$-\alpha_0^* \tau^* = 0 \quad (\text{A.5})$$

$$-\alpha_k^* d_k^* = 0 \quad k = 1, \dots, K \quad (\text{A.6})$$

$$\sum_{k=1}^K d_k^* - d = 0 \quad (\text{A.7})$$

$$-\nabla \tau^* + \sum_{k=1}^K \lambda_k^* \nabla (C_k^2 \tau^* d_k^* + C_k^1 d_k^* + C_k^0 - T) + \nu^* \nabla \left(\sum_{k=1}^K d_k^* - d \right) - \alpha_0^* \nabla \tau^* - \nabla \left(\sum_{k=1}^K \alpha_k^* d_k^* \right) = 0 \quad (\text{A.8})$$

From the conditions in (A.2), we can see that the batch size at user k must satisfy (5.4). Moreover, it can be inferred from (A.4) that the bound in (5.4) holds with equality for

any learner k having $\lambda_k^* \geq 0$. The upper bound will be equal to the optimal solution (i.e. strong duality will hold) for some feasible τ^* when strictly speaking, $\lambda_k^* > 0$ for $k = 1, \dots, K$ because in that case, the second term in (A.4) must be equal to zero. By re-writing the bound on d_k^* in (5.4) as an equality and substituting it back in (A.7), we have the following relation:

$$d = \sum_{k=1}^K d_k^* = \sum_{k=1}^K \left[\frac{T - C_k^0}{\tau^* C_k^2 + C_k^1} \right] = \sum_{k=1}^K \left[\frac{a_k}{\tau^* + b_k} \right] \quad (\text{A.9})$$

The expression on the right-most hand-side has the form of a partial fraction expansion of a rational polynomial function of τ^* where $a_k, b_k \in \mathcal{R}^{++}$. Therefore, we can expand (A.9) in the following way:

$$\begin{aligned} \frac{a_1}{\tau^* + b_1} + \frac{a_2}{\tau^* + b_2} + \dots + \frac{a_k}{\tau^* + b_k} + \dots + \frac{a_K}{\tau^* + b_K} = \\ \frac{1}{(\tau^* + b_1)(\tau^* + b_2) \dots (\tau^* + b_k) \dots (\tau^* + b_K)} \times \\ \left[a_1(\tau^* + b_2)(\tau^* + b_3) \dots (\tau^* + b_k) \dots (\tau^* + b_K) + \right. \\ a_2(\tau^* + b_1)(\tau^* + b_3) \dots (\tau^* + b_k) \dots (\tau^* + b_K) + \dots + \\ \left. a_k(\tau^* + b_1)(\tau^* + b_2) \dots (\tau^* + b_{k-1})(\tau^* + b_{k+1}) \dots (\tau^* + b_K) \right. \\ \left. + \dots + a_K(\tau^* + b_1)(\tau^* + b_2) \dots (\tau^* + b_{K-1}) \right] \quad (\text{A.10}) \end{aligned}$$

Finally, the expanded form can be cleaned up in the form of a rational function with respect to τ , which is equal to the total dataset size d .

$$d = \frac{\sum_{k=1}^K a_k \prod_{\substack{l=1 \\ l \neq k}}^K (\tau^* + b_l)}{\prod_{k=1}^K (\tau^* + b_k)} \quad (\text{A.11})$$

Please note that the degrees of the numerator and denominator will be $K - 1$ and K , respectively. Furthermore, the poles of the system will be $-b_k$, and, since $b_k \geq 0$, the system will be stable. Furthermore, $\tau^* = -b_k$ is not a feasible solution for the problem, because it

is eliminated by the $\tau \geq 0$ constraint. Therefore, we can re-write (A.11) as shown in (5.5). By solving this polynomial, we obtain a set of solutions for τ^* , one of them is feasible. The problem being non-convex, this feasible solution τ^* will constitute the upper bound to the solution of the relaxed problem.

As a last step, to ensure that the solution set is feasible, it must be noted that according to (A.5) and (A.6), α_0^* and $\alpha_k^* \forall k$ must be equal to 0. Expanding the vanishing gradient condition in (A.8), it can be shown that the following two relations can be derived (representing $K + 1$ equations):

$$\lambda_k^* C_k^2 \tau^* + \lambda_k^* C_k^1 + \nu^* = \alpha_k^*, \quad k \in \mathcal{K} \quad (\text{A.12})$$

$$-1 + \sum_{k=1}^K \lambda_k^* C_k^2 d_k^* = \alpha_0^* \quad (\text{A.13})$$

By setting $\alpha_0^* = 0$ and $\alpha_k^* = 0$ for $k = 1, \dots, K$, we can write λ_k^* in terms of ν^* as shown in (A.14) and substitute the resulting expression in (A.13) to find ν^* using the values of d_k^* and τ^* obtained from (5.4) and (5.5), respectively.

$$\lambda_k^* = -\frac{\nu^*}{C_k^2 \tau^* + C_k^1}, \quad k \in \mathcal{K} \quad (\text{A.14})$$

$$\nu^* = -\frac{1}{\sum_{k=1}^K \frac{C_k^2 d_k^*}{C_k^2 \tau^* + C_k^1}} \quad (\text{A.15})$$

The values of λ_k^* for $k = 1, \dots, K$ can be obtained by back-substituting ν^* in (A.14). As one can observe, as long as there exists a τ^* greater than zero, ν^* will be negative and hence, λ_k^* for $k = 1, \dots, K$ will be strictly greater than zero. Hence, as long as there exists a $\tau^* > 0$ in the feasible set such that $d_k^* > 0$, there will exist a set of $\lambda_k^* > 0$ for $k = 1, \dots, K$. This fact can be used to verify the feasibility of the solution. This step is also helpful when there may exist multiple values of τ greater than zero for choosing the optimal τ^* .

Appendix B: Proof of Theorem 2

From the KKT optimality conditions, we have the following condition on the Lagrangian in (A.1):

$$\begin{aligned} \nabla L_{z, \tau_k^*, d_k^*} \forall k \in \mathcal{K} = & \nabla z + \sum_{k=1}^K \lambda_k \nabla (C_k^2 \tau_k^* d_k^* + C_k^1 d_k^* + C_k^0 - T) - \sum_{k=1}^K \nabla \alpha_k \tau_k^* + \\ & \sum_{k=1}^K \nu_k \nabla (-d_k^* + d_l) + \sum_{k=1}^K \nu'_k \nabla (d_k^* - d_u) + \sum_{n=1}^N \mu_n \nabla (-z + \tau_{c_{n,1}}^* - \tau_{c_{n,2}}^*) + \\ & \sum_{n=1}^N \mu'_n \nabla (-z - \tau_{c_{n,1}}^* + \tau_{c_{n,2}}^*) + \omega \nabla \left(\sum_{k=1}^K d_k^* - d \right) = 0 \quad (\text{B.1}) \end{aligned}$$

After taking the partial derivative of the Lagrangian function in (B.1) with respect to τ_k and d_k^* , the following sets of equations can be obtained as shown in (B.2) and (B.3).

$$\lambda_k C_k^2 \tau_k^* + \lambda_k C_k^1 + \nu_k + \nu'_k + \omega = 0, \quad \forall k \quad (\text{B.2})$$

$$\lambda_k C_k^2 d_k^* + u_k + u'_k + \alpha_k = 0, \quad \forall k \quad (\text{B.3})$$

Solving for τ_k^* and d_k^* will give the results shown in (5.12) and (5.13). The procedure to obtain u_k and u'_k is given below.

B.1 Obtaining \mathbf{u} and \mathbf{u}'

The maximum staleness constraint in (5.9a) can be re-written as two separate inequalities as shown below:

$$-z + \tau_k - \tau_l \leq 0 \quad (\text{B.4})$$

$$-z - \tau_k + \tau_l \leq 0 \quad (\text{B.5})$$

The k^{th} element of the vector \mathbf{u} denoted as u_k is associated with the Lagrange multipliers of the maximum staleness constraint inequality in (B.4) whereas u'_k is associated with the

inequality in (B.5), and the way to calculate them is shown in (B.6) and (B.7), respectively.

$$u_k = \nabla_{\tau_k} \sum_{n=1}^N \mu_n (-z + \tau_k - \tau_l) \quad (\text{B.6})$$

$$u'_k = \nabla_{\tau_k} \sum_{n=1}^N \mu'_n (-z - \tau_k + \tau_l) \quad (\text{B.7})$$

As defined earlier, $k \in \mathcal{K}$ and $l \in \{\mathcal{K} \mid l > k \forall k\}$.

In this case, after some manipulations, u_k can be defined as the following:

$$u_k = \sum_{j=n_k}^{N_k} \mu_j - \sum_{j=1}^{K-1} \mu_{n_j+(k-j)} \quad (\text{B.8})$$

The start index and end indices of the first summation in (B.8) are defined in (B.9) and (B.10), respectively.

$$n_k = 1 + \sum_{m=0}^{k-1} (K - m) \quad (\text{B.9})$$

$$N_k = \sum_{m=1}^k (K - m) \quad (\text{B.10})$$

On the other hand, u'_k can be simply be defined as the following:

$$u'_k = - \sum_{j=n_k}^{N_k} \mu'_j + \sum_{j=1}^{K-1} \mu'_{n_j+(k-j)} \quad (\text{B.11})$$

Appendix C: Proof of Theorem 3

The optimization variables are denoted by \mathbf{x} where $\mathbf{x} = [\tau \ d_1 \ d_2 \ \dots \ d_k \ \dots \ d_K]^T$. Then, the relaxed program in (6.10) can be re-written in the form of a QCQP as shown below:

$$\min_{\mathbf{x}} \quad \mathbf{x}^T \mathbf{F} \mathbf{x} + \mathbf{f}^T \mathbf{x} + f_0 \quad (\text{C.1})$$

$$\text{s.t.} \quad \mathbf{x}^T \mathbf{P}_k \mathbf{x} + \mathbf{p}_k^T \mathbf{x} + p_k^0 \leq 0, \quad \forall k \in \mathcal{K} \quad (\text{C.1a})$$

$$\mathbf{x}^T \mathbf{Q}_k \mathbf{x} + \mathbf{q}_k^T \mathbf{x} + q_k^0 \leq 0, \quad \forall k \in \mathcal{K} \quad (\text{C.1b})$$

$$\mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{a}^T \mathbf{x} + a_0 \leq 0 \quad (\text{C.1c})$$

$$\mathbf{x}^T \bar{\mathbf{A}} \mathbf{x} + \bar{\mathbf{a}}^T \mathbf{x} + \bar{a}_0 \leq 0 \quad (\text{C.1d})$$

$$\mathbf{x}^T \mathbf{U} \mathbf{x} + \mathbf{U}^T \mathbf{x} + u_0 \leq 0 \quad (\text{C.1e})$$

$$\mathbf{x}^T \mathbf{V}_k \mathbf{x} + \mathbf{v}_k^T \mathbf{x} + v_k^0 \leq 0, \quad \forall k \in \mathcal{K} \quad (\text{C.1f})$$

Constraints (C.1a) and (C.1b) represent the time and energy constraints, respectively, and constraints (C.1c) and (C.1d) represent the total batch size constraint as two inequalities. The non-negative constraints on τ and d_k are given in (C.1e) and (C.1f), respectively. The constants associated with the time and energy constraints can be defined as $p_k^0 = C_k^0 - T$ and $q_k^0 = G_k^0 - E_k^0$, respectively, $\forall k$. The variables $a_0 = -d$, $\bar{a}_0 = d$ and $v_k^0 = d_l, \forall k$ whereas $u_0 = 0$ and $f_0 = 0$.

$$\mathbf{f} = [-1 \ 0 \ 0 \ \dots \ C_k^1 \ \dots \ 0]^T \quad (\text{C.2})$$

$$\mathbf{p}_k = [0 \ 0 \ 0 \ \dots \ C_k^1 \ \dots \ 0]^T, \forall k$$

$$\mathbf{q}_k = [0 \ 0 \ 0 \ \dots \ G_k^1 \ \dots \ 0]^T, \forall k$$

$$\mathbf{a} = [0 \ 1 \ 1 \ \dots \ 1 \ \dots \ 1]^T$$

$$\bar{\mathbf{a}} = [0 \ -1 \ -1 \ \dots \ -1 \ \dots \ -1]^T$$

$$\mathbf{u} = [-1 \ 0 \ 0 \ \dots \ 0 \ \dots \ 0]^T$$

$$\mathbf{v}_k = [0 \ 0 \ 0 \ \dots \ -1 \ \dots \ 0]^T, \forall k$$

The coefficients associated with the linear terms in the objective and constraints (\mathbf{f} and \mathbf{p}_k , \mathbf{q}_k , \mathbf{a} , $\bar{\mathbf{a}}$, \mathbf{u} , and \mathbf{v}_k , respectively) are given in (C.2) as column vectors.

The quadratic matrices associated with the time and energy constraints, \mathbf{P}_k and \mathbf{Q}_k , respectively, are given in (C.3) and (C.4).

$$\mathbf{P}_k(i, j) = \begin{cases} 0.5C_k^2, & \text{if } i = 1 \ \& \ j = k + 1 \\ & i = k + 1 \ \& \ j = 1 \\ 0, & \text{otherwise} \end{cases} \quad (\text{C.3})$$

$$\mathbf{Q}_k(i, j) = \begin{cases} 0.5G_k^2, & \text{if } i = 1 \ \& \ j = k + 1 \\ & i = k + 1 \ \& \ j = 1 \\ 0, & \text{otherwise} \end{cases} \quad (\text{C.4})$$

The remaining quadratic matrices \mathbf{F} , \mathbf{A} , $\bar{\mathbf{A}}$, \mathbf{U} and \mathbf{V}_k are all $\mathbf{0}_{(K+1) \times (K+1)}$.

The functions $\mathbf{F}^2(\mathbf{\Gamma})$, $\mathbf{f}^1(\mathbf{\Gamma})$ and $f_0(\mathbf{\Gamma})$ can now be defined as [100]:

$$\mathbf{F}^2(\mathbf{\Gamma}) = \sum_{k=1}^K \lambda_k \mathbf{P}_k + \gamma_k \mathbf{Q}_k \quad (\text{C.5})$$

$$\mathbf{f}^1(\mathbf{\Gamma}) = \sum_{k=1}^K (\lambda_k \mathbf{p}_k + \gamma_k \mathbf{q}_k + \nu_k \mathbf{v}_k) + \alpha \mathbf{a} + \bar{\alpha} \bar{\mathbf{a}} + \omega \mathbf{u} \quad (\text{C.6})$$

$$f_0(\mathbf{\Gamma}) = \sum_{k=1}^K (\lambda_k p_k^0 + \gamma_k q_k^0 + \nu_k v_k^0) + \alpha a_0 + \bar{\alpha} \bar{a}_0 \quad (\text{C.7})$$

Appendix D: Proof of Lemma 1

Let's assume the orchestrator will train the MEL model for a total of L epochs where a global aggregation can occur at any update step $l + 1$ for $l = 0, \dots, L - 1$, while local updates occur at each step $l + 1$. In the synchronous model of [76, 77, 87], between any two global updates, each learner k performs τ updates whereas it performs τ_k updates in the proposed HA-Asyn. Let us assume, to facilitate the analysis, that the global aggregations occur at integer multiples of the $\tau_m = \max(\tau_k)$; which represents the maximum possible updates that would be done by the highest performing learner. We can now define the interval $[g]$ over $[g\tau_m, (g + 1)\tau_m]$ for $g = 0, 1, 2, \dots$

Assuming a global aggregation were to occur at each iteration $l + 1$, let us define an auxiliary set of global model parameters denoted by $\hat{\mathbf{w}}_{[g]}$ for any interval $[g]$ which would be calculated if a global update step took place. (Note that at the beginning of any interval $[g]$, a global aggregation does occur.) Then, the update rule for this auxiliary set can be given by:

$$\hat{\mathbf{w}}_{[g]}[l + 1] = \hat{\mathbf{w}}_{[g]}[l] - \eta \nabla F(\hat{\mathbf{w}}_{[g]}[l]) \quad (\text{D.1})$$

We assume that the local loss function at learner k given by $F_k(\mathbf{w})$ is:

1. convex
2. ρ -Lipschitz $\|F_k(\mathbf{w}) - F_k(\bar{\mathbf{w}})\| \leq \rho \|\mathbf{w} - \bar{\mathbf{w}}\|$
3. β -smooth $\|\nabla F_k(\mathbf{w}) - \nabla F_k(\bar{\mathbf{w}})\| \leq \beta \|\mathbf{w} - \bar{\mathbf{w}}\|$

These assumptions will hold for ML models with convex loss function such as linear regression and SVM. By simulations, we will show that the proposed solutions work for non-convex models such as DNN with ReLU activation. It has been shown that for such a model, the difference between the global optimal model and the auxiliary model for any iteration $l + 1$ within an interval g , for $l = 0, \dots, L - 1$ and $g = 0, 1, 2, \dots$, can be bounded by:

$$\|\mathbf{w}[l + 1] - \hat{\mathbf{w}}_g[l + 1]\| \leq \|\mathbf{w}[l] - \hat{\mathbf{w}}_g[l]\| + \frac{\eta\beta}{d} \sum_{k=1}^K f_k[l - g\tau_m] \quad (\text{D.2})$$

The function $f_k(t) = \frac{\delta_k}{\beta} [(\eta\beta + 1)^t - 1]$ which relates the local model $\mathbf{w}_k \forall k \in \mathcal{K}$ to the auxiliary model set $\hat{\mathbf{w}}_g[l]$ as follows:

$$\|\mathbf{w}_k[l+1] - \hat{\mathbf{w}}_g[l+1]\| \leq f_k(l - g\tau_m) \quad (\text{D.3})$$

Assume that each learner has performs τ_k updates and for a particular interval, $l \in [g\tau_k, (g+1)\tau_k] \forall k \in \mathcal{K}$ where $l+1$ is the progression of the index of the best performing learner. Then, the upper bound on the model divergence can be given by the following expression:

$$\|\mathbf{w}[l+1] - \hat{\mathbf{w}}[l+1]\| \leq \|\mathbf{w}[l] - \hat{\mathbf{w}}[l]\| + \frac{\eta\beta}{d} \sum_{k=1}^K f_k(l - g\tau_k) \quad (\text{D.4})$$

The learning rate η can be selected such that $0 \leq \eta\beta \leq 1$ which is necessary to satisfy the assumptions in [12]. In that case $1 \leq \eta\beta + 1 \leq 2$ and because $t_k = l - g\tau_k \geq 0$, the function $f_k(t_k)$ grows exponentially greater as t increases because the dominating term is $(\eta\beta + 1)^{t_k}$. So, as the staleness $s_{k,l} = \tau_k - \tau_l \quad k, l \in \mathcal{K} \mid l \neq k$ increases, $f_k(t_k)$ will be higher for more learners which will result in a higher bound on the divergence. Therefore, as the auxiliary model diverges further from the globally optimal model, the loss will increase and hence, it can be expected that the accuracy will decrease.

To sum up, the learning will progress faster as $l+1$ is higher which can be done maximizing the τ_m . Alternatively, if we want to keep $t_k \forall k$ low, we can maximize the $\min(\tau_k)$ while controlling staleness $s_{k,l}$.

Appendix E: Proof of Theorem 4

In this proof, we will show that the objective function O is strictly convex under certain conditions because $\frac{\partial^2 O}{\partial \tau^2} > 0$. Because τ is a positive integer, the optimal value τ^* is the argument that minimizes $O(\tau)$. The reciprocal of $L(\tau)$ in (7.13) can be separated into two terms $M(\tau)$ and $N(\tau)$ as follows:

$$M(\tau) = \frac{d}{KT} \frac{\prod_{k=1}^K (\tau + b_k)}{\sum_{k=1}^K \tau \prod_{\substack{l=1 \\ l \neq k}}^K (\tau + b_l)} \quad (\text{E.1a})$$

$$N(\tau) = \frac{1}{KT} \frac{\sum_{k=1}^K a_k \prod_{\substack{l=1 \\ l \neq k}}^K (\tau + b_l)}{\sum_{k=1}^K \tau \prod_{\substack{l=1 \\ l \neq k}}^K (\tau + b_l)} \quad (\text{E.1b})$$

Therefore, the objective function can be re-written as $O(\tau) = O_1(\tau) + O_2(\tau)$ where $O_1(\tau) = M(\tau)P(\tau)$ and $O_2(\tau) = N(\tau)P(\tau)$. Moreover, the term $P(\tau)$ can be written as the reciprocal of $\nu(\tau)$ where

$$\nu(\tau) = A - B \frac{C^\tau - 1 - (C - 1)\tau}{\tau} \quad (\text{E.2})$$

The constants $A = \eta \left(1 - \frac{\beta\eta}{2}\right)$, $B = \frac{\delta}{\beta} \frac{\rho}{\omega\epsilon^2}$ and $C = \eta\beta + 1$. We can say that $B = \frac{\delta}{\beta} B_0$ where $B_0 = \frac{\rho}{\omega\epsilon^2} > 0$ is a control parameter that can be set empirically.

For brevity, we will represent $f(\tau)$ as f where f may be O , O_1 , O_2 , M , N or P . We will also represent $\frac{\partial f}{\partial \tau}$ and $\frac{\partial^2 f}{\partial \tau^2}$ as f' and f'' , respectively. Using this new notation, O'' can be given as follows:

$$O'' = P(M'' + N'') + 2P'(M' + N') + (M + N)P'' \quad (\text{E.3})$$

By definition $M > 0$ and $N > 0$ because they are related to the time consumed by user k , and $P > 0$ assuming that the constraint in (7.9f) holds. Hence, if we can show that each of the three terms in (E.3) are strictly positive, then O will be strictly convex. We need to show that $P'' > 0$ and $M'' + N'' > 0$. Furthermore, if we can show that $M' + N'$ and P' follow the same sign, $O'' > 0$.

The first derivatives of M , N , and P can be given by:

$$M' = -\frac{d}{KT} \frac{\sum_{k=1}^K \frac{b_k}{(\tau+b_k)^2}}{\left(\sum_{k=1}^K \frac{\tau}{(\tau+b_k)}\right)^2} \quad (\text{E.4a})$$

$$N' = \frac{M'}{d} - \frac{1}{KT} \frac{2 \sum_{k=1}^K \frac{b_k}{(\tau+b_k)^3}}{\left(\sum_{k=1}^K \frac{\tau}{(\tau+b_k)}\right)^2} \quad (\text{E.4b})$$

$$P' = -B \frac{C^\tau [1 - (\ln C)\tau] - 1}{[A\tau - B(C^\tau - 1 - (C - 1)\tau)]^2} \quad (\text{E.4c})$$

The variables $a_k = \frac{C_k^1}{C_k^2}$ and $b_k = \frac{C_k^0}{C_k^2}$, respectively, and both are positive quantities. For M' , the first term outside the square bracket is always negative whereas the term inside is a sum of positive quantities whereas N' is a sum of negative quantities. Therefore, $M' < 0$ and $N' < 0$. Hence, we need to show that $P' < 0$ or find the domain on which $P' < 0$.

The complete expressions of M'' and N'' can be given by:

$$M'' = \frac{d}{KT} \frac{1}{\left(\sum_{k=1}^K \frac{\tau}{(\tau+b_k)}\right)^2} \left[2 \sum_{k=1}^K \frac{b_k}{(\tau+b_k)^3} + \sum_{k=1}^K \frac{b_k}{(\tau+b_k)^2} \right] \quad (\text{E.5a})$$

$$N'' = \frac{1}{KT} \sum_{k=1}^K \frac{a_k}{(\tau+b_k)^3 \left(\sum_{l=1}^K \frac{\tau}{(\tau+b_l)}\right)^4} \left[\sum_{l=1}^K \frac{\tau}{\tau+b_l} + 2(\tau+b_k) \sum_{l=1}^K \frac{1}{(\tau+b_l)^2} + \sum_{l=1}^K \frac{2}{(\tau+b_l)^3} \right] \quad (\text{E.5b})$$

It can be shown that M'' and N'' are strictly greater than zero because they are both a sums of positive terms. Hence, the left-most term in (E.3) is strictly positive. Thus we need to show that $P'' > 0$ or find the domain for which this is true.

Although the complete expression for P'' is omitted for brevity, it can be shown that the necessary and sufficient conditions to achieve $P' < 0$ and $P'' > 0$ is 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 can show that we need to check the following inequality:

$$\tau [(\ln C)\tau - C(\ln C) - 1 + C] > 0 \quad (\text{E.6})$$

We know that as long as a feasible $\tau^* > 0$ is found, we need to satisfy the second term enclosed by the square brackets. Writing τ as a function of C and, we can see that the condition on τ is the following:

$$\tau > \frac{C(\ln C) + 1 - C}{\ln C} \quad (\text{E.7})$$

Recall that $C = \eta\beta + 1$ where η is chosen such that $\eta\beta \leq 1$, and $\eta, \beta > 0$. Hence, it follows that $1 \leq \eta\beta + 1 \leq 2$. If we plot, $f(C) = \frac{C(\ln C) + 1 - C}{\ln C}$ against the domain of C , we notice that for $\nu'' < 0$ and hence, for P' to be strictly negative and P'' to be strictly positive, it is sufficient for $\tau > 0$. Hence, we have now proved that $O(\tau)$ is strictly convex because $\frac{\partial^2 O}{\partial \tau^2} > 0$ as long as τ is a positive integer and the ML model variables are selected as defined.

Appendix F: List of Published, Accepted, or Submitted Papers

F.1 Journal Papers

The following journal papers have been submitted as part of this dissertation:

1. U Mohammad, S Sorour, and M Hefeida, “Dynamic Task Allocation for Mobile Edge Learning,” submitted to the *IEEE Transactions on Mobile Computing*.
2. U Mohammad, S Sorour, and M Hefeida, “Asynchronous Task Allocation for Federated and Parallelized Mobile Edge Learning,” submitted to the *IEEE Transactions on Vehicular Technology* as a correspondence paper.
3. U Mohammad, S Sorour, and M Hefeida, “Asynchronous Task Allocation for Mobile Edge Learning with Time and Energy Constraints,” submitted to the *IEEE Transactions on Parallel and Distributed Systems: Special Issue on Parallel and Distributed Computing for AI/ML/DL*.

F.2 Conference Papers

The following papers have been either published in (or submitted to) peer-reviewed IEEE Communications Society (ComSoc) conferences:

1. Umair Mohammad, Sameh Sorour, and Mohamed Hefeida, “Optimal Task Allocation for Mobile Edge Learning with Time Constraints,” submitted to the *2020 IEEE Global Communications Conference (IEEE GLOBECOM 2020)*.
2. Umair Mohammad, Sameh Sorour, and Mohamed Hefeida, “Task Allocation for Federated Mobile Edge Learning with Delay and Energy Constraints,” in Proceedings of the *2020 IEEE International Conference on Communications Workshops on Edge Machine Learning for 5G Mobile Networks and Beyond (IEEE ICC’20 Workshop - EML5G)*.


3. U. Y. Mohammad and S. Sorour, “Adaptive Task Allocation for Mobile Edge Learning,” in Proceedings of the *2019 IEEE Wireless Communications and Networking Conference Workshop (WCNCW)*.
4. U. Y. Mohammad and S. Sorour, “Multi-Objective Resource Optimization for Hierarchical Mobile Edge Computing,” in Proceedings of the *2018 IEEE Global Communications Conference: Mobile and Wireless Networks (Globecom2018 MWN)*.

F.3 Other Journal Publications

The following papers two papers were published as part of group projects not related directly to the dissertation:

1. F Madkour, U Mohammad, S Sorour, M Hefeida and A Abdel-Rahim, “A Vendor-Independent Reliability Testing Model for Vehicle-to-Infrastructure Communications,” published in the *Transportation Research Record (TRR) Journal*.
2. Tarar, A.; Mohammad, U.; Srivastava, S., “Wearable skin sensors and its challenges: A review of transdermal, optical, and mechanical sensors,” published in *Biosensors (MDPI)*.

Appendix G: IEEE Formal Reuse Licenses



Multi-Objective Resource Optimization for Hierarchical Mobile Edge Computing

Conference Proceedings: 2018 IEEE Global Communications Conference (GLOBECOM)

Author: Umair Yaqub

Publisher: IEEE

Date: Dec. 2018

Copyright © 2018, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:


- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK
CLOSE WINDOW



Adaptive Task Allocation for Mobile Edge Learning

Conference Proceedings: 2019 IEEE Wireless Communications and Networking Conference Workshop (WCNCW)

Author: Umair Mohammad

Publisher: IEEE

Date: April 2019

Copyright © 2019, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK
CLOSE WINDOW



Task Allocation for Mobile Federated and Offloaded Learning with Energy and Delay Constraints

Conference Proceedings: 2020 IEEE International Conference on Communications Workshops (ICC Workshops)

Author: Umair Mohammad

Publisher: IEEE

Date: June 2020

Copyright © 2020, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK

CLOSE WINDOW