Resource Allocation for Federated Learning Enabled Edge Intelligence

LIM WEI YANG BRYAN

Interdisciplinary Graduate School

(Alibaba-NTU Joint Research Institute)

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

2022

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

15/10/2021
.....
Date

.....
LIM WEI YANG BRYAN

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

15/10/2021

.....

Date

.....

Prof. Miao Chunyan

Authorship Attribution Statement

This thesis contains materials from 4 papers published in the following peer-reviewed journals in which I am listed as an author.

Chapter 2 is published as Lim, W.Y.B., Luong, N.C., Hoang, D.T., Jiao, Y., Liang, Y.C., Yang, Q., Niyato, D. and Miao, C., 2020. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 22(3), pp.2031-2063.

The contributions of the co-authors are as follows:

- Prof. Niyato and Prof. Miao provided the initial research direction.
- I conducted the literature review and prepared the manuscript draft.
- Mr. Luong, Dr. Hoang, and Dr. Jiao searched for relevant papers, plotted figures, and provided feedback from their areas of expertise.
- Prof. Ying, Prof. Yang, Prof. Niyato, and Prof. Miao contributed to the revision and provided expertise to refine the paper.

Chapter 3 is published as Lim, W.Y.B., Xiong, Z., Kang, J., Niyato, D., Leung, C.S., Miao, C. and Shen, S., 2020. When Information freshness meets service latency in federated learning: A task-aware incentive scheme for smart industries. *IEEE Transactions on Industrial Informatics*, Early Access Article.

The contributions of the co-authors are as follows:

- Prof. Niyato, Prof. Leung, and Prof Miao provided the initial research direction.
- I prepared the manuscript draft, proposed the system model, solved the optimization problem, and implemented the simulations.
- Prof. Xiong and Dr. Kang provided help in the mathematical formulation.
- Prof. Niyato, Prof. Leung, Prof Miao, and Prof. Shen edited and refined the paper.

Chapter 4 is published as Lim, W.Y.B., Huang, J., Xiong, Z., Kang, J., Niyato, D., Hua, X.S., Leung, C. and Miao, C., 2021. Towards federated learning in UAV-enabled internet of vehicles: A multi-dimensional contract-matching approach. *IEEE Transactions on Intelligent Transportation Systems*, 22(8), pp.5140-5154.

The contributions of the co-authors are as follows:

- Prof. Niyato and Prof. Miao provided the initial research direction.
- I prepared the manuscript draft, proposed the system model, solved the optimization problem, and implemented the simulations.

- Mr. Huang, Prof. Xiong, and Dr. Kang provided help in the mathematical formulation.
- Prof. Niyato, Prof. Hua, Prof. Leung, and Prof. Miao edited and refined the paper.

Chapter 5 is published as Lim, W.Y.B., Ng, J.S., Xiong, Z., Jin, J., Zhang, Y., Niyato, D., Leung, C.S. and Miao, C., 2021. Decentralized Edge Intelligence: A Dynamic Resource Allocation Framework for Hierarchical Federated Learning. *IEEE Transactions on Parallel and Distributed Systems*, 33(3), pp.536-550.

The contributions of the co-authors are as follows:

- Prof. Niyato and Prof. Leung provided the initial research direction.
- I prepared the manuscript draft, proposed the system model, solved the optimization problem, and implemented the simulations.
- Ms. Ng, Prof. Xiong, Dr. Zhang, and Dr. Jin provided help in the mathematical formulation.
- Prof. Niyato, Prof. Leung, and Prof. Miao edited and refined the paper.

15/10/2021

.....

Date

.....

LIM WEI YANG BRYAN

Acknowledgements

I wish to express my greatest gratitude to my advisors and mentors, Prof. Miao, Prof. Niyato, Prof. Leung, Prof. Xu, and Dr. Wang Di for their guidance and supervision, without which this journey will be impossible.

My heartfelt gratitude goes to my friends, Jer Shyuan, Hongchao, Jonathan, and Han Yue, all of whom gave me great memories and laughter along the way. My now-graduated seniors, Prof. Zehui Xiong and Prof. Jiawen Kang also provided me with great advice and accompaniment in this journey. I will always reminisce the "good old days" we spent together, in spite of the unfortunate COVID years.

I am ever thankful to my parents, Lim Ann and Toh Yen Choo, and my wife, Foo Feng Lin, for their unconditional support and love. Last but not least, this thesis is dedicated to the newest member of our family, my son Theodore Lim Chen Huan.

Contents

List of Figures

List of Tables

Abstract

The confluence of Edge Computing and Artificial Intelligence (AI) has driven the rise of Edge Intelligence, which leverages the storage, communication, and computation capabilities of end devices and edge servers to empower AI implementation at scale closer to where data is generated. An enabling technology of Edge Intelligence is the privacy preserving machine learning paradigm known as Federated Learning (FL). In FL, end users carry out model training locally before transmitting the model parameters or gradient updates, rather than the raw data, to a model owner for aggregation. Amid the increasingly stringent privacy regulations, FL has enabled the development of applications that have to be built using sensitive user data and will continue to revolutionize service delivery in the Finance, Internet of Things (IoT), healthcare, and transport industries, among others.

However, the implementation of FL is envisioned to involve thousands of heterogeneous distributed end devices that differ in terms of communication and computation resources, as well as the levels of willingness to participate in the collaborative model training process. The potential node failures, device dropouts, and stragglers effect are key bottlenecks that impede the effective, sustainable, and scalable implementation of FL.

In this thesis, I will first present a tutorial and survey on FL and highlight its role in enabling Edge Intelligence. This tutorial and survey provide readers with a comprehensive introduction to the forefront challenges and state-of-the-art approaches towards implementing FL at the edge. In consideration of resource heterogeneity, I then provide multifaceted solutions towards improving the efficiency of resource allocation for implementing FL at scale amid information asymmetry. In the first study, I propose a vanilla contract-theoretic optimization approach towards balancing the tradeoffs of information freshness and service latency in a federated crowdsensing scenario. In the second study, I devise a multi-dimensional contract-matching approach for optimized resource allocation amid multiple sources of heterogeneities. The first two studies leverage the self-revealing properties of contract

theory to solve the resource allocation problem in the information asymmetric edge network. Through performance evaluation, it is shown that even when the worker types are unknown, the resource allocation in the edge network is optimized. In the third study, in face of bounded rationality and dynamic decisions of the workers, I propose a two-level evolutionary game theoretic and auction approach to allocate and price resources to facilitate efficient edge intelligence. The performance evaluation shows the convergence and uniqueness of solutions, as well as the profit maximization aspect of the proposed solution. The studies presented in the thesis are formulated via the interdisciplinary interplay of concepts derived from network economics, optimization, game theory, and machine learning. Finally, I outline promising research directions for future works.

Chapter 1

Introduction

1.1 Background

The enhanced perception capabilities of state-of-the-art sensors deployed on, e.g., the Internet of Things (IoT), Unmanned Aerial Vehicles (UAV), and smart vehicles have enabled a wealth of raw data to be captured at the edge of the network today. Empowered with the rise of Artificial Intelligence (AI) based models, which outperform conventional hand-engineered methods when there is an abundance of training data, effective model training, e.g, for personalized recommendation, facial recognition, and trajectory optimization, has been successfully deployed on end devices, e.g., smartphones. However, with the ubiquity of interconnected end devices, the burden on communication networks necessitates the proposal of Edge Computing as an alternative to current cloud-centric approaches in which raw data have to be transmitted to centralized remote servers for processing.

The confluence of Edge Computing and AI gives rise to Edge Intelligence, which leverages the storage, communication, and computation capabilities of end devices and edge servers to enable edge caching, model training, and inference [1] closer to where data are produced. Specifically, edge caching refers to the collection and storage of data, edge model training refers to AI model training at the edge, and edge inference refers to the deployment of trained models at the edge and end devices for computation of desired outputs, e.g., image classification, given some input data.

The entire pipeline from data collection to model training is often referred to as *Edge Learning*. The Edge Learning framework offers two main benefits over that of the cloud-centric approach. Firstly, the raw data collected by end devices can be processed without the need for the transmission to the remote cloud, thus relieving the burden on communication networks. Secondly, the computation capabilities of proximal edge servers, e.g., roadside units (RSU), can be leveraged to complete time-sensitive tasks, e.g., multi-object detection for autonomous driving.

However, there exist implementation challenges that have to be addressed to enable efficient Edge Learning. Firstly, privacy regulations that govern the sharing of data, e.g., the General Data Protection Regulation (GDPR), are increasingly stringent [2]. Given that Edge Learning involves the utilization of user data for decentralized model training, privacy preserving collaborative learning schemes have to be developed to ensure sustainable participation of end users. Secondly, even though end devices are equipped with improving computation capabilities, they are still much more computationally-constrained relative to edge or cloud servers. As such, effective computation offloading strategies to enable the end-edge-cloud collaboration across the network are required. Thirdly, AI models are increasingly complex and over-parameterized. This necessitates the design of lightweight models for deployment on end devices for computation and communication efficiency.

In response, the confluence of Edge Computing and AI has led to the rise of Edge Intelligence, which leverages the storage, communication, and computation capabilities of end devices and edge servers to enable edge caching, model training, and model inference closer to where data are produced. One of the enabling technologies of Edge Intelligence is the privacy preserving machine learning paradigm known as Federated Learning (FL) [3]. In FL, data owners, i.e., end users, carry out model training locally before transmitting the model parameters or gradient updates, rather than the raw data, to a model owner for aggregation. This enables privacy-preserving collaborative machine learning while leveraging on the computation capabilities of end users. Moreover, model parameters are smaller in size than raw data, thus alleviating the burden on backbone communication networks. FL has enabled the development of several applications that have to be built using sensitive user data, e.g., next-word-prediction models for text messaging applications and Internet of Things (IoT) empowered healthcare solutions. Recently, FL has found many successes in applications for mobile edge computing.

As highlighted by the authors in [4], the increasing complexity and heterogeneity of wireless networks enhance the appeal of adopting a data-driven ML based approach [5] for optimizing system designs and resource allocation decision making for mobile edge networks. However, the private data of users may be sensitive in nature. As such, existing learning based approach can be combined with FL for privacy-preserving applications.

For brevity, the related studies are not fully reviewed. Table 1.1 provides a brief description of the studies that address the four applications of FL for resource optimization at the wireless edge networks:

i *Cyberattack Detection:* The ubiquity of IoT devices and increasing sophistication of cyberattacks [6] imply that there is a need to improve existing cyberattack detection tools. Recently, Deep Learning has been widely successful in cyberattack detection. Coupled with FL, cyberattack detection models can be learned collaboratively while maintaining user privacy.

ii *Edge Caching and Computation Offloading:* Given the computation and storage capacity constraints of edge servers, some computationally intensive tasks of end devices have to be offloaded to the remote cloud server for computation. In addition, commonly requested files or services should be placed on edge servers for faster retrieval, i.e., users do not have to communicate with the remote cloud when they want to access these files or services. As such, an optimal caching and computation offloading scheme can be collaboratively learned and optimized with FL.

iii *Base Station Association:* In a dense network, it is important to optimize base station association so as to limit interference faced by users. However, traditional learning based approaches that utilize user data often assume that such data are centrally available. Given user privacy constraints, an FL based approach can be adopted instead.

iv *Vehicular Networks:* The Internet of Vehicles (IoV) [7] features smart vehicles with data collection, computation and communication capabilities for relevant functions, e.g., navigation and traffic management. However, this wealth of knowledge is again, private and sensitive in nature since it can reveal the driver's location and personal information. An FL based approach

is therefore useful, e.g., in traffic queue length prediction and energy demand in electric vehicle charging stations done at the edge of IoV networks.

However, despite the many advantages and application use cases, the FL network is envisioned to involve thousands of heterogeneous distributed devices, e.g., smartphones and IoT. Given that synchronous cloud-aggregation remains the predominant approach for global model aggregation in FL, node failures, device dropouts, and the stragglers effect due to the computation and communication inefficiencies of end users have proven to be key bottlenecks that impede the effective and scalable implementation of FL. In consideration of resource heterogeneity at the network edge, we provide multifaceted solutions towards improving the efficiency of resource allocation (i.e., computation, communication, and incentive resources) for implementing FL at scale over the wireless edge networks in this thesis. In Section 1.2, we provide a concise summary of our contribution and the organization of the thesis.

1.2 Summary of Contributions and Organization of Thesis

In Chapter 2, we motivate FL as an important paradigm shift towards enabling collaborative machine learning model training at the wireless edge. Then, we provide a concise tutorial on FL implementation and present to the reader a list of useful open-source frameworks that pave the way for future research on FL and its applications. Next, we discuss the unique features of FL relative to a centralized machine learning approach and the resulting implementation challenges. For each of these challenges, we present to the reader a comprehensive discussion of existing solutions and approaches explored in the FL literature. Part of Chapter 2 is published under: Wei Yang Bryan Lim et al. "Federated learning in mobile edge networks: A comprehensive survey." *IEEE Communications Surveys & Tutorials* (2020).

In Chapter 3, we introduce the adoption of FL in an industrial and crowdsensing setting, in which independent data owners may be reluctant to share their private raw data with industrial competitors. The adoption of FL enables collaborative

TABLE 1.1: FL based approaches for mobile edge network optimization.

Applications	**Ref.**	**Description**
Cyberattack Detection	[9]	Cyberattack detection with edge nodes as workers
	[10]	Cyberattack detection with IoT gateways as workers
	[11]	Blockchain to store model updates
Edge caching and computation offloading	[12]	DRL for caching and offloading in UEs
	[13]	DRL for computation offloading in IoT devices
	[14]	Stacked autoencoder learning for proactive caching
	[15]	Greedy algorithm to optimize service placement schemes
Base station assoication	[16]	Deep echo state networks for VR application
	[17]	Mean field game with imitation for cell association
Vehicular networks	[18]	Extreme value theory for large queue length prediction
	[19]	Energy demand learning in electric vehicular networks

model training across a federation of such industrial data owners. In conventional FL studies, it is usually assumed that the worker either has the data ready for model training, or needs to collect the data upon request. As such, the crucial tradeoff between *service latency* and *Age of Information (AoI)* is under-explored. In particular, the AoI is defined as the time elapsed between data collection, i.e., collection or aggregation of data from the distributed Industrial IoT (IIoT) devices, to completion of the FL based training, and thus captures information freshness [8], whereas the service latency is defined as the time elapsed between the initiation of the FL training request to the completion of the FL based training. On one hand, if the data are aggregated only upon request, the information is at its freshest, i.e., the AoI is low. On the other hand, the service latency incurred during data collection can be intolerable especially if the model owner prefers faster task completion. In addition, there exists an incentive mismatch between the workers and model owner in managing this tradeoff. For example, while the model owner may prefer more frequent data updating to ensure a low AoI, the update expense, e.g. in terms of computation, communication, and energy consumption, incurred can be prohibitive for the worker. Moreover, the model owners are unaware of the worker types, i.e., data update cost, due to information asymmetry. To this end, we propose a contract-theoretic task-aware incentive mechanism to motivate workers to update the data accordingly in consideration of the different preferences of the model owner towards AoI and service latency. We leverage the self-revealing properties of the contract-theoretic incentive mechanism design to appropriately reward workers based on their data updating cost amid information asymmetry. Chapter 3 is published under: Wei Yang Bryan Lim et al. "When Service Latency Meets Information Freshness in Federated Learning," *IEEE Transactions on Industrial Informatics* (2020).

In Chapter 4, motivated by the increasing popularity of UAVs as mobile base stations, aerial sensing devices, or enablers of Intelligent Transport Systems, we devise a multi-dimensional contract-matching resource allocation scheme for a network with independent Drones-as-a-Service (DaaS) operators collaborating in data collection and FL model training as a service provision. There exists an incentive mismatch between the model owner and the UAVs. On one hand, the model owners aim to maximize their profits by selecting the optimal UAVs which can complete the stipulated task at the lowest cost, e.g., in terms of sensing, transmission, and

computation costs. On the other hand, the UAVs can take advantage of the information asymmetry and misreport their types so as to seek higher compensation. To that end, we leverage the self-revealing properties of contract theory [20] as an incentive mechanism design to reward the UAVs based on their actual types. In particular, given the complexity of the sensing and collaborative learning task, we devise a multi-dimensional contract to account for the multi-dimensional sources of heterogeneity in terms of UAV sensing, learning, and transmission capabilities. Chapter 4 is published under: Wei Yang Bryan Lim et al. "Towards Federated Learning in UAV-Enabled Internet of Vehicles: A Multi-Dimensional Contract-Matching Approach," *IEEE Transactions on Intelligent Transportation Systems* (2021).

In Chapter 5, a *hierarchical* FL (HFL) framework is proposed in which the workers (i.e. data owners) do not communicate directly with a central controller, i.e., the model owner (macro base stations). Instead, the local parameter values are first uploaded to edge servers, e.g., at micro base stations, for intermediate aggregation. Then, communication with the model owner is further established for global aggregation. Besides reducing the instances of global communications with the remote servers of the model owner, this relay approach reduces the dropout rate of devices. We consider a two-level resource allocation and incentive design problem. At the lower level, to encourage the participation of workers, the cluster heads offer reward pools to be shared among workers based on their data contribution in the cluster. We model the cluster selection decisions of the workers as an evolutionary game to capture the bounded rationality and worker dynamics of gradual strategy adjustment and adaptation. At the upper-level, there may be multiple model owners in the network that aim to train a model for their respective usage collaboratively with the participation of the workers and cluster heads. However, at any point in time, each worker and cluster head can only participate in the training process with a single model owner. To derive the allocation of cluster head to the model owner, as well as the optimal pricing of the services of the cluster head by the competitive model owners, we adopt a deep learning based auction mechanism that preserves the properties of the truthfulness of the bidders while simultaneously achieving revenue maximization for the cluster heads. Chapter 5 is published under: Wei Yang Bryan Lim et al. "Decentralized Edge Intelligence: A Dynamic Resource Allocation Framework for Hierarchical Federated Learning," *IEEE Transactions on Parallel and Distributed Systems* (2021).

In Chapter 6, we provide a summary of the thesis and present a discussion of the future research directions and open questions.

Chapter 2

Literature Review

2.1 Introduction

Currently, there are nearly 7 billion connected Internet of Things (IoT) devices and 3 billion smartphones around the world [12]. These devices are equipped with increasingly advanced sensors, computing, and communication capabilities. As such, they can potentially be deployed for various crowdsensing tasks, e.g., for medical purposes [21] and air quality monitoring [22]. Coupled with the rise of Deep Learning (DL) [23], the wealth of data collected by end devices opens up countless possibilities for meaningful research and applications.

In the traditional cloud-centric approach, data collected by mobile devices is uploaded and processed centrally in a cloud-based server or data center. In particular, data collected by IoT devices and smartphones such as measurements [24], photos [25], videos [26], and location information [27] are aggregated at the data center [28]. Thereafter, the data are used to provide insights or produce effective inference models. However, this approach is no longer sustainable for the following reasons. Firstly, data owners are increasingly privacy sensitive. Following privacy concerns among consumers due to high profile cases of data leaks and data misuse, policy makers have responded with the implementation of data privacy legislation such as the European Commission's General Data Protection Regulation (GDPR) [29] and Consumer Privacy Bill of Rights in the US [30]. In particular, the consent (GDPR Article 6) and data minimalization principle (GDPR Article 5) limits data collection and storage only to what is consumer-consented and absolutely necessary

for processing. Secondly, a cloud-centric approach involves long propagation delays and incurs unacceptable latency [31] for applications in which real-time decisions have to be made, e.g., in self-driving car systems [32]. Thirdly, the transfer of raw data to the cloud for processing burdens the backbone networks[1]. This is especially so in tasks involving unstructured data, e.g., in video analytics [33]. This is exacerbated by the fact that cloud-centric training is relatively reliant on wireless communications [34]. As a result, this can potentially impede the development of new technologies.

With data sources mainly located outside the cloud today, Mobile Edge Computing (MEC) has naturally been proposed as a solution. In MEC, the computing and storage capabilities [31] of end devices and edge servers are leveraged to bring model training closer to where data are produced [35]. As defined in [34], an end-edge-cloud computing network comprises: (a) end devices, (b) edge nodes, and (c) cloud server. For model training in conventional MEC approaches, a collaborative paradigm has been proposed in which training data are first sent to the edge servers for model training up to lower level DNN layers, before more computation intensive tasks are offloaded to the cloud [36], [37] (Fig. 2.1). However, this arrangement incurs significant communication costs and is unsuitable for applications that require persistent training [34]. In addition, computation offloading and data processing at edge servers still involve the transmission of potentially sensitive personal data. This can discourage privacy-sensitive consumers from taking part in model training or even violate increasingly stringent privacy laws [29]. Although various privacy preservation methods, e.g., Differential Privacy (DP) [38], have been proposed, a number of users are still not willing to expose their private data for fear that their data may be inspected by external servers. In the long run, this discourages the development of technologies as well as new applications.

To guarantee that training data remains on personal devices and to facilitate collaborative machine learning of complex models among distributed devices, a decentralized ML approach called Federated Learning (FL) is introduced in [3]. In FL, mobile devices[2] use their local data to cooperatively train an ML model required by an FL server. They then send the model updates, i.e., the model's weights, to

[1]Note that in some cases, the communication cost of FL is not insignificant due to complexity of large models, whereas the relatively fewer data samples a worker possesses is less costly to transmit. As such, we discuss communication cost reduction as well in this thesis.

[2]We use the term mobile devices and mobile edge in this Chapter given that many of the works reviewed have focused on how to implement FL on resource-constrained edge devices such as IoT.

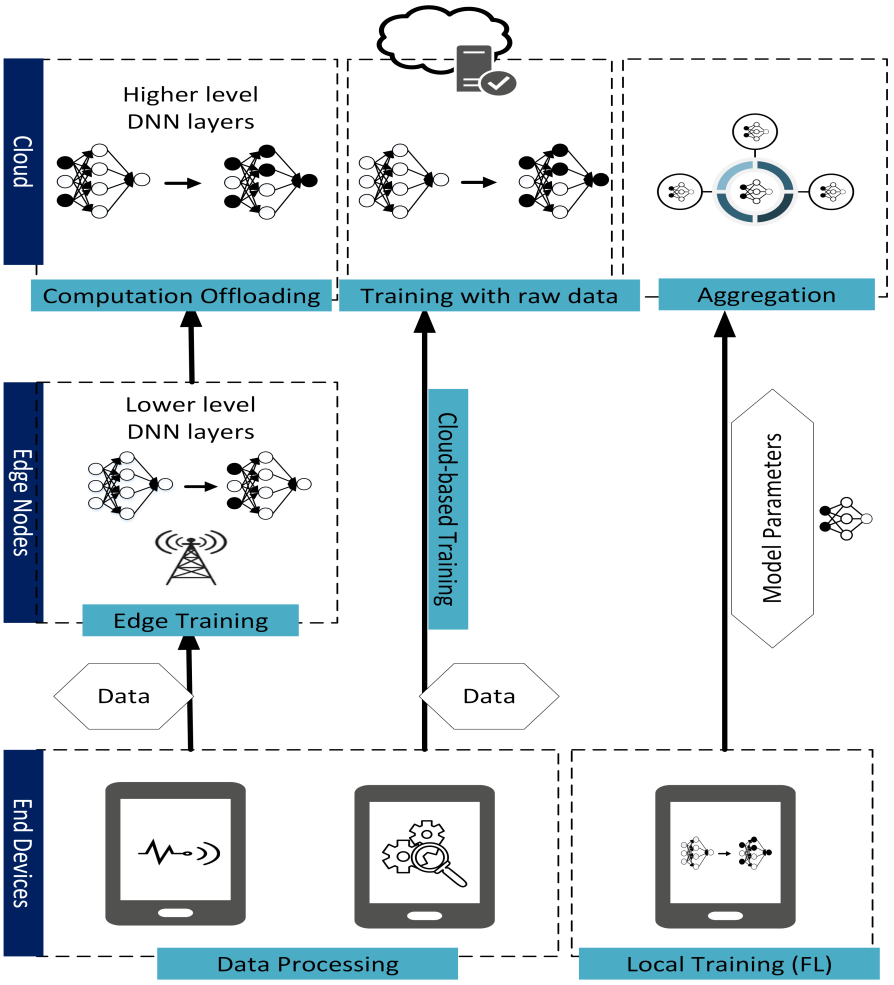

FIGURE 2.1: Edge AI approach brings AI processing closer to where data are produced. In particular, FL allows training on devices where the data are produced.

the FL server for aggregation. The steps are repeated in multiple rounds until a desirable accuracy is achieved. This implies that FL can be an enabling technology for ML model training at mobile edge networks. As compared to conventional cloud-centric ML model training approaches, the implementation of FL for model training at mobile edge networks features the following advantages.

i *Minimizing requirement of network bandwidth:* Less information is required to be transmitted to the cloud. For example, instead of sending the raw data over for processing, participating devices only send the updated model parameters for aggregation. As a result, this significantly reduces costs of data communication and relieves the burden on backbone networks.

However, note that the insights from this chapter can be similarly applied on edge networks in general.

ii *Privacy:* Following the above point, the raw data of users need not be sent to the cloud. Under the assumption that FL workers and servers are non-malicious, this enhances user privacy and reduces the probability of eavesdropping to a certain extent. In fact, with enhanced privacy, more users will be willing to take part in collaborative model training and so, better inference models can be built.

iii *Low latency:* With FL, ML models can be consistently trained and updated. Meanwhile, in the MEC paradigm, real-time decisions, e.g., event detection[39], can be made locally at the edge nodes or end devices. Therefore, the latency is much lower than when decisions are made in the cloud before transmitting them to the end devices. This is vital for time critical applications such as self-driving car systems in which the slightest delays can potentially be life threatening [32].

Given the aforementioned advantages, FL has seen recent successes in several applications. For example, the Federated Averaging algorithm (*FedAvg*) proposed in [3] has been applied to Google's Gboard [40] to improve next-word prediction models. In addition, several studies have also explored the use of FL in a number of scenarios in which data are sensitive in nature, e.g., to develop predictive models for diagnosis in health AI [41] and to foster collaboration across multiple hospitals [42] and Government agencies [43].

Besides being an enabling technology for ML model training *at* mobile edge networks, FL has also been increasingly applied as an enabling technology *for* mobile edge network optimization. Given the computation and storage constraints of increasingly complex mobile edge networks, conventional network optimization approaches that are built on static models fare relatively poorly in modelling dynamic networks [34]. As such, a data-driven Deep Learning (DL) based approach [5] for optimizing resource allocation is increasingly popular. For example, DL can be used for representation learning of network conditions [44] whereas Deep Reinforcement Learning (DRL) can optimize decision making through interactions with the dynamic environment [45]. However, the aforementioned approaches require user data as an input and these data may be sensitive or inaccessible in nature due to regulatory constraints. As such, in this chapter, we also briefly discuss FL's potential to function as an enabling technology for optimizing wireless edge

networks, e.g., in cell association [17], computation offloading [12], and vehicular networks [18].

However, there are several challenges to be solved before FL can be implemented at scale. Firstly, even though raw data no longer need to be sent to the cloud servers, communication costs remain an issue due to the high dimensionality of model updates and limited communication bandwidth of participating mobile devices. In particular, state-of-the-art DNN model training can involve the communication of millions of parameters for aggregation. Secondly, in a large and complex mobile edge network, the heterogeneity of participating devices in terms of data quality, computation power, and willingness to participate have to be well managed from the resource allocation perspective. Thirdly, FL does not guarantee privacy in the presence of malicious workers or aggregating servers. In particular, recent research works have clearly shown that a malicious worker may exist in FL and can infer the information of other workers just from the shared parameters alone. As such, privacy and security issues in FL still need to be considered.

In this chapter, we focus on the literature review and address the following topics:

i We motivate the importance of FL as an important paradigm shift towards enabling collaborative ML model training. Then, we provide a concise tutorial on FL implementation and present to the reader a list of useful open-source frameworks that pave the way for future research on FL and the the related applications.

ii We discuss the unique features of FL relative to a centralized ML approach and the resulting implementation challenges. For each of these challenges, we present to the reader a comprehensive discussion of existing solutions and approaches explored in the FL literature.

iii We discuss FL as an enabling technology for mobile edge network optimization. In particular, we discuss the current and potential applications of FL as a privacy-preserving approach for applications in edge computing.

Besides, we also consolidate the existing surveys on FL and mobile edge computing in Table 2.1.

TABLE 2.1: An overview of selected surveys in FL and MEC.

Ref.	Subject	Contribution
[46]	FL	Introductory tutorial on categorization of FL architectures, e.g., vertical FL
[4]		FL in optimizing resource allocation for wireless networks while preserving data privacy
[47]		Tutorial on FL and discussions of implementation challenges in FL
[33]	MEC	Computation offloading strategy to optimize DL performance in edge computing
[48]		Survey on architectures and frameworks for edge intelligence
[49]		ML for IoT management, e.g., network management and security
[37]		Survey on computation offloading in MEC
[45]		Survey on DRL approaches to address issues in communications and networking
[50]		Survey on techniques for computation offloading
[51]		Survey on architectures and applications of MEC
[52]		Survey on computing, caching, and communication issues at mobile edge networks
[53]		Survey on the phases of caching and comparison among the different caching schemes
[31]		Survey on joint mobile computing and wireless communication resource management in MEC

The rest of this chapter is organized as follows. Section 2.2 introduces the background and fundamentals of FL. Section 2.3 reviews solutions provided to reduce communication costs. Section 2.4 discusses resource allocation approaches in FL. Section 2.5 concludes the chapter.

2.2 Background and Fundamentals of Federated Learning

2.2.1 Federated Learning

Motivated by privacy concerns among data owners, the concept of FL is introduced in [3]. FL allows users to collaboratively train a shared model while keeping personal data on their devices, thus alleviating their privacy concerns. As such, FL can serve as an enabling technology for ML model training at mobile edge networks. For an introduction to the categorizations of different FL settings, e.g., vertical and horizontal FL, we refer the interested readers to [46].

In general, there are two main entities in the FL system: the data owners (i.e. *workers*) and the model owner (i.e. *FL server*). Let $\mathcal{N} = \{1, \ldots, N\}$ denote the set of N data owners, each of which has a private dataset $D_{i \in \mathcal{N}}$. Each data owner i uses its dataset D_i to train a *local model* $\mathbf{w}_i$ and send only the local model parameters to the FL server. Then, all collected local models are aggregated $\mathbf{w} = \cup_{i \in \mathcal{N}} \mathbf{w}_i$ to generate a *global model* $\mathbf{w}_G$. This is different from the traditional centralized training which uses $\mathbf{D} = \cup_{i \in \mathcal{N}} D_i$ to train a model $\mathbf{w}_T$, i.e., data from each individual source is aggregated first before model training takes place centrally.

A typical architecture and training process of an FL system is shown in Fig. 2.2. In this system, the data owners serve as the FL workers which collaboratively train an ML model required by an aggregate server. An underlying assumption is that the data owners are honest, which means they use their real private data to do the training and submit the true local models to the FL server. Of course, this assumption may not always be realistic [54] and we discuss the proposed solutions subsequently in Section 2.4.

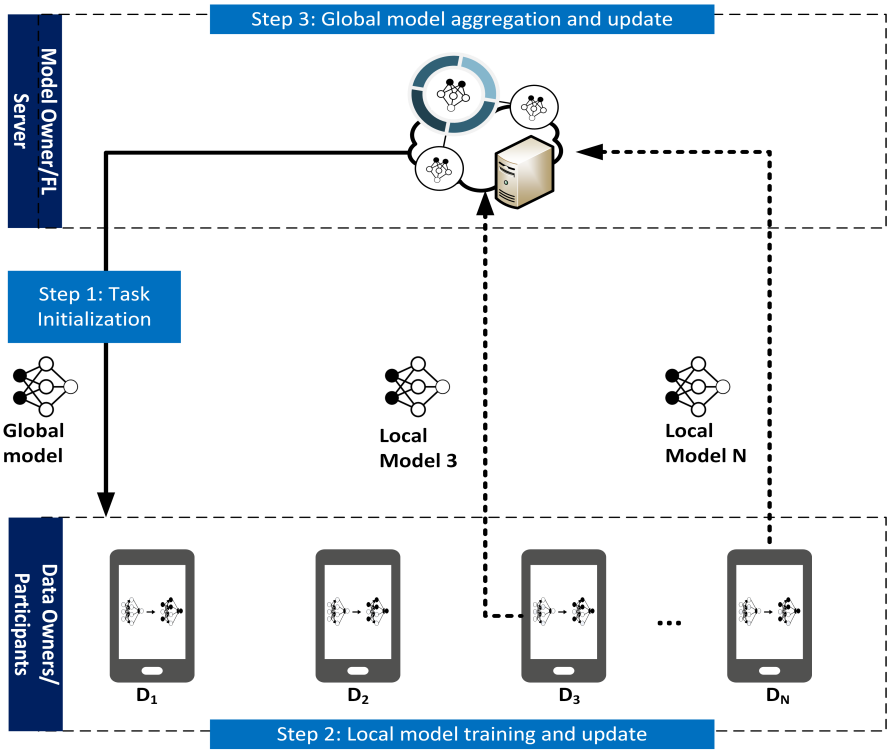

FIGURE 2.2: General FL training process involving N workers.

In general, the FL training process includes the following three steps. Note: the *local* model refers to the model trained at each participating device, whereas the *global* model refers to the model aggregated by the FL server.

i *Step 1 (Task initialization)*: The server decides the training task, i.e., the target application, and the corresponding data requirements. The server also specifies the hyperparameters of the global model and the training process, e.g., learning rate. Then, the server broadcasts the initialized global model $\mathbf{w}_G^0$ and task to selected workers.

ii *Step 2 (Local model training and update)*: Based on the global model $\mathbf{w}_G^t$, where t denotes the current iteration index, each worker respectively uses its local data and device to update the local model parameters $\mathbf{w}_i^t$. The goal of worker i in iteration t is to find optimal parameters $\mathbf{w}_i^t$ that minimize the loss function $L(\mathbf{w}_i^t)$, i.e.,

$$\mathbf{w}_i^{t^*} = \arg\min_{\mathbf{w}_i^t} L(\mathbf{w}_i^t). \tag{2.1}$$

The updated local model parameters are subsequently sent to the server.

iii *Step 3 (Global model aggregation and update)*: The server aggregates the local models from workers and then sends the updated global model parameters $\mathbf{w}_G^{t+1}$ back to the data owners.

The server wants to minimize the global loss function $L(\mathbf{w}_G^t)$, i.e.,

$$L(\mathbf{w}_G^t) = \frac{1}{N}\sum_{i=1}^{N} L(\mathbf{w}_i^t). \tag{2.2}$$

Steps 2-3 are repeated until the global loss function converges or a desirable training accuracy is achieved.

Note that the FL training process can be used for different ML models that essentially use the SGD method such as Support Vector Machines (SVMs) [55], neural networks, and linear regression [56]. A training dataset usually contains a set of n data feature vectors $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and a set of corresponding data labels[3] $\mathbf{y} = \{y_1, \ldots, y_n\}$. In addition, let $\hat{y_j} = f(\mathbf{x}_j; \mathbf{w})$ denote the predicted result from the model $\mathbf{w}$ updated/trained by data vector x_j. Table 2.2 summarizes several loss functions of common ML models [57].

TABLE 2.2: Loss functions of common ML models.

Model	Loss function $L(\mathbf{w}_i^t)$
Neural network	$\frac{1}{n}\sum_{j=1}^{n}(y_i - f(\mathbf{x}_j; \mathbf{w}))^2$ (Mean Squared Error)
Linear regression	$\frac{1}{2}\left\|y_j - \mathbf{w}^T\mathbf{x}_j\right\|^2$
K-means	$\sum_j \left\|\mathbf{x}_j - f(\mathbf{x}_j; \mathbf{w})\right\|$ ($f(\mathbf{x}_j; \mathbf{w})$ is the centroid of all objects assigned to x_j's class)
squared-SVM	$[\frac{1}{n}\sum_{j=1}^{n}\max(0, 1 - y_j(\mathbf{w}^T\mathbf{x}_j - bias))]$ $+\lambda\left\|\mathbf{w}^T\right\|^2$($bias$ is the bias parameter and λ is const.)

Global model aggregation is an integral part of FL. A straightforward and classical algorithm for aggregating the local models is the *FedAvg* algorithm proposed in [3], which is similar to that of local SGD [58]. The pseudocode for *FedAvg* is given in Algorithm 2.1. As described in Step 1 above, the server first initializes the task

[3]In the case of unsupervised learning, there is no data label.

Algorithm 2.1 Federated averaging algorithm [3]

Require: Local minibatch size B, number of workers m per iteration, number of local epochs E, and learning rate η.
Ensure: Global model $\mathbf{w}_G$.
1: [Worker i]
2: **LocalTraining**(i, $\mathbf{w}$):
3: Split local dataset D_i to minibatches of size B which are included into the set $\mathcal{B}_i$.
4: **for** each local epoch j from 1 to E **do**
5: **for** each $b \in \mathcal{B}_i$ **do**
6: $\mathbf{w} \leftarrow \mathbf{w} - \eta \Delta L(\mathbf{w}; b)$ (η is the learning rate and ΔL is the gradient of L on b.)
7: **end for**
8: **end for**
9:
10: [Server]
11: Initialize $\mathbf{w}_G^0$
12: **for** each iteration t from 1 to T **do**
13: Randomly choose a subset $\mathcal{S}_t$ of m workers from $\mathcal{N}$
14: **for** each partipant $i \in \mathcal{S}_t$ **parallely do**
15: $\mathbf{w}_i^{t+1} \leftarrow$ **LocalTraining**(i, $\mathbf{w}_G^t$)
16: **end for**
17: $\mathbf{w}_G^t = \frac{1}{\sum_{i \in \mathcal{N}} D_i} \sum_{i=1}^{N} D_i \mathbf{w}_i^t$ (Averaging aggregation)
18: **end for**

(lines 11-16). Thereafter, in Step 2, the worker i implements the local training and optimizes the target in (2.1) on minibatches from the original local dataset (lines 2-8). Note that a minibatch refers to a randomized subset of each worker's dataset. At the t^{th} iteration (line 17), the server minimizes the global loss in (2.2) by the averaging aggregation which is formally defined as

$$\mathbf{w}_G^t = \frac{D_i}{\sum_{i \in \mathcal{N}} D_i} \sum_{i=1}^{N} \mathbf{w}_i^t. \tag{2.3}$$

The FL training process is iterated till the global loss function converges or a desirable accuracy is achieved.

2.2.2 Statistical Challenges of FL

In traditional distributed ML, the central server has access to the whole training dataset. As such, the server can split the dataset into subsets that follow similar distributions. The subsets are subsequently sent to participating nodes for distributed training. However, this approach is impractical for FL since the local dataset is only accessible by the worker.

In the FL setting, the workers may have local datasets that follow different distributions, i.e., the datasets of workers are non-IID. While the authors in [3] show that the aforementioned *FedAvg* algorithm is able to achieve desired accuracy even

when data are non-IID across workers, the authors in [59] found otherwise. For example, the accuracy of a *FedAvg*-trained CNN model has 51% lower accuracy than the centrally-trained CNN model for CIFAR-10 [60]. This deterioration in accuracy is further shown to be quantified by the earth mover's distance (EMD) [61], i.e., the difference in FL worker's data distribution as compared to the population distribution. As such, when data are non-IID and highly skewed, data-sharing is proposed in which a shared dataset with uniform distribution across all classes is sent by the FL server to each FL worker. Then, the worker trains its local model on its private data together with the received data. The simulation result shows that accuracy can be increased by 30% with 5% shared data due to reduced EMD. However, a common dataset may not always be available for sharing by the FL server. An alternative solution to gather contributions towards building the common dataset is subsequently discussed in section 2.4.

The authors in [62] also find that global imbalance, i.e., the situation in which the collection of data held across all FL workers is class imbalanced, leads to a deterioration in model accuracy. As such, the Astraea framework is proposed. On initialization, the FL workers first send their data distribution to the FL server. A rebalancing step is introduced before training begins in which each worker performs data augmentation [63] on the minority classes, e.g., through random rotations and shifts. After training on the augmented data, a mediator is created to coordinate intermediate aggregation, i.e., before sending the updated parameters to the FL server for global aggregation. The mediator selects workers with data distributions that best contribute to a uniform distribution when aggregated. This is done through a greedy algorithm approach to minimize the Kullback-Leibler Divergence [64] between local data and uniform distribution. The simulation results show accuracy improvement when tested on imbalanced datasets.

Given the heterogeneity of data distribution across devices, there has been an increasing number of studies that borrow concepts from multi-task learning [65] to learn separate, but structurally related models for each worker. Instead of minimizing the conventional loss function presented previously in Table 2.2 of Section 2.2.1 (Page 15), the loss function is modified to also model the relationship amongst tasks [66]. Then, the MOCHA algorithm is proposed in which an alternating optimization approach [67] is used to approximately solve the minimization problem. Interestingly, MOCHA can also be calibrated based on the resource constraints of a

participating device. For example, the quality of approximation can be adaptively adjusted based on network conditions and CPU states of the participating devices. However, MOCHA cannot be applied to non-convex DL models.

Similarly, the authors in [68] also borrow concepts from multi-task learning to deal with the statistical heterogeneity in FL. The FEDPER approach is proposed in which all FL workers share a set of base layers trained using the *FedAvg* algorithm. Then, each worker separately trains another set of personalization layers using its own local data. In particular, this approach is suitable for building recommender's systems given the diverse preferences of workers. The authors show empirically using the Flickr-AES dataset [69] that the FEDPER approach outperforms a pure *FedAvg* approach since the personalization layer is able to represent the personal preference of an FL worker. However, it is worth noting that the collaborative training of the base layers is still important to achieve a high test accuracy, since each worker has insufficient local data samples for personalized model training.

Apart from data heterogeneity, the convergence of a distributed learning algorithm is always a concern. A higher convergence rate helps to save a large amount of time and resources for the FL workers, and also significantly increases the success rate of the federated training since fewer communication rounds imply reduced worker dropouts. To ensure convergence, the study in [70] proposes *FedProx*, which modifies the loss function to also include a tunable parameter that restricts how much local updates can affect the prevailing model parameters. The *FedProx* algorithm can be adaptively tuned, e.g., when training loss is increasing, model updates can be tuned to affect the current parameters less. Similarly, the authors in [71] also propose the *LoAdaBoost FedAvg* algorithm to complement the aforementioned data-sharing approach [59] in ML on medical data. In *LoAdaBoost FedAvg*, workers train the model on their local data and compare the cross-entropy loss with the median loss from the *previous* training round. If the current cross-entropy loss is higher, the model is retrained before global aggregation so as to increase learning efficiency. The simulation results show that faster convergence is achieved as a result.

In fact, the statistical challenges of FL coexist with other issues that we explore in subsequent sections. For example, the communication costs incurred in FL can be reduced by faster convergence. Similarly, resource allocation policies can also

be designed to solve statistical heterogeneity. As such, we revisit these concepts in greater detail subsequently.

2.2.3 FL protocols and frameworks

The main open-source frameworks for FL that have been developed are as follows:

i *TensorFlow Federated (TFF)*: TFF [72] is a framework based on Tensorflow developed by Google for decentralized ML and other distributed computations. TFF consists of two layers (a) FL and (b) Federated Core (FC). The FL layer is a high-level interface that allows the implementation of FL to existing TF models without the user having to apply the FL algorithms personally. The FC layer combines TF with communication operators to allow users to experiment with customized and newly designed FL algorithms.

ii *PySyft*: PySyft [73] is a framework based on PyTorch for performing encrypted, privacy-preserving DL and implementations of related techniques, such as Secure Multiparty Computation (SMPC) and DP, in untrusted environments while protecting data. Pysyft is developed such that it retains the native Torch interface, i.e., the ways to execute all tensor operations remain unchanged from that of Pytorch. When a SyftTensor is created, a LocalTensor is automatically created to also apply the input command to the native Pytorch tensor. To simulate FL, workers are created as *Virtual Workers*. Data, i.e., in the structure of tensors, can be split and distributed to the Virtual Workers as a simulation of a practical FL setting. Then, a PointerTensor is created to specify the data owner and storage location. In addition, model updates can be fetched from the Virtual Workers for global aggregation.

iii *LEAF*: An open source framework [74] of datasets that can be used as benchmarks in FL, e.g., Federated Extended MNIST (FEMNIST), and Sentiment140 [75]. In these datasets, the writer or user is assumed to be a worker in FL, and their corresponding data are taken to be the local data held in their personal devices. The implementation of newly designed algorithms on these benchmark datasets allows reliable comparison across studies.

iv *FATE*: Federated AI Technology Enabler (FATE) is an open-source framework by WeBank [76] that supports the federated and secure implementation of several ML models.

2.2.4 Unique characteristics and issues of FL

Besides the statistical challenges we have presented in Section 2.2.2, FL has some unique characteristics and features [77] as compared to other distributed ML approaches:

i *Slow and unstable communication*: In the traditional distributed training in a data center, the communication environment can be assumed to be perfect where the information transmission rate is very high and there is no packet loss. However, these assumptions are not applicable to the FL environment where heterogeneous devices are involved in training. For example, the Internet upload speed is typically much slower than download speed [78]. Also, some workers with unstable wireless communication channels may consequently drop out due to disconnection from the Internet.

ii *Heterogeneous devices*: Apart from bandwidth constraints, FL involves heterogeneous devices with varying resource constraints. For example, the devices can have different computing capabilities, i.e., CPU states and battery levels. The devices can also have different levels of *willingness* to participate, i.e., FL training is resource consuming and given the distributed nature of training across numerous devices, there is a possibility of free ridership.

iii *Privacy and security concerns*: As we have previously discussed, data owners are increasingly privacy sensitive. However, malicious workers are able to infer sensitive information from shared parameters, which potentially negates privacy preservation. In addition, we have previously assumed that all workers and FL servers are trustful.

These unique characteristics of FL lead to several practical issues in FL implementation mainly in three aspects that we now proceed to discuss, i.e., (a) statistical challenges (b) communication costs and (c) resource allocation. In the following sections, we review related work that addresses each of these issues.

2.3 Communication Cost

In FL, a number of rounds of communications between the workers and the FL server may be required to achieve a target accuracy. For complex DL model training involving, e.g. CNN, each update may comprise millions of parameters [79]. The high dimensionality of the updates can result in the incurrence of high communication costs and can lead to a training bottleneck. In addition, the bottleneck can be worsened due to (a) unreliable network conditions of participating devices [80] and (b) the asymmetry in Internet connection speeds in which upload speed is faster than download speed, resulting in delays in model uploads by workers [78]. As such, there is a need to improve the communication efficiency of FL. The following approaches to reduce communication costs are considered:

i *Edge and End Computation*: In the FL setup, the communication cost often dominates computation cost [3]. The reason is that on-device dataset is relatively small whereas mobile devices of workers have increasingly fast processors. On the other hand, workers may be willing to participate in the model training only if they are connected to Wi-Fi [78]. As such, more computation can be performed on edge nodes or on end devices before each global aggregation so as to reduce the number of communication rounds needed for the model training. In addition, algorithms to ensure faster convergence can also reduce number of communication rounds involved, at the expense of more computation on edge servers and end devices.

ii *Model Compression*: This is a technique commonly used in distributed learning [81]. Model or gradient compression involves the communication of an update that is transformed to be more compact, e.g., through sparsification, quantization or subsampling [82], rather than the communication of full update. However, since the compression may introduce noise, the objective is to reduce the size of update transferred during each round of communication while maintaining the quality of trained models [83].

iii *Importance-based Updating*: This strategy involves selective communication such that only the important or relevant updates [84] are transmitted in each communication round. In fact, besides saving on communication costs, omitting some updates from workers can even improve the global model performance.

2.3.1 Edge and End Computation

To decrease the number of communication rounds, the additional computation can be performed on participating end devices before each iteration of communication for global aggregation (Fig. 2.3(a)). The authors in [3] consider two ways to increase computation on participating devices: (a) increasing parallelism in which more workers are selected to participate in each round of training and (b) increasing computation per worker whereby each worker performs more local updates before communication for global aggregation. A comparison is conducted for the FederatedSGD (*FedSGD*) algorithm and the proposed *FedAvg* algorithm. For the *FedSGD* algorithm, all workers are involved and only one pass is made per training round in which the minibatch size comprises the entirety of the worker's dataset. This is similar to the full-batch training in centralized DL frameworks. For the proposed *FedAvg* algorithm, the hyperparameters are tuned such that more local computations are performed by the workers. For example, the worker can make more passes over its dataset or use a smaller local minibatch size to increase computation before each communication round. The simulation results show that increased parallelism does not lead to significant improvements in reducing communication cost, once a certain threshold is reached. As such, more emphasis is placed on increasing computation per worker while keeping the fraction of selected workers constant. For MNIST CNN simulations, increased computation using the proposed *FedAvg* algorithm can reduce communication rounds when the dataset is IID. For non-IID dataset, the improvement is less significant using the same hyperparameters. For Long Short Term Memory (LSTM) simulations [85], improvements are more significant even for non-IID data. However, as we will discuss later, the non-IID data distribution is a key concern in FL. The empirical observation of improvements found in [3] is not consistent when explored in other studies. Instead, more precise methods are proposed to deal with non-IID distributions.

As an extension, the authors in [86] also validate that a similar concept as that of [3] works for vertical FL. In vertical FL, collaborative model training is conducted across the same set of workers with different data features. The Federated Stochastic Block Coordinate Descent (FedBCD) algorithm is proposed in which each participating device performs multiple local updates first before communication for global aggregation. In addition, convergence guarantee is also provided

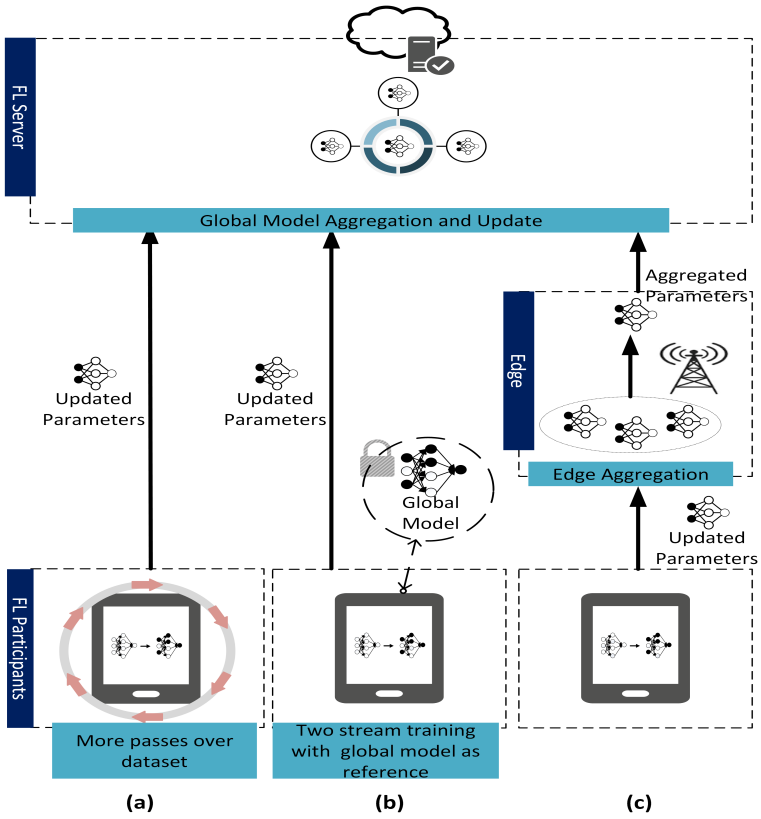

FIGURE 2.3: Approaches to increase computation at edge and end devices include (a) Increased computation at end devices, e.g., more passes over dataset before communication (b) Two-stream training with global model as a reference and (c) Intermediate edge server aggregation.

with an approximate calibration of the number of local updates per interval of communication.

Another way to decrease communication cost can also be through modifying the training algorithm to increase convergence speed, e.g., through the aforementioned *LoAdaBoost FedAvg* in [71]. Similarly, the authors in [87] also propose increased computation on each participating device by adopting a two-stream model (Fig. 2.3(b)) commonly used in transfer learning and domain adaptation [88]. During each training round, the global model is received by the worker and fixed as a reference in the training process. During training, the worker learns not just from local data, but also from other workers with reference to the fixed global model. This is done through the incorporation of Maximum Mean Discrepancy (MMD) into the loss function. MMD measures the distance between the means of two data distributions [88], [89]. By minimizing MMD loss between the local and global models, the worker can extract more generalized features from the global model,

thus accelerating the convergence of the training process and reducing communication rounds. The simulation results on the CIFAR-10 and MNIST dataset using DL models such as AlexNet [90] and 2-CNN respectively show that the proposed two-stream FL can reach the desirable test accuracy in 20% fewer communication rounds even when data are non-IID. However, while convergence speed is increased, more computation resources have to be consumed by end devices for the aforementioned approaches. Given the energy constraints of participating mobile devices in particular, this necessitates resource allocation optimization that we subsequently discuss in Section 2.4.

While the aforementioned studies consider increasing computation on participating *devices*, the authors in [91] propose an edge computing inspired paradigm in which proximate edge *servers* can serve as intermediary parameter aggregators given that the propagation latency from worker to the edge server is smaller than that of the worker-cloud communication (Fig. 2.3(c)). A hierarchical FL (*HierFAVG*) algorithm is proposed whereby for every few local worker updates, the edge server aggregates the collected local models. After a predefined number of edge server aggregations, the edge server communicates with the cloud for global model aggregation. As such, the communication between the workers and the cloud occurs only once after an interval of multiple local updates. Comparatively, for the *FedAvg* algorithm proposed in [3], the global aggregation occurs more frequently since no intermediate edge server aggregation is involved. The authors further prove the convergence of *HierFAVG* for both convex and non-convex objective functions given non-IID user data. The simulation results show that for the same number of local updates between two global aggregations, more intermediate edge aggregations before each global aggregation can lead to reduced communication overhead as compared to the *FedAvg* algorithm. This result holds for both IID and non-IID data, implying that intermediate aggregation on edge servers may be implemented on top of *FedAvg* to reduce communication costs. However, when applied to non-IID data, the simulation results show that *HierFAVG* fails to converge to the desired accuracy level (90%) in some instances, e.g., when edge-cloud divergence is large or when there are many edge servers involved. As such, further study is required to better understand the tradeoffs between adjusting local and edge aggregation intervals, so as to ensure that the parameters of the *HierFAVG* algorithm can be optimally calibrated to suit other settings. Nevertheless, *HierFAVG* is a promising approach for the implementation of FL at mobile edge

networks, since it leverages the proximity of intermediate edge servers to reduce communication costs, and potentially relieve the burden on the remote cloud.

2.3.2 Model Compression

To reduce communication costs, the authors in [78] propose structured and sketched updates to reduce the size of model updates sent from workers to the FL server during each communication round.

Structured updates restrict worker updates to have a pre-specified structure, i.e., low rank and random mask. For the low rank structure, each update is enforced to be a low rank matrix expressed as a product of two matrices. Here, one matrix is generated randomly and held constant during each communication round whereas the other is optimized. As such, only the optimized matrix needs to be sent to the server. For the random mask structure, each worker update is restricted to be a sparse matrix following a predefined random sparsity pattern generated independently during each round. As such, only the non-zero entries are required to be sent to the server.

On the other hand, *sketched updates* refer to the approach of encoding the update in a compressed form before communication with the server, which subsequently decodes the updates before aggregation. One example of the sketched update is the subsampling approach, in which each worker communicates only a random subset of the update matrix. The server then averages the subsampled updates to derive an unbiased estimate of the true average.

The simulation results on the CIFAR-10 image classification task show that for structured updates, the random mask performs better than that of the low rank approach. The random mask approach also achieves higher accuracy than sketching approaches since the latter involves the removal of some information obtained during training. However, the combination of all three sketching tools, i.e., subsampling, quantization, and rotation, can achieve a higher compression rate and faster convergence, albeit with some sacrifices in accuracy. This suggests that for practical implementation of FL where there are many workers available, more workers can be selected for training per round so that subsampling can be more aggressive to reduce communication costs.

In addition to the subsampling and quantization approaches, the federated dropout approach is also considered in which a fixed number of activation functions at each fully-connected layer is removed to derive a smaller sub-model. The sub-model is then sent to the workers for training. The updated sub-model can then be mapped back to the global model to derive a complete DNN model with all weights updated during subsequent aggregation. This approach reduces the server-to-worker communication cost, and also the size of worker-to-server updates. In addition, local computation is reduced since fewer parameters have to be updated. The simulations are performed on MNIST, CIFAR-10, and EMNIST [92] datasets. For the lossy compression, it is shown that the subsampling approach taken by [78] does not reach an acceptable level of performance. The reason is that the update errors can be averaged out for worker-to-server uploads but not for server-to-worker downloads. On the other hand, quantization with Kashin's representation can achieve the same performance as the baseline without compression while having communication cost reduced by nearly 8 times when the model is quantized to 4 bits. For federated dropout approaches, the results show that a dropout rate of 25% of weight matrices of fully-connected layers (or filters in the case of CNN) can achieve acceptable accuracy in most cases while ensuring around 43% reduction in the size of models communicated. However, if dropout rates are more aggressive, the convergence of the model can be slower.

The aforementioned two studies suggest useful model compression approaches in reducing communication costs for both server-to-worker and worker-to-server communications. As one may expect, the reduction in communication costs comes with sacrifices in model accuracy. It will thus be useful to formalize the compression-accuracy tradeoffs since this varies for different tasks.

2.3.3 Importance-based Updating

Based on the observation that most parameter values of a DNN model are sparsely distributed and close to zero [93], the authors in [84] propose the edge Stochastic Gradient Descent (eSGD) algorithm that selects only a small fraction of important gradients to be communicated to the FL server for parameter update during each communication round. The eSGD algorithm keeps track of loss values at two consecutive training iterations. If the loss value of the current iteration is smaller

than the preceding iteration, this implies that current training gradients and model parameters are important for training loss minimalization. As a result, their respective hidden weights are assigned a positive value. In addition, the gradient is also communicated to the server for parameter updates. Once this does not hold, i.e., the loss increases as compared to the previous iteration, other parameters are selected to be updated based on their hidden weight values. A parameter with a larger hidden weight value is more likely to be selected since it has been labeled as important many times during training. To account for small gradient values that can delay convergence if they are ignored and not updated completely [94], these gradient values are accumulated as residual values. Since the residuals may arise from different training iterations, each update to the residual is weighted with a discount factor using the momentum correction technique [95]. Once the accumulated residual gradient reaches a threshold, they are chosen to replace the least important gradient coordinates according to the hidden weight values. The simulation results show that eSGD with a 50% drop ratio can achieve higher accuracy than that of the thresholdSGD algorithm proposed in [93], which uses a fixed threshold value to determine which gradient coordinates to drop. eSGD can also save a large proportion of gradient size communicated. However, eSGD still suffers from accuracy loss as compared to standard SGD approaches.

While [84] studies the selective communication of gradients, the authors in [80] propose the Communication-Mitigated Federated Learning (CMFL) algorithm that uploads only relevant local model updates to reduce communication costs while guaranteeing global convergence. In each iteration, a worker's local update is first compared with the global update to identify if the update is relevant. A relevance score is computed where the score equates to the percentage of same-sign parameters in the local and global update. In fact, the global update is not known in advance before aggregation. As such, the global update made in the previous iteration is used as an estimate for comparison since it was found empirically that more than 99% of the normalized difference between two sequential global updates are smaller than 0.05 in both MNIST CNN and Next-Word-Prediction LSTM. An update is considered to be irrelevant if its relevance score is smaller than a predefined threshold. The simulation results show that CMFL requires 3.47 times and 13.97 times fewer communication rounds to reach 80% accuracy for MNIST CNN and Next-Word-Prediction LSTM, respectively, as compared to the benchmark *FedAvg* algorithm. In addition, CMFL can save significantly more communication rounds

as compared to Gaia. Note that Gaia is a geo-distributed ML approach suggested in [96] which measures relevance based on *magnitude* of updates rather than sign of parameters. When applied with the aforementioned MOCHA algorithm 2.2.2 [66], CMFL can reduce communication rounds by 5.7 times for the Human Activity Recognition dataset [97] and 3.3 times for the Semeion Handwritten Digit dataset [98]. In addition, CMFL can achieve slightly higher accuracy since it involves the elimination of irrelevant updates that are outliers that harm training.

As a summary, we present the discussed works in this subsection in Table 2.3.

2.4 Resource Allocation

FL involves the participation of heterogeneous workers that have different dataset qualities, computation capabilities, energy states, and willingness to participate. Given the worker heterogeneity and resource constraints, i.e., in device energy states and communication bandwidth, resource allocation has to be optimized to maximize the efficiency of the training process. In particular, the following resource allocation issues need to be considered:

i *Worker Selection:* As part of the FL protocol presented in Section 2.2.3, worker selection refers to the selection of devices to participate in each training round. Typically, a set of workers is randomly selected by the server to participate. Then, the server has to aggregate parameter updates from all participating devices in the round before taking a weighted average of the models [3]. As such, the training progress of FL is limited by the training time of the slowest participating devices, i.e., stragglers [99]. New worker selection protocols are thus investigated in order to address the training bottleneck in FL.

ii *Joint Radio and Computation Resource Management:* Even though computation capabilities of mobile devices have grown rapidly, many devices still face a scarcity of radio resources [100]. Given that local model transmission is an integral part of FL, there has been a growing number of studies that focus on developing novel wireless communication techniques for efficient FL.

TABLE 2.3: Approaches to communication cost reduction in FL.

Approaches	Ref.	Key Ideas	Tradeoffs and Shortcomings
Edge and End Computation	[3]	More local updates in between communication for global aggregation, to reduce instances of communication	Increased computation cost and poor performance in non-IID setting
	[86]	Similar to the ideas of [3], but with convergence guarantees for vertical FL	Increased computation cost and delayed convergence if global aggregation is too infrequent
	[87]	Transfer learning-inspired two-stream model for FL workers to learn from the fixed global model so as to accelerate training convergence	Increased computation cost and delayed convergence
	[91]	MEC-inspired edge server assisted FL that aids in intermediate parameter aggregation to reduce instances of communication	System model is not scalable when there are more edge servers
Model Compression	[78]	Structured and sketched updates to compress local models communicated from worker to FL server	Model accuracy and convergence issues
	[83]	Similar to the ideas of [78], but for communication from FL server to workers	Model accuracy and convergence issues
Importance-based Updating	[84]	Selective communication of gradients that are assigned importance scores, i.e., to reduce training loss	Only empirically tested on simple datasets and tasks, with fluctuating results
	[80]	Selective communication of local model updates that have higher relevance scores when compared to previous global model	Difficult to implement when global aggregations are less frequent

iii *Adaptive Aggregation:* As discussed in Section 2.2.1, FL involves global aggregation in which model parameters are communicated to the FL server for aggregation. The conventional approach to global aggregation is a synchronous one, i.e., global aggregations occur in fixed intervals after all workers complete a certain number of rounds of local computation. However, adaptive calibrations of global aggregation frequency can be investigated to increase training efficiency subject to resource constraints [99].

iv *Incentive Mechanism:* In the practical implementation of FL, workers may be reluctant to participate in a federation without receiving compensation since training models are resource-consuming. In addition, there exists information asymmetry between the FL server and workers since workers have greater knowledge of their available computation resources and data quality. Therefore, incentive mechanisms have to be carefully designed to both incentivize participation and reduce the potential adverse impacts of information asymmetry.

2.4.1 Worker Selection

To mitigate the training bottleneck, the authors in [101] propose a new FL protocol called *FedCS*. This protocol is illustrated in Fig. 2.4. The system model is a MEC framework in which the operator of the MEC is the FL server that coordinates training in a cellular network that comprises participating mobile devices that have heterogeneous resources. Accordingly, the FL server first conducts a *Resource Request* step to gather information such as wireless channel states and computing capabilities from a subset of randomly selected workers. Based on this information, the MEC operator selects the maximum possible number of workers that can complete the training within a prespecified deadline for the subsequent global aggregation phase. By selecting the maximum possible number of workers in each round, accuracy and efficiency of training are preserved. To solve the maximization problem, a greedy algorithm [102] is proposed, i.e., workers that take the least time for model upload and update are iteratively selected for training. The simulation results show that compared with the FL protocol which only accounts for training deadline without performing worker selection, *FedCS* can achieve higher accuracy since *FedCS* is able to involve more workers in each training round [3].

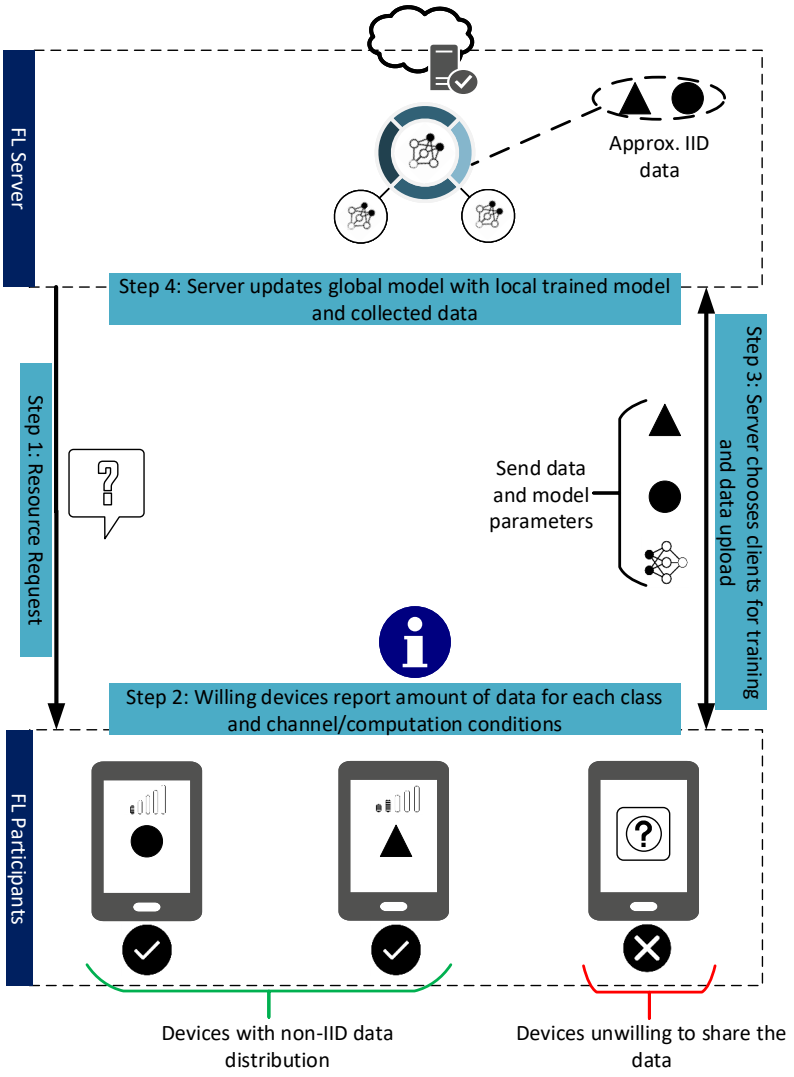

FIGURE 2.4: Worker selection under the FedCS and Hybrid-FL protocol.

However, *FedCS* has been tested only on simple DNN models. When extended to the training of more complex models, it may be difficult to estimate how many workers should be selected. In addition, there is a bias towards selecting workers with devices that have better computing capabilities. These workers may not hold data that is representative of the population distribution. In particular, we revisit the fairness issue [103] subsequently in this section.

While *FedCS* addresses heterogeneity of resources among workers in FL, the authors in [104] extend their work on the *FedCS* protocol with the Hybrid-FL protocol

that deals with differences in data distributions among workers. The dataset of workers participating in FL may be non-IID since it is reflective of each individual user's specific characteristics. As we have discussed in Section 2.2.2, the non-IID dataset may significantly degrade the performance of the *FedAvg* algorithm [59]. One proposed measure to address the non-IID nature of the dataset is to distribute publicly available data to workers, such that the EMD between their on-device dataset and the population distance is reduced. However, such a dataset may not always exist, and workers may not download them for security reasons. Thus, an alternative solution is to construct an approximately IID dataset using inputs from a limited number of privacy insensitive workers [104]. In the Hybrid-FL protocol, during the *Resource Request* step (Fig. 2.4), the MEC operator asks random workers if they permit their data to be uploaded. During the worker selection phase, apart from selecting workers based on computing capabilities, workers are selected such that their uploaded data can form an approximately IID dataset in the server, i.e., the amount of collected data in each class has close values (Fig. 2.4). Thereafter, the server trains a model on the collected IID dataset and merges this model with the global model trained by workers. The simulation results show that even with just 1% of workers sharing their data, classification accuracy for non-IID data can be significantly improved as compared to the aforementioned *FedCS* benchmark where data are not uploaded at all. However, the recommended protocol can violate the privacy and security of users, especially if the FL server is malicious. In addition, the proposed measure can be costly especially in the case of videos and images. As such, it is unlikely that workers will voluntarily upload data when they can free ride on the efforts of other volunteers. For feasibility, well-designed incentive and reputation mechanisms are needed to ensure that only trustworthy workers are allowed to upload their data.

In general, the mobile edge network environment in which FL is implemented is dynamic and uncertain with variable constraints, e.g., wireless network and energy conditions. Thus, this can lead to training bottlenecks. To this end, Deep Q-Learning (DQL) can be used to optimize resource allocation for model training as proposed in [105]. The system model is a Mobile Crowd Machine Learning (MCML) setting that enables workers in a mobile crowd network to collaboratively train DNN models required by an FL server. The participating mobile devices are constrained by energy, CPU, and wireless bandwidth. Thus, the server needs to determine the proper amounts of data, energy, and CPU resources that the mobile

devices use for training to minimize energy consumption and training time. Under the uncertainty of the mobile environment, a stochastic optimization problem is formulated. In the problem, the server is the agent, the state space includes the CPU and energy states of the mobile devices, and the action space includes the number of data units and energy units taken from the mobile devices. To achieve the objective, the reward function is defined as a function of the accumulated data, energy consumption, and training latency. To overcome the large state and action space issues of the server, the DQL technique based on Double Deep Q-Network (DDQN) [106] is adopted to solve the server's problem. The simulation results show that the DQL scheme can reduce energy consumption by around 31% compared with the greedy algorithm, and training latency is reduced up to 55% compared with the random scheme. However, the proposed scheme is applicable only in federations with few participating mobile devices.

The aforementioned resource allocation approaches focus on improving the training efficiency of FL. However, this may cause some FL workers to be left out of the aggregation phase because they are stragglers with limited computing or communication resources.

One consequence of this *unfair* resource allocation, a topic that is commonly explored in resource allocation for wireless networks [107] and ML [108]. For example, if the worker selection protocol selects mobile devices with higher computing capabilities to participate in each training round [101], the FL model will be over-represented by the distribution of data owned by workers with devices that have higher computing capabilities. Therefore, the authors in [103] and [109] consider fairness as an additional objective in FL. Fairness is defined in [109] to be the *variance* of performance of an FL model across workers. If the variance of the testing accuracy is large, this implies the presence of more bias or less fairness, since the learned model may be highly accurate for certain workers and less so for other underrepresented workers. The authors in [109] propose the q-Fair FL (q-FFL) algorithm that reweighs the objective function in *FedAvg* to assign higher weights in the loss function to devices with higher loss. The modified objective function is as follows:

$$\min_{w} F_q(w) = \sum_{k=1}^{m} \frac{p_k}{q+1} F_k^{q+1}(w) \tag{2.4}$$

where F_k refers to the standard loss functions presented in Table 2.2, q refers to the calibration of fairness in the system model, i.e., setting $q = 0$ returns the

formulation to the typical FL objective, and p_k refers to the ratio of local samples to the total number of training samples. In fact, this is a generalization of the Agnostic FL (AFL) algorithm proposed in [103], in which the device with the highest loss dominates the entire loss function. The simulation results show that the proposed q-FFL can achieve lower variance of testing accuracy and converges more quickly than the AFL algorithm. However, as expected, for some calibrations of the q-FFL algorithm, there can be convergence slowdown since stragglers can delay the training process. As such, an asynchronous aggregation approach can potentially be considered for use with the q-FFL algorithm. As an alternative, the authors in [110] propose a neural network based approach to estimate the local models of FL workers that are left out during training, so as to reduce inference bias.

2.4.2 Joint Radio and Computation Resource Management

While most FL studies have previously assumed orthogonal-access schemes such as Orthogonal Frequency-division Multiple Access (OFDMA) [111], the authors in [112] propose a multi-access Broadband Analog Aggregation (BAA) design for the communication-latency reduction in FL. Instead of performing communication and computation separately during global aggregation at the server, the BAA scheme builds on the concept of over-the-air computation [113] to *integrate* computation and communication through exploiting the signal superposition property of a multiple-access channel. The proposed BAA scheme allows the reuse of the whole bandwidth (Fig. 2.5(a)) whereas OFDMA orthogonalizes bandwidth allocation (Fig. 2.5(b)). As such, for orthogonal-access schemes, communication latency increases in direct proportion with the number of workers whereas for multi-access schemes, latency is independent of the number of workers. The bottleneck of signal-to-noise ratio (SNR) during BAA transmission is the participating device with the longest propagation distance given that devices that are nearer have to lower their transmission power for amplitude alignment with devices located further. To increase SNR, workers with longer propagation distances have to be dropped. However, this leads to the truncation of model parameters. As such, to manage the SNR-truncation tradeoff, three scheduling schemes are considered namely a) *Cell-interior scheduling*: workers beyond a distance threshold are not

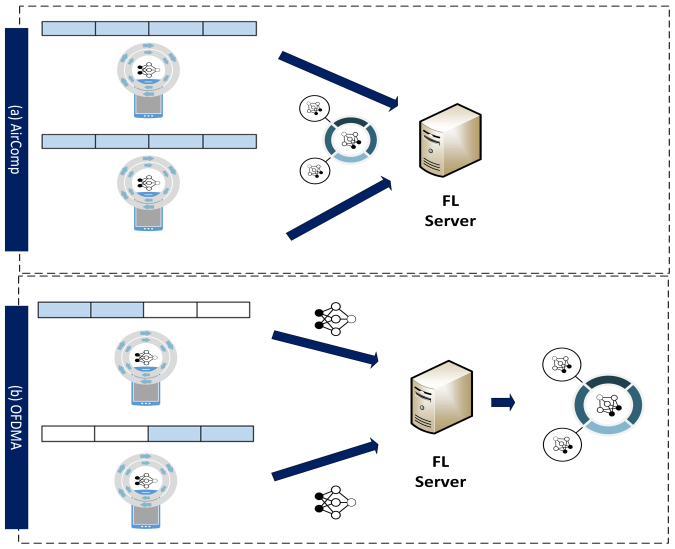

FIGURE 2.5: A comparison between (a) BAA by over-the-air computation which reuses bandwidth (above) and (b) OFDMA (below) which uses only the allocated bandwidth.

scheduled, b) *All-inclusive scheduling*: all workers are considered, and c) *Alternating scheduling*: edge server alternates between the two aforementioned schemes. The simulation results show that the proposed BAA scheme can achieve similar test accuracy as the OFDMA scheme while achieving latency reduction from 10 times to 1000 times. As a comparison between the three scheduling schemes, the cell-interior scheme outperforms the all-inclusive scheme in terms of test accuracy for high mobility networks where workers have rapidly changing locations. For low mobility networks, the alternating scheduling scheme outperforms cell-interior scheduling.

As an extension, the authors in [114] also introduce error accumulation and gradient sparsification in addition to over-the-air computation. In [112], gradient vectors that are not transmitted as a result of power constraints are completely dropped. To improve the model accuracy, the untransmitted gradient vectors can first be stored in an error accumulation vector. In the next round, local gradient estimates are then corrected using the error vector. In addition, when there are bandwidth limitations, the participating device can apply gradient sparsification to keep only elements with the highest magnitudes for transmission. The elements that are not transmitted are subsequently added to the error accumulation vector for gradient estimate correction in the next round. The simulation results show that the

proposed scheme can achieve higher test accuracy than over-the-air computation without error accumulation or gradient sparsification since it corrects gradient estimates with the error accumulation vector and allows for more efficient utilization of the bandwidth.

Similar to [112] and [114], the authors in [115] propose an integration of computation and communication via over-the-air computation. However, it is observed that aggregation error incurred during over-the-air computation can lead to a drop in model accuracy [116] as a result of signal distortion. As such, a worker selection algorithm is proposed in which the number of devices selected for training is maximized so as to improve statistical learning performance [3] while keeping the signal distortion below a threshold. Due to the nonconvexity [117] of the mean-square-error (MSE) constraint and intractability of the optimization problem, a difference-of-convex functions (DC) algorithm [118] is proposed to solve the maximization problem. The simulation results show that the proposed DC algorithm is scalable and can also achieve near-optimal performance that is comparable to global optimization, which is non-scalable due to its exponential time complexity. In comparison with other state-of-the-art approaches such as the semidefinite relaxation technique (SDR) proposed in [119], the proposed DC algorithm can also select more workers, thus also achieving higher model accuracy.

2.4.3 Adaptive Aggregation

The proposed *FedAvg* algorithm synchronously aggregates parameters as shown in Fig. 2.6(a) and is thus susceptible to the straggler effect, i.e., each training round only progresses as fast as the slowest device since the FL server waits for *all* devices to complete local training before global aggregation can take place [99]. In addition, the model does not account for workers that can join halfway when the training round is already in progress. As such, the asynchronous model is proposed to improve the scalability and efficiency of FL. For asynchronous FL, the server updates the global model whenever it receives a local update (Fig. 2.6(b)). The authors in [99] find empirically that an asynchronous approach is robust to workers joining halfway during a training round, as well as when the federation involves participating devices with heterogeneous processing capabilities. However, the model convergence is found to be significantly delayed when data are non-IID

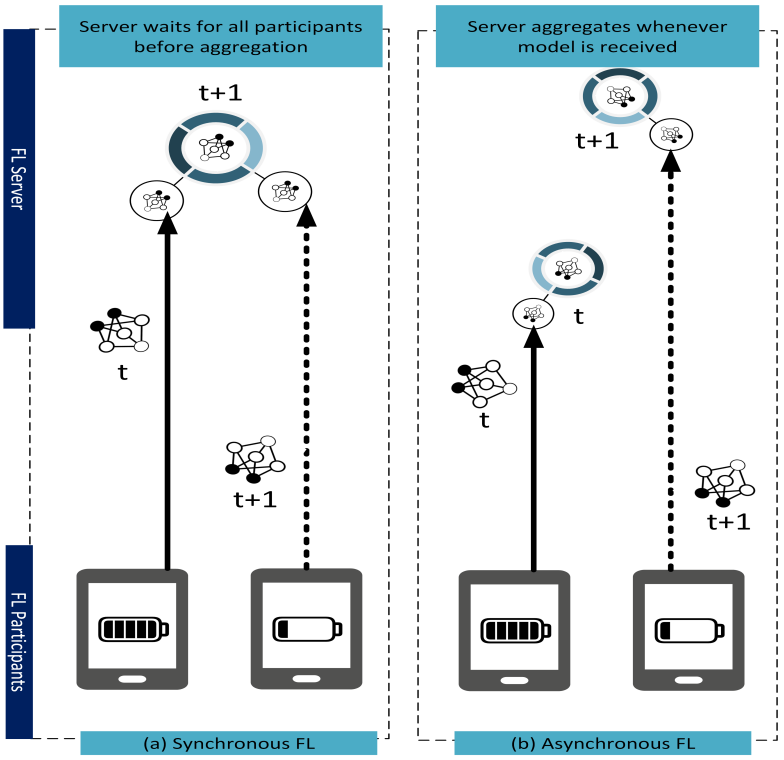

FIGURE 2.6: A comparison between (a) synchronous and (b) asynchronous FL.

and unbalanced. As an improvement, [120] proposes the *FedAsync* algorithm in which newly received local updates are adaptively weighted according to staleness, that is defined as the difference between the current epoch and iteration in which the received update belongs to. For example, a stale update from a straggler is outdated since it should have been received in previous training rounds. As such, it is weighted less. In addition, the authors also prove the convergence guarantee for a restricted family of non-convex problems. However, the current hyperparameters of the *FedAsync* algorithm still have to be tuned to ensure convergence in different settings. As such, the algorithm is still unable to generalize to suit the dynamic computation constraints of heterogeneous devices. In fact, given the uncertainty surrounding the reliability of asynchronous FL, synchronous FL remains to be the approach most commonly used today [121].

For most existing implementations of the *FedAvg* algorithm, the global aggregation phase occurs after a fixed number of training rounds. To better manage the dynamic resource constraints, the authors in [57] propose an adaptive global aggregation scheme that varies the global aggregation frequency so as to ensure desirable model performance while ensuring efficient use of available resources, e.g., energy, during the FL training process. In [57], the MEC system model used consists of (a)

the local update phase where the model is trained using local data, (b) edge aggregation phase where the intermediate aggregation occurs and (c) global aggregation phase where the updated model parameters are received and aggregated by the FL server. In particular, the authors study how the training loss is affected when the total number of edge servers aggregations and local updates between global aggregation intervals varies. For this, a convergence bound of gradient descent with non-IID data is first derived. Then, a control algorithm is subsequently proposed to adaptively choose the optimal global aggregation frequency based on the most recent system state. For example, if global aggregation is too time consuming, more edge aggregations will take place before communication with the FL server is initiated. The simulation results show that the adaptive aggregation scheme outperforms the fixed aggregation scheme in terms of loss function minimization and accuracy within the same time budget. However, the convergence guarantee of the adaptive aggregation scheme is only considered for convex loss functions currently.

2.4.4 Incentive Mechanism

The authors in [122] propose a service pricing scheme in which workers serve as training service providers for a model owner. In addition, to overcome energy inefficiency in the transfer of model updates, a cooperative relay network is proposed to support model update transfer and trading. The interaction between workers and model owner is modelled as a Stackelberg game [123] in which the model owner is the buyer and workers are the sellers. In the Stackelberg game, each rational worker can noncooperatively decide on its own profit maximization price. In the lower-level subgame, the model owner determines the size of training data to maximize profits considering the increasing concave relationship between the learning accuracy of the model and the size of training data. In the upper-level subgame, the workers decide the price per unit of data to maximize their individual profits. The simulation results show that the proposed mechanism can ensure the uniqueness of the Stackelberg equilibrium. For example, model updates that contain valuable information is priced higher at the Stackelberg equilibrium. In addition, model updates can be transferred cooperatively, thus reducing congestion in communication and improving energy efficiency. However, the simulation environment only involves relatively few mobile devices.

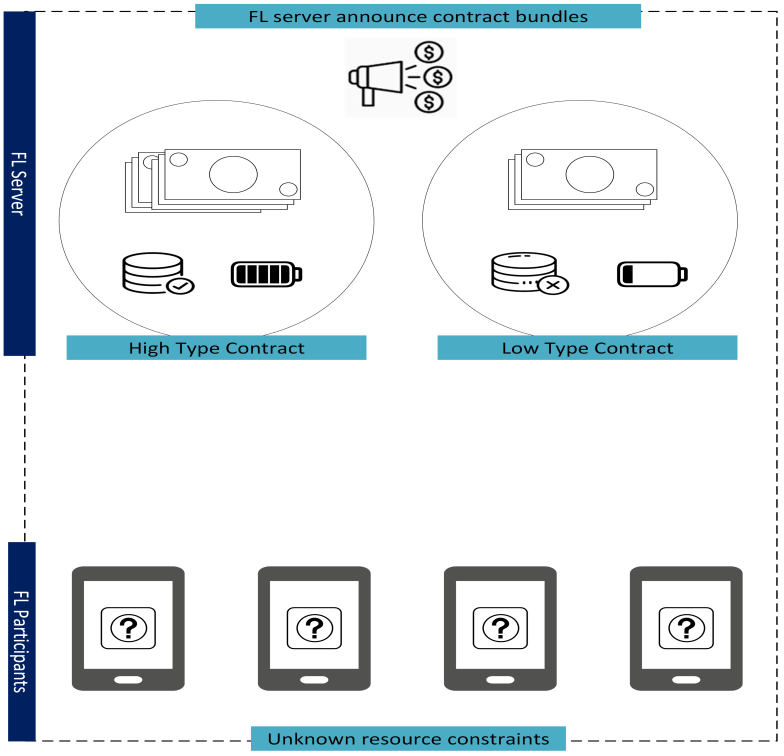

FIGURE 2.7: Workers with unknown resource constraints maximize their utility only if they choose the bundle that best reflects their constraints.

Similar to [122], the authors in [124, 125] also model the interaction between workers and model owner as a Stackelberg game, which is well-suited to represent the FL server-worker interaction involved in FL.

However, unlike the aforementioned conventional approaches to solving Stackelberg formulations, a DRL-based approach is adopted together with the Stackelberg game by the authors in [126]. In the DRL formulation, the FL server acts as an agent that decides a payment in response to the participation level and payment history of edge nodes, with the objective of minimizing incentive expenses. Then, the edge nodes determine an optimal participation level in response to the payment policy. This learning based incentive mechanism design enables the FL server to derive an optimal policy in response to its observed state, without requiring any prior information.

In contrast to [122, 124–126], the authors in [127] propose an incentive design using a contract theoretic [20] approaches to attract workers with high-quality data for FL. In particular, well-designed contracts can reduce information asymmetry through self-revealing mechanisms in which workers select only the contracts

specifically designed for their types. For feasibility, each contract must satisfy the Individual Rationality (IR) and Incentive Compatibility (IC) constraints. For IR, each worker is assured of a positive utility when the worker participates in the federation. For IC, every utility maximizing worker only chooses the contract designed for its own type. The model owner aims to maximize its own profits subject to IR and IC constraints. As illustrated in Fig. 2.7, the optimal contracts derived are self-revealing such that each high-type worker with higher data quality only chooses contracts designed for its type, whereas each low-type worker with lower data quality does not have the incentive to imitate high-type workers. The simulation results show that all types of workers only achieve maximum utility when they choose the contract that matches their types. In addition, the proposed contract theory approach also has better performance in terms of profit for the model owner compared with the Stackelberg game-based incentive mechanism. This is because under the contract theoretic approach, the model owner can extract more profits from the workers whereas under the Stackelberg game approach, the workers can optimize their individual utilities. In fact, the information asymmetry between FL servers and workers makes contract theory a powerful and efficient tool for mechanism design in FL.

The authors in [127] further introduce reputation as a metric to measure the reliability of FL workers and design a reputation-based worker selection scheme for reliable FL [54]. In this setting, each worker has a reputation value [128] derived from two sources, (a) direct reputation opinions from past interactions with the FL server and (b) indirect reputation opinions from other task publishers, i.e., other FL servers. The indirect reputation opinions are stored in an open-access reputation blockchain [129] to ensure secure reputation management in a decentralized manner. Before model training, the workers choose a contract that best fits its dataset accuracy and resource conditions. Then, the FL server chooses the workers that have reputation scores that are larger than a prespecified threshold. After the FL task is completed, i.e., a desirable accuracy is achieved, the FL server updates the reputation opinions, which are subsequently stored in the reputation blockchain. The simulation results show that the proposed scheme can significantly improve the accuracy of the FL model since unreliable workers are detected and not selected for FL training.

As a summary, we present the works reviewed in this chapter in Table 2.4.

TABLE 2.4: Approaches to resource allocation in FL.

Approaches	Ref.	Key Ideas	Tradeoffs and Shortcomings
Worker Selection	[101]	FedCS to select workers based on computation capabilities so as to complete FL training before specified deadline	Difficult to estimate training duration accurately for complex models
	[104]	Following [101], Hybrid-FL to select workers so as to accumulate IID, distributable data for FL model training	Request of data sharing may defeat the original intent of FL
	[105]	DRL to determine resource consumption by FL workers	DRL models are difficult to train especially when the number of FL workers are large
	[130]	Following [105], DRL for resource allocation with mobility-aware FL workers	
	[109]	Fair resource allocation to reduce variance of model performance	Convergence delays with more fairness
Joint Radio and Computation Resource Management	[112]	Propose BAA to integrate computation and communication through exploiting the signal superposition property of multiple-access channel	Signal distortion can lead to drop in accuracy, the scalability is also an issue when large heterogeneous networks are involved
	[114]	Improves on [112] by accounting for gradient vectors that are not transmitted due to power constraints	
	[115]	Improves on [112] using the DC algorithm to minimize aggregation error	
Adaptive Aggregation	[99]	Asynchronous FL where model aggregation occurs whenever local updates are received by FL server	Significant delay in convergence in non-IID and unbalanced dataset
	[57]	Adaptive global aggregation frequency based on resource constraints	Convergence guarantees are limited to restrictive assumptions
Incentive Mechanism	[122, 124–126]	Stackelberg game for incentivizing higher quantities of training data or compute resource contributed	FL server derives lower profits. Also, assumption that there is only one FL server
	[127]	Contract theoretic approach to incentivize FL workers	Assumption that there is only one FL server
	[54]	Reputation mechanism to select effective workers	

2.5 Conclusion and Chapter Discussion

This chapter has presented a tutorial of FL and a comprehensive survey on the issues regarding FL implementation. Firstly, we begin with an introduction to the motivation for MEC, and how FL can serve as an enabling technology for collaborative model training at mobile edge networks. Then, we describe the fundamentals of FL model training. Afterwards, we provide detailed reviews, analyses, and comparisons of approaches for emerging implementation challenges in FL. Furthermore, we also briefly discuss the implementation of FL for privacy-preserving mobile edge network optimization. Below, we list the lessons learned from our review of the literature, which paves the subsequent chapters ahead in this thesis.

i The study of incentive mechanism design is a particularly important aspect of FL. Similar to crowdsensing, the process of FL requires the workers to incur costs in data collection and model training. However, existing studies in incentive design [122, 127] does not consider FL from a crowdsensing perspective, i.e., the data has been assumed to be stored in the worker device before training occurs. As a result, the operational tradeoffs that occur during data collection and model training, e.g., between information freshness and service latency, for a crowdsensing-FL framework is under-explored. To this end, we leverage a contract optimization approach to devise a task-aware incentive scheme in Chapter 3.

ii Incentivizing workers is a challenging task. In a large network, the workers have varying costs, which they are not obliged to report truthfully. With the use of self-revealing mechanisms in contract theory [54, 127], the hidden types of workers are accounted in devising incentive schemes for FL. However, existing studies consider only the single dimensional contract, which is insufficient to capture the complexities and heterogeneities required for workers in the network. In Chapter 4, we devise the multi-dimensional contract incentive scheme to account for multiple sources of information asymmetry in an FL network. Specifically, we utilize this mechanism to incentivize FL in the UAV network, which is characterized by heterogeneity in traversal cost, communication cost, and computation costs.

iii Existing studies of incentive mechanism design generally assume that a federation enjoys a monopoly. In particular, each FL network is assumed to only

consist of multiple individual workers collaborating with a sole FL server. There can be exceptions to this setting as follows: (a) the workers may be competing data owners who are reluctant to share their model parameters since the competitors also benefit from a trained global model and (b) the FL servers may compete with other FL servers, i.e., model owners. The FL servers may also restrict the participation of workers in more than one FL training. In this case, the formulation of the incentive mechanism design will be vastly different from that proposed. In Chapter 5, we devise a deep learning based auction mechanism to model the competition among multiple model owners for the restricted participation of model owners.

iv In heterogeneous mobile networks, the consideration of resource allocation is important to ensure efficient FL. In Section 2.4, we have explored different dimensions of resource heterogeneity for consideration, e.g., varying computation and communication capabilities, willingness to participate, and quality of data for local model training. In addition, we have explored various tools that can be considered for resource allocation. Naturally, traditional optimization approaches have also been well explored in radio resource management for FL, given the high dependency on communications efficiency in FL. However, the assumption of static system constraints has been common. In fact, most studies do not consider the dynamic and self-organizing aspects of workers in the FL network. In Chapter 5, we consider the resource optimization in FL networks subjected to *dynamic* system states.

v In Section 2.3, we learn that communication cost reduction using model compression schemes comes with sacrifices in terms of either higher computation costs or lower inference accuracy. Similarly, there exist different tradeoffs to be considered in resource allocation. A scalable model is thus one that enables customization to suit varying needs. For example, the study of [109] allows the FL server to calibrate levels of fairness when allocating training importance, whereas the study in [131] enables the tradeoffs between training completion time and energy expense to be calibrated by the FL system administrator. Apart from working to directly reduce the size of the model communicated, studies on FL can draw inspiration from applications and approaches in the MEC paradigm. In Chapter 5, we propose a hierarchical FL framework that leverages the computation and communication capacities of

edge servers towards scalable and efficient FL. This circumvents the need for model compression that results in accuracy loss.

Chapter 3

Task-aware Contract Design to Tradeoff the Service Latency and Information Freshness in Federated Learning

3.1 Introduction

In recent years, the enhanced sensing and communication capabilities of modern Internet of Things (IoT) devices have promoted a growing interest in the deployment of the Industrial IoT (IIoT) for various applications, e.g., in smart agriculture [132], supply chain [133], and logistics [134].

The wealth of data collected by the IIoT devices[1] enables effective AI [23] based models to raise the productivity of labor-intensive industries such as agriculture [135]. In smart agriculture, the wireless sensor network can be deployed in crop fields [136] to capture humidity and temperature readings, as well as images for model training, e.g., for rapid identification of insect infestation [137]. This contributes to food security for consumers and income stability for producers.

[1]While we use the example of IIoT in this chapter, note that our findings apply to other scenarios as well.

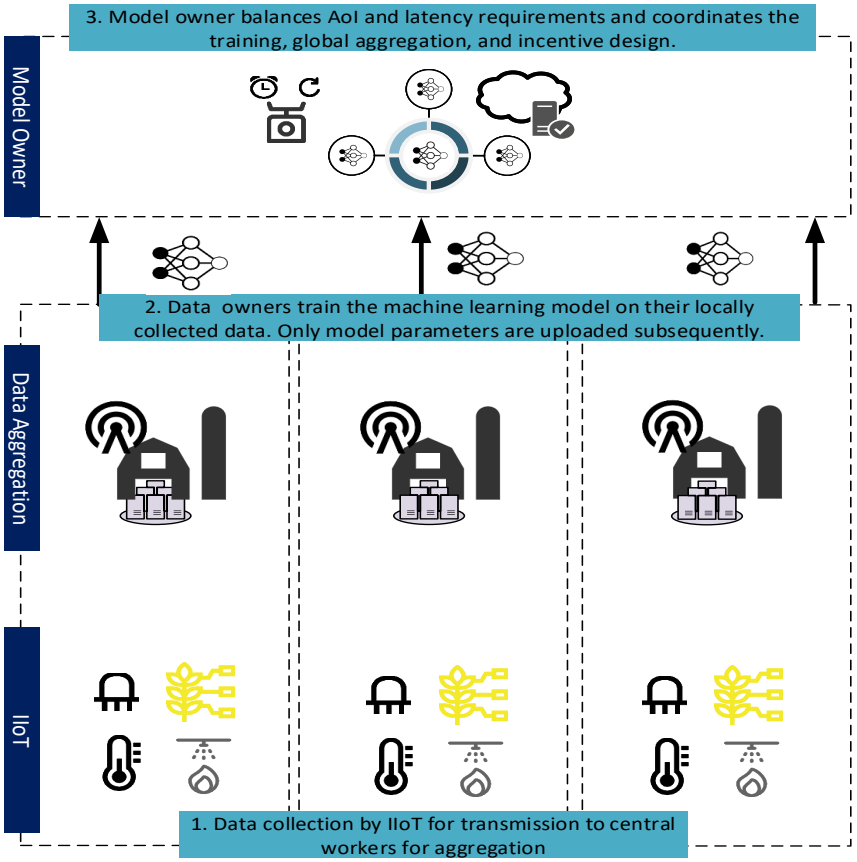

FIGURE 3.1: The system model.

However, the state-of-the-art representation-learning based models, e.g., Deep Learning (DL), typically require large quantities of training data to outperform conventional hand-crafted analytical methods. An IIoT network deployed by a single data owner may not be able to capture sufficient training samples across all classes to build effective models that can generalize well [137]. For example, a single farm may lack sufficient X-ray images of damaged wheat kernels [138] for model training to effectively identify grains affected by granary weevils. Naturally, independent data owners can collaborate by sharing their data. However, the data owners may be reluctant to share their private raw data with industrial competitors. To this end, we propose the adoption of a FL [78] based approach to enable collaborative model training across a federation of industrial data owners.

Our system model (Fig. 3.1) consists of multiple data owners, hereinafter *workers*, and a model owner, e.g., a company that develops smart agriculture applications. The worker first collects the data, e.g., temperature readings or crop images, with their deployed IIoT devices for transmission to a personal data aggregator, e.g., the

IoT gateway. Whenever the model owner requires model training, a request and an initialized set of model parameters are transmitted to the worker. Then, the worker trains the model on its local data and transmits only the updated parameters to the model owner for aggregation. The local training and transmission for global aggregation iterates till a desired accuracy is achieved.

In conventional FL studies, it is usually assumed that the worker either has the data ready for model training, or needs to collect the data upon request. As such, the crucial operational tradeoff experienced during data collection (sensing) and FL model training between *service latency* and *Age of Information (AoI)* is under-explored. In particular, the AoI is defined as the time elapsed between data collection, i.e., collection or aggregation of data from the distributed IIoT devices, to completion of the FL based training, and thus captures information freshness of the model parameters received as derived from the collected data [8], whereas the service latency is defined as the time elapsed between the initiation of the FL training request to the completion of the FL based training. On one hand, if the data are aggregated only upon request, the information is at its freshest, i.e., the AoI is low. On the other hand, the service latency incurred during data collection can be intolerable especially if the model owner prefers faster task completion. For certain tasks, e.g., automated grain quality evaluation [137], ensuring a low AoI is less crucial given that the training and test data remain structurally similar across time. However, for other tasks, e.g., the identification of rapidly mutating virus strains in crops [139], a low AoI is more crucial.

In addition, there exists an incentive mismatch between the workers and model owner in managing this tradeoff. For example, while the model owner may prefer more frequent data updating to ensure a low AoI, the update cost, e.g. in terms of computation, communication, and energy consumption, incurred can be prohibitive for the worker. Moreover, the model owners are unaware of the worker types, i.e., data update cost, due to information asymmetry.

To this end, we propose a contract-theoretic task-aware incentive mechanism to motivate workers to update the data accordingly in consideration of the different preferences of the model owner towards AoI and service latency. For example, if a low AoI is more crucial to the model owner, the contract bundles will be designed such that the workers update the data more frequently. Otherwise, if a low service

latency is more crucial, the workers update the data less frequently. The self-revealing property of the contract-theoretic mechanism design ensures that the workers can be appropriately rewarded based on their types, which is otherwise hidden information. The main contributions of this chapter are summarized as follows:

- We introduce the AoI and service latency tradeoff in a system model involving the FL-based model training.
- We leverage on the self-revealing properties of the contract-theoretic incentive mechanism design to appropriately reward workers based on their data updating cost, amid information asymmetry. In addition to the distinct worker types, our contract also holds when it is applied to the continuous worker types, thus validating its feasibility for practical applications.
- We show that our incentive mechanism design can be calibrated to suit the varying preferences of the model owner for AoI and service latency.

The organization of this chapter is as follows. In Section 3.2, we discuss the system model and problem formulation. In Section 3.3, we study and analyze the contract theoretic formulation. In Section 3.4, we provide the performance evaluation and Section 3.5 concludes the chapter.

3.2 System Model and Problem Formulation

The model owner initiates a synchronous FL task involving a set $\mathcal{I} = \{1, \ldots, i, \ldots, I\}$ of I workers that last for a fixed duration T. During the FL task, there can be more than one instance of model training request initiated by the model owner, e.g., to ensure that the global model is kept up-to-date, through model training with updated data. We assume that each instance of request arrival follows the Poisson process[2] [140]. An FL based model training is first initiated through the request of the model owner. Each model training takes place over K iterations to minimize the global loss $F^K(\boldsymbol{w})$ where K is stipulated by the model owner

[2]Note that the Poisson process is used since there is no way to know the exact time of the request arrival. However, from historical data, we can know the average interval duration between the request arrivals.

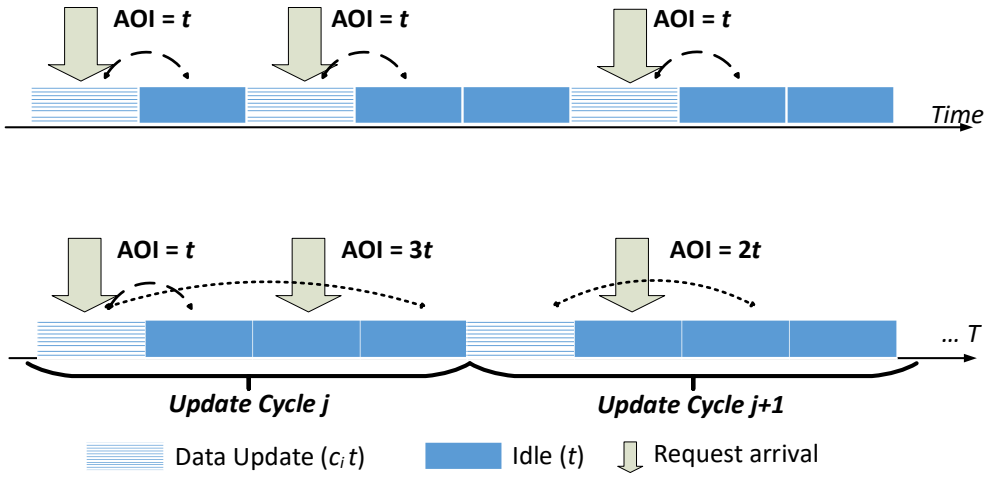

FIGURE 3.2: AoI model of FL training (a) without and (b) with caching. Note that $c_i \geq 1$

and $\mathcal{K} = \{1, \ldots, k, \ldots, K\}$. Each k^{th} iteration in turn consists of three steps [78] namely: (a) *Local Computation*: The worker trains the received global model $\boldsymbol{w}^{(k)}$ locally using the processed data, (b) *Wireless Transmission*: The worker transmits the model parameter update to the model owner, and (c) *Global Model Parameter Update*: All parameter updates received from the I workers are aggregated to derive an updated global model $\boldsymbol{w}^{(k+1)}$, which is then transmitted back to the worker for the $(k+1)^{th}$ iteration.

Following [140], we denote the time taken to complete K iterations of local computation and wireless transmission, i.e., one instance of model training, as t, i.e., period. Note that t applies across all workers given that in the synchronous FL scheme, model training duration is constrained by the slowest worker, i.e., a worker has to wait for others to complete the training before the model can be aggregated [2]. Moreover, each duration T can be represented in terms of instances of t (Fig. 3.2), and the time taken for worker i to collect and process the data for model training is denoted by a constant $c_i t$, $c_i \in \mathbb{N}$. Note that the data collection cycle can be in any number of blocks of t. Without loss of generality, a model training request arrives at the beginning of each period. In the following, we consider the AoI and service latency of the conventional FL scheme and the FL with caching scheme.

3.2.1 AoI and Service Latency Model

In the conventional FL scheme, the worker collects and processes the data on-demand upon the request of the model owner. As such, there is a constant minimum AoI regardless of the period in which the request arrives since data are collected only when it is required. The AoI is the time taken to complete FL training where:

$$\bar{A}_i^c = t. \tag{3.1}$$

Moreover, the service latency of the conventional FL scheme is given by the summation of periods taken for data collection and model training where:

$$D_i^c = c_i t + t. \tag{3.2}$$

In contrast, with caching, the worker updates the cached data periodically every θ_i interval, independent of the period in which the request arrives, where:

$$\theta_i = c_i t + a_i t, \ a_i \in \mathbb{N}, \tag{3.3}$$

and responds to the arriving request through local model training on the cached data. Note that we assume the ideal cache, i.e., each data owner is able to cache and update all the data, and leave the design of specific caching schemes for our future works. For each worker i, the T duration spans across $J_i = \frac{T}{\theta_i}$ update cycles.

Following the characteristics of the Poisson process, the probability of a request arrival is identical across periods and is given by $\frac{1}{T}$. If a request arrives at the n^{th} period during the data collection, the service latency is $c_i t + t - (n-1)t$. Otherwise, if the request arrives at any of the remaining periods within an update cycle, the service latency is t. As such, the average service latency $\bar{D}_s$ of the FL scheme with caching is:

$$\begin{aligned}
\bar{D}_s &= \left[\frac{1}{c_i + a_i}\Big((c_i t + t) + \cdots + (c_i t + t - (c_i - 1)t)\Big) + \frac{a_i}{c_i + a_i} t\right] \\
&= \frac{1}{c_i + a_i}\left[c_i(c_i + t) - t_i \sum_{q=1}^{c_i - 1} q\right] + \frac{a_i}{c_i + a_i} t \\
&= \frac{1}{c_i + a_i}\left[\frac{c_i t}{2}(c_i + 3)\right] + \frac{a_i}{c_i + a_i} t.
\end{aligned} \tag{3.4}$$

For content caching, the AoI of the data is at the minimum t only if the request comes during the data collection phase, or at the beginning of phase $(c_i + 1)t$. Otherwise, the AoI for a request that arrives at period lt will be $[l - (c_i + 1) + 1]t$, where $l \geq (c_i + 2)t$, i.e., the periods after data collection have been completed. As such, the average AoI is given by:

$$\begin{aligned} \bar{A}_s &= \frac{(c_i+1)}{c_i+a_i} \cdot t + \sum_{l=c_i+2}^{a_i+c_i} \frac{1}{c_i+a_i} \cdot (l-c_i)t \\ &= \frac{t}{c_i+a_i}\left(c_i + 1 + \frac{(a_i-1)(a_i+2)}{2}\right). \end{aligned} \tag{3.5}$$

In comparison with the conventional FL scheme, the FL with caching is a flexible system model that enables the model owner's management of the AoI and service latency tradeoffs. From (3.4) and (3.5), we also observe the tradeoff between service latency and AoI for our choice of cycle length θ_i. Intuitively, a lower θ_i, i.e., shorter cycle length or more update cycles J_i over T, enables a lower average AoI given that data are more frequently updated. However, the service latency increases as well since the updating takes time.

Given that the choice of θ_i involves a variation in resource expense of the workers, e.g, a lower θ_i represents higher data collection and caching cost incurred for worker i, an appropriate incentive mechanism design is required to motivate the workers towards a choice of θ_i that benefits the model owner. In the following, we model the profit functions of the workers and model owner respectively.

3.2.2 Worker and Model Owner Profit Function

The data update cost η_i incurred for worker i per update cycle is given as follows:

$$\eta_i = \alpha_i \left(E_i^T + E_i^C\right) + \beta_i, \tag{3.6}$$

where α_i refers to the unit cost of energy, E_i^T refers to the energy consumed for transmission from the IIoT network to the gateway, E_i^C represents energy consumption for caching of the data [141], and β_i refers to data collection cost.

With statistical information, the workers can be categorized into a set $\mathcal{N} = \{\eta_m : 1 \leq m \leq M\}$ of M data update cost types by data mining tools, e.g., k-means. The

worker types η_m can be characterized by a probability mass function $p(\eta_m)$, where the cost types are indexed in a non-decreasing order $0 < \eta_1 \leq \cdots \eta_m \leq \cdots \leq \eta_M$. Therefore, the utility u_m of the worker type m is given as follows:

$$u_m(\omega_m) = R_m - \eta_m J_m, \tag{3.7}$$

where ω_m indicates the contract pair that consists of the rewards-update cycles bundle (R_m, J_m) designed for the type m worker, R_m refers to the contract rewards, and J_m refers to the number of update cycles.

To model the tradeoff between preferences for service latency and AoI, the model owner profit function can be expressed by

$$\Pi = \sum_{m=1}^{M} Ip(\eta_m)\left(\sigma\left(w_a \Upsilon\left(1 + \frac{\mu}{\bar{A}_m(J_m)}\right) + w_d \Gamma\left(\frac{\phi}{\bar{D}_m(J_m)}\right)\right) - R_m\right), \tag{3.8}$$

where w_a and w_d represent the weighted preferences for information freshness, i.e., the inverse of AoI, and faster task completion, i.e., the inverse of service latency, respectively. Both the inverse of AoI and service latency are functions of J_i. Moreover, $w_a + w_d = 1$ and $w_d, w_a \in [0, 1]$. As an illustration, $w_a > w_d$ represents a model owner that values fresh information over faster task completion. In this regard, the model owner requires workers to have a higher J_m, i.e., more frequent data updating or a shorter cycle length θ_m equivalently. $\Upsilon(\cdot)$ is an increasing concave function with respect to the inverse of AoI to indicate the diminishing returns from information freshness, whereas $\Gamma(\cdot)$ is a linear function with respect to the inverse of service latency following [142, 143]. In addition, σ refers to the profit conversion parameter from AoI and service latency, whereas μ and ϕ are system model parameters.

3.3 Contract Design

In this section, we discuss the conditions for contract feasibility. Then, we relax the constraints to derive the optimal contract $\Omega(\mathcal{N}) = \{\omega_m, 1 \leq m \leq M\}$.

3.3.1 Feasibility Conditions

A feasible contract must satisfy the following constraints:

Definition 3.1. Individual Rationality (IR): Each type m worker achieves non-negative utility if it chooses the contract item designed for its type, i.e., contract item ω_m.

$$u_m(\omega_m) \geq 0, 1 \leq m \leq M. \tag{3.9}$$

Definition 3.2. Incentive Compatibility (IC): Each type m worker achieves the maximum utility if it chooses the contract item designed for its type, i.e., ω_m. As such, it has no incentive to choose contracts designed for other types.

$$u_m(\omega_m) \geq u_m(\omega_{m'}), m \neq m', \ 1 \leq m \leq M. \tag{3.10}$$

The contract formulation is shown as follows:

$$\begin{aligned} &\max_{\Omega} \Pi(\Omega(\mathcal{N})) \\ &\text{s.t. } (3.9), (3.10). \end{aligned} \tag{3.11}$$

However, the optimization problem in (3.11) involves M IR constraints and $M(M-1)$ IC constraints, all of which are non-convex. As such, we proceed to reduce and relax the conditions that guarantee a feasible contract.

Lemma 3.1. For any feasible contract $\Omega(\mathcal{N})$, we have $J_m < J_{m'}$ if and only if $R_m < R_{m'}$, $m \neq m'$.

Proof. We first prove the sufficiency, i.e., if $R_m < R_{m'} \Rightarrow J_m < J_{m'}$. Rearranging the IC constraint:

$$\eta_m J_{m'} - \eta_m J_m \geq R_{m'} - R_m > 0,$$

which implies $\eta_m J_{m'} > \eta_m J_m$ and hence $J_{m'} > J_m$. Next, we prove the necessity, i.e., $J_m < J_{m'} \Rightarrow R_m < R_{m'}$. We consider the IC constraint of the type m' worker

and rearrange the terms to have:

$$\eta_m(J_m - J_{m'}) \geq R_m - R_{m'} \Rightarrow R_m - R_{m'} < 0$$

Hence it follows that $R_m < R_{m'}$. The proof is now completed. □

Lemma 3.2. Monotonicity: For any feasible contract $\Omega(\mathcal{N})$, if $\eta_m > \eta_{m'}$, it follows that $J_m \leq J_{m'}$.

Proof. We adopt the proof by contradiction, i.e., the lemma is incorrect if there exists $J_m > J_{m'}$ such that $\eta_m > \eta_{m'}$.

We consider the IC constraints for type m and m' worker:

$$R_m - \eta_m J_m \geq R_{m'} - \eta_m J_{m'}, \tag{3.12}$$

$$R_{m'} - \eta_{m'} J_{m'} \geq R_m - \eta_{m'} J_m. \tag{3.13}$$

Then, we add the constraints together and rearrange the terms to obtain:

$$-\eta_m J_m - \eta_{m'} J_{m'} \geq -\eta_m J_{m'} - \eta_{m'} J_m, \tag{3.14}$$

$$\underbrace{(\eta_m - \eta_{m'})}_{>0}\underbrace{(J_m - J_{m'})}_{>0} \leq 0. \tag{3.15}$$

Given that $J_m > J_{m'}$ and $\eta_m > \eta_{m'}$, $(\eta_m - \eta_{m'})(J_m - J_{m'}) > 0$, this contradicts with (3.15). As such, there does not exist $J_m > J_{m'}$ such that $\eta_m > \eta_{m'}$ for the feasible contract, which confirms that the lemma is correct. □

From Lemma 3.1, we show the intuitive result that the IC contract offers higher rewards to workers which update the data more frequently, whereas Lemma 3.2 indicates that workers with lower cost of updating are willing to update the data more frequently. This gives us the necessary constraints.

Theorem 3.1. A feasible contract must meet the following conditions:

$$\begin{cases} J_1 \geq J_2 \geq \cdots \geq J_m \geq \cdots \geq J_M \\ R_1 \geq R_2 \geq \cdots \geq R_m \geq \cdots \geq R_M \end{cases} \tag{3.16}$$

Next, we further relax the IR and IC constraints. Intuitively, the minimum utility worker is the worker that incurs the highest cost of data update, i.e., the type M worker.

Lemma 3.3. *(Reduce IR Constraints):* If the IR constraint of the minimum utility worker, i.e., type M, is satisfied, the other IR constraints will also hold.

Proof. From the IC constraint and $\eta_m \geq \eta_M$, we have:

$$R_m - \eta_m J_m \geq R_M - \eta_m J_M \geq R_M - \eta_M J_M \geq 0. \tag{3.17}$$

As such, as long as the IR constraint of the type M worker is satisfied, the IR constraints of other workers will hold. □

Lemma 3.4. *(Reduce IC Constraints):* The IC constraints can be reduced into the Local Downward Incentive Constraints (LDIC).

Proof. Consider three worker types $\eta_{m-1} < \eta_m < \eta_{m+1}$. The two LDICs [144], i.e., constraints between type m and type $m-1$ workers , are provided as follows:

$$R_{m+1} - \eta_{m+1} J_{m+1} \geq R_m - \eta_{m+1} J_m, \text{ and} \tag{3.18}$$

$$R_m - \eta_m J_m \geq R_{m-1} - \eta_m J_{m-1}. \tag{3.19}$$

From Lemma 3.1, we have $R_m \geq R_{m+1}$ when $J_m \geq J_{m+1}$. As such, we can rewrite the LDICs as follows:

$$\begin{aligned} &\eta_{m+1}(J_{m-1} - J_m) \geq \eta_m(J_{m-1} - J_m) \geq R_{m-1} - R_m \Rightarrow \\ &R_{m+1} - \eta_{m+1} J_{m+1} \geq R_m - \eta_{m+1} J_m \geq R_{m-1} - \eta_{m+1} J_{m-1}. \end{aligned} \tag{3.20}$$

As such, we have

$$R_{m+1} - \eta_{m+1} J_{m+1} \geq R_{m-1} - \eta_{m+1} J_{m-1}. \tag{3.21}$$

Hence, if the LDIC constraint holds for type-m worker, it will also hold for type $m-1$ worker. This process can be extended downward from type $m-1$ to type 1

worker, i.e., all DICs hold, as follows:

$$\begin{aligned} R_{m+1} - \eta_{m+1} J_{m+1} &\geq R_{m-1} - \eta_{m+1} J_{m-1} \\ &\geq \cdots \\ &\geq R_1 - \eta_{m+1} J_1. \end{aligned}$$

A similar procedure can be taken to show that if the Local Upward Incentive Constraint (LUIC) holds, all UICs are also satisfied. Given the monotonicity condition in Theorem 3.1 (Page 56), the LDIC also implies the Local Upward Incentive Constraint (LUIC) as follows:

$$R_m - \eta_{m-1} J_m \leq R_{m-1} - \eta_{m-1} J_{m-1}. \tag{3.22}$$

Therefore, the IC constraints can be reduced to the LDIC constraint, which ensures that all UIC and DIC constraints hold. □

With Lemma 3.3, we reduce M IR constraints into a single constraint, i.e., as long as the minimum utility worker has a non-negative utility, the other IR constraints hold. With Lemma 3.4, we reduce $M(M-1)$ IC constraints into $M-1$ constraints, i.e., as long as the LDICs hold, it follows that the other IC constraints hold. We are thus able to derive a tractable set of sufficient conditions for the feasible contract.

Theorem 3.2. A feasible contract must meet the following sufficient conditions:

i $R_1 - \eta_1 J_1 \geq 0$,

ii $R_{m+1} - \eta_{m+1} J_{m+1} + \eta_m J_m \geq R_m \geq R_{m+1} - \eta_m J_{m+1} + \eta_m J_m$.

3.3.2 Contract Optimality

To solve the optimal contract rewards R_m^*, we first establish the dependence of optimal contract rewards $\mathbf{R}$ on the number of updates J. Thereafter, we solve the problem in (3.11) with $\mathbf{J}$ only. Specifically, we obtain the optimal rewards $R^*(\mathbf{J})$ given a set of feasible number of updates for each worker which satisfies the monotonicity constraint $J_1 \geq J_2 \geq \cdots \geq J_m \geq \cdots \geq J_M$.

Theorem 3.3. For a known set of number of update cycles $\mathbf{J}$ satisfying $J_1 \geq \cdots \geq J_m \geq \cdots \geq J_M$ in a feasible contract, the optimal reward is given by:

$$R_m^* = \begin{cases} \eta_m J_m, & \text{if } m = M, \\ R_{m+1} - \eta_m J_{m+1} + \eta_m J_m, & \text{otherwise.} \end{cases} \tag{3.23}$$

Proof. We first assume there exists some $\mathbf{R}^\dagger$ that yields greater profit for the model owner, meaning that the theorem is incorrect, i.e., $\Pi(R^\dagger) > \Pi(R^*)$. This implies there exists at least a $t \in \{1, 2, \ldots, M\}$ that satisfies the inequality $R_t^\dagger < R_t^*$.

According to the LDIC constraint of Lemma 3.4,

$$R_t \geq R_{t-1} - \eta_t J_{t-1} + \eta_t J_t. \tag{3.24}$$

In contrast from Theorem 3.2,

$$R_t^* = R_{t+1} - \eta_t J_{t+1} + \eta_t J_t. \tag{3.25}$$

From (3.24) and (3.25), we can deduce that $R_{t+1}^\dagger < R_{t+1}^*$. Continuing the process up to $t = M$, we obtain $R_M^\dagger \leq R_M^* = \eta_m J_m$, which violates the IR constraint. As such, there does not exist the rewards $\mathbf{R}^\dagger$ that yields greater profit for the model owner. Intuitively, the lowest reward that satisfies the IR and IC constraints is chosen for profit maximization. □

Following Theorem 3.3, we can re-express the optimal rewards as:

$$R_m^* = \eta_M J_M + \sum_{m=i}^{M} \Delta_t, \tag{3.26}$$

where $\Delta_M = 0$, $\Delta_t = -\eta_m J_{m+1} + \eta_m J_m$, and $t = 1, 2, \ldots, M-1$. By substituting the optimal rewards in (3.26) into the profit function of the model owner in (3.8) to derive $G_m(J_m)$, we obtain the following optimization problem:

$$\begin{aligned} &\max_{(R_m^*, J_m^*)} \Pi\left(\Omega\left(\mathcal{N}\right)\right) = \sum_{m=1}^{M} G_m(J_m), \\ &\text{s.t. } J_1 \geq J_2 \geq \cdots \geq J_m \geq \cdots \geq J_M. \end{aligned} \tag{3.27}$$

As such, J_m^* can be derived by separately optimizing each $G_m(J_m)$, e.g., through convex optimization tools, as follows:

$$J_m^* = \arg\max_{J_m} p(\eta_m)\Bigg(\sigma\Bigg(w_a \Upsilon\left(1+\frac{\mu}{\bar{A}_m(J_m)}\right)+ w_d\Gamma\left(\frac{\phi}{\bar{D}_m(J_m)}\right)\Bigg)+\eta_{m-1}J_m\sum_{t=1}^{m-1}p(\eta_t)-\eta_m J_m\sum_{t=1}^{m}p(\eta_t). \tag{3.28}$$

The derived solutions are feasible if and only if they satisfy the monotonicity constraint. Otherwise, we adopt the "Bunching and Ironing" algorithm [145] to adjust the solutions iteratively. Given the concavity of $G_m(J_m)$, the adjusted solutions are globally optimal.

3.3.3 Continuum Worker Types

We have only considered discrete worker types so far, i.e., workers with a fixed M types. In practice, there may be a continuum of worker types [144] with probability density function $f(\eta)$ and cumulative distribution function $F(\eta)$ bounded by $[\underline{\eta}\ , \bar{\eta}]$. The optimization problem can be rewritten as:

$$\begin{aligned}&\max_{\{J(\eta),R(\eta)\}}\int_{\underline{\theta}}^{\bar{\theta}}[R(\eta)-\eta J(\eta)]f(\eta)d\eta\\ &\text{s.t.}\\ &\text{IR:}\quad R(\eta)-\eta J(\eta)\geq 0\\ &\text{IC:}\quad R(\eta)-\eta J(\eta)\geq R(\eta')-\eta J(\eta')\ ,\eta'\neq\eta,\eta'\in[\underline{\eta},\bar{\eta}]\end{aligned} \tag{3.29}$$

Similarly, the IR constraint can be reduced to a single constraint involving the lowest utility worker, i.e., $u(\bar{\eta})\geq 0$, whereas the IC constraints can be reduced as follows.

Lemma 3.5. The IC constraints can be reduced into the monotonicity and Local IC (LIC) constraints :

i Monotonicity: $\frac{dJ(\eta)}{d\eta}\geq 0$,

ii LIC: $R'(\eta)=\eta J'(\eta)\frac{dJ(\eta)}{d\eta}$.

Proof. The monotonicity constraint can be derived following the procedures in Section 3.3.1. The LIC constraint [144] can be proven by contradiction, i.e., we assume there exists at least one η' which violates the IC constraint where:

$$R(\eta) - \eta J(\eta) < R(\eta') - \eta J(\eta'), \tag{3.30}$$

which implies:

$$\int_{\eta}^{\eta'} \left[R'(x) - \eta J'(x) \frac{dJ(x)}{dx} \right] dx > 0. \tag{3.31}$$

From the LIC, we have $\int_{\eta}^{\eta'} \left[R'(x) - xJ'(x) \frac{dJ(x)}{dx} \right] dx = 0$. For $\eta < x < \eta'$, the monotonicity condition implies $\eta J'(\eta) \frac{dJ(\eta)}{d\eta} \leq xJ'(\eta) \frac{dJ(\eta)}{d\eta}$. As such, it follows that:

$$\int_{\eta}^{\eta'} \left[R'(x) - \eta J'(x) \frac{dJ(x)}{dx} \right] dx < 0, \tag{3.32}$$

which contradicts with (3.31). Therefore, there does not exist a η' that violates the IC constraint, i.e., the lemma is correct. □

With the constraints relaxed, the optimization problem in (3.29) can be solved to derive the contract pairs $(J^*(\eta), R^*(\eta))$ using a similar approach for (3.27). This validates that our solution holds even in the case of continuum types.

3.4 Performance Evaluation

In this section, we first compare the AoI and service latency for the FL with caching and the conventional FL scheme. Then, we evaluate the optimality of our designed contract, with the additional consideration of how the contract bundle is calibrated across different weight preferences.

Unless otherwise stated, the key simulation parameters are provided in Table 3.1. We assume that there are 50 workers in our system model, with varying update costs η_i modeled by the normal distribution. Then, the k-means is adopted to obtain $m = 5$ worker types for contract design. Note that we keep c constant, i.e., $c_i = c, \forall i \in \mathcal{I}$ to enable a focused discussion on data update frequencies. In other words, the workers vary a_i, i.e., periods between data collection, to respond to the

TABLE 3.1: Table of Key Simulation Parameters.

Simulation Parameters	Value
Update cost η	$\mathcal{N}(500, 300)$
Profit conversion parameter σ	$100,000$
AoI to profit conversion μ	$100,000$
Latency to profit conversion ϕ	2
Total time period T	$1000s$
Unit of time taken for data aggregation c	5
Time taken for FL training t	$3s$

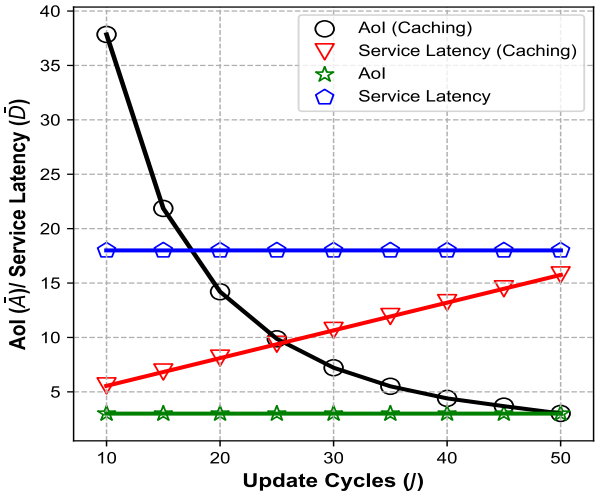

FIGURE 3.3: AoI and Service Latency for the FL schemes.

stipulated J_m in a contract bundle. Moreover, following (3.8) in Section 3.2.2, we use the logarithmic and linear function to model returns to the inverse of AoI and service latency respectively.

To compare the FL schemes, we first set the designed contract aside for further discussion and do not factor in incentive expense for now. We consider a set of update cycles from $J = [10, 50]$, with increments of 5. Note that a higher J implies more frequent data collection and updating.

Fig. 3.3 illustrates the AoI and service latency of the two schemes across a range of J values. Note that the AoI and service latency of the conventional FL scheme is independent of the update cycles given that the data are only updated when a request arrives. In contrast, more frequent updates lead to lower AoI and higher service latency for the caching scheme.

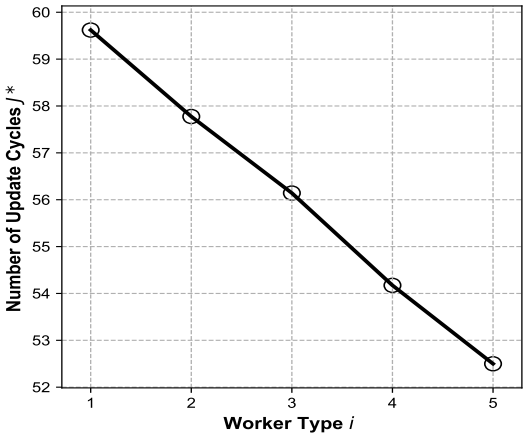

FIGURE 3.4: Number of update cycles vs. worker types.

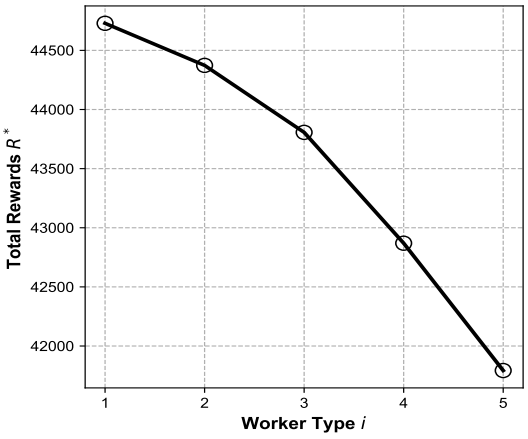

FIGURE 3.5: Total rewards vs. worker types.

In particular, we observe that the AoI of the caching scheme approaches that of the conventional scheme when J is large enough, at the expense of incurring greater service latency. Similarly, the service latency is lower when updates are less frequent, at the expense of having a higher AoI. The FL with caching enables the advantage of flexibility in managing the tradeoffs between service latency and AoI. For example, the delay-sensitive tasks with less consideration for information freshness can be performed with fewer update cycles, whereas tasks that require information freshness can be performed with more frequent updating.

3.4.1 Feasibility of the Contract

3.4.2 Comparison Between the FL Schemes

To study the contract feasibility, we set $w_a = w_d = 0.5$, i.e., both AoI and service latency are of equal importance to the model owner.

The simulation results in Fig. 3.4 and Fig. 3.5 validate the monotonic condition of the contract. In particular, a contract bundle has higher contractual rewards if and only if it requires the worker to update the data more frequently, which is consistent with Lemma 3.1 (Section 3.3.1). In addition, the higher rewards are distributed to worker types that incur lower marginal cost of data updating which is consistent with Lemma 3.2 (Section 3.3.1).

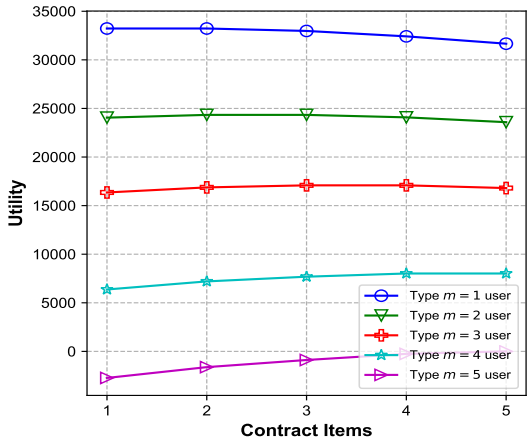

FIGURE 3.6: Utility for each worker type vs. contract items.

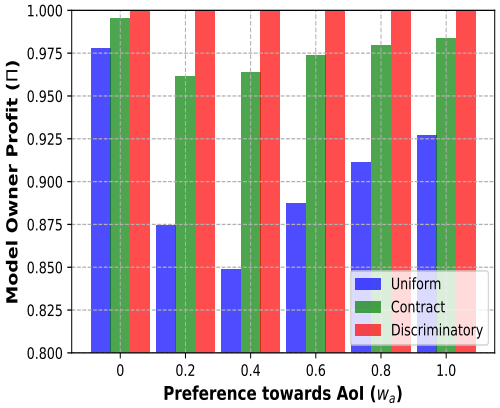

FIGURE 3.7: AoI and Service Latency for different preferences.

The IC of our contract is also demonstrated in Fig. 3.6. In Fig 3.6, the utilities of each worker type are computed with the assumption that it is assigned to each of the contract items $1 - 5$. Clearly, each worker only derives the maximum utility when it is assigned to the contract item designed for its type. For example, type 5 worker, i.e., the minimum utility worker, derives negative utility if it is assigned to the contract items designed for other types. This also validates the IR constraint, i.e., workers have non-negative utilities.

We further compare the proposed incentive scheme with the uniform and discriminatory pricing scheme. In the uniform scheme, workers are offered the same contract bundle regardless of their types, i.e., the contract bundle for the minimum utility worker type to ensure participation across all types. In the discriminatory scheme, we assume a hypothetical situation in which all worker types are known to the model owner, i.e., information asymmetry does not exist. Then, we set the discriminatory scheme as the benchmark to compare the model owner profits among the three schemes in Fig. 3.7. We observe that our proposed contract design allows a model owner to derive greater profits as compared to the uniform scheme, given that the self-revealing mechanism of our incentive scheme distinguishes between the worker types. In addition, the contract scheme is able to derive profits that are close to the perfect information case, thus validating that the adverse effects of information asymmetry is reduced.

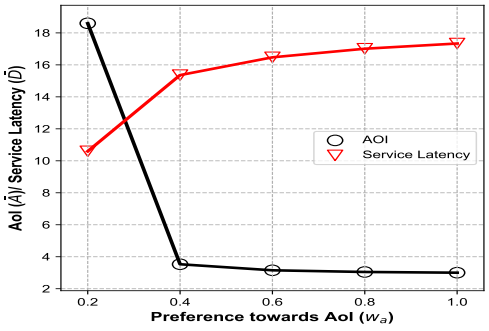

FIGURE 3.8: AoI and Service Latency for different preferences.

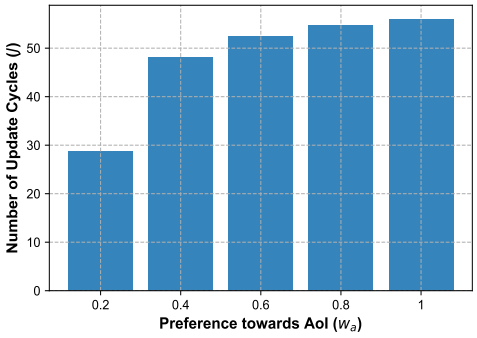

FIGURE 3.9: Number of update cycles for different preferences.

3.4.3 Managing the AoI-Service Latency Tradeoff

In practice, a model owner may have different preferences for varying tasks. We vary the weights w_a and w_d within the range $[0.2, 1]$ to study the changes in AoI, service latency, and update cycles when the model owner preferences vary.

In Fig. 3.8, the AoI and service latency across varying preferences towards the AoI is plotted. For example, the first AoI value represents the AoI when $w_a = 0.2$ and $w_d = 0.8$. Similarly, the first value of service latency corresponds to the aforementioned weight values. Intuitively, a model owner that does not value information freshness but values that the request is met with lower latency will have a high AoI and low service latency. Fig. 3.9 depicts the changes in the number of update cycles as the preference towards AoI varies. As expected, when the preference towards AoI is high, e.g., $w_a = 1$, the number of update cycles is the highest and the corresponding AoI is close to t.

Fig. 3.8 and Fig. 3.9 show the flexibility of our FL scheme. The model owner is able to cater towards different requirements for different tasks, through varying the number of update cycles accordingly for profit maximization. Moreover, with the contract-theoretic incentive mechanism design we discuss in Section 3.4.1, the contract bundles can be appropriately calibrated to achieve the desirable outcomes.

3.5 Conclusion and Chapter Discussion

In this chapter, we have proposed the FL with caching and studied its tradeoff between AoI and service latency. We have also designed the contract-theoretic incentive mechanism for both the discrete and continuous workers. Performance evaluation has shown the IC and the flexibility towards varying AoI and service latency requirements of our incentive scheme.

As we have observed in this chapter, the methods to derive the optimal solution are not complex after the reduction of the IC and IR constraints. Thereafter, the contract optimization can be solved using optimization tools, e.g., cvxpy [146]. As such, the execution of the contract theoretic incentive mechanism is scalable. However, there exists drawbacks in our approach that we will discuss as follows.

Firstly, the single dimensional contract formulation we have assumed in this chapter implies an over-simplification of the utility and cost function of the workers. In this chapter, we only consider the update cost per iteration of workers. However, we have not considered the computation or communication overheads incurred by workers, which can be significant for some complex AI models. An iterative upload of parameters can also be costly for workers in rural areas. As such, we address this in Chapter 4 through a *multi-dimensional* contract, to account for various sources of heterogeneity in the network.

Secondly, we have assumed that the worker types are static and non-changing throughout the entire process. In practice, a worker's cost may vary from one period to another. In this case, the worker may belong to one type in a period, and have type that varies in the next period. The choice of the worker's contract is therefore, no longer one that maximizes its utility. However, in this chapter, we have assumed that the type is constant, and the choice of the contract remains the same throughout. To address this problem, we propose the dynamic contract in our work [147]. Specifically, each worker type, e.g., update cost, is allowed to vary from one time period to another. Then, a single contract is negotiated for the entire period, while taking into account the time correlation of worker types. On one hand, the worker is able to choose a contract while taking into account its expected dynamic type. On the other hand, no extra communication overhead is incurred since the contract can be negotiated right at the beginning and lasts for

the entire duration. Moreover, the *intertemporal* IC constraints are not violated, since the dynamic types are taken into account.

Thirdly, we have assumed that the model owner profit follows the logarithmic function with respect to service latency and AoI. Our assumption follows that of some works on FL, e.g., [126], and AoI, e.g., [140]. The concavity of the model owner utility that we have assumed is reasonable. Besides, it is differentiable and continuous, thus contributing to the tractability we observe in this chapter. However, the profit function may not be easily derived for model owners in reality. The reason is that the influence of data quality/information freshness on model accuracy is not obvious. As neural networks become over-parametrized, the explainable aspects have deteriorated. It is not easy to quantify how much influence data (and its corresponding AoI) has on model accuracy. As such, our future works may consider an approximation of the model owner profit function when a closed form function is not available.

Finally, in this chapter, we have assumed that the model owner is the sole model owner in the FL network. In particular, the FL network we consider is assumed to only consist of multiple individual workers collaborating with a sole FL server. Following our discussion in Chapter 2, there can be exceptions to this setting as follows: (a) the workers may be competing data owners who are reluctant to share their model parameters since the competitors also benefit from a trained global model and (b) the FL servers may compete with other FL servers, i.e., model owners. The FL servers may therefore restrict the participation of workers on more than one FL training. In this case, the formulation of the incentive mechanism design will be vastly different from that proposed. In Chapter 5, we will revisit this notion and devise a deep learning based auction mechanism in response, to model the competition among multiple model owners for the restricted participation of model owners.

Chapter 4

Multi-Dimensional Contract Matching Design for Federated Learning in UAV Networks

4.1 Introduction

Following the advancements in the Internet of Things (IoT) and edge computing paradigm, traditional Vehicular Ad-Hoc Networks (VANETs) that focus mainly on Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communications [148, 149] are gradually evolving into the Internet of Vehicles (IoV) paradigm [150, 151].

The IoV is an open and integrated network system which leverages the enhanced sensing, communication, and computation capabilities of its component data sources, e.g., vehicular sensors, IoT devices, and Roadside Units (RSUs) [152], to build data-driven applications for Intelligent Transport Systems, e.g., for traffic prediction [153], traffic management [154], route planning [155], and other smart city applications [156]. Coupled with the rise of Deep Learning, the wealth of data and enhanced computation capabilities of IoV components enable effective Artificial Intelligence (AI) based models to be built.

Beyond ground data sources, aerial platforms are increasingly important today given that modern day traffic networks have grown in complexity. In particular, Unmanned Aerial Vehicles (UAVs) are commonly used today to provide data collection and computation offloading support in the IoV paradigm. The UAVs feature the benefits of high mobility, flexible deployment, cost effectiveness [157], and can also provide more comprehensive coverage as compared to ground users. UAVs can be deployed, e.g., to capture images of car parks for the management and analysis of parking occupancy [158], to capture images of roads and highways for traffic monitoring applications [159–161], and also to aggregate data from stationary vehicles and roadside units that in turn collect data of other passing vehicles periodically [162]. Apart from data collection, the UAVs have also been used to provide computation offloading support for resource constrained IoV components [163, 164].

As such, studies proposing the Internet of Drones (IoD) and Drones-as-a-Service (DaaS) [165–167] have gained traction recently. Moreover, the DaaS industry is a rapidly growing one [168] that comprises independent drone owners that provide on-demand data collection and model training for businesses and city planners.

Naturally, to build a better inference model, the independently owned UAV companies can collaborate by sharing their data collected from various sources, e.g., carparks, RSUs, and highways, for collaborative model training. However, in recent years, the regulations governing data privacy, e.g., GDPR, are increasingly stringent. As such, this can potentially prevent the sharing of data across DaaS providers. To this end, we propose the adoption of FL based [3] approach to enable privacy-preserving collaborative ML across a federation of independent DaaS providers.

In our system model (Fig. 4.1), a client, hereinafter model owner, is interested in collecting data from a region for model training, e.g., for traffic prediction. Given the energy constraints of UAVs [169], the region is further divided into smaller subregions. The model owner then announces an FL task, e.g., the capturing of real-time traffic flow over highways or the collection of data from RSUs for model training [161]. Then, only the DaaS providers, hereinafter UAVs, that are able to complete the task within the stipulated time and energy constraints respond to the model owner. Thereafter, the model owner assigns an optimal UAV to each subregion. After the UAV collects the sensing data, model training takes place

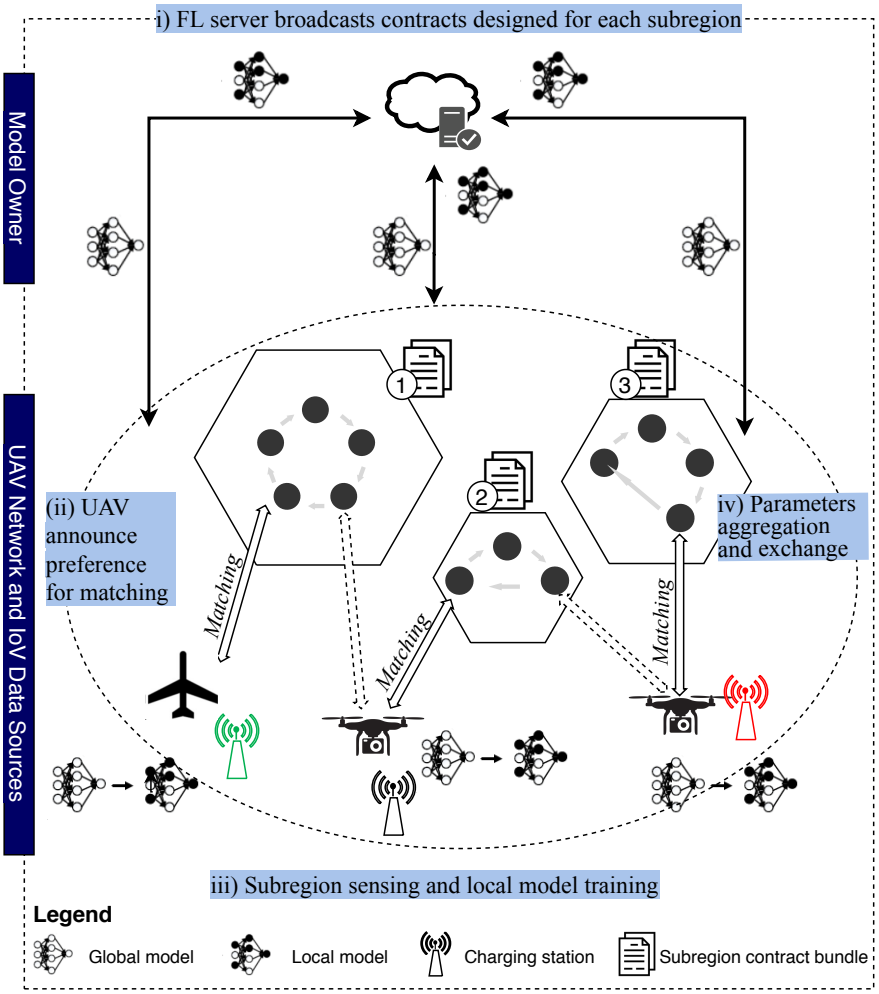

FIGURE 4.1: System model involving UAV-subregion contract-matching.

on each UAV separately, following which only the updated model parameters are transmitted to the model owner for global aggregation.

Our proposed approach has three advantages. Firstly, the resource constrained IoV components are aided by the UAV deployment for completion of time sensitive sensing and model training tasks. Secondly, it preserves the privacy of the UAV-collected data by eliminating the need for data sharing across UAVs. Thirdly, it is communication efficient. The reason is that traditional methods of data sharing will require the raw data to be uploaded to an aggregating cloud server. With FL, only the model parameters need to be transmitted by the UAVs.

However, there exists an incentive mismatch between the model owner and the UAVs. On one hand, the model owners aim to maximize their profits by selecting the optimal UAVs which can complete the stipulated task at the lowest cost, e.g., in terms of sensing, transmission, and computation costs. On the other hand, the UAVs can take advantage of the information asymmetry and misreport their types so as to seek higher compensation. To that end, we leverage the self-revealing properties of contract theory [20] as an incentive mechanism design to appropriately reward the UAVs based on their actual types. In particular, different from existing works and Chapter 3, given the complexity of the sensing and collaborative learning task, we consider a multi-dimensional contract to account for the multi-dimensional sources of heterogeneity in terms of UAV sensing, learning, and transmission capabilities.

After deriving optimal contracts to which the UAVs respond , the possibility that multiple UAVs prefer a particular subregion still remains. To that end, we leverage the Gale-Shapley (GS) [170] matching-based algorithm to assign the optimal UAVs to each subregion.

The contribution of this chapter is as follows:

i We propose an FL based sensing and collaborative learning scheme in which UAVs collect the data and participate in privacy-preserving collaborative model training for applications in the IoV paradigm towards the development of an Intelligent Transport System.

ii In consideration of the incentive mismatches and information asymmetry between the UAVs and model owner, we propose a multi-dimensional contract-matching based incentive mechanism design that aims to leverage on the self-revealing properties of an optimal contract, such that the most optimal UAV can be matched to a subregion.

iii Our incentive mechanism design considers a general UAV sensing, computation, and transmission model, and thus can be extended to specific FL based applications in the IoV paradigm.

Due to the heavy notations used in this chapter, we provide a list of common notations used in this chapter in Table 4.1. The organization of this chapter is as

TABLE 4.1: Table of commonly used notations.

Notation	Description
n	Subregion
j	UAV
C_j	Base of UAV j
l_j^n	Total sensing distance
$l_{C_j}^n$	Total traversal distance
$\tau_P^{j,n}$	Total duration taken for traversal and sensing
$E_P^{j,n}$	Total energy taken for traversal and sensing
α_j^n	Marginal cost of node coverage for sensing
ψ_j^n	Traversal cost
$\tau_C^{j,n}$	Local computation duration
$E_C^{j,n}$	Total energy taken for computation
β_j	Marginal cost of node coverage for computation
$\tau_T^{j,n}$	Total duration for transmission
ζ_j^n	Energy taken for transmission
u_j^n	UAV utility
R_j^n	Contractual rewards
ϕ	Unit cost of energy for the UAV
Π	Model owner profit
Ω^n, ω^n	Contract set and individual contract
$\tilde{R}$	Compensation for sensing and computation costs
$\hat{R}$	Compensation for traversal and transmission costs
$\upsilon(\alpha_y, \beta_z)$	Marginal cost of node coverage
Φ_i	UAV auxiliary type

follows. Section 4.2 introduces the system model and problem formulation, Section 4.3 discusses the multi-dimensional contract formulation, Section 4.4 considers a matching-based UAV-subregion assignment, Section 4.5 presents the performance evaluation of our proposed incentive mechanism design, and Section 4.6 concludes.

4.2 System Model and Problem Formulation

We consider a network in which a model owner aims to collect data from stipulated nodes, e.g., from RSUs or images of segments in the highway, in a target sensing region to fulfill a time-sensitive task. One UAV is selected by the task publisher to cover each of the subregions. Given information asymmetry and the multiple sources of heterogeneity in UAV cost types, the model owner leverages the self-revealing properties of a multi-dimensional contract theoretic approach to choose

one UAV suited to cover each of the subregion. After data collection, the UAV returns to their respective UAV bases for FL based model training.

Following [171], the target sensing region can be modeled as a graph and divided into N smaller graphs, i.e., subregions whose set is denoted $\mathcal{N} = \{1, \ldots, n, \ldots, N\}$, e.g., through the multilevel graph partition algorithm [172]. The set of nodes in subregion n is denoted $\mathcal{I}_n = \{I_1, \ldots, I_n, \ldots, I_N\}$ with the node i in subregion n located at $\boldsymbol{x}_i^n \in \mathbb{R}^3$. The Euclidean distance between two nodes i and i' located within subregion n, $\forall i, i' \in \mathcal{I}_n, i \neq i'$ is expressed as $l_{i,i'}^n$ where $l_{i,i'}^n = ||\boldsymbol{x}_i^n - \boldsymbol{x}_{i'}^n|| < \infty$, i.e., all nodes are inter-accessible.

A set $\mathcal{J} = \{1, \ldots, j, \ldots, J\}$ of J unmanned aerial vehicles (UAVs) are located at bases situated around the target sensing region. Without loss of generality, we assume that each base owns a single UAV and $J \geq N$. Moreover, our model can be easily extended to scenarios in which a UAV swarm is required for sensing in each subregion. Denote $C_j = \{C_1, \ldots, C_j, \ldots, C_J\}$ as the set of bases where C_j refers to the base of UAV j located at $\boldsymbol{y}_{C_j} \in \mathbb{R}^3$. The Euclidean distance between the base of UAV j and subregion n is expressed as $l_{C_j}^n$, where $l_{C_j}^n = ||\boldsymbol{y}_{C_j} - \boldsymbol{x}_{\bar{i}}^n|| < \infty$ and $\bar{i}$ denotes the centre of the subregion.

There are two stages in our system model as follows:

i **Multi-Dimensional Contract Design:** The UAV types, e.g., sensing, traversal, and transmission costs, are private information not known to the model owner. As such, the model owner designs a multi-dimensional contract to leverage the self-revealing mechanism so as to select the optimal UAV to cover each subregion. In particular, the model owner can maximize its profits by choosing the lowest cost UAV among all feasible UAVs that can complete the task within the time constraint.

ii **UAV-Subregion Assignment:** Each UAV reports its type and ranks the N subregions based on its preferences of coverage to the FL model owner. Then, a stable UAV-subregion matching is derived using the Gale-Shapley (GS) algorithm. Note that each UAV's preference can vary across different subregions. As an illustration, we consider a representative UAV j at base C_j and two subregions n and n' where $l_{C_j}^{n'} \gg l_{C_j}^n, \forall i_n \in I_n$ and $\forall i_{n'} \in I_{n'}$. In this case, UAV j has to traverse a longer distance to reach subregion n'. As

such, it is able to cover a smaller proportion of subregion n' relative to n due to energy and time constraints.

In the following, we consider the sensing, computation, and data transmission model of a representative UAV.

4.2.1 UAV Sensing Model

We consider a representative UAV j tasked by the model owner to cover a proportion of nodes in the subregion n. Denote the node coverage assignment of UAV j in subregion n to be $\mathcal{A}^{j,n} = \left\{a_{i,i'}^{j,n} | \forall i, i' \in \mathcal{I}_n, i \neq i'\right\}$ where $a_{i,i'}^{j,n} = 1$ represents that the UAV has to fly through the segment between nodes i and i', and $a_{i,i'}^{j,n} = 0$ implies otherwise.

The total distance l_j^n traveled for sensing by UAV j under assignment $\mathcal{A}^{j,n}$ is as follows:

$$l_j^n = \sum_{i' \neq i, i' \in \mathcal{I}_n} a_{i,i'}^{j,n} l_{i,i'}^n. \tag{4.1}$$

Denote $\theta_j^n = \frac{\sum_{i' \neq i, i' \in \mathcal{I}_n} a_{i,i'}^{j,n}}{|\mathcal{I}_n|}$ where $|\cdot|$ indicates cardinality, i.e., θ_j^n refers to the proportion of node coverage by UAV j in subregion n where $0 \leq \theta_j^n \leq 1$.

Apart from traveling between the nodes, the UAV has to travel to and from its base. Denote the total distance traveled by the UAV as $L_j^n = l_j^n + l_{C_j}^n$. Hereinafter, we refer to l_j^n as the *sensing* distance, whereas $l_{C_j}^n$ refers to the *traversal* distance.

Following the works of [173, 174], each UAV travels with an average velocity v_j and expends a fixed propulsion power $p_j = c_{j,1} v_j^3 + \frac{c_{j,2}}{v_j}$ throughout the task for tractability, where $c_{j,1}$ and $c_{j,2}$ refers to the required power to balance the parasitic drag caused by skin friction and required power to balance the drag force of air redirection respectively[1]. Note that the propulsion power consumed by the UAV when it changes its direction is negligible [169]. The total duration taken for traversal and sensing is denoted $\tau_P^{j,n} = \frac{L_j^n}{v_j}$, whereas the total energy consumed to cover the traversal and sensing distance is as follows:

[1]In practice, the propulsion power is in turn a function of other factors, e.g., reference area of the UAV and wing aspect ratio and weight. For simplicity, we consider that p_j accounts for these factors.

$$
\begin{aligned}
E_P^{j,n} = \frac{L_j^n}{v_j} p_j &= \frac{\theta_j^n l^n + l_{C_j}^n}{v_j} p_j \\
&= \frac{p_j l^n}{v_j} \theta_j^n + \frac{l_{C_j}^n}{v_j} p_j \\
&= \alpha_j^n \theta_j^n + \psi_j^n,
\end{aligned} \tag{4.2}
$$

where l^n is the distance traveled by the UAV if it covers all nodes, i.e., $\theta_j^n = 1$, $\alpha_j^n = \frac{p_j l^n}{v_j}$ and $\psi_j^n = \frac{l_{C_j}^n}{v_j} p_j$ for notation simplicity. Note that α_j^n represents the sensing cost, i.e., marginal cost of node coverage for sensing in the subregion, whereas ψ_j^n refers to the traversal cost, i.e., the energy cost of traveling to and from the base. A higher α_j^n can imply that the UAV j requires greater propulsion power to complete the task, e.g., due to its larger weight or wing-aspect ratio, whereas a higher ψ_j^n implies either a greater propulsion power to move, or a greater traversal cost, i.e., the subregion is farther away from the base. While the value of α_j^n varies across subregions due to the varying l_n, i.e., the marginal cost of node coverage varies according to the sensing area of the subregion, the *ordering* of the UAV types based on the sensing costs is retained. On the other hand, the order of UAVs by traversal costs varies across subregions, based on the distance between the UAV base and each of the subregions.

4.2.2 UAV Computation Model

After the UAV j covers its assigned set of nodes following assignment $\mathcal{A}^{j,n}$, it returns to the base C_j for an FL based model training over K global iterations where $\mathcal{K} = \{1, \ldots, k, \ldots, K\}$ to minimize the global loss $F^K(\boldsymbol{w})$. Following [78], each training iteration k consists of three steps namely: (i) *Local Computation*, i.e., the UAV trains the received global model $\boldsymbol{w}^{(k)}$ locally using the sensing data[2], (ii) *Wireless Transmission*, i.e., the UAV transmits the model parameter update $\boldsymbol{h}_j^{(k)}$ to the model owner, and (iii) *Global Model Parameter Update*, i.e., all parameter updates derived from the N subregions are aggregated to derive an updated global model $\boldsymbol{w}^{(k+1)}$, where $\boldsymbol{w}^{(k+1)} = \cup_{j \in \mathcal{N}} (\boldsymbol{w}_j^{(k)} + \boldsymbol{h}_j^{(k)})$, which is then transmitted back to the UAVs for the $(k+1)^{th}$ training iteration.

[2]In practice, note that the model parameters received may be initialized and trained with a generic dataset off-line in the cloud first. This set of model parameters are updated afterwards during the FL process.

In general, a series of local model training is performed by the UAV to minimize an L-Lipschitz and γ-strongly convex local loss function G_j up to the target accuracy A^* defined by the model owner to derive the parameter update. Note that a larger value of A^* implies a greater deviation from the optimal value. Moreover, $0 < A^* < 1$, i.e., the local solution $\boldsymbol{h}_j^{(k)}$ does not have to be trained to optimality, e.g., to reduce local computation duration especially for time sensitive tasks. In particular, following the formulation in [131]:

$$\begin{aligned} &G_j\left(\boldsymbol{w}^{(k)}, \boldsymbol{h}_j^{(k)}\right) - G_j\left(\boldsymbol{w}^{(k)}, \boldsymbol{h}_j^{(k)*}\right) \\ &\quad \leq A^*\left(G_j\left(\boldsymbol{w}^{(k)}, \mathbf{0}\right) - G_j\left(\boldsymbol{w}^{(k)}, \boldsymbol{h}_j^{(k)*}\right)\right). \end{aligned} \tag{4.3}$$

The FL training is completed after $K = \frac{a}{1-A^*}$ global iterations where $a = \frac{2L^2}{\gamma^2 \xi}$ and $0 \leq \xi \leq \frac{\gamma}{L}$. The total local computation duration $\tau_C^{j,n}$ is as follows:

$$\tau_C^{j,n} = K\left(\frac{V C_j \theta_j^n D^n \log_2(1/A^*)}{f_j}\right), \tag{4.4}$$

whereas the energy consumption of UAV j for computation is as follows:

$$E_C^{j,n} = K\left(\kappa C_j \theta^{j,n} D^n V \log_2(1/A^*) f_j^2\right) = \beta_j \theta_j^n. \tag{4.5}$$

Note that κ is the effective switched capacitance that depends on the chip architecture [175], C_j is the number of cycles per bit for computing one sample data of UAV j, $\theta^{j,n} D^n$ is the unit of data samples collected by UAV j, $V \log_2(1/A^*)$ refers to the lower bound on the number of local iterations required to achieve local accuracy A^* [131] where $V = \frac{2}{(2-L\delta)\delta\gamma}$, and f_j refers to the computation capacity of the UAV j, measured by CPU cycles per second. For ease of notation, we denote $\beta_j = \kappa K C_j D^n V \log_2(1/A^*) f_j^2$, i.e., a higher β_j implies greater energy cost for computation per additional node coverage. Similar to α_j^n, the value of β_j^n varies across subregions due to the different units of data samples available for computation. However, the ordering of the UAV types based on computation costs is retained.

4.2.3 UAV Transmission Model

After local computation, the wireless transmission takes place from the selected UAVs to the model owner. For simplicity, we denote the achievable rate of the

UAV j to be a product of its transmit power ρ_j and a scaling factor λ_j^n that covers other considerations, e.g., bandwidth allocation and channel gain.

The total time $\tau_T^{j,n}$ taken by the UAV to upload parameter update $\boldsymbol{h}_j^{(k)}$ of size H is as follows: $\tau_T^{j,n} = K\frac{H}{\lambda_j^n \rho_j}$. Note that the model upload size is constant regardless of the number of global iterations or quantity of data collected, given the fixed dimensions of the model update. The transmission energy consumption, denoted ζ_j^n, is as follows:

$$E_T^{j,n} = \tau_T^{j,n} \rho_j = \zeta_j^n. \tag{4.6}$$

4.2.4 UAV and Model Owner Utility Modeling

The utility function of a representative UAV j covering subregion n can be expressed as follows:

$$\begin{aligned} u_j^n(\theta_j^n) &= R_j^n(\theta_j^n) - \phi\left(E_P^{j,n} + E_C^{j,n} + E_T^{j,n}\right) \\ &= R_j^n(\theta_j^n) - \phi\left(\alpha_j^n \theta_j^n + \psi_j^n + \beta_j^n \theta_j^n + \zeta_j^n\right), \end{aligned} \tag{4.7}$$

where $R_j^n(\theta_j^n)$ refers to the contractual rewards and ϕ refers to the unit cost of energy.

Following [125, 126, 176], the FL model accuracy $\Upsilon(\sum_{n=1}^{N} \theta_{j*}^n D^n)$ is a concave function of the aggregate data collected across N subregions by the N selected UAVs. In particular, the inference accuracy of the model is improved when more nodes are covered, i.e., a model trained using data across a more comprehensive coverage of classes may be built. Without loss of generality, we consider the aggregate model performance to be an average of node coverage across all regions, analogous to the Federated Averaging algorithm:

$$\Upsilon\left(\sum_{n=1}^{N} \theta_{j*}^n D^n\right) = \frac{1}{N}\sum_{n=1}^{N} log(1 + \mu \theta_{j*}^n D^n), \tag{4.8}$$

where $\mu > 0$ is the system parameter. The total profit obtained from all UAVs is thus as follows:

$$\Pi(\Omega) = \sigma\Upsilon\left(\sum_{n=1}^{N}\theta_{j^*}^{n}D^{n}\right) - \sum_{n=1}^{N}R_{j*}^{n} \tag{4.9}$$

where $\sigma > 0$ refers to the conversion parameter from model performance to profits, and the contractual reward expense for each selected UAV is denoted as $R_{j^*}^{n}$. In the next section, we devise the optimal contract which satisfies the Individual Rationality and Incentive Compatibility constraints.

4.3 Multi-Dimensional Contract Design

In this section, we first consider a multi-dimensional contract formulation. To solve the multi-dimensional contract, we sort the UAV types according to an auxiliary variable that reflects the marginal cost of node coverage. Then, we relax the constraints for contract feasibility and include a fixed compensation component for traversal and transmission costs so as to solve for the optimal contract.

4.3.1 Contract Condition Analysis

Given that the sensing cost α, traversal cost ψ, computation cost β, and transmission cost ζ are all private information that are not precisely known by the model owner, we consider the multi-dimensional contract theoretic incentive mechanism design to leverage on its self-revealing properties. The reason is that without the self-revealing properties, the UAV operators may misreport their costs in order to seek overly high compensations.

The UAVs can be classified into different types to characterize their heterogeneity. In particular, the UAVs can be categorized into a set $\Psi = \{\psi_x^n : 1 \leq x \leq X\}$ of X traversal cost types, set $\mathcal{A} = \{\alpha_y^n : 1 \leq y \leq Y\}$ of Y sensing cost types, set $\mathcal{B} = \{\beta_z^n : 1 \leq z \leq Z\}$ of Z computation cost types, and set $\mathcal{C} = \{\zeta_q^n : 1 \leq q \leq Q\}$ of Q transmission cost types.

Without loss of generality, we also assume that the user types are indexed in non-decreasing orders in all four dimensions: $0 < \psi_1 \leq \psi_2 \leq \cdots \leq \psi_X$, $0 < \alpha_1^n \leq$

$\alpha_2^n \leq \cdots \leq \alpha_Y^n$, $0 < \beta_1^n \leq \beta_2^n \leq \cdots \leq \beta_Z^n$, and $0 < \zeta_1^n \leq \zeta_2^n \leq \cdots \leq \zeta_Q^n$. For ease of notation, we represent a UAV of traversal cost type x, sensing cost type y, computation cost type z, and transmission cost type q to be that of type-(x, y, z, q).

To enforce the UAVs to truthfully reveal their private information, we adopt a two-step procedure for the contract design:

i *Multi-Dimensional Contract Design:* We convert the multi-dimensional problem into a single-dimensional contract formulation following the approach in [177]. In particular, we sort the UAVs by an auxiliary, one-dimensional type $\Phi(\alpha_y^n, \beta_z^n)$ in the ascending order based on the marginal cost of node coverage, i.e., sensing and computation cost types. Then, we solve for the optimal contract for each subregion n denoted $\Omega^n(\mathcal{A}, \mathcal{B}) = \{\omega_{y,z}^n : 1 \leq y \leq Y, 1 \leq z \leq Z\}$ where $n \in \mathcal{N}$ to derive the optimal node coverage-contract reward bundle $\{\theta_{y,z}^n, \tilde{R}_{y,z}^n\}$.

ii *Traversal Cost Compensation:* In contrast to existing works on multi- dimensional contracts, the UAVs also incur the additional traversal cost and transmission cost components, both of which are not coupled with the marginal cost of node coverage. In other words, these costs have to be incurred regardless of the number of nodes a UAV decides to cover in the subregion. For each contractual reward, we add in a fixed compensation $\hat{R}$ to derive the final contract bundle $\{\theta_{y,z}^n, (\tilde{R}_{y,z}^n + \hat{R})\}$.

We first discuss the multi-dimensional contract formulation as follows. A contract is feasible only if the Individual Rationality (IR) and Incentive Compatibility (IC) constraints hold simultaneously.

Definition 4.1. Individual Rationality (IR): Each type-(y, z) UAV achieves non-negative utility if it chooses the contract item designed for its type, i.e., contract item $\omega_{y,z}$.

$$u_{y,z}(\omega_{y,z}) \geq 0, 1 \leq y \leq Y, 1 \leq z \leq Z. \tag{4.10}$$

Definition 4.2. Incentive Compatibility (IC): Each type-(y, z) UAV achieves the maximum utility if it chooses the contract item designed for its type, i.e., contract item $\omega_{y,z}$. As such, it has no incentive to choose contracts designed for other types.

$$\begin{aligned} u_{y,z}(\omega_{y,z}) \geq u_{,y,z}(\omega_{y',z'}), 1 \leq y \leq Y, 1 \leq z \leq Z, \\ y \neq y', z \neq z'. \end{aligned} \tag{4.11}$$

The multi-dimensional contract formulation is as follows:

$$\begin{aligned} &\max_{\Omega} \Pi(\Omega^n(\mathcal{A}, \mathcal{B})) \\ &\text{s.t. } (4.10), (4.11). \end{aligned} \tag{4.12}$$

However, the optimization problem in (4.12) involves YZ, i.e., IR constraints and $YZ(YZ-1)$, i.e., IC constraints, all of which are non-convex. Therefore, we first convert the contract into a single-dimensional formulation in the next section.

4.3.2 Conversion Into A Single-Dimensional Contract

In order to account for the marginal cost of node coverage, we consider a revised utility $\tilde{u}_{y,z}$ of the UAV type-(y,z) that excludes the traversal and transmission costs as follows:

$$\tilde{u}_{y,z}(\theta(y,z), \tilde{R}_{y,z}) = \eta(\alpha_y, \beta_z) + \tilde{R}_{y,z}, \tag{4.13}$$

where we denote $\eta(\alpha_y, \beta_z) = -\phi(\alpha_y \theta(y,z) + \beta_z \theta(y,z))$ for ease of notation, and $\tilde{R}_{y,z}$ refers to the contractual reward arising from the multi-dimensional contract design. To focus on a representative contract, we drop the n superscripts for now. Given that the ranking of marginal cost types does not change across subregion, note that our contract design is a general one applicable to all subregions.

We derive the marginal cost of node coverage $\upsilon(\alpha_y, \beta_z)$ for the type-(y,z) UAV as follows:

$$\upsilon(\alpha_y, \beta_z) = -\frac{\partial \eta(\alpha_y, \beta_z)}{\partial \theta(y,z)} = \phi(\alpha_y + \beta_z). \tag{4.14}$$

Intuitively, $\frac{\partial \eta(\alpha_y, \beta_z)}{\partial \theta(y,z)} < 0$ since the coverage of an additional node results in the additional expenses of sensing and computation costs. A larger value of $\upsilon(\alpha_y, \beta_z)$ implies a larger marginal cost of node coverage, due to the greater sensing and computation costs incurred for a particular UAV type.

We can now sort the YZ UAVs according to their marginal cost of node coverage in a non-decreasing order as follows:

$$\Phi_1(\theta), \Phi_2(\theta), \ldots, \Phi_i(\theta), \ldots, \Phi_{YZ}(\theta), \tag{4.15}$$

where $\Phi_i(\theta)$ denotes the auxiliary type-$\Phi_i(\theta)$ user. Given the sorting order, the UAV types are in an ascending order based on their marginal cost of node coverage:

$$\upsilon(\theta, \Phi_1) \leq \upsilon(\theta, \Phi_2) \leq \cdots \leq \upsilon(\theta, \Phi_i) \leq \cdots \leq \upsilon(\theta, \Phi_{YZ}), \tag{4.16}$$

Note that for ease of notation, we use type-Φ_i or type-i interchangeably to represent the auxiliary type-i user. In addition, we refer to $\eta_i(\theta_i)$ and $\eta(\theta_i, \Phi_i)$ interchangeably to represent the new ordering subsequently. Similarly, to represent the marginal cost of node coverage, we use $\upsilon_i(\theta_i)$ and $\upsilon(\theta_i, \Phi_i)$. In the next section, we derive the necessary and sufficient conditions for the contract design.

4.3.3 Conditions For Contract Feasibility

We derive the necessary conditions to guarantee contract feasibility based on the IR and IC constraints as follows.

Lemma 4.1. For any feasible contract $\Omega\{\mathcal{A}, \mathcal{B}\}$, we have $\theta_i < \theta_{i'}$ if and only if $\tilde{R}_i < \tilde{R}'_i$, $i \neq i'$.

Proof. We first prove the sufficiency, i.e., if $R_i < R'_i \Rightarrow \theta_i < \theta_{i'}$. From the IC constraint of type-Φ_i UAV we have:

$$\eta(\theta_i, \Phi_i) + \tilde{R}_i \geq \eta(\theta_i, \Phi_{i'}) + \tilde{R}_{i'},$$
$$\eta(\theta_i, \Phi_i) - \eta(\theta_i, \Phi_{i'}) \geq \tilde{R}_{i'} - \tilde{R}_i > 0,$$

which implies:

$$\eta(\theta_i, \Phi_i) \geq \eta(\theta_i, \Phi_{i'}),$$

Given that $\frac{\partial \eta(\theta_i, \Phi_i)}{\partial \theta_i} < 0$, we can deduce $\theta_i < \theta_{i'}$.

Next, we prove the necessity, i.e., $\theta_i < \theta_{i'} \Rightarrow \tilde{R}_i < \tilde{R}'_i$. Similarly, we consider the IC constraint of the type-Φ_i UAV:

$$\eta(\theta_i, \Phi_i) + \tilde{R}_i \geq \eta(\theta_{i'}, \Phi_i) + \tilde{R}_{i'},$$
$$\eta(\theta_i, \Phi_i) - \eta(\theta_{i'}, \Phi_i) \geq \tilde{R}_{i'} - \tilde{R}_i.$$

Given $\theta_i < \theta_{i'}$, we deduce $\eta(\theta_i, \Phi_i) < \eta(\theta_{i'}, \Phi_i)$, which follows that $\tilde{R}_{i'} < \tilde{R}_i$. The proof is now completed. □

Lemma 4.2. Monotonicity: For any feasible contract $\Omega\{\mathcal{A}, \mathcal{B}\}$, if $\upsilon(\theta_i, \Phi_i) > \upsilon(\theta_i, \Phi_{i'})$, it follows that $\theta_i \leq \theta_{i'}$.

Proof. We adopt the proof by contradiction to validate the monotonicity condition. We first assume that there exists $\theta_i > \theta_{i'}$ such that $\upsilon(\theta_i, \Phi_i) > \upsilon(\theta_i, \Phi_{i'})$, i.e., the lemma is incorrect.

We consider the IC constraints for the type Φ_i and $\Phi_{i'}$ UAV:

$$\eta(\theta_i, \Phi_i) + \tilde{R}_i \geq \eta(\theta_{i'}, \Phi_i) + \tilde{R}_{i'}, \tag{4.17}$$
$$\eta(\theta_{i'}, \Phi_{i'}) + \tilde{R}_{i'} \geq \eta(\theta_i, \Phi_{i'}) + \tilde{R}_i. \tag{4.18}$$

Then, we add the constraints together and rearrange the terms to obtain:

$$[\eta(\theta_i, \Phi_i) - \eta(\theta_{i'}, \Phi_i)] - [\eta(\theta_i, \Phi_{i'}) - \eta(\theta_{i'}, \Phi_{i'})] \geq 0. \tag{4.19}$$

By the fundamental theorem of calculus, we have:

$$\begin{aligned} &[\eta(\theta_i, \Phi_i) - \eta(\theta_{i'}, \Phi_i)] - [\eta(\theta_i, \Phi_{i'}) - \eta(\theta_{i'}, \Phi_{i'})] \\ &= \int_{\theta_{i'}}^{\theta_i} \frac{\partial \eta(\theta, \Phi_i)}{\partial \theta} d\theta - \int_{\theta_{i'}}^{\theta_i} \frac{\partial \eta(\theta, \Phi_{i'})}{\partial \theta} d\theta \\ &= \int_{\theta_{i'}}^{\theta_i} \left[\frac{\partial \eta(\theta, \Phi_i)}{\partial \theta} - \frac{\partial \eta(\theta, \Phi_{i'})}{\partial \theta} \right] d\theta \\ &= -\int_{\theta_{i'}}^{\theta_i} \left[\upsilon(\theta, \Phi_i) - \upsilon(\theta, \Phi_{i'}) \right] d\theta. \end{aligned} \tag{4.20}$$

Given (4.12) from Section 4.3.1, as well as the assumption $\theta_i > \theta_{i'}$ and $\eta(\theta_i, \Phi_i) > \eta(\theta_i, \Phi_{i'})$, we can deduce that (4.20) is negative, which contradicts with (4.19). As

such, there does not exist $\theta_i > \theta_{i'}$ and $\eta(\theta_i, \Phi_i) > \eta(\theta_i, \Phi_{i'})$ for the feasible contract, which confirms that the lemma is correct. The proof is now completed. □

As such, Lemmas 4.1 and 4.2 give us the necessary conditions of the feasible contract in the following theorem.

Theorem 4.1. A feasible contract must meet the following conditions:

$$\begin{cases} \theta_1 \geq \theta_2 \geq \cdots \geq \theta_i \geq \cdots \geq \theta_{YZ} \\ \tilde{R}_1 \geq \tilde{R}_2 \geq \cdots \geq \tilde{R}_i \geq \cdots \geq \tilde{R}_{YZ} \end{cases} \tag{4.21}$$

Next, we further relax the IR and IC constraints. Due to the independence of Φ_i on the contract item $\{\theta, \tilde{R}\}$, i.e., $\Phi_i(\theta, \tilde{R}) = \Phi_i(\theta', \tilde{R}')$, $\theta \neq \theta', \tilde{R} \neq \tilde{R}'$, the UAV type does not change with the node coverage and contract rewards. In addition, the ordering of the type by marginal costs does not change with the subregion n. As such, we are able to deduce the minimum utility UAV Φ_{max}, i.e., the UAV type that incurs the highest marginal cost of node coverage, as follows:

$$\Phi_{max}(\theta) = \arg\min_{\Phi_i} \tilde{u}\left(\theta, \tilde{R}, \Phi_i\right). \tag{4.22}$$

Intuitively $\Phi_{max} = \Phi_{YZ}$, i.e., the UAV characterized by $\{\alpha_Y, \beta_Z\}$ is the UAV which incurs the highest marginal cost of node coverage, and hence it is the minimum utility UAV.

Lemma 4.3. If the IR constraint of the minimum utility UAV type Φ_{YZ} is satisfied, the other IR constraints will also hold.

Proof. From the IC constraint and the sorting order $\eta_1(\theta_1) \geq \cdots \geq \eta_i(\theta_i) \cdots \geq \eta_{YZ}(\theta_{YZ})$, we have the following relation:

$$\eta_i(\theta_i) + \tilde{R}_i \geq \eta_i(\theta_{YZ}) + \tilde{R}_{YZ} \geq \eta_{YZ}(\theta_{YZ}) + \tilde{R}_{YZ} \geq 0. \tag{4.23}$$

As such, as long as the IR constraint of the UAV type Φ_{YZ} is satisfied, it follows that the IR constraints of the other UAVs will also hold. □

Lemma 4.4. For a feasible contract, if $\omega_{i-1} \overset{PIC}{\Leftrightarrow} \omega_i$ and $\omega_i \overset{PIC}{\Leftrightarrow} \omega_{i+1}$, then $\omega_{i-1} \overset{PIC}{\Leftrightarrow} \omega_{i+1}$.

Note that the relation $\omega_i \overset{PIC}{\Leftrightarrow} \omega_{i'}, i \neq i'$ implies the *Pairwise Incentive Compatibility* (PIC), which is fulfilled under the following condition:

$$\begin{cases} u_i\left(\omega_i\right) \geq u_i\left(\omega_{i'}\right), \\ u_{i'}\left(\omega_{i'}\right) \geq u_{i'}\left(\omega_i\right). \end{cases} \tag{4.24}$$

Proof. Suppose we have three UAV types Φ_{i-1}, Φ_i, and Φ_{i+1} where $i-1 < i < i+1$. The Local Upward Incentive Constraint (LUIC), i.e., IC constraint between the i^{th} and $(i+1)^{th}$ UAV is as follows:

$$\eta\left(\theta_{i-1}, \Phi_{i-1}\right) + \tilde{R}_{i-1} \geq \eta\left(\theta_i, \Phi_{i-1}\right) + \tilde{R}_i \tag{4.25}$$

$$\eta\left(\theta_i, \Phi_i\right) + \tilde{R}_i \geq \eta\left(\theta_{i+1}, \Phi_i\right) + \tilde{R}_{i+1}. \tag{4.26}$$

In addition, we consider:

$$\begin{aligned} &\left[\eta\left(\theta_{i+1}, \Phi_i\right) - \eta\left(\theta_i, \Phi_i\right)\right] - \left[\eta\left(\theta_{i+1}, \Phi_{i-1}\right) - \eta\left(\theta_i, \Phi_{i-1}\right)\right] \\ &= \int_{\theta_i}^{\theta_{i+1}} \frac{\partial \eta\left(\theta, \Phi_i\right)}{\partial \theta} d\theta - \int_{\theta_i}^{\theta_{i+1}} \frac{\partial \eta\left(\theta, \Phi_{i-1}\right)}{\partial \theta} d\theta \\ &= \int_{\theta_i}^{\theta_{i+1}} \frac{\partial \eta\left(\theta, \Phi_i\right)}{\partial \theta} - \frac{\partial \eta\left(\theta, \Phi_{i-1}\right)}{\partial \theta} d\theta \\ &= -\int_{\theta_i}^{\theta_{i+1}} \left[\upsilon\left(\theta, \Phi_i\right) - \upsilon\left(\theta, \Phi_{i-1}\right)\right] d\theta \\ &= \int_{\theta_{i+1}}^{\theta_i} \left[\upsilon\left(\theta, \Phi_i\right) - \upsilon\left(\theta, \Phi_{i-1}\right)\right] d\theta. \end{aligned} \tag{4.27}$$

Given the order of marginal cost of node coverage in (4.16) in Section 4.3.2, it follows that the LUIC is positive. As such, we have:

$$\eta\left(\theta_{i+1}, \Phi_i\right) - \eta\left(\theta_i, \Phi_i\right) \geq \eta\left(\theta_{i+1}, \Phi_{i-1}\right) - \eta\left(\theta_i, \Phi_{i-1}\right). \tag{4.28}$$

Adding the LUIC inequalities presented in (4.26) together with that of (4.28), we have:

$$\eta(\theta_{i-1}, \Phi_{i-1}) + \tilde{R}_{i-1} \geq \eta(\theta_{i+1}, \Phi_{i-1}) + \tilde{R}_{i+1}. \tag{4.29}$$

By considering the Local Downward Incentive Constraint (LDIC), i.e., IC constraint between the i^{th} and $(i-1)^{th}$ UAV, we are able to derive that:

$$\eta(\theta_{i+1}, \Phi_{i+1}) + \tilde{R}_{i+1} \geq \eta(\theta_{i-1}, \Phi_{i+1}) + \tilde{R}_{i-1}. \tag{4.30}$$

As such, given that the LUIC and LDIC hold, we have proven that the PIC of the contracts hold, i.e., $\omega_{i-1} \overset{PIC}{\Leftrightarrow} \omega_{i+1}$. □

With Lemma 4.3, we are able to reduce YZ IR constraints into a single constraint, i.e., as long as the minimum utility type Φ_{YZ} UAV has a non-negative utility, it follows that the other IR constraints will hold. Moreover, with Lemma 4.4, we are able to reduce $YZ(YZ-1)$ constraints into $YZ-1$ constraints, i.e., as long as the PIC constraint of the type Φ_i and type Φ_{i+1} UAV holds, it follows that the IC constraints between the type Φ_i and all other UAV types will hold.

With this, we are able to derive a tractable set of sufficient conditions for the feasible contract in Theorem 4.2 as follows. The first condition refers to the reduced IR condition corresponding to Lemma 4.3, whereas the second condition refers to the PIC condition between the type Φ_i and type Φ_{i+1} UAV corresponding to Lemma 4.4.

Theorem 4.2. A feasible contract must meet the following sufficient conditions:

i $\eta_{YZ}(\theta_{YZ}) + \tilde{R}_{YZ} \geq 0$

ii $\tilde{R}_{i+1} + \eta(\theta_{i+1}, \Phi_{i+1}) - \eta(\theta_i, \Phi_{i+1}) \geq \tilde{R}_i \geq \tilde{R}_{i+1} + \eta(\theta_{i+1}, \Phi_i) - \eta(\theta_i, \Phi_i)$.

4.3.4 Contract Optimality

To solve for the optimal contract rewards $\tilde{R}_i^*$, we first establish the dependence of optimal contract rewards $\mathbf{R}$ on route coverage θ. Thereafter, we solve the problem in (4.12) from Section 4.3.1 with $\boldsymbol{\theta}$ only. Specifically, we obtain the optimal rewards $R^*(\boldsymbol{\theta})$ given a set of feasible node coverages from each UAV which satisfies the monotonicity constraint $\theta_1 \geq \theta_2 \geq \cdots \geq \theta_i \geq \cdots \geq \theta_{YZ}$.

In addition, the multi-dimensional contract formulation that we have thus far only considered the self-revelation for two types, i.e., sensing and computation costs.

To account for traversal and transmission cost types, we add an additional fixed compensation $\hat{R}$ into the contract rewards. The traversal cost can be derived from the historical information of the UAV, and can be calibrated based on the response that the model owner receives. In the following theorem, we prove that the addition of a fixed rewards compensation does not violate the IC constraints, i.e., the self-revealing properties of the contract are still preserved, whereas it is inconsequential even if the IR constraint is violated, given that $\tilde{R}$ has already been designed to sufficiently compensate marginal costs, and only one optimal UAV is required to serve each subregion. The optimal rewarding scheme is summarized as follows.

Theorem 4.3. For a known set of node coverage $\boldsymbol{\theta}$ satisfying $\theta_1 \geq \theta_2 \geq \cdots \geq \theta_i \geq \cdots \geq \theta_{YZ}$ in a feasible contract, the optimal reward is given by:

$$R_i^* = \begin{cases} \hat{R} - \eta(\theta_i, \Phi_i), & \textit{if } i = YZ, \\ \hat{R} + \tilde{R}_{i+1} + \eta(\theta_{i+1}, \Phi_i) - \eta(\theta_i, \Phi_i), & \textit{otherwise.} \end{cases} \tag{4.31}$$

Proof. There are two parts for the proof. Firstly, we prove that the reward design for the two-dimensional contract is optimal. Therefore, we adopt the proof by contradiction. We first assume there exists some $\mathbf{R}^\dagger$ that yields greater profit for the model owner, meaning that the theorem is incorrect, i.e., $\Pi(R^\dagger) > \Pi(R^*)$. For simplicity, we need to consider only the reward portion of the model owner's profit function in this proof, i.e., $\sum_{i=1}^{YZ} R_i^\dagger < \sum_{i=1}^{YZ} R_i^*$. This implies there exists at least a $t \in \{1, 2, \ldots, YZ\}$ that satisfies the inequality $R_t^\dagger < R_t^*$.

According to the PIC constraint of Lemma 4.4, we have:

$$R_t^\dagger \geq R_{t+1}^\dagger + \eta\left(\theta_{t+1}, \Phi_{t+1}\right) - \eta\left(\theta_t, \Phi_t\right). \tag{4.32}$$

In contrast from Theorem 4.3, we have:

$$R_t^* = R_{t+1}^* + \eta\left(\theta_{t+1}, \Phi_{t+1}\right) - \eta\left(\theta_t, \Phi_t\right). \tag{4.33}$$

From (4.32) and (4.33), we can deduce that $R_{t+1}^\dagger < R_{t+1}^*$. Continuing the process up to $t = YZ$, we have $R_{YZ}^\dagger \leq R_{YZ}^* = -\eta(\theta_i, \Phi_i)$, which violates the IR constraint. As such, there does not exist the rewards $\mathbf{R}^\dagger$ in the feasible contract that yields greater profit for the model owner. Intuitively, the model owner chooses the lowest reward that satisfies the IR and IC constraints for profit maximization.

Secondly, we show that adding a fixed traversal cost reward does not violate the IC constraint. Within a subregion, when we consider the complete utility function of the auxiliary UAV with type i, $i \neq i'$,

$$\begin{aligned} \eta\left(\theta_i, \Phi_i\right) + \tilde{R}_i + \hat{R} - \psi_i^n - \zeta_i^n \geq \\ \eta\left(\theta_{i'}, \Phi_i\right) + \tilde{R}_i + \hat{R} - \psi_i^n - \zeta_i^n. \end{aligned} \tag{4.34}$$

Intuitively, the traversal and transmission cost is structurally separate from the marginal costs, i.e., sensing and computation cost types of the UAV within a subregion n. As such, the fixed reward terms cancel out and the self-revealing properties of the contract is preserved.

Note that the IR constraint may no longer hold for some i where

$$\psi_i > \eta\left(\theta_i, \Phi_i\right) + \tilde{R}_i + \hat{R}^n. \tag{4.35}$$

However, this is inconsequential given that unlike the conventional contract theoretic formulations, we only require a type of UAV to serve a subregion. Moreover, $\tilde{R}$ is already designed such that the IR constraints hold to compensate marginal costs sufficiently. For certain subregions, $\hat{R}$ can be calibrated upwards if no UAV responds to the model owner.

□

Following (4.31), we can re-express the optimal rewards as:

$$R_i^* = \hat{R} - \eta(\theta_{YZ}, \Phi_{YZ}) + \sum_{t=i}^{YZ} \Delta_t, \tag{4.36}$$

where $\Delta_{YZ} = 0$, $\Delta_t = \eta(\theta_{i+1}, \Phi_i) - \eta(\theta_i, \Phi_i)$, and $t = 1, 2, \ldots, YZ - 1$.

Unlike conventional contract theoretic formulations, the model owner only requires a single contract per subregion n for the optimal UAV. From (4.16) in Section 4.3.2, we can deduce that the optimal type-i^* UAV to serve each subregion is $i^* = \arg\max_{\Phi_i \in \mathbf{\Phi}} \Pi(\Omega\{\mathcal{A}, \mathcal{B}\}) = \arg\min_{\Phi_i \in \mathbf{\Phi}} \Phi_i = \Phi_1$. Intuitively, for each region, the model owner leverages the self-revealing properties of the multi-dimensional contract formulation to obtain an optimal UAV with the lowest marginal cost of node coverage for profit maximization. In other words, this is the UAV that can

cover the largest proportion of the subregion at the lowest cost, among all feasible UAVs that can complete the task. We can substitute the optimal rewards into the profit function of the model owner and rewrite the profit maximization problem as follows:

$$\begin{aligned} &\max_{(R,\theta_{i^*}^n)} \Pi(\Omega^n) = \sum_{n=1}^{N} G^n(\theta_1^n), \\ &\text{s.t.} \\ &C1: \ \theta_1^n \geq \theta_2^n \geq \cdots \geq \theta_i^n \geq \cdots \geq \theta_{YZ}^n, \\ &C2: \ 0 \leq \theta_1^n \leq 1, \end{aligned} \tag{4.37}$$

where:

$$\begin{aligned} G^n &= \frac{\sigma}{N} log(1 + \mu\theta_1^n D^n) - R_1^* \\ &= \frac{\sigma}{N} log(1 + \mu\theta_1^n D^n) - \hat{R} - \phi(\alpha_1^n \theta_1^n + \beta_1 \theta_1^n). \end{aligned} \tag{4.38}$$

Note that $C1$ refers to the monotonicity constraint of the contract whereas $C2$ specifies the upper and lower bounds of the route coverage. Taking the first order derivative of (4.37) to solve for θ_1^{n*}, we can derive the closed form solution of the optimal node coverage-contract reward pair as follows:

$$\begin{cases} \theta_1^{n*} = \frac{1}{\mu D^n} \left(\frac{\sigma}{N\phi(\alpha_1 + \beta_1)} - 1 \right), \\ R_i^* = \hat{R} - \eta(\theta_{YZ}, \Phi_{YZ}) + \sum_{t=i}^{YZ} \Delta_t. \end{cases} \tag{4.39}$$

Following (4.39), we are able to derive an optimal node coverage-contract reward pair upon the announcement of the UAV types. If the derived contract pairs satisfy the monotonicity conditions, they are the optimal contract formulation. Otherwise, we use the iterative adjusted algorithm, i.e., "Bunching and Ironing" algorithm [145], to return the results that satisfy the monotonicity constraint.

Note that to derive the optimal rewards for any type-i^* UAV, it is necessary to obtain the rewards for type-$i^*+1, \ldots,$ type-YZ UAV. However, since there can only be one selected UAV for each subregion, the reward for the $(i^*+1)^{th}$ UAV and onward has to be zero in practice. As such, to compute the rewards, we first assume the type-YZ UAV is selected to obtain a hypothetical R_{YZ}. With this, we can then work from backwards, towards deriving $R_{YZ-1}, \ldots, R_{i^*}$. Thereafter,

the model owner specifies route assignment $\mathcal{A}^{1,n}$ for the optimal UAV such that $\theta_1^{n*} = \frac{\sum_{i' \neq i, i' \in \mathcal{I}_n} a_{i,i'}^{1,n}}{|\mathcal{I}_n|}$.

In the next section, we consider the UAV-subregion assignment.

4.4 UAV-Subregion Assignment

In this section, we consider the matching-based UAV-subregion assignment using the GS algorithm. In Section 4.3, we note that the optimal UAV for each subregion has to be the lowest UAV type, i.e., type-1 UAV. However, given that all subregions will prefer type-1 UAV, there exists a need to consider a two-side matching such that the optimal UAVs are matched to the subregions efficiently.

4.4.1 Matching Rules

We introduce a complete, reflexive, and transitive binary preference relation [178], i.e., "$\succ$" to study the preferences of the UAV. For example, $n \succ_j n'$ implies that the UAV j strictly prefers subregion n to n', whereas $n \succeq_j n'$ indicates that the UAV j prefers the subregion n at least as much as UAV j'. We also consider the core definitions as follows:

Definition 4.3. *Matching:* For the formulated matching problem $(\mathcal{J}, \mathcal{N}, \mathcal{P}_n, \mathcal{P}_j)$, where $\mathcal{J}$ and $\mathcal{N}$ denote the set of UAVs and the set of subregions respectively, whereas $\mathcal{P}_n$ and $\mathcal{P}_j$ denote the preferences of the subregions and UAVs respectively. The matching $\mathcal{M}(j) = n$ indicates that the UAV j has been matched to subregion n, whereas the matching $\mathcal{M}(j) = \emptyset$ implies that the UAV j has not been matched to any subregion.

Definition 4.4. *Propose Rule:* The UAV $j \in \mathcal{J}$ announces its type to all the eligible subregions based on feasibility of task completion. Then, the contract is formulated and the subregion proposes to its most preferred UAV j^* in its preference set $\mathcal{P}_n$, i.e., $n^* \succ_j n^{*\prime}$, $n^* \forall n \in \mathcal{N}$, $n^* \neq n^{*\prime}$. Note that the preference of the subregion is managed by the FL model owner.

Definition 4.5. *Reject Rule:* The UAV $j \in \mathcal{J}$ rejects the subregion if a better matching candidate exists. Otherwise, the subregion that is not rejected will be retained as a matching candidate.

However, given that the subregion preference for UAV type is only based on two of four type dimensions, i.e., marginal costs, some subregions may have multiple preferred UAVs with the same marginal costs. To that end, adopting the approach in [179], we introduce the rewards calibration rule by adjusting the traversal and transmission compensation downward to further reduce the number of eligible UAVs matched to each subregion.

Definition 4.6. *Rewards Calibration Rule:* For the subregion $n \in \mathcal{N}$ that has more than one optimal UAV matched, the contractual rewards can be adjusted downwards, following which the preference of the UAVs are renewed for another iteration of matching. The adjustment is as follows:

$$\tilde{R}^n = \tilde{R}^n - \Delta\tilde{R}^n. \tag{4.40}$$

4.4.2 Matching Implementation and Algorithm

We now explain the implementation procedure of the UAV-subregion assignment.

Phase 1: Initialization

i The model owner announces the sensing subregions and time constraint $\bar{\tau_n}$ for task completion to all UAVs in $\mathcal{J}$. Since θ_j^n is not known apriori, the time limit for task completion can be a function of the variable $\hat{\theta}$ specified by the model owner, e.g., $\hat{\theta} = 0.8$, i.e., only UAVs that can complete 80% of the route coverage should announce their types (Line 4 of Algorithm 4.1).

ii The UAVs announce their types α_j^n and β_j^n to the feasible subregions (Line 5), i.e., subregions in which the UAVs are able to meet the requirements for task completion.

iii We initialize (Lines 6- 7):

  i $\mathcal{M}$ as an empty set

  ii $\mathcal{R}$ as the set of subregions that have yet to be matched, i.e., $\mathcal{R} = \mathcal{N}$ at initialization

  iii $\mathcal{P}_n$ as the set of subregion preferences based on the ascending order of marginal costs $\Phi(\alpha_j^n, \beta_j^n)$ as discussed in Section 4.3.4.

Algorithm 4.1 GS Algorithm for UAV-Subregion Assignment

1: **Input:** $\mathcal{N}, \bar{\tau_n}, \mathcal{J}, \Psi, \mathcal{A}, \mathcal{B}, \mathcal{C}, E_j$
2: **Output:** $\mathcal{M}^*(j), \forall j \in \mathcal{J}$
3: Phase I: Initialization
4: **if** $\tau_P^{j,n}(\hat{\theta}) + \tau_C^{j,n}(\hat{\theta}) + \tau_T^{j,n} \leq \bar{\tau_n}(\hat{\theta})$ **then**
5: UAV $j \in \mathcal{J}$ report types α_j^n, β_j^n to subregion n
6: **end if**
7: Sort $\Phi(\alpha_j^n, \beta_j^n)$ in an ascending order to derive $\mathcal{P}_n$
8: **Set** $\mathcal{R} = \mathcal{N}$, $\mathcal{M} = \emptyset$
9: Phase II: Iterative Matching
10: **while** $\mathcal{R} \neq \emptyset$ and $\mathcal{P}_n \neq \emptyset$, $\forall n \in \mathcal{R}$ **do**
11: **for** n in $\mathcal{R}$ **do**
12: $j^* = \arg\min_{\Phi_i \in \mathbf{\Phi}} \Phi_i$
13: Formulate $(\theta_{j^*}^n, R_{j^*}^n)$ and propose contract
14: **while** n has more than one optimal UAV **do**
15: $\tilde{R}^n = \tilde{R}^n - \Delta\tilde{R}^n$
16: **end while**
17: **if** $u_{j^*}^n(\theta_{j^*}^n) > u_{j^*}^{n'}(\theta_{j^*}^{n'})$ **then**
18: $\mathcal{M}(j^*) = n$
19: Remove n from $\mathcal{R}$ and add n' to $\mathcal{R}$
20: **else**
21: Remove j^* from $\mathcal{P}_n$
22: **end if**
23: **end for**
24: **end while**

Phase 2: Iterative Matching

Each iteration of matching consists of four stages.

i *Proposal:* For each subregion $n \in \mathcal{R}$, the node coverage-contract reward pair is formulated for the most optimal UAV $j^* = \arg\min_{\Phi_i \in \mathbf{\Phi}} \Phi_i$ in the preference set $\mathcal{P}_n$. Then, the subregion proposes to an optimal UAV (Lines 11-12).

ii *Rewards Calibration:* If the subregion has more than one optimal UAV, the contract reward is calibrated downward until a one-to-one matching is achieved (Line 13-14).

iii *Rejection:* If the UAV has a better matching candidate $u_{j^*}^n(\theta_{j^*}^n) < u_{j^*}^{n'}(\theta_{j^*}^{n'})$, the subregion is rejected. If not, the UAV keeps the subregion as a matching candidate.

iv *Update:* If a subregion has been matched, remove it from $\mathcal{R}$ (Line 17). If a prevailing matching candidate has been rejected, add it back to $\mathcal{R}$ (line 17). Update $\mathcal{P}_n$ by removing the UAV that has issued a rejection in this iteration (Line 19).

The iterations are repeated until all subregions have been matched, i.e., $\mathcal{R} = \emptyset$, or the remaining subregion has been rejected by all UAVs in its preference list, i.e., $\mathcal{P}_n = \emptyset$. The pseudocode is presented in Algorithm 4.1.

The stability and optimality properties of the GS algorithm are ensured following the proofs in [180]. As such, the self-revealing properties of our multi-dimensional contract design assure truthful type reporting, whereas the matching algorithm ensures that only one optimal UAV is matched to each region. In the following, we perform the performance evaluation of our incentive design.

4.5 Performance Evaluation

In this section, we consider the optimality of our devised contract. To illustrate the contract optimality, we first consider the case of six UAVs and a single subregion. Then, we study a single iteration of matching between 5 UAVs and 3 subregions. Finally, we study the GS matching-based UAV-subregion assignment that involves up to 7 UAVs and 6 subregions. Unless otherwise stated, the list of value ranges for the key simulation parameters is as summarized in Table 4.2. The key parameters we use are with reference to studies involving UAV and FL optimization [131, 173].

4.5.1 Contract Optimality

To illustrate the optimality of our multi-dimensional contract design, we first consider a highly simplified and demonstrative case of a single subregion and six UAVs of ascending marginal cost of route coverage. The values of α for the UAVs lie in the range of $[250, 875]$, with increments of 125, whereas the β values lie in the range of $[20, 70]$, with increments of 10. The values are varied as presented in Table 4.2. Then, the auxiliary types are derived following (4.16) from Section 4.3.2, and are arranged in an ascending order for contract derivation. Type-1 UAV has the lowest

TABLE 4.2: Table of Key Simulation Parameters.

Simulation Parameters	**Value**
UAV Sensing and Traversal Parameters	
p	10 - 35
v	10 - 20 m/s
l^n	1000 - 2000 m
$l^n_{C_j}$	500 - 1000 m
UAV Computation Parameters	
L	4
ε	$\frac{1}{3}$
δ	$\frac{1}{4}$
γ	2
A^*	0.6
ε	$\frac{1}{3}$
C	$10 - 30$ cycles/bit
K	24
V	4
κ	10^{-28}
f	2 GHz
UAV Transmission Parameters	
D^n	$500 - 1000$ MB
λ^n	$10000 - 15000$
H	1 MB
ρ	$8 - 18$
UAV and Model Owner Utility Parameters	
ϕ	0.05
μ	1
σ	100000

marginal cost of node coverage, whereas type-6 UAV has the highest marginal cost of node coverage. To focus our study on the optimality of our contract design, we hold the traversal and transmission cost types of the UAV constant for now. In addition, we assume that all the UAVs can complete the task within the time constraints.

Fig. 4.2 and Fig. 4.3 consider the hypothetical scenarios in which each particular UAV type takes turn to be matched to the subregion. As an illustration, if UAV type-1 has been matched to serve the subregion, the optimal node coverage is 1, whereas the contractual reward is 35. In contrast, if UAV type-6 is matched, the optimal node coverage is close to 0.4, whereas the reward is 20. From Fig. 4.2 and Fig. 4.3, we can observe that the monotonicity condition discussed in Theorem 4.1

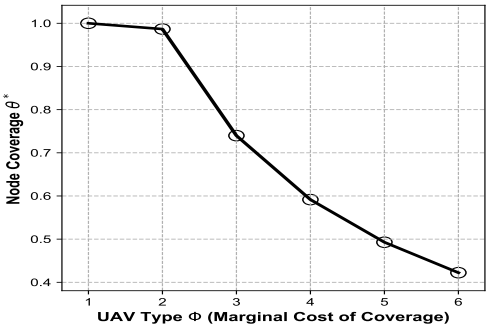

FIGURE 4.2: UAV node coverage vs. auxiliary types

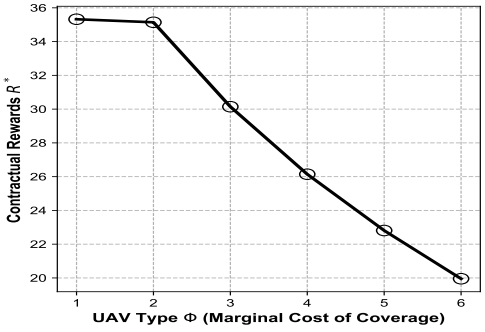

FIGURE 4.3: Contract rewards vs. auxiliary types.

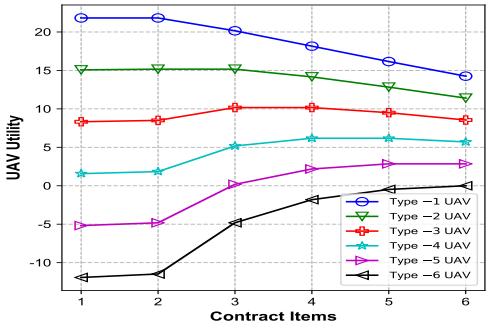

FIGURE 4.4: Contract items vs. UAV utilities.

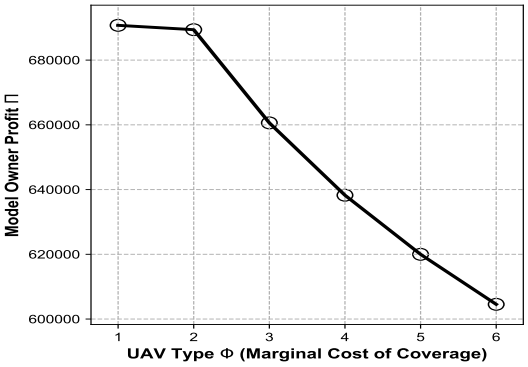

FIGURE 4.5: The model owner profits vs. UAV auxiliary types.

from Section 4.3.3 holds. In other words, the higher is the marginal cost of node coverage, the lower is the optimal node coverage and contract rewards.

Fig. 4.4 shows that the IC constraints of the contract hold. As an illustration, we consider the type-6 UAV, i.e., the UAV with the maximum marginal cost of node coverage. The type-6 UAV derives negative utility if it misreports its type, i.e., to imitate any other lower marginal cost UAV types $1-5$. As discussed in Definition 4.2 from Section 4.3.1, each UAV derives the highest utility only if it reports its type truthfully to the model owner. This validates the self-revealing mechanism of our contract.

Fig. 4.5 shows that the model owner profits are the highest when it is able to be matched with the UAV of the lowest marginal cost of node coverage. This validates our discussion in Section 4.3.4, and confirms the model owner preference. In other

TABLE 4.3: UAV Types For Preference Analysis.

UAV Type	Coordinates	α	β	Subregion Preference
1	$(100, 100)$	500	20	$(2, 3, 1)$
2	$(900, 900)$	500	20	$(1, 3, 2)$
3	$(400, 400)$	750	30	$(3, 2, 1)$
4	$(450, 450)$	750	30	$(3, 2, 1)$
5	$(500, 500)$	1000	40	(3)

words, among UAVs that are able to complete the task, the model owner prefers the UAV that incurs the lowest cost.

4.5.2 UAV-Subregion Preference Analysis

To analyze the preferences of the UAVs and subregions before we proceed with matching, we consider 5 UAV types and 2 subregions. In particular, the auxiliary types of the UAVs are shown in Table 4.3. Similarly, the types are derived from the calibration of the parameters listed in Table 4.3. The UAVs are sorted in the ascending order based on marginal cost of node coverage. Besides, we consider three subregions $1, 2, 3$ of coordinates $(1000, 1000)$, $(50, 50)$, $(500, 500)$. The subregion preference for each UAV is also presented in the last column of Table 4.3.

Following our discussion in Section 4.3.4, the subregions prefer the UAV types with lower marginal cost. Naturally, the preferences for all three subregions are similar as follows: $(1, 2, 3, 4, 5)$. Note that the subregions are indifferent between types 1 and 2, as well as types 4 and 5 given that the pairs have the same marginal cost of node coverage.

To consider the UAV preferences, we plot the potential profits that each UAV may gain from covering the different subregions in Fig. 4.6. Note that this profit is a hypothetical one in some cases, since the profits can only be realized *if* the UAV has been matched to cover the subregion. However, given that the UAV is not aware if it will be matched to the subregion apriori, the preference list of the UAV can only be constructed with the assumption that it is indeed matched to the subregion. We note that UAV 1 prefers subregion 2, whereas UAV 2 prefers subregion 1 and so on. Intuitively, the preference for subregions relies on the traversal costs, i.e.,

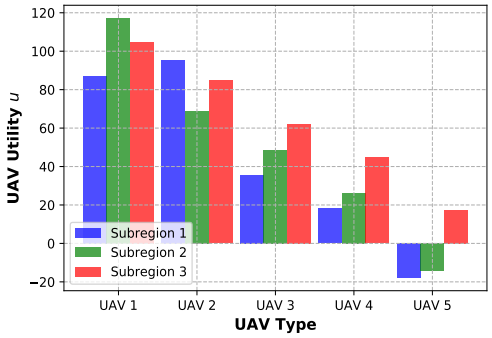

FIGURE 4.6: The UAV utility for each subregion vs. types.

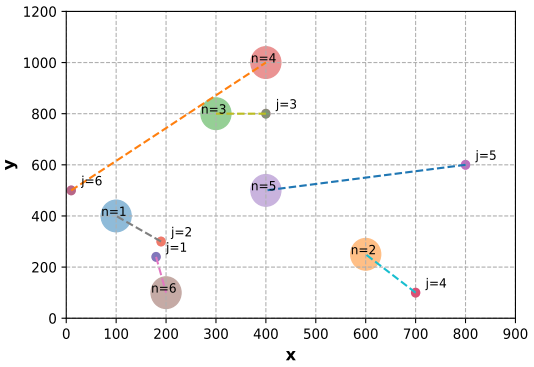

FIGURE 4.7: UAV matching for homogeneous subregions.

the cost of traveling to and from the subregion. As such, the preferences for UAVs 1 and 2 are $(2, 3, 1)$ and $(1, 3, 2)$ respectively.

On the other hand, UAV 5 prefers only the closest region 3, given the potential negative profits derived if it serves the other two subregions, as a result of the high marginal costs incurred for task completion. As such, we are able to derive the matching of (Subregion 2, UAV 1) and (Subregion 1, UAV 2) given that the UAV-subregion preferences match perfectly.

The consideration for subregion 3 is clearly more challenging than that of subregions 1 and 2 given that the subregion is indifferent between the two remaining UAVs 3 and 4, and that the UAVs also rank the subregion highest, in terms of preference. To that end, we consider the rewards calibration rule proposed in Definition 4.6 from Section 4.4.1. The contract rewards are calibrated downwards till a UAV emerges as the only choice left. In this case, after the downward calibration of rewards $\tilde{R}$, UAV 4 will clearly be matched with subregion 3, given its close proximity to the subregion.

Through this relatively straightforward example, we are able to derive an insight, i.e., a successful match will have the lowest marginal cost type UAVs matched to the subregion that it is situated closest to. Clearly, Fig. 4.6 also validates the efficiency of our incentive mechanism design, i.e., the best available UAV is matched to the respective subregion.

TABLE 4.4: UAV Type and Preference for Subregions.

UAV Type	Subregion Preference
1	$(6, 1, 5, 2, 3, 4)$
2	$(6, 1, 5, 3, 2, 4)$
3	$(3, 4, 5, 1, 2, 6)$
4	$(2, 5, 6, 1, 3, 4)$
5	$(2, 5, 3, 4, 1, 6)$
6	$(1, 5, 3, 6, 4, 2)$

4.5.3 Matching-Based UAV-Subregion Assignment

In this section, we consider the matching-based UAV-subregion assignment. In particular, we consider three scenarios to illustrate the matching-based assignment.

In the *first* scenario, six UAVs of ascending marginal cost types are initialized to choose among six subregions that are of varying distances from each UAV. Each of the subregions is calibrated to hold the same quantities of data (D^n) and sensing area (l^n) for coverage. For ease of exposition, the UAVs are all able to complete their tasks within their energy capacities and stipulated time constraints. The coordinates of the subregions and UAVs, as well as the matching outcomes, are presented in Fig. 4.7. The preference list of the UAVs is presented in Table 4.4. Note that the preference list of each subregion is simply $(1, 2, 3, 4, 5, 6)$, i.e., among all feasible UAVs that can cover the subregion within the time and energy constraints, the UAV with the lowest marginal cost is preferred.

From Fig. 4.7, we observe that the UAV 1 is matched to its most preferred subregion 6. Though UAV 2 also prefers subregion 6, it is unable to be matched to the subregion given that UAV 1 is higher up on the list of preferences of subregion 6. As such, UAV 2 is matched to its second choice. Naturally, the matching between UAV 3 and subregion 3, UAV 4 and subregion 2, as well as UAV 5 and subregion 5 is intuitive, given the unavailability of the other more preferred UAVs for the subregions to match with. We observe that UAV 6 is finally matched with its fifth choice, given that the UAV 6 has the lowest priority among subregions.

In the *second* scenario, we consider the same UAV types but with heterogeneous subregions of different data quantities and sensing areas for coverage. As is expected, the sizes of the regions do not affect the matching outcomes and the matching remains the same (Fig. 4.8). This is given that the preference rankings of the

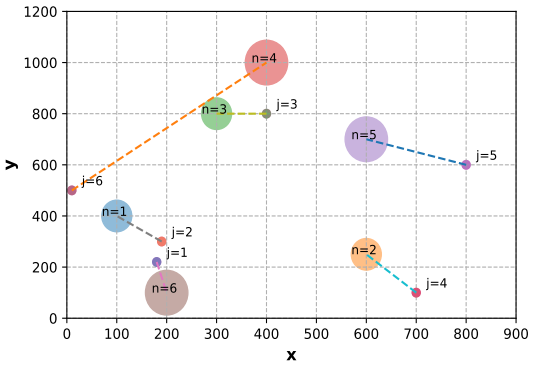

FIGURE 4.8: UAV matching for subregions with different data quantities and coverage area.

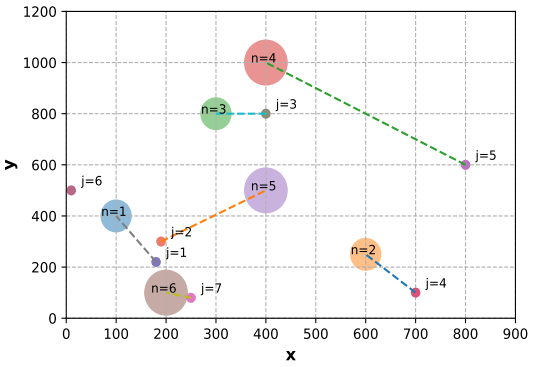

FIGURE 4.9: UAV matching where $J > N$.

subregion and the UAV remain constant. While the varying values of D^n and l^n affect the *magnitude* of UAV types, the *ordering* of the UAV types, and thus their preferences, is retained. This is important to ensure that the monotonicity of our contract design holds across subregions, so as to preserve the contract optimality.

In the *third* scenario, we consider the case where $J > N$, i.e., the number of UAVs exceeds the number of subregions available. As an illustration, we add in the UAV 7, which has the lowest marginal cost of node coverage relative to that of the other six available UAVs from the aforementioned scenarios. We observe from Fig. 4.9 that the matching outcomes have changed. UAV 7 is now matched with its most preferred subregion, i.e., subregion 6, in place of UAV 1. Naturally, this affects the assignment for the other UAVs. For example, UAV 1 has to be matched to its second choice now, whereas UAV 2 has to be matched to its third choice. We observe that the UAV of the largest type, i.e., UAV 6 is left out of the assignment as a result.

The simulation results allow us to validate the efficiency of our mechanism design. Firstly, the contract design ensures truthful type reporting and the incentive compatibility of our contract is validated. Secondly, with consideration of the preferences, the available UAV with the lowest marginal cost of node coverage is matched to the subregion. This ensures the profit maximization of the model owner.

4.6 Conclusion and Chapter Discussion

In this chapter, we have considered an FL based sensing and collaborative learning scheme involving UAVs for applications in the IoV paradigm. Given the incentive mismatches between the UAVs and the model owners, we have proposed a multi-dimensional contract-matching incentive design such that the UAV with the lowest marginal cost of node coverage is assigned to each subregion for task completion.

As we have observed in this chapter, we have utilized the multi-dimensional contract optimization to account for various sources of heterogeneity in the network. While we have considered a three dimensional contract in this chapter, it is possible to further extend this to account for other sources of heterogeneity. When utilized in the UAV network which is characterized by various operational complexities, the multi-dimensional contract has indeed proven to be a powerful tool for self-revelation in a network with information asymmetry. However, the chapter has the following drawbacks.

Firstly, even though UAVs have proven to be relatively useful and advantageous due to their flexibility in deployment and low cost, the reliability of UAVs is still uncertain. In particular, with weather uncertainties, the UAVs may fail to operate. However, in this chapter, we have not accounted for this uncertainty. One solution is to incorporate a backup UAV for the coverage of a region after the initially chosen UAV fails. The sensing and model training tasks of the UAVs can be conducted in multiple stages. After the first stage failure, a backup UAV can be deployed in the second stage and so forth. This step for the backup request can be added in Section 4.5.3, together with the matching phase. However, the backup UAV's addition implies additional cost incurred for the model owner. As such, to minimize the total network cost while accounting for the backup UAV reservation and corrective cost, we can utilize a *z-stage* stochastic optimization approach [181].

Secondly, in this chapter, we have assumed that the UAVs only train the FL model at their respective charging stations. For time sensitive tasks, the delay incurred for the UAVs to return to their charging stations may be intolerable. In this chapter, we have omitted the discussion for brevity. Several works have introduced the concept of aerial computing, in which the UAVs perform model training onboard. However, with its limited battery constraint, this may be challenging. As a solution, the UAVs can also leverage offloading to edge servers to share this computation

burden. To mitigate straggling edge servers, the work of [182] proposes a coded distributed computing approach in which a computation task can be partitioned into subtasks for offloading to the edge. Moreover, to preserve the privacy of FL, coding techniques such as random linear coding [183] can be utilized such that colluding edge servers are unable to recover the raw data of the UAVs.

Finally, in this chapter, we have assumed that the model owner is the sole model owner in the FL network. Following our discussion in Chapters 2 and 3, there can be exceptions to this setting as follows: (a) the workers may be competing data owners who are reluctant to share their model parameters since the competitors also benefit from a trained global model and (b) the FL servers may compete with other FL servers, i.e., model owners. In this case, the formulation of the incentive mechanism design will be vastly different from that proposed. In Chapter 5, we will revisit this notion and devise a deep learning based auction mechanism in response, to model the competition among multiple model owners for the restricted participation of model owners.

Chapter 5

Evolutionary Edge Association and Auction in Hierarchical Federated Learning

5.1 Introduction

Today, the predominant approach for AI based model training is cloud-centric, i.e., the data owners transmit the training data to a public cloud server for processing. However, this is no longer desirable due to the following reasons. Firstly, privacy laws, e.g., GDPR [2], are increasingly stringent. In addition, the privacy-sensitive data owners can opt out of data sharing with third parties. Secondly, the transfer of massive quantities of data to the distant cloud burdens the communication networks and incurs unacceptable latency especially for time-sensitive tasks. As such, this necessitates the proposal of Edge Computing [35] as an alternative, in which raw data are processed at the edge of the network, closer to where data are produced.

The confluence of Edge Computing and AI gives rise to Edge Intelligence, which leverages the storage, communication, and computation capabilities of end devices and edge servers to enable edge caching, model training, and inference [1] closer to where data are produced. One of the enabling technologies [184] of Edge Intelligence is the privacy preserving machine learning paradigm termed FL [3]. In FL,

only the updated model parameters, rather than the raw data, need to be transmitted back to the model owner for global aggregation. The main advantages of FL are: (a) FL enables privacy preserving collaborative machine learning, (b) FL leverages on the computation capabilities of IoT devices for local model training, thus reducing the computation workload of the cloud, and (c) Model parameters are often smaller in size than raw data, thus alleviating the burden on backbone communication networks. This has enabled several practical applications.

However, the FL network is envisioned to involve thousands of heterogeneous distributed devices, e.g., smartphones and Internet of Thing (IoT) devices [121]. In this case, communication inefficiency remains a key bottleneck in FL. Specifically, node failures and device dropouts due to communication failures can lead to inefficient FL. Moreover, workers, i.e., data owners, with severely limited connectivity are unable to participate in the FL training, thus adversely affecting the model's ability to generalize. As such, solutions from edge computing have recently been incorporated to solve the communication bottleneck in FL. In [184–186], a *hierarchical* FL (HFL) framework is proposed in which the workers do not communicate directly with a central controller, i.e., the model owner. Instead, the local parameter values are first uploaded to edge servers, e.g., at base stations, for intermediate aggregation. Then, communication with the model owner is further established for global aggregation. Besides reducing the instances of global communications with the remote servers of the model owner, this relay approach reduces the dropout rate of devices.

While [185] discusses convergence guarantees and presents empirical results to show that the HFL approach does not compromise on model performance, the challenges of resource allocation and incentive mechanism design have not yet been well-addressed in the HFL framework. In 5G and Beyond networks, the resource sharing and incentive mechanism design for end-edge-cloud collaboration are of paramount importance to facilitate efficient Edge Intelligence [184].

In this chapter, we consider a decentralized learning based system model inspired by the HFL. In our system model, there exist data owners, hereinafter referred to as workers, that participate in the FL model training facilitated by different cluster heads, e.g., base stations that support the intermediate aggregation of model parameters and efficient relaying to the model owners (Fig. 5.1). We consider a two-level resource allocation and incentive design problem as follows:

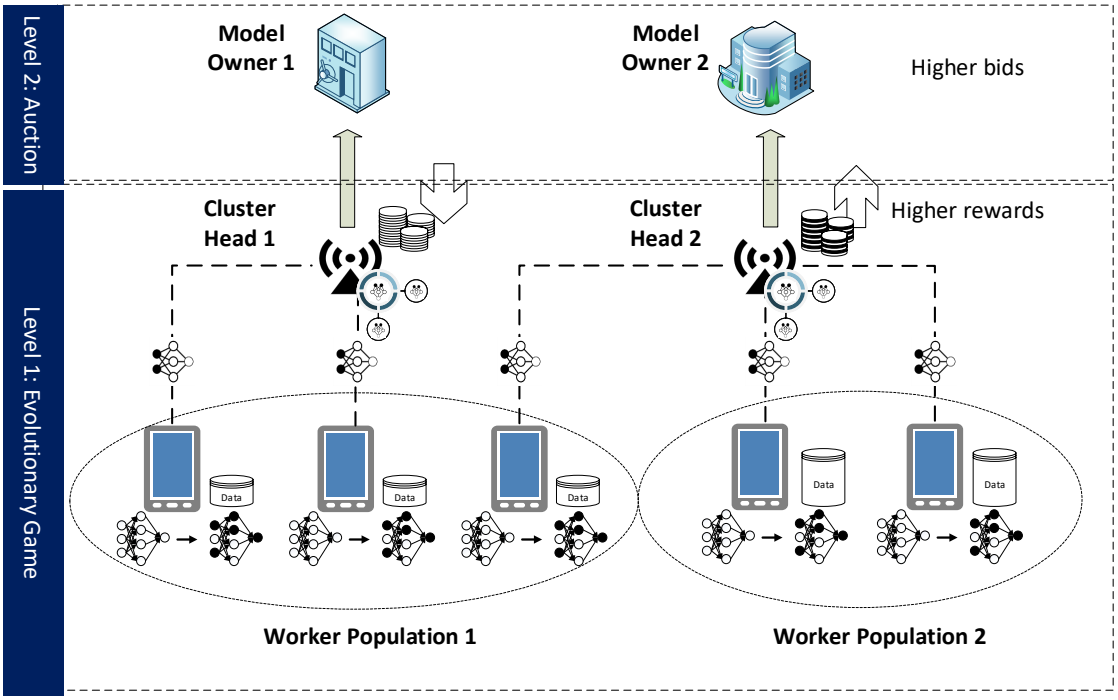

FIGURE 5.1: An illustration of our system model involving two populations of workers. Within each population, all workers have the same data quantities. The workers may choose to join either cluster head. The dynamics are modeled at the first level using the evolutionary game. Cluster head 2 eventually has higher data coverage across the network given that it offers the workers higher rewards. Thus, the services of cluster head 2 are valued higher at the auction.

i *Lower level (Between workers and cluster heads):* Each worker can freely choose which cluster to join. To encourage the participation of workers, the cluster heads offer reward pools to be shared among workers based on their data contribution in the cluster. For example, a worker that has contributed more data[1] during its local training will receive a larger share of the reward pool. Moreover, the cluster heads offer the workers resource blocks, i.e., bandwidth, to facilitate the efficient uplink transmission of the updated model parameters. However, as more workers join a cluster, the payoffs are inevitably reduced due to the division of rewards over a larger number of workers and the increased communication congestion. Thus, the cluster selection strategies of each worker can affect the payoffs of other workers. Accordingly, the workers may slowly adapt their strategies in response to other workers. In contrast to conventional optimization approaches, we use the evolutionary game theory

[1]Note that this knowledge is available to the cluster heads given that the workers have to report their available resources during the client selection procedure [101]. In this chapter, we omit discussions regarding the client selection phase.

[187] to derive the equilibrium composition of the clusters. Our game formulation enables bounded rationality and worker dynamics to be captured. Specifically, the workers gradually adapt their strategies in response to other non-cooperative workers. To achieve their objectives, they observe each others' strategies and gradually adjust their strategies accordingly. The solution is therefore, not immediately derived.

ii *Upper level (Between cluster heads and model owners):* There may be multiple model owners in the network that aim to train a model for their respective usage collaboratively with the participation of the workers and cluster heads. However, at any point in time, each worker and cluster head can only participate in the training process with a single model owner. To derive the allocation of cluster head to the model owner, as well as the optimal pricing of the services of the cluster head by the competitive model owners, we adopt a deep learning based auction mechanism that preserves the properties of truthfulness of the bidders, while simultaneously achieving revenue maximization for the cluster heads.

The main contributions of this chapter are as follows:

i We propose a joint resource allocation and incentive design framework for the HFL. The "Edge for AI" [188] approach supports decentralized Edge Intelligence, i.e., FL at the edge with reduced reliance on a central controller.

ii We model the cluster selection decisions of the workers as an evolutionary game. Then, we provide proofs for the uniqueness and stability of the evolutionary equilibrium. In contrast to conventional optimization tools that assume that the players are perfectly rational, our model enables us to capture the dynamics and bounded rationality of player decisions.

iii To assign the cluster heads to model owners, we use a deep learning based auction mechanism. In contrast to conventional auctions, the deep learning based auction ensures seller revenue maximization while satisfying the individual rationality and incentive compatibility constraints.

The organization of this chapter is as follows. In Section 5.2, we discuss the system model and problem formulation. In Section 5.3, we study and analyze the

evolutionary game. In Section 5.4, we discuss the deep learning based auction mechanism. Section 5.5 provides the performance evaluation and Section 5.6 concludes the chapter.

5.2 System Model and Problem Formulation

5.2.1 System Model

We consider a network that consists of a set $\mathcal{N} = \{1, \ldots, n, \ldots, N\}$ of workers. There exist L distinct model owners, each of which desires to develop an AI model for its own purposes, e.g., traffic crowdsensing [159] or location based recommendation [189]. Given the communication constraints of individual workers, the central controller-reliant conventional FL architecture is prone to high device dropout rates [186]. Moreover, we have J cluster heads, e.g., base stations, employed across the network to facilitate the HFL task, the set of which is denoted by $\mathcal{J} = \{1, \ldots, j, \ldots, J\}$. Each worker can choose to associate with any one $j \in \mathcal{J}$ cluster head.

Without loss of generality, each cluster head $j \in \mathcal{J}$ can only serve a single model owner $l \in \mathcal{L}$ at a time and facilitates the HFL process for a cluster j of $p_j N$ workers, where $\sum_j^J p_j = 1$.

In HFL, a cluster head j first receives an initial global model, i.e., parameters denoted by the vector $\boldsymbol{w}$, from a model owner that has chosen its services. It then relays the global model to its workers. The FL model training takes place over K iterations to minimize the global loss $F^K(\boldsymbol{w})$ where K is stipulated by the model owner. Each k^{th} iteration, $k \in \{1, \ldots, k, \ldots, K\}$, consists of three steps [78] namely:

i *Local Computation*: Each worker trains the received global model $\boldsymbol{w}^{(k)}$ locally.

ii *Wireless Transmission*: The worker transmits the model parameter update to its cluster head j.

iii *Intermediate Model Parameter Update*: All parameter updates received from its $p_j N$ workers are aggregated by the cluster head to derive an updated

intermediate model $\boldsymbol{w}_j^{(k+1)}$, which is then transmitted back to the worker for the $(k+1)^{th}$ training iteration.

After K iterations, the intermediate model $\boldsymbol{w}_j^{(K)}$ is transmitted to the model owner for global aggregation with the intermediate parameters collected from other clusters. A new set of updated global parameters is derived by the model owner that sends it out to its cluster heads for another round of local model training.

In this chapter, we assume that the cluster heads are pre-determined, e.g., through the cluster head selection algorithms based on energy efficiency [190, 191], trust [192], and social effects [193]. Instead, we focus our study on a two-level optimization problem as follows: a) in the *lower level*, we adopt an evolutionary game approach to study the dynamics of cluster selection by the workers to derive the dynamics of the composition of each cluster, and b) in the *upper level*, we adopt a deep learning based auction to value each cluster head's worth to a model owner.

5.2.2 Lower-Level Evolutionary Game

In the lower level, the cluster formation is derived given J cluster heads. Each cluster head has the objective of attracting more workers to join its cluster, since this ensures that the cluster will have a larger *data coverage* across the network. With a larger data coverage, the cluster value is increased, e.g., due to the fact that the model performance increases with more training data [2].

To encourage the participation of the workers, each cluster head offers a reward pool to be shared by all workers in the cluster. The reward to be distributed to each worker is based on the proportion of the worker's contributions in the cluster, i.e., its data quantity relative to the total amount of data in the cluster. On one hand, a cluster that offers a high reward pool is more attractive to workers. On the other hand, when more workers join that cluster, the reward pool has to be shared among a larger number of workers. Thus, the worker decisions as to which cluster they choose to join are interrelated with the decisions of other workers. We adopt an evolutionary game theory approach to model the dynamics of cluster formation.

5.2.3 Upper-Level Deep Learning Based Auction

The lower-level evolutionary game gives us the data coverage of each cluster. For example, a cluster head that has greater data coverage will be deemed more valuable to an FL model owner, since the model performance, e.g., inference accuracy, is improved [126]. However, recall from Section 5.2.1 that there exists more than one model owner in the network seeking to secure the services of the cluster head to facilitate the HFL. In consideration of the competition among model owners, we adopt an auction mechanism in which L model owners bid for the services of each cluster head $j \in \mathcal{J}$. Specifically, we utilize the deep learning based auction mechanism which has the attractive properties of ensuring the truthfulness of the bidders, as well as revenue maximization for the seller (i.e., cluster head), as discussed in Section 5.4.

5.3 Lower-Level Evolutionary Game

5.3.1 Evolutionary Game Formulation

In the following, we formulate the cluster selection as an evolutionary game:

i *Players:* The set of workers $\mathcal{N} = \{1, \ldots, n, \ldots, N\}$ in the FL network are the players of the evolutionary game. For clarity, we use the terms "workers" hereinafter.

ii *Population:* We partition the workers into $\mathcal{M} = \{1, \ldots, m, \ldots, M\}$ populations based on the data quantities[2] that each worker owns or the data coverage proportion of the worker using conventional data mining tools, e.g., k-means. The data coverage proportion can be a reflection of the workers' market share in the case when organizations are considered, e.g., based on the proportion of users a bank has, or usage frequency in the case in which individuals are considered, e.g., based on how often the worker uses an IoT device. Each worker of population m owns d_m training data samples,

[2]For simplicity, we only consider data quantity as the criterion for clustering here. Note that we may cluster the workers based on other important factors such as geolocation (as a proxy for data distribution) and reputation metrics (as a proxy for quality of contribution). Our work can easily be extended to cover such measures.

whereas the total data quantity in a population is denoted D_m. We denote the number of workers in each population as $n_m = p_m N$ where $p_m \in [0, 1]$ and $\sum_{m=1}^{M} p_m = 1$. In other words, we have M populations of workers in the network, where all workers within a population own the same number of data samples.

iii *Strategy:* The strategy of each worker in population m is the selection of a cluster to join so as to achieve utility maximization. The strategy space of each worker n in population m is denoted by $\mathcal{S}_n^{(m)} = \{a_{n,1}^{(m)}, \ldots, a_{n,j}^{(m)}, \ldots, a_{n,J}^{(m)}\}$ in which $a_{n,j}^{(m)}$ is a binary variable where $a_{n,j}^{(m)} = 1$ represents that the worker n in population m chooses the cluster j, whereas $a_{n,j}^{(m)} = 0$ indicates otherwise.

iv *Population Share:* We denote the fraction of population m that selects strategy j, i.e., cluster j, by $x_j^{(m)}$ where $\sum_{j=1}^{J} x_j^{(m)} = 1$. The population state [194] is denoted by the vector $\mathbf{x}^{(m)} = [x_1^{(m)}, \ldots, x_j^{(m)}, \ldots, x_J^{(m)}]^T \in \mathbb{X}$.

v *Payoff:* The expected payoff of each worker is determined by its net utility, which is the difference between the reward that it derives from joining a cluster, and the cost of participating in the FL model training. We further discuss payoffs in Section 5.3.2.

As an illustration, the system model and game formulation are illustrated in Fig. 5.1, for the case of two populations. Each worker in population 1 has fewer data samples than each worker in population 2. Each worker in the population can also choose to join either cluster head. Eventually, each cluster head is associated with a certain level of data coverage, and has its worth evaluated using the auction mechanism discussed in Section 5.4.

Note that in this chapter, we consider that each worker can only join a single cluster, as the worker device is unable to support two instances of model training in parallel. However, our model can be extended to the situation in which each worker can join more than one cluster at a time. In this case, the worker can be modeled to decide, in an evolutionary process, on how its limited resources can be divided among the model owners. Then, $x_j^{(m)} \in [0, 1]$ is denoted to represent the share of resources a worker from population m contributes to cluster head j, where $\sum_{j=1}^{J} x_j^{(m)} = 1$.

5.3.2 Worker Utility and Replicator Dynamics

The rewards derived by workers of population m, from joining a cluster j for K iterations of FL model training, are given by:

$$p_j^{(m)} = \alpha_j \frac{x_j^{(m)} D_m}{\sum_{m=1}^{M} x_j^{(m)} D_m} + R_j, \tag{5.1}$$

where α_j is the reward pool to be divided across all workers in cluster j based on their data contributions, $\frac{x_j^{(m)} D_m}{\sum_{m=1}^{M} x_j^{(m)} D_m}$ is the share of rewards based on the worker's data contribution[3], and R_j is a fixed reward offered to workers in cluster j based on the compensation for the workers' participation costs.

The cost of workers of population m incurred from joining a cluster j is given by the addition of the computation and communication cost[4] over the K iterations of the model training. The computation cost is as follows:

$$c_m^{cmp} = \eta \kappa \theta_m f_m^2, \tag{5.2}$$

where η is the unit cost of energy consumption, κ is the coefficient of the value that is determined by the circuit architecture of the worker Central Processing Unit (CPU) [131], θ_m is the number of CPU cycles required to perform local computation, i.e., model training, and f_m^2 refers to the computation capability of the worker which is determined by the clock frequency of the worker CPU. Without loss of generality, we have the computation cost held constant throughout for all workers, i.e., $c_m^{cmp} = c^{cmp}$, $\forall m \in \mathcal{M}$. To account for the varying computation and communication capabilities, we can straightforwardly extend our work to include multiple heterogeneous populations with varying computation capabilities. For example, if there are Λ varying computation costs, we can have ΛM populations accordingly.

The main benefit of HFL is that devices with communication constraints are able to participate in FL. To facilitate the communication of parameters, the cluster

[3]In this chapter, we make a simplifying assumption that the workers are not malicious. In practice, the parameter contributions of the workers can be randomly verified, e.g. detecting model parameters with outlying magnitudes, and malicious workers with anomalous contributions can be penalized from participating in the future FL training.

[4]For ease of presentation, we have used a simple computation and communication cost model in this Chapter. However, a more specific modeling of the cost can be referenced from Chapter 4. The use of the model does not affect the findings of this chapter.

head, e.g., a base station, distributes communication resource blocks to all participants within the cluster. On the one hand, clusters with more communication resources are attractive to participants since the participants can benefit from a higher achievable uplink transmission rate. On the other hand, with more participants attracted to join the cluster, the increased competition for resource blocks leads to more congestion. As such, following [195], we model the disutilities arising from network congestion effects as

$$c_{m,j}^{com}(t) = \zeta_j \left(\sum_{m \in \mathcal{M}} x_j^{(m)}(t) \right)^2, \tag{5.3}$$

where ζ_j is the congestion coefficient determined by the resource constraints of the cluster head [195], whereas $\left(\sum_{m \in \mathcal{M}} x_j^{(m)}\right)^2$ represents the usage profile across populations in the network for a particular cluster. Specifically, a cluster head with more resources will have a lower congestion coefficient. Moreover, workers in a less-populated cluster experience less congestion.

The total cost of participation incurred by worker n_m (of population m) in cluster j is obtained as

$$c_j^{(m)}(t) = c^{cmp} + c_{m,j}^{com}(t). \tag{5.4}$$

At time t, the net utility that the workers in class m receive for their participation in cluster j is:

$$u_j^{(m)}(t) = \mathcal{U}\left(p_j^{(m)}(t) - c_j^{(m)}(t) \right), \tag{5.5}$$

where we assume $\mathcal{U}(\cdot)$ to be a linear utility function indicating the risk neutrality of workers without loss of generality [196, 197].

Accordingly, at time t, the average utility of workers in population m across all J clusters is

$$\bar{u}^{(m)}(t) = \sum_{j=1}^{J} x_j^{(m)} u_j^{(m)}(t). \tag{5.6}$$

In practice, information regarding the utility derived from joining different clusters can be exchanged and compared among workers within the network [198]. Workers may thus switch from one cluster to another to seek higher utilities. To capture the dynamics of the cluster selection and model the strategy adaptation process, we define the replicator dynamics [199] as follows:

$$\dot{x}_j^{(m)}(t) = f_j^{(m)}(\mathbf{x}^{(m)}(t)) = \delta x_j^{(m)}(t)\left(u_j^{(m)}(t) - \bar{u}^{(m)}(t)\right), \quad \forall m \in \mathcal{M}, \forall j \in \mathcal{J}, \forall t, \tag{5.7}$$

where δ refers to the positive learning rate of the population that controls the speed at which workers adapt their strategies. For example, in a network with communication bottlenecks [32] or negative network effects [26], the learning rate tends to be slower as the worker requires more time to collect the information required to change its decision.

The replicator dynamics is a series of ordinary differential equations (ODEs) with the initial condition $\mathbf{x}^{(m)}(0) \in \mathbb{X}$ [200]. Specifically, based on the replicator dynamics, workers in population m can adapt their strategy, i.e., switch from one cluster to another if their utilities are lower than the expected utility. The *evolutionary equilibrium* is a fixed point in (5.7) that is reached in a particular t when $\dot{x}_j^{(m)}(t) = 0, \forall m \in \mathcal{M}, \forall j \in \mathcal{J}$. In other words, at the evolutionary equilibrium, workers from all clusters derive an identical payoff such that there is no longer a need to deviate from their prevailing clusters.

In a dynamic system, it is of paramount importance that the equilibrium is stable and unique. In terms of stability, an evolutionary equilibrium remains to be $\dot{x}_j^{(m)}(t) = 0$ for all time periods after the equilibrium is first reached. In terms of uniqueness, the same evolutionary equilibrium is reached regardless of the initial conditions. In Section 5.3.3, we prove the existence, uniqueness, and stability of the solution to (5.7).

5.3.3 Existence, Uniqueness, and Stability of the Evolutionary Equilibrium

In this section, we first prove the boundedness of (5.7) in Lemma 5.1.

Lemma 5.1. *The first order derivatives of* $f_j^{(m)}(\mathbf{x}^{(m)}(t))$ *with respect to* $x_v^{(m)}(t)$ *is bounded for all* $v \in J$.

Proof. For ease of presentation, we omit the notations of t and (m) in this proof. The first order derivative of $f_j(\mathbf{x})$ with respect to x_v, where $v \in \mathcal{J}$, is given by

$$\frac{\mathrm{d}f_j(\mathbf{x})}{\mathrm{d}x_v} = \delta \left[\frac{\mathrm{d}x_j}{\mathrm{d}x_v} (u_j - \bar{u}) + x_j \left(\frac{\mathrm{d}u_j}{\mathrm{d}x_v} - \frac{\mathrm{d}\bar{u}}{\mathrm{d}x_v} \right) \right]. \tag{5.8}$$

For ease of notation, denote $A(\mathbf{x}_j) = \sum_{m=1}^{M} x_j D_m$. Next, we derive $\frac{\mathrm{d}u_j}{\mathrm{d}x_v}$ as follows:

$$\frac{\mathrm{d}u_j}{\mathrm{d}x_v} = \alpha_j \left(\frac{\frac{\mathrm{d}x_j}{\mathrm{d}x_v} D_m}{A(\mathbf{x}_j)} - \frac{x_j D_m^2}{A^2(\mathbf{x}_j)} \right) - 2\zeta_j \left(\sum_{m \in \mathcal{M}} x_j \right). \tag{5.9}$$

It follows that $\left| \frac{\mathrm{d}u_j}{\mathrm{d}x_v} \right|$ and thus $\left| \frac{\mathrm{d}\bar{u}}{\mathrm{d}x_v} \right|$ are clearly bounded $\forall v \in \mathcal{J}$. Therefore, this represents that $\left| \frac{\mathrm{d}f_j(\mathbf{x})}{\mathrm{d}x_v} \right|$ is bounded. This proof also applies to all M populations and T time periods. □

Theorem 5.1. *For any initial condition* $\mathbf{x}^{(m)}(0) \in \mathbb{X}$, *there exists a unique evolutionary equilibrium to the dynamics defined in (5.7).*

Proof. From Lemma 1, we have proven that the replicator dynamics $f_j^{(m)}(\mathbf{x}^{(m)}(t))$ is bounded and continuously differentiable $\forall \mathbf{x}^{(m)}(t) \in \mathbb{X}, \forall m \in \mathcal{M}, \forall j \in \mathcal{J}, \forall t$. Therefore, the maximum absolute value of its partial derivative given in (8) is a Lipschitz constant. According to the Mean Value Theorem, there exists a constant c between $x_1^{(m)}(t)$ and $x_2^{(m)}(t)$ such that $\frac{\left| f_j^{(m)}(x_1^{(m)}(t)) - f_j^{(m)}(x_2^{(m)}(t)) \right|}{(x_1^{(m)}(t) - x_2^{(m)}(t))} = \frac{\mathrm{d}f_j(c)}{\mathrm{d}x_v}$. Therefore, we can define the relation

$$\begin{aligned} \left| f_j^{(m)}(x_1^{(m)}(t)) - f_j^{(m)}(x_2^{(m)}(t)) \right| \leq \Gamma \left| x_1^{(m)}(t) - x_2^{(m)}(t) \right|, \\ \forall (x_1^{(m)}, x_2^{(m)}) \in \mathbb{X}, \forall m \in \mathcal{M}, \forall t. \end{aligned} \tag{5.10}$$

where $\Gamma = \max \left\{ \left| \frac{\mathrm{d}f_j(c)}{\mathrm{d}x_v} \right| \right\}$. Following the Lipschitz condition [201], this implies that the replicator dynamics, i.e., an initial value problem with $\mathbf{x}^{(m)}(0) \in \mathbb{X}$, in (7) has a unique solution $x_j^{(m)\star} \in \mathbb{X}$. □

Next, we prove the stability of the evolutionary equilibrium in the following theorem.

Theorem 5.2. *For any initial condition* $\mathbf{x}^{(m)}(0) \in \mathbb{X}$*, the evolutionary equilibrium to the dynamics defined in (5.7) is stable.*

Proof. We define the Lyapunov function

$$G(\mathbf{x}^{(m)}(t)) = \left(\sum_{m=1}^{M} \sum_{j=1}^{J} x_j^{(m)}(t) \right)^2, \tag{5.11}$$

which is positive definite since:

$$G(\mathbf{x}^{(m)}(t)) \begin{cases} = 0 & \text{if } \mathbf{x}(t) = \mathbf{0} \\ > 0 & \text{otherwise.} \end{cases} \tag{5.12}$$

Taking the first-order derivative with of $G(\mathbf{x}^{(m)}(t))$ with respect to t,

$$\frac{\mathrm{d}G(\mathbf{x}^{(m)}(t))}{\mathrm{d}t} = 2 \left(\sum_{m=1}^{M} \sum_{j=1}^{J} x_j^{(m)}(t) \right) \left(\sum_{m=1}^{M} \sum_{j=1}^{J} \dot{x}_j^{(m)}(t) \right). \tag{5.13}$$

Note that at any point of time $\sum_{m=1}^{M} \sum_{j=1}^{J} x_j^{(m)}(t) = M$. Thus, the replicator dynamics have to equate to zero for this to hold, i.e., the net movements and strategy adaptations across clusters are zeroed in order for the population to remain constant. Specifically,

$$\sum_{m=1}^{M} \sum_{j=1}^{J} \dot{x}_j^{(m)}(t) = 0, \forall t. \tag{5.14}$$

Therefore, (5.14) ensures that $\frac{\mathrm{d}G(\mathbf{x}^{(m)}(t))}{\mathrm{d}t} = 0$, which satisfies the Lyapunov conditions required for stability, as defined in the Lyapunov's second method for stability [202]. □

As such, we have proven the uniqueness and stability of the evolutionary equilibrium.

Next, we discuss the procedures to derive the equilibrium cluster data coverage based on the replicator dynamics in (5.7). In contrast to the population evolution

algorithm [198] which involves the intervention of a centralized controller, e.g., in disseminating information of potential payoffs that can be derived from joining a particular cluster, we consider the implementation of a decentralized cluster selection algorithm in Algorithm 5.1.

At the initialization phase, workers in each population m are randomly assigned to j clusters, where $m \in \mathcal{M},\ j \in \mathcal{J}$. At each time period t, the workers compute their utilities and the average utility of workers in the population. Note that in practice, the workers may not have the complete information of all workers belonging to the same population in a large network. As such, its knowledge of the average utility in the population is based on the worker's "best guess", i.e., the expected average utility. This procedure is simply a comparison between (a) the worker's own utility from joining a particular cluster j, i.e., $u_j^{(m)}(t)$ and (b) the expected average utilities of other workers from the same population which has chosen to join other clusters, i.e., $E(\bar{u}^{(m)}(t))$. Thereafter, the evolution of the population state can be derived following the replicator dynamics.

The output of Algorithm 5.1 is the population state that is observed after t_{max} iterations. Then, we are eventually able to derive the data coverages of the cluster head, i.e., the proportion of data across the network that each cluster head can cover, as follows:

$$D_j = \sum_{m=1}^{M} x_j^{(m)}(t_{max}) D_m. \tag{5.15}$$

5.4 Deep Learning Based Auction for Valuation of Cluster Head

5.4.1 Auction Formulation

Based on the cluster formations from the evolutionary game, we are able to derive the data coverage D_j of the cluster head $j \in \mathcal{J}$.

As each cluster head can only offer its services to a single model owner, i.e., the workers' participation in the FL model training, the model owners need to compete for the services of the cluster heads. Each model owner $l \in \mathcal{L}$ has a different

Algorithm 5.1 Cluster Selection for HFL

Input: Worker and cluster characteristics
Initialization: Workers in population m each assigned to a random cluster
while $t < t_{max}$ **do**
  for $m \in \mathcal{M}$ **do**
    Payoff Computation
    for $j \in \mathcal{J}$ **do**
      Derive $u_j^{(m)}(t) = p_j^{(m)}(t) - c_j^{(m)}(t)$
    end for
    Compute $E(\bar{u}^{(m)}(t)) = E(\sum_{j=1}^{J} x_j^{(m)} u_j^{(m)}(t))$
    Cluster Selection
    for $j \in \mathcal{J}$ **do**
      Derive $\dot{x}_j^{(m)}(t) = \delta x_j^{(m)}(t) \left(u_j^{(m)}(t) - \bar{u}^{(m)}(t) \right)$ and $x_j^{(m)}(t)$
    end for
  end for
end while
for $j \in \mathcal{J}$ **do**
  Compute $D_j = \sum_{m=1}^{M} x_j^{(m)}(t_{max}) D_m$
end for

preference for the accuracy of their models, e.g., applications for accident warning and prediction [161] require higher accuracy than the route planning and navigation systems. Following the work in [203], the FL model accuracy A_l of model owner l, can be expressed as a power law function that is denoted as follows:

$$A_l(\mu_l) = \sigma - v\mu_l^{-r}, \tag{5.16}$$

where σ, v, and r are calibrable parameters depending on the model to be trained. μ_l is the data coverage required by model owner l to achieve its required model accuracy, σ is the upper bound of the accuracy that can be derived from historical data, whereas v and r are the fixed parameters of the function.

In general, when the requirement for data coverage of the model owner μ_l is larger, the model owner has more incentive to pay a higher price for the services of the cluster heads which have more data coverage. In contrast, a model owner that already has some pre-existing training data d_l will have less incentive to bid for the services of a cluster head. Therefore, the valuation b_l of model owner l for the services offered by the cluster heads can be expressed as $b_l = \mu_l - d_l$.

In order to maximize the revenue of the cluster heads and to ensure that the services from the cluster heads are allocated to the model owners that value them most,

we model the allocation problem as multiple rounds of single-item auctions. In this auction, the cluster heads are the sellers, i.e., auctioneers, while the model owners are the buyers, i.e., bidders. The cluster head with the highest amount of data coverage is the first to auction its services to the model owners. All model owners submit their bids to compete for the service. Then, the cluster head collects the bid profile $(b_1, \ldots, b_l, \ldots, b_L)$ to decide on the winning model owner $l^\star$ and the corresponding payment price $\theta_{l^\star}$. After each round of auction, the winning model owner has higher data coverage, i.e., higher d_l. Thus, its valuation in the next round of auction naturally has to be updated and decreases. The auction ends when all cluster heads have been allocated to the model owners. Note that the model owners may participate in more than one round of auction to fulfill their data coverage requirement, e.g., if the data coverage that a single cluster has is insufficient to fulfill its needs.

Accordingly, the utility of the model owner in each round of the auction is as follows

$$u_l = \begin{cases} b_l - \theta_{l^\star} & \text{if the model owner wins the bid,} \\ 0 & \text{otherwise.} \end{cases} \tag{5.17}$$

An optimal auction [204] has two characteristics:

i *Individual Rationality (IR):* By participating in the auction, the model owners receive non-negative payoff, i.e., $u_l \geq 0$.

ii *Incentive Compatibility (IC):* There is no incentive for the model owners to submit bids other than their true valuations, i.e., the bidders always bid truthfully.

In each round of the auction, in order to determine the payment price $\theta_{l^\star}$ of the winning model owner, traditional auction schemes such as the first-price auction and second-price auction (SPA) may be adopted. However, each of these traditional auction schemes has its own drawbacks.

The traditional first-price auction, in which the highest bidder pays the exact bid it submits, maximizes the revenue of the seller but does not ensure that the bidders submit their true valuations. On the other hand, the SPA, in which the highest bidder pays the price offered by the second highest bidder, ensures that the bidders

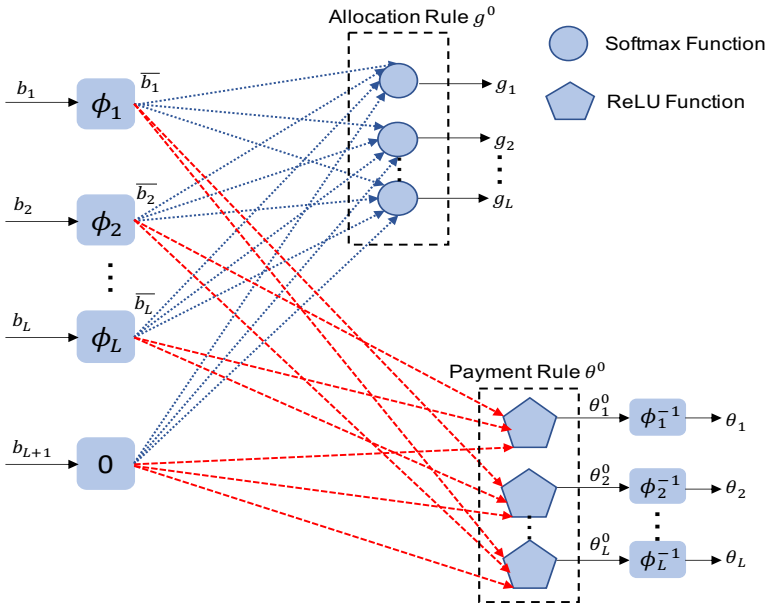

FIGURE 5.2: Neural Network Architecture for the Optimal Auction.

submit their true valuations, i.e., ensures IC, but does not maximize the revenue of the seller. Therefore, in order to ensure that both conditions of truthfulness and revenue maximization of the seller are satisfied, we design an optimal auction using the Deep Learning approach with reference to the study in [205].

5.4.2 Deep Learning Based Auction for Valuation of Cluster Heads

In this section, we illustrate the neural network architecture for the design of an optimal auction. Following the procedure in [205], we describe the neural network architecture (Fig. 5.2) which renders the design of an optimal auction. Then, we elaborate on the proposed implementation of multiple round single-item auctions for the valuation of the services of cluster heads.

By adopting the SPA scheme to determine the payment price of the winning model owner, the revenue of the cluster head is not maximized, especially when the bid of the second highest bidder is low. Thus, in order to maximize the revenue of the cluster heads, the monotonically increasing functions are applied to the bids of the model owners to transform the bids into transformed bids, which are used to determine the allocation and the corresponding payment of the model owners in the network. The input bids and the transformed bids of model owner l are

denoted as b_l and $\bar{b}_l$ respectively. The transform function for the bids submitted by model owner l is denoted as ϕ_l. In order to determine the allocation and conditional payment of the model owners, the SPA with zero reserve price (SPA-0) is applied to transform the bids. The reserve price is the minimum price that the cluster head requires to offer its service. The SPA-0 allocation rule and the SPA-0 payment rule of the model owner l are represented by $g_l^0(\bar{\boldsymbol{b}})$ and $\theta_l^0(\bar{\boldsymbol{b}})$, respectively, where $\bar{\boldsymbol{b}}$ is the vector of the transformed bids. The SPA-0 allocation rule $g_l^0(\bar{\boldsymbol{b}})$ determines the winning model owner which has the highest bid if the bid is greater than zero. The SPA-0 payment rule $\theta_l^0(\bar{\boldsymbol{b}})$ determines the conditional payment price θ_l of model owner l by applying the inverse transform function which is represented by ϕ_l^{-1}.

Theorem 5.3. *For any set of strictly monotonically increasing function* $\{\phi_1, \ldots, \phi_L\}$, *an auction which is defined by the allocation rule* $g_l = g_l^0 \circ \phi_l$ *and the payment rule* $\theta_l = \phi_l^{-1} \circ \theta_l^0 \circ \phi_l$ *satisfies the properties of IC and IR, where* g^0 *and* θ^0 *represent the SPA-0 allocation rule and the SPA-0 payment rule, respectively [205].*

Based on the theorem, for any choices of strictly monotonically increasing transform functions, the proposed auction with the allocation rule g_l and conditional payment rule θ_l satisfy the characteristics of the optimal auction, i.e., IR and IC. Therefore, the monotone transform functions are used in the neural network to ensure the IR and IC properties of the auction. In addition to this, the cluster heads want to maximize their revenues. Based on the allocation and the conditional payment rules, the revenue of the cluster head is determined. In particular, the revenue of the cluster head is equivalent to the payment price of the winning model owner. Thus, the objective of the cluster heads is to maximize their individual revenues while fulfilling the properties of IR and IC of the optimal auction. In order to do so, the neural network architecture learns the appropriate transform functions for the optimal auction to minimize the loss function which is defined as the expectation of the negated revenue of the cluster head. The minimization of the loss function is equivalent to the maximization of the revenue of the cluster head. With this, the optimal auction design based on the neural network architecture maximizes the revenues of the cluster heads while satisfying the necessary and sufficient conditions for IC and IR.

The algorithm for the implementation of the optimal auction based on the neural network architecture is illustrated in Algorithm 5.2.

Algorithm 5.2 Algorithm for Deep-Learning Based Optimal Auction.

Require: Set of cluster heads $\mathcal{J} = \{1, \ldots, j, \ldots, J\}$, bids of model owners $b^i = (b_1^i, \ldots, b_l^i, \ldots, b_L^i)$
Ensure: Revenue of the cluster heads
while $\mathcal{J} \neq \emptyset$ **do**
  Identify the cluster head the highest data coverage, D_j
  Initialization: $\mathbf{w} = [w_{qs}^l] \in \mathbb{R}_+^{I \times QS}$, $\boldsymbol{\beta} = [\beta_{qs}^l] \in \mathbb{R}^{I \times QS}$
  Deep-Learning Based Optimal Auction:
  while Loss function $\hat{R}(\mathbf{w}, \boldsymbol{\beta})$ is not minimized **do**
    Compute the transformed bids $\bar{b}_l^i = \theta_l(b_l^i) = \min_{q \in \mathcal{Q}} \max_{s \in \mathcal{S}} (w_{qs}^l b_l + \beta_{qs}^l)$
    Compute the allocation probabilities $g_l(\bar{\mathbf{b}}) = softmax_l(\bar{b}_1, \ldots, \bar{b}_l, \ldots, \bar{b}_{L+1}; \gamma)$
    Compute the SPA-0 payments $\theta_l^0(\bar{\mathbf{b}}) = ReLU(\max_{s \neq l} \bar{b}_s)$
    Compute the conditional payments $\theta_l = \phi_l^{-1}(\theta_l^0(\bar{\mathbf{b}}))$
    Compute the loss function $\hat{R}(\mathbf{w}, \boldsymbol{\beta})$
    Update the network parameters $\mathbf{w}$ and $\boldsymbol{\beta}$ using the SGD solver
  end while
  Update the data coverage of the winning model owner $d_l^{new} = d_l^{old} + D_j$
  Update the valuation of the winning model owner $b_l = \mu_l - d_l$
  Remove the cluster head from set $\mathcal{J}$
end while
  return The revenue gain by the cluster heads

In the following, we discuss the three important functions in the neural network architecture

i the monotone transform function ϕ_l,

ii the allocation rule g_l,

iii the conditional payment rule θ_l.

5.4.3 Monotone Transform Functions

In the auction, the valuation, i.e., bid b_l of each model owner l is the input to the transform function ϕ_l. The transform function maps the input to its transformed bid, $\bar{b}_l = \phi_l(b_l)$. Each transform function ϕ_l is modelled as a two-layer feed forward network that consists of the min and max operators over several linear functions, as shown in Fig. 5.3. There are Q groups of S linear functions $h_{qs}(b_l) = w_{qs}^l b_l + \beta_{qs}^l$, $\mathcal{Q} = \{1, \ldots, q, \ldots, Q\}$, $\mathcal{S} = \{1, \ldots, s, \ldots, S\}$, $w_{qs}^l \in \mathbb{R}^+$ and $\beta_{qs}^l \in \mathbb{R}$ are the

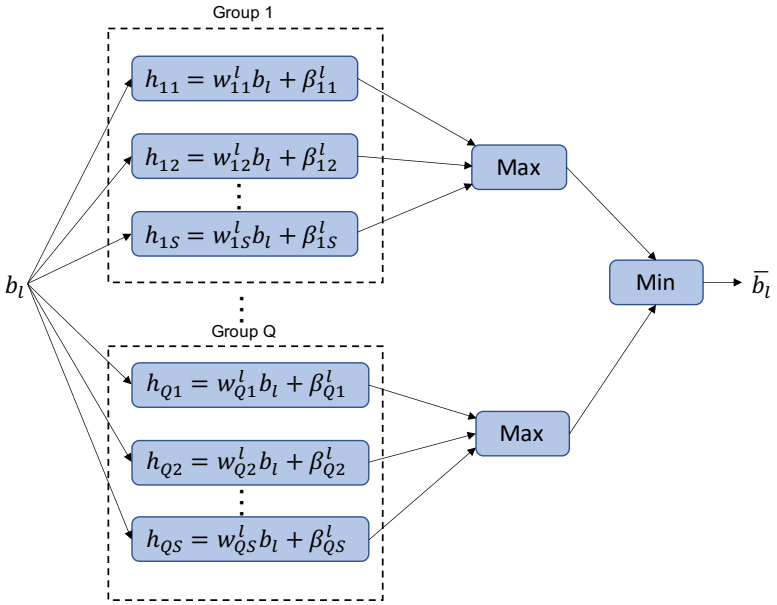

FIGURE 5.3: Monotone Transform Functions.

positive weight and bias respectively. With these linear functions, the transform function ϕ_l of each model owner l is defined as follows:

$$\phi_l(b_l) = \min_{q \in \mathcal{Q}} \max_{s \in \mathcal{S}} (w_{qs}^l b_l + \beta_{qs}^l). \tag{5.18}$$

Based on the parameters for the forward transform function ϕ_l, the inverse function ϕ_l^{-1} can be derived as follows:

$$\phi_l^{-1}(y) = \max_{q \in \mathcal{Q}} \min_{s \in \mathcal{S}} (w_{qs}^l)^{-1} (y - \beta_{qs}^l). \tag{5.19}$$

5.4.4 Allocation Rule

The allocation rule in the neural network architecture is based on the SPA-0 allocation rule. In particular, the data coverage D_j of the cluster head j is allocated to the model owner with the highest transformed bid if the transformed bid is more than zero. Otherwise, the cluster head does not sell its service to any model owner. In order to model the competition among the model owners, we use a *softmax* function on the vector of transformed bids $\bar{\mathbf{b}} = (\bar{b}_1, \ldots, \bar{b}_l, \ldots, \bar{b}_L)$ and the additional dummy input $\bar{b}_{L+1} = 0$ in the allocation network. The output of the *softmax* function is a vector of allocation probabilities, which is represented by

$\mathbf{g} = (g_1, \ldots, g_l, \ldots, g_L)$. The *softmax* function used in the neural network architecture is defined as follows:

$$\begin{aligned} g_l(\bar{\mathbf{b}}) &= softmax_l(\bar{b}_1, \ldots, \bar{b}_l, \ldots, \bar{b}_{L+1}; \gamma) \\ &= \frac{e^{\gamma \bar{b}_l}}{\sum_{l=1}^{L+1} e^{\gamma \bar{b}_l}}, \ \gamma > 0, \forall l \in \mathcal{L}. \end{aligned} \tag{5.20}$$

The parameter γ in the *softmax* function measures the quality of the approximation where the higher the γ, the more accurate the approximation of the function. However, a better quality of approximation results in a less smooth allocation function.

5.4.5 Conditional Payment Rule

The conditional payment rule determines the price θ_l that needs to be paid by the winning model owner l. The conditional payment rule is carried out in two steps. Firstly, the SPA-0 payment θ_l^0 for each model owner l is calculated. Specifically, the SPA-0 payment θ_l^0 is the maximum of the transformed bids of other model owners and zero which is determined by using the *ReLU* activation unit function as follows:

$$\theta_l^0(\bar{\mathbf{b}}) = ReLU(\max_{s \neq l} \bar{b}_s), \ \forall l \in \mathcal{L}. \tag{5.21}$$

The $ReLU(x) = \max(x, 0)$ activation function guarantees that the SPA-0 payment of each model owner is non-negative. Secondly, based on the SPA-0 payment θ_l^0, the conditional payment θ_l of model owner l is calculated as follows:

$$\theta_l = \phi_l^{-1}(\theta_l^0(\bar{\mathbf{b}})), \tag{5.22}$$

where the inverse transform function from (5.19) is applied to the SPA-0 payment of the model owner l.

5.4.6 Neural Network Training

The aim of the neural network is to optimize the weights and biases of the linear functions in the neural network such that the loss function is minimized. In the

neural network, the loss function is defined as the expectation of the negated revenue of the cluster head. The loss function of the neural network is formulated based on the inputs, i.e., the training dataset and the outputs, i.e., the allocation probabilities and the conditional payments of the model owners. The training dataset of the neural network consists of the bidders' valuation profiles of which the bidders' valuation profile i is denoted as $\mathbf{b}^i = (b_1^i, \ldots, b_L^i)$, $\mathcal{I} = \{1, \ldots, i, \ldots, I\}$ where I is the size of the training dataset. b_l^i is the valuation of model owner l for the data coverage of cluster head is drawn from a valuation distribution function $f_B(b)$. Since the valuation b_l^i of the model owner l depends on the data coverage requirement μ_l and the current amount of data coverage d_l of the model owner, i.e., $b_l^i = \mu_l^i - d_l^i$, the distribution function $f_B(b)$ can be determined based on the distribution of the data coverage requirement, which is represented by $f\mu(\mu)$. In this chapter, we assume that the data coverage requirement of the model owners follows a uniform distribution, i.e., $\mu \sim U[\mu_{min}, \mu_{max}]$.

The parameters of the monotone transform functions, i.e., weights w_{qs}^l and biases β_{qs}^l are the entries of matrices $\mathbf{w}$ and $\boldsymbol{\beta}$. The matrices are needed to determine the allocation probability and conditional payment of model owner l, which are represented by $g_l^{(\mathbf{w},\boldsymbol{\beta})}$ and $\theta_l^{(\mathbf{w},\boldsymbol{\beta})}$ respectively.

The objective of the training is to find the optimal weight $\mathbf{w}^*$ and bias $\boldsymbol{\beta}^*$ matrices that minimize the loss function of the neural network, i.e., the expectation of the negated revenue of the cluster head j. Specifically, the approximation of the loss function, $\hat{R}$ is defined as follows:

$$\hat{R}(\mathbf{w}, \boldsymbol{\beta}) = -\frac{1}{I} \sum_{l=1}^{I} g_l^{(\mathbf{w},\boldsymbol{\beta})}(\mathbf{b}^i) \theta_l^{(\mathbf{w},\boldsymbol{\beta})}(\mathbf{b}^i). \tag{5.23}$$

For the optimization of the loss function $\hat{R}(\mathbf{w}, \boldsymbol{\beta})$ over the parameters $(\mathbf{w}, \boldsymbol{\beta})$, a stochastic gradient descent (SGD) solver is used.

5.5 Performance Evaluation

In this section, we present the performance evaluation of the evolutionary game based cluster formation and deep learning based auction for the valuation of cluster

TABLE 5.1: Simulation Parameters.

Parameter	**Values**
Total number of workers in the network N	90
Number of data samples of population m D_m	$[2400, 4800]$
Reward pool offered in cluster α_j	$[100, 300]$
Fixed rewards offered in cluster R_j	80
Congestion coefficient ζ_j	$[10, 20]$
Computation cost c^{cmp}	0.1
Rate of strategy adaptation δ	0.001
Total Number of Model Owners, L	10
Size of Training Dataset, I	1000
Number of Groups for Linear Function Q	5
Linear Functions in Each Group S	10
Learning Rate	0.001
Quality of Approximation	1000, 2000
Range of the model owners' data coverage requirement, μ_l	$\sim U[0.5, 0.9]$, $\sim U[0, 0.4]$

data coverage. Unless otherwise stated, the simulation parameters are as shown in Table 5.1 [203, 206]. Note that we use the terms "cluster" and "cluster heads" interchangeably.

5.5.1 Lower-level Evolutionary Game

For the first part of our simulations, we analyze the lower-level evolutionary game. We consider a network which consists of 90 workers. The workers have different data quantities that follow a uniform distribution. Using the k-means algorithm, we derive 3 populations[5] of 30 workers each based on the data quantities that they possess. In the first population $m = 1$, each worker has 80 data samples. In the

[5]For ease of exposition, we use only three populations for comparison. Our model can easily be extended to multiple populations.

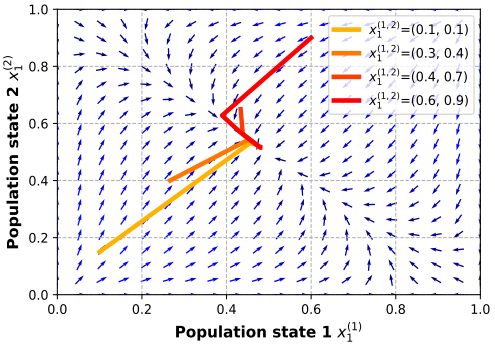

FIGURE 5.4: Phase plane of the replicator dynamics.

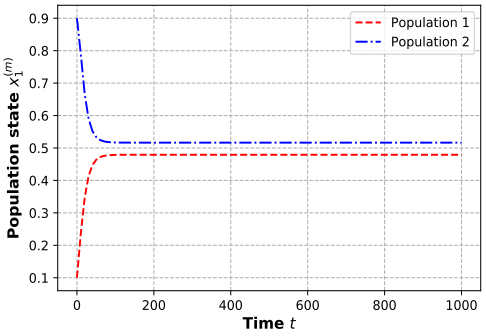

FIGURE 5.5: Evolutionary equilibrium of population states for cluster 1.

second population $m = 2$, each worker has 100 data samples. In the third population $m = 3$, each worker has 160 samples. Without loss of generality, the data samples that each worker owns are assumed to be characterized by the populations that they belong to. Thus, the populations are arranged in ascending order based on the data samples each worker has.

Besides, there exist 3 cluster heads in the network with each offering different reward pools α_j, as well as congestion coefficients ζ_j. Recall from Section 5.3.2 that a higher value of α_j indicates that a cluster offers a larger reward pool for the workers to share, whereas a higher value of ζ_j represents that a cluster head has more limited communication resources. The clusters are arranged in ascending order based on the reward pool they offer, i.e., cluster 3 offers the highest reward pool to its workers.

Accordingly, in each time period, workers in the populations choose one of the clusters to join. Then, following Algorithm 5.1, the strategy adaptation is performed and evolved such that the workers evaluate their payoffs and churn to another cluster with higher payoffs with some probabilities. Eventually, the evolutionary equilibrium is achieved.

5.5.1.1 Stability and Uniqueness of the Evolutionary Equilibrium

To demonstrate the uniqueness of the evolutionary equilibrium, i.e., the solution to the replicator dynamics defined in (5.7) from Section 5.3.2, we first derive the phase

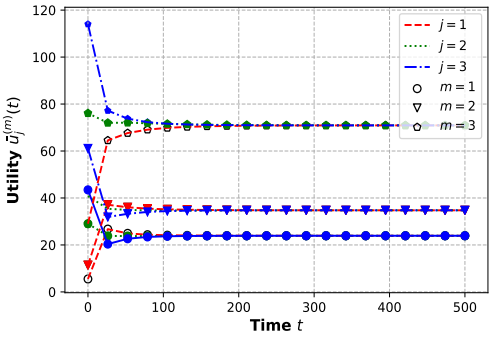

FIGURE 5.6: Evolution of population utilities.

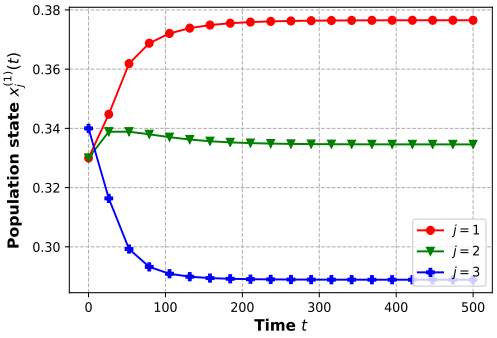

FIGURE 5.7: Evolution of population states for population 1.

plane of the replicator dynamics in Fig. 5.4. For ease of exposition, population 3 is excluded initially. As such, only the first and second populations of workers are considered to choose among the three clusters to join. Figure 5.4 shows the population states of populations 1 and 2, i.e., the proportion of workers in each population that join cluster 1. We consider varying initial conditions in Fig. 5.4 and plot the corresponding dynamics. For example, for the first condition, we have 10% of the workers from both populations choose cluster 1. Clearly, despite varying initial conditions, the evolutionary equilibrium always converges to a unique solution.

To evaluate the stability of this evolutionary equilibrium, we consider that 30% and 70% of workers in populations 1 and 2 respectively are initially allocated to cluster 1. Then, we plot the evolution of population states in Fig. 5.5. We observe that the proportion of workers from population 2 joining cluster 1 declines as more workers from population 1 join the cluster. This is due to the division of rewards and congestion effects as more workers join the cluster. Eventually, the evolutionary equilibrium is reached and the population states no longer change.

Next, we consider the situation in which there are three populations and three clusters. We set the initial conditions such that a third of the workers from each population is assigned to each cluster initially. Then, we plot the evolution of utilities in Fig. 5.6. Specifically, within each population, we plot the utilities derived by workers which have chosen each of the three clusters. We observe that for each population, the utilities derived from choosing the varying clusters eventually converge with time. This implies that an evolutionary equilibrium is

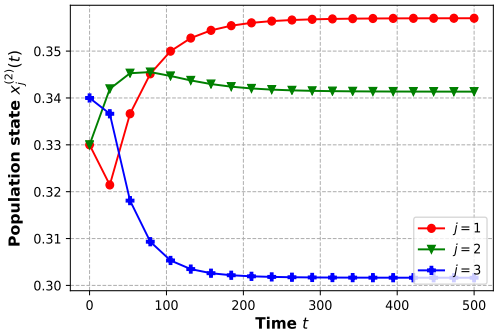

FIGURE 5.8: Evolution of population states for population 2.

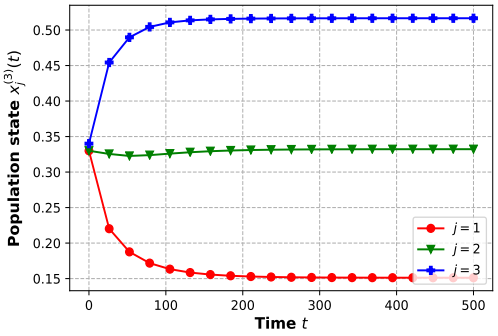

FIGURE 5.9: Evolution of population states for population 3.

reached whereby at the equilibrium, the workers no longer have the incentive to adapt their cluster selection strategies. Moreover, at the equilibrium, workers which belong to population 3 derive the greatest utilities, given that they are compensated for having larger data shares in the clusters.

In Figs. 5.7-5.9, we plot the evolution of population states of each population respectively. We observe that cluster 3, which offers the highest reward pool for distribution across workers, has the largest proportion of workers from population 3. The reason is that the workers from population 3 have the largest number of data samples. Hence, they can have the largest shares of the large reward pool if they join cluster 3. In contrast, the lowest proportion of workers from population 1 joins cluster 3 because they have the lowest reward shares. However, there is an upper limit to how many workers can join cluster 3. Even though cluster 3 offers the highest reward pool, the distribution of rewards and congestion effects set in eventually when more workers have joined the cluster. Thus, workers of population 3 may also join clusters 1 and 2 as well when this occurs.

5.5.1.2 Evolutionary Equilibrium Under Varying Parameters and Conditions

In this section, we vary the simulation parameters to study the evolutionary equilibrium under varying conditions. In Fig. 5.10, we vary the learning rate of the population, which controls the speed of strategy adaptation. Naturally, when the learning rate is low, the evolutionary equilibrium can only be reached after a longer

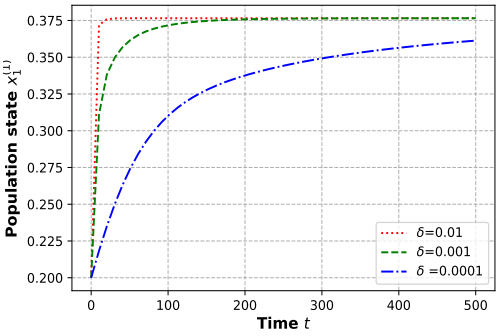

FIGURE 5.10: Evolutionary dynamics under different learning rates.

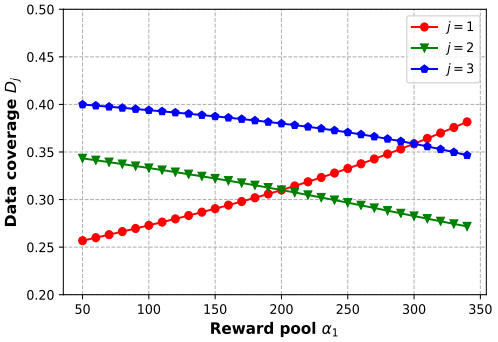

FIGURE 5.11: Data coverage vs. varying fixed rewards in cluster 1.

time period. However, we can observe that the stability of the evolutionary equilibrium is not compromised. The speed of convergence depends on how fast the workers can observe and adapt their strategies, e.g., they have more accurate information about the system.

In Fig. 5.11, we vary the reward pool offered by cluster 1 between $[50, 350]$ while keeping the reward pool for clusters 2 and 3 constant at 200 and 300 respectively. Then, we plot the changes in data coverage of each cluster, i.e., how much data coverage a cluster has as a result of worker contribution. Naturally, the data coverage of cluster 1 increases with an increment of the reward pool. The reason is that the cluster is more attractive to workers as shareable rewards increase. We further note that at the points $\alpha_1 = 200$ and $\alpha_1 = 300$, the data coverage for cluster 1 is identical to those of clusters 2 and 3 respectively. The reason is that at these points, cluster 1 is identical to the corresponding clusters and thus, workers are indifferent between choosing the cluster to join.

In Fig. 5.12, we vary the congestion coefficient of cluster 1 between $[2, 18]$ while keeping the other clusters' constant. Then, we plot the changes in data coverage of each cluster. We observe that as the congestion coefficient increases, the cluster has lower data coverage. Instead, the workers that used to join cluster 1 adapt their strategies and churn to clusters 2 and 3. This is given that with a large congestion coefficient, the cluster head has more limited communication resources and can no longer support as many workers without them having to incur higher communication costs, e.g., due to device interference.

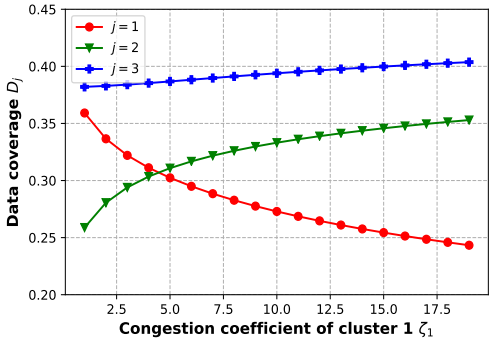

FIGURE 5.12: Data coverage vs. varying congestion coefficient in cluster 1.

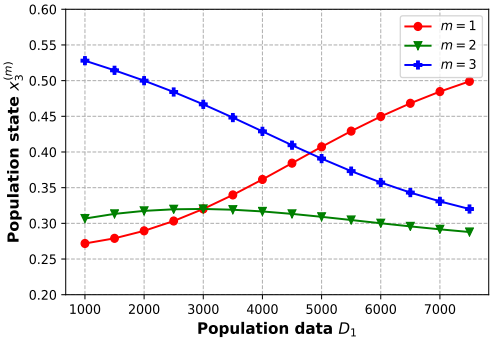

FIGURE 5.13: Population states in cluster 3 vs. varying population data for population 1.

In Fig. 5.13, we vary the data quantities of workers in population 1 while keeping the other populations constant. Then, we plot the population states of all populations with respect to participation in cluster 3. Specifically, the figure shows the proportion of workers from each population which has joined cluster 3 as the data quantities of population 1 vary. Clearly, as the data owned by workers in population 1 increases, more workers from population 1 are able to join cluster 3, which is the cluster with the largest reward pool as mentioned in Section 5.5.1.1. The reason is that with more data, the workers in the population can gain a larger proportion of the pooled rewards, relative to that of workers from other populations.

5.5.2 Upper-Level Deep Learning Based Auction

In this section, we perform simulations to evaluate the performance of the Deep-Learning based auction. For comparison, the classic SPA is chosen as the baseline scheme. The TensorFlow Deep Learning library is used to implement the optimal auction design. We consider a network with the formations of 3 clusters and 10 model owners. We first evaluate the performance of the Deep-Learning based auction against the traditional SPA. Then, we proceed to study the impacts of (a) data coverage of the cluster heads, (b) model owners with varying distribution for data coverage requirement, and (c) different quality of approximation, on the revenues of the cluster heads.

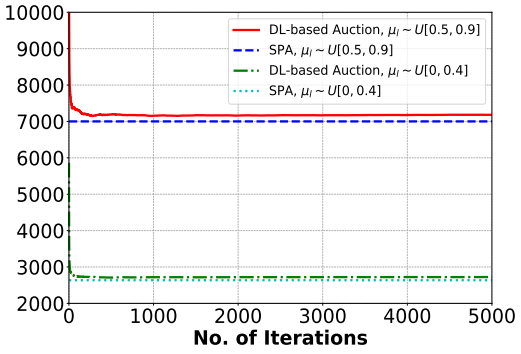

FIGURE 5.14: Revenue of cluster head 1 under different distribution of model owners.

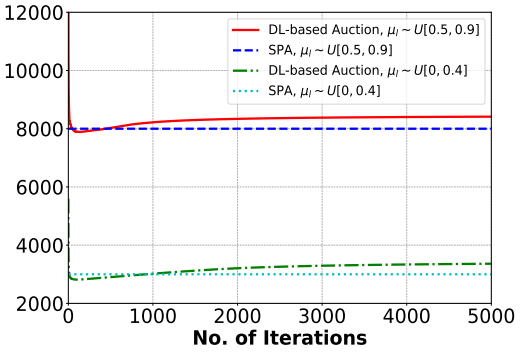

FIGURE 5.15: Revenue of cluster head 2 under different distribution of model owners.

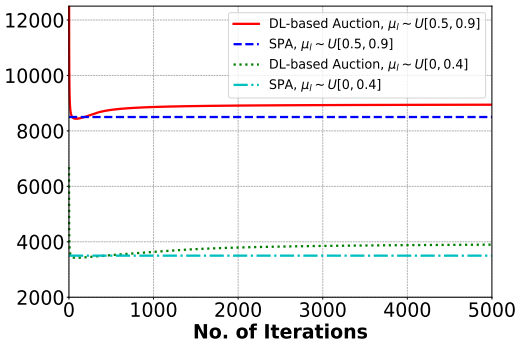

FIGURE 5.16: Revenue of cluster head 3 under different distribution of model owners.

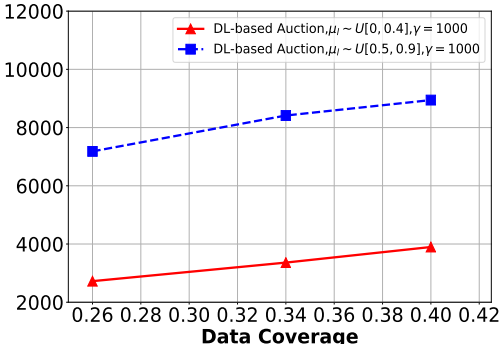

FIGURE 5.17: Revenue vs data coverage of cluster heads.

5.5.2.1 Evaluation of the Deep Learning Based Auction

From Figs. 5.14 to 5.16, we observe that the revenue of the cluster heads determined by the deep-learning based auction is consistently higher than that of the conventional SPA scheme. The reason is that the SPA scheme guarantees IC, but does not guarantee that the revenue of the seller is maximized. While preserving the properties of IC and IR of the traditional auction, the deep-learning based auction maximizes the revenue earned by the cluster heads by providing their services to the model owners that value their services the most.

We examine the impacts on the cluster heads' revenues when they are presented with model owners with data coverage requirements of different distribution ranges. Specifically, model owners can take on two distribution ranges, i.e., $\mu_l \sim U[0, 0.4]$

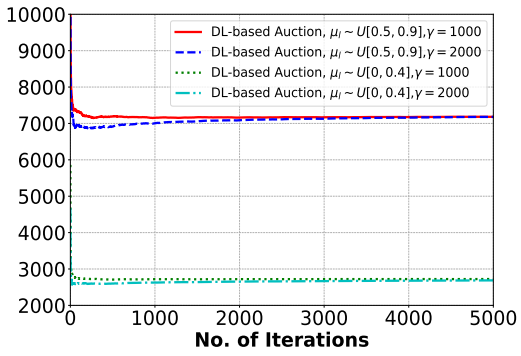

FIGURE 5.18: Revenue of cluster head 1 under different approximation qualities.

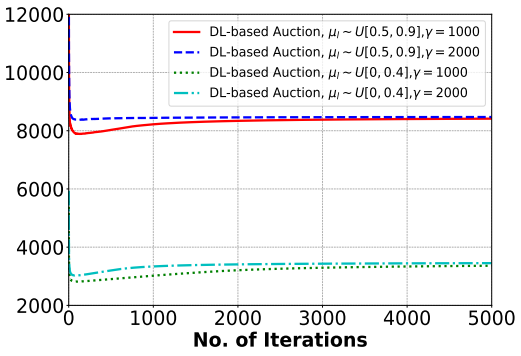

FIGURE 5.19: Revenue of cluster head 2 under different approximation qualities.

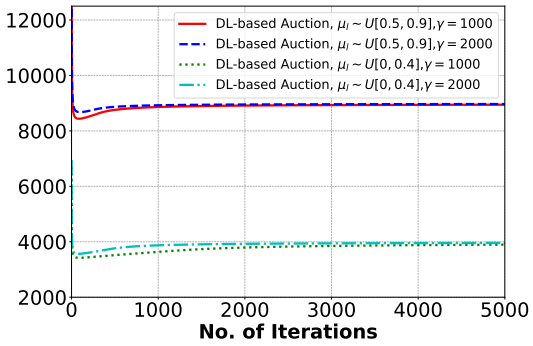

FIGURE 5.20: Revenue of cluster head 3 under different approximation qualities.

and $\mu_l \sim U[0.5, 0.9]$. In Fig. 5.14, cluster head 1 which has a total data coverage of 0.26 earns a revenue of 2724 when model owners have data coverage requirements that range between 0 and 0.4. On the other hand, when model owners have a higher range of data coverage requirements, i.e., between 0.5 and 0.9, cluster head 1 earns a higher revenue of 7182. Since the model owners with higher data coverage requirements value the cluster head more, they have more incentive to pay a higher price which results in the higher revenue earned by the cluster head. Similar trends are observed for cluster head 2 and cluster head 3 in Fig. 5.15 and Fig. 5.16 respectively.

The revenues of the cluster heads are affected by the amount of data coverage. In Fig. 5.17, we observe that when the data coverage of the cluster head is higher, the revenue earned is also higher. In particular, when the model owners have high requirement for data coverage, i.e., $\mu_l \sim U[0.5, 0.9]$, cluster head 3 with the highest data coverage of 0.4 earns a revenue of 8944 whereas cluster head 1 with the lowest

data coverage of 0.26 earns a revenue of 7182. This is due to the fact that the cluster head with the highest data coverage is allowed to conduct auction for its services first. Hence, it will be able to offer its service to the model owner with the highest data coverage requirement in which the model owner is willing to pay the highest price as compared to other model owners in the network. Intuitively, this serves to compensate the cluster head for the higher rewards expense it incurs.

To further evaluate the performance of the deep-learning based auction, we consider the cluster heads' revenues under the different quality of approximation, γ. The quality of approximation is used in the *softmax* function in the determination of the winning model owner. We consider two values for the quality of approximation, i.e., 1000 and 2000. We observe in Fig. 5.18 to 5.20, the revenue of the cluster head increases slightly when the quality of approximation is higher. Specifically, given the model owners with a high requirement for data coverage, i.e., $\mu_l \sim U[0.5, 0.9]$, the revenues of cluster head 3 are 8944 and 8969 when the values of γ are 1000 and 2000, respectively. This follows that with a greater value of approximation quality, the neural network is able to solve the optimization problem which maximizes the revenue of the cluster heads.

5.6 Conclusion and Chapter Discussion

In this chapter, we proposed a resource allocation and incentive mechanism design framework for HFL. We considered a two-level problem and leveraged the evolutionary game theory to derive the equilibrium solution for the cluster selection phase. Then, we introduced a deep learning based auction mechanism to value the cluster head's services. The performance evaluation shows the uniqueness and stability of the evolutionary equilibrium, as well as the revenue maximizing property of the auction mechanism.

Following our discussions in Chapters 3 and 4, this chapter differs in that there is no monopoly for the FL model owner. In other words, there is competition among the FL model owners. To model the competition, we have utilized an auction mechanism in this chapter. Moreover, while Chapters 3 and 4 assume the static network composition, we have considered the case of dynamic network formation in a self-organized HFL network in this chapter. As such, this challenge is solved

using the evolutionary game theory aptly. On top of that, the convergence of the algorithm is proven and shown numerically. In the following, we discuss the drawbacks of the chapter and the potential future works.

Firstly, in this chapter, we have assumed that the workers are only able to join a single model owner. The practical reason is that at each instance, the worker is only able to support one instance of model training. Moreover, given that the model owners may be competing, and that the model itself has value, it is unlikely that the workers will be allowed to join multiple instances of training. However, some workers may deviate from this assumption and join other instances of model training at separate time slots. To circumvent this shortfall, one method is to maintain the workers' reputations and memberships on a reputation blockchain [54]. In this case, workers that cheat by joining more than one instance of training will be penalized and prevented from joining other model owners.

Secondly, we have assumed that the cluster head's reward pools are static in this chapter. This assumption is not unrealistic, given that in reality, a cluster head may have fixed budget constraints. To further account for the cases in which the cluster heads may have varying budget constraints, we can explore the use of a Stackelberg differential game. As an illustration, in the lower-level game, the cluster head association strategies of the workers can still be modelled using an evolutionary game. However, in the upper-level game, a Stackelberg differential game [207] can be adopted in which the cluster heads decide an optimal reward scheme given the dynamic association strategies of the workers in the lower-level.

Thirdly, the model owners only purchase the intermediate model from the cluster heads in this chapter. In reality, the cluster heads' communication and computation resources have also been purchased. To cover this resource expense, a single item auction cannot be used. Instead, we can consider a combinatorial auction with varying valuation profiles. The combinatorial auction can be solved in two phases. Firstly, the winners' determination problem can be solved using a Graph Neural Network based approach [208]. Secondly, the pricing problem can be solved using our approach discussed in this chapter.

Fourthly, in this chapter, we have made the assumption that a worker has a comparatively small subset of data as compared to other workers. This may be the case, e.g., for IoT collected data in a large-scale network of users. The utility

gained from training the derived model is therefore not incorporated within in this chapter. However, for some cases, the utility gained from cooperatively training a better model may be significant. One such example is FL for finance, in which a coalition of financial institutions may band together to train a model. If an institution does not take part, it may not be allowed to use the model. In such a case, the utility may therefore be modeled differently, and the focus is on contributions to receive a common model, rather than an individual payment.

Finally, in a large scale HFL network, we recognize that it is difficult to track the malicious updates by the workers. For example, the model poisoning attacks by malicious workers may corrupt the inference accuracy of the derived intermediate model, and in turn, affect the global model that the FL model owner attempts to build. As such, consensus mechanisms such as the proof of verification [209] can be considered. This can be implemented in tandem with the reputation blockchain [54] that we have discussed previously.

Chapter 6

Conclusion and Future Works

In this thesis, we began with a comprehensive survey of FL enabled edge intelligence in Chapter 2. In the chapter, we highlighted the enabling role that FL has in future edge intelligence networks as a tool to facilitate and empower scalable AI at the edge. Then, we discussed the drawbacks and limitations of FL. In particular, the challenges of communication inefficiency, resource misallocation, and privacy and security intrusions are crucial issues to be solved before the scalable implementation of FL.

Given the wide scope of these pressing challenges, we began the discussion of technical works in Chapter 3 by addressing the incentive mechanism design problem in an industrial crowdsensing and collaborative AI model supported by FL. Different from existing works in the literature, we study the operational tradeoffs between service latency and information freshness in collaborative AI. Then, we utilize the contract theory to solve for the case of both the distinct and continuous worker types.

Recognizing the limitations of vanilla contract optimization, we formulate a multi-dimensional contract and matching optimization framework in Chapter 4. Specifically, we consider a scenario of UAV and AI empowered edge intelligence networks. The UAVs are characterized by multiple sources of resource heterogeneity. As such, the multi-dimensional contract is useful to account for this aspect.

However, the works presented in Chapters 3 and 4 do not capture the self-organizing and dynamic aspect of an FL network. Moreover, the works presented in the two

chapters still assume the existence of a single model owner. This is restrictive, since the FL servers may compete with other FL servers, i.e., model owners, to impose restrictive memberships on the workers in the collaborative AI process. As such, we propose a hierarchical resource allocation framework for communication efficient hierarchical FL in Chapter 5. In the lower level, we devise an evolutionary game model to derive the equilibrium composition of the clusters. In the upper level, to derive the allocation of cluster head to the model owner, as well as the optimal pricing of the services of the cluster head by the competitive model owners, we adopt a deep learning-based auction mechanism.

As a conclusion to the thesis, we discuss the challenges and research directions in deploying FL at scale to be discussed as follows.

i *Dropped participants:* The approaches discussed in Section 2.4, e.g., [101], [104], and [105], propose new algorithms for participant selection and resource allocation to address the training bottleneck and resource heterogeneity. In these approaches, the wireless connections of participants are assumed to be always available. However, in practice, participating mobile devices may go offline and can drop out from the FL system due to connectivity or energy constraints. A large number of dropped devices from the training participation can significantly degrade the performance [3], e.g., accuracy and convergence speed, of the FL system. New FL algorithms need to be robust to device drop out in the networks and anticipate the scenarios in which only a small number of participants are left connected to participate in a training round. One potential solution is that the FL model owner provides free dedicated/special connections, e.g., cellular connections, as an incentive to the participants to avoid drop out.

ii *Privacy concerns:* FL is able to protect the data privacy of each participants since the raw data of the participant is not exposed to a third-party server, with just the model parameters exchanged with the FL server. However, as specified in [210], [211], and [212], communicating the model updates during the training process can still reveal sensitive information to an adversary or a third-party. The current approaches propose security solutions such as DP, e.g., [38], [213], and [214], and collaborative training, e.g., [215] and [216]. However, the adoption of these approaches sacrifices the performance, i.e.,

model accuracy. They also require significant computation on participating mobile devices. Thus, the tradeoff between privacy guarantee and system performance has to be well balanced when implementing the FL system.

iii *Unlabeled data:* It is important to note that the approaches reviewed in the survey are proposed for supervised learning tasks. This means that the approaches assume that labels exist for all of the data in the federated network. However, in practice, the data generated in the network may be unlabeled or mislabeled [217]. This poses a big challenge to the server to find participants with appropriate data for model training. Tackling this challenge may require the challenges of scalability, heterogeneity, and privacy in the FL systems to be addressed. One possible solution is to enable mobile devices to construct their labeled data by learning the "labeled data" from each other. Emerging studies have also considered the use of semi-supervised learning inspired techniques [218].

iv *Interference among mobile devices:* The existing resource allocation approaches, e.g., [101] and [105], address the participant selection based on the resource states of their mobile devices. In fact, these mobile devices may be geographically close to each other, i.e., in the same cell. This introduces an interference issue when they update local models to the server. As such, channel allocation policies may need to be combined with the resource allocation approaches to address the interference issue. While studies in [112], [114], and [115] consider multi-access schemes and over-the-air computation, it remains to be seen if such approaches are scalable, i.e., able to support a large federation of many participants. To this end, data driven learning based solutions, e.g., federated DRL, can be considered to model the dynamic environment of mobile edge networks and make optimized decisions.

v *Comparisons with other distributed learning methods:* Following the increased scrutiny on data privacy, there has been a growing effort on developing new privacy preserving distributed learning algorithms. One study proposes *split learning* [219], which also enables collaborative ML without requiring the exchange of raw data with an external server. In split learning, each participant first trains the neural network up to a cut layer. Then, the outputs from training are transmitted to an external server that completes the other layers of training. The resultant gradients are then backpropagated up to the

cut layer, and eventually returned to the participants to complete the local training. In contrast, FL typically involves the communication of full model parameters. The authors in [220] conduct an empirical comparison between the communication efficiencies of split learning and FL. The simulation results show that split learning performs well when the model size involved is larger, or when there are more participants involved, since the participants do not have to transmit the weights to an aggregating server. However, FL is much easier to implement since the participants and FL server are running the same global model, i.e., the FL server is just in charge of aggregation and thus FL can work with one of the participants serving as the master node. As such, more research efforts can be directed towards guiding system administrators to make an informed decision as to which scenario warrants the use of either learning method.

vi *Further studies on learning convergence:* One of the essential considerations of FL is the convergence of the algorithm. FL finds weights to minimize the global model aggregation. This is actually a distributed optimization problem, and the convergence is not always guaranteed. Theoretical analysis and evaluations on the convergence bounds of the gradient descent based FL for convex and non-convex loss functions are important research directions. While existing studies have covered this topic, many of the guarantees are limited to restrictions, e.g., the convexity of the loss function.

vii *Usage of tools to quantify statistical heterogeneity:* Mobile devices typically generate and collect data in a non-IID manner across the network. Moreover, the number of data samples among the mobile devices may vary significantly. To improve the convergence of FL algorithm, the statistical heterogeneity of the data needs to be quantified. Recent works, e.g., [221], have developed tools for quantifying statistical heterogeneity through metrics such as local dissimilarity. However, these metrics cannot be easily calculated over the federated network before training begins. The importance of these metrics motivates future directions such as the development of efficient algorithms to quickly determine the level of heterogeneity in federated networks.

viii *Combined algorithms for communication reduction:* Currently, there are three common techniques of communication reduction in FL as discussed in Section 2.3. It is important to study how these techniques can be combined with

each other to improve the performance further. For example, the model compression technique can be combined with the edge server-assisted FL. The combination is able to significantly reduce the size of model updates, as well as the instances of communication with the FL server. However, the feasibility of this combination has not been explored. In addition, the tradeoff between accuracy and communication overhead for the combination technique needs to be further evaluated. In particular, for simulation results we discuss in Section 2.3, the accuracy-communication cost reduction tradeoff is difficult to manage since it varies for different settings, e.g., data distribution, quantity, number of edge servers, and number of participants.

ix *Cooperative mobile crowd ML:* In the existing approaches, mobile devices need to communicate with the server directly and this may increase the energy consumption. In fact, mobile devices nearby can be grouped in a cluster, and the model downloading/uploading between the server and the mobile devices can be facilitated by a "cluster head" that serves as a relay node [222]. The model exchange between the mobile devices and the cluster head can then be done in Device-to-Device (D2D) connections. Such a model can improve the energy efficiency significantly. Efficient coordination schemes for the cluster head can thus be designed to further improve the energy efficiency of a FL system.

x *Applications of FL:* Given the advantages of guaranteeing data privacy, FL has an increasingly important role to play in many applications, e.g., healthcare, finance and transport systems. For most current studies on FL applications, the focus mainly lies in the federated training of the learning model, with the implementation challenges neglected. For future studies on the applications of FL, besides a need to consider the aforementioned issues in the survey, i.e., communication costs, resource allocation, and privacy and security, there is also a need to consider the specific issues related to the system model in which FL will be adopted in. For example, for delay critical applications, e.g, in vehicular networks, there will be more emphasis on training efficiency and less on energy expense.

List of Publications

Journal Articles

1. **Wei Yang Bryan Lim**, Jianqiang Huang, Zehui Xiong, Jiawen Kang, Dusit Niyato, Xian-Sheng Hua, Cyril Leung, and Chunyan Miao, "Towards Federated Learning in UAV-enabled Internet of Vehicles: A multi-dimensional contract-matching approach", *IEEE Transactions on Intelligent Transportation Systems (2021)*, 22(8), pp.5140-5154.

2. **Wei Yang Bryan Lim**, Jer Shyuan Ng, Zehui Xiong, Dusit Niyato, Cyril Leung, and Chunyan Miao, "Dynamic edge association and resource allocation in self-organizing hierarchical federated learning networks", *IEEE Journal on Selected Areas in Communications (2021)*, Accepted.

3. **Wei Yang Bryan Lim**, Jer Shyuan Ng, Zehui Xiong, Jiangming Jin, Yang Zhang, Dusit Niyato, Cyril Leung, and Chunyan Miao, "Decentralized edge intelligence: A dynamic resource allocation framework for hierarchical federated learning", *IEEE Transactions on Parallel and Distributed Systems (2021)*, 33(3), pp.536-550.

4. **Wei Yang Bryan Lim**, Sahil Garg, Zehui Xiong, Yang Zhang, Dusit Niyato, Cyril Leung, and Chunyan Miao, "UAV-assisted commmunication efficient federated learning in the era of the Artificial Intelligence of Things", *IEEE Network (2021)*, Accepted.

5. **Wei Yang Bryan Lim**, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao, "Federated learning in mobile edge networks: A comprehensive survey", *IEEE Communications Surveys and Tutorials (2020)*, 22(3), pp.2031-2063.

6. **Wei Yang Bryan Lim**, Zehui Xiong, Chunyan Miao, Dusit Niyato, Q. Yang, C. Leung, and H. V. Poor, "Hierarchical incentive mechanism design for federated machine learning in mobile networks", *IEEE Internet of Things Journal (2020)*, 7(10), pp.9575-9588.

7. **Wei Yang Bryan Lim**, Sahil Garg, Zehui Xiong, Dusit Niyato, Cyril Leung, Chunyan Miao, and Mohsen Guizani, "Dynamic contract design for federated learning in smart healthcare applications", *IEEE Internet of Things Journal (2020)*, Early Access Article.

8. **Wei Yang Bryan Lim**, Zehui Xiong, Dusit Niyato, Cyril Leung, Chunyan Miao, and Xuemin (Sherman) Shen, "A task-aware incentive scheme for federated learning", *IEEE Transactions on Industrial Informatics (2020)*, Early Access Article.

9. Jer Shyuan Ng, **Wei Yang Bryan Lim**, Nguyen Cong Luong, Zehui Xiong, Alia Asherleva, Dusit Niyato, Cyril Leung, and Chunyan Miao, "A comprehensive survey on coded distributed computing: Fundamentals, challenges, and networking applications", *IEEE Communications Surveys & Tutorials (2021)*, 23(3), pp.1800-1837.

10. Jer Shyuan Ng, **Wei Yang Bryan Lim**, Zehui Xiong, Dusit Niyato, Cyril Leung, and Chunyan Miao, "A double auction mechanism for resource allocation in coded vehicular edge computing", *IEEE Transactions on Vehicular Technology (2021)*, Accepted.

11. Zehui Xiong, Yang Zhang, **Wei Yang Bryan Lim**, Jiawen Kang, Dusit Niyato, Cyril Leung, and Chunyan Miao, "UAV-assisted wireless energy and data transfer with deep reinforcement learning", *IEEE Transactions on Cognitive Communications and Networking (2020)*, 7(1), pp.85-99.

12. Jer Shyuan Ng, **Wei Yang Bryan Lim**, Hong-Ning Dai, Zehui Xiong, Jianqiang Huang, Dusit Niyato, Xian-Sheng Hua, Cyril Leung, Chunyan Miao, "Joint auction-coalition formation framework for communication-efficient federated learning in UAV-enabled Internet of Vehicles", *IEEE Transactions on Intelligent Transportation Systems (2020)*, 22(4), pp.2326-2344.

13. Yi Liu, Jiangtian Nie, Xuandi Li, Syed Hassan Ahmed, **Wei Yang Bryan Lim**, and Chunyan Miao, "Federated Learning in the Sky: Aerial-Ground

Air Quality Sensing Framework with UAV Swarms", *IEEE Internet of Things Journal (2020)*, 8(12), pp.9827-9837.

Conference Proceedings

1. **Wei Yang Bryan Lim**, Jer Shyuan Ng, Zehui Xiong, Sahil Garg, Yang Zhang, Dusit Niyato, and Chunyan Miao, "Dynamic edge association in hierarchical federated learning networks", in *Proceedings of IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, Shenyang, China, August 2021.

2. **Wei Yang Bryan Lim**, Zehui Xiong, Dusit Niyato, Jianqiang Huang, Xian-Sheng Hua, and Chunyan Miao, "Incentive mechanism design for federated learning in vehicular ad hoc networks", in *IEEE Vehicular Technology Conference (VTC)*, Fall, Victoria, Canada, October 2020.

3. **Wei Yang Bryan Lim**, Jianqiang Huang, Zehui Xiong, Jiawen Kang, Dusit Niyato, Xian-Sheng Hua, Cyril Leung, and Chunyan Miao, "Multi-dimensional contract-matching for federated learning in UAV-enabled Internet of vehicles", in *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, Taipei, December 2020.

4. **Wei Yang Bryan Lim**, Zehui Xiong, Jiawen Kang, Dusit Niyato, Yang Zhang, Cyril Leung, and Chunyan Miao, "An incentive scheme for federated learning in the sky", in *Proceedings of ACM MobiCom Workshop on Drone Assisted Wireless Communications for 5G and Beyond (DroneCom)*, London, UK, September 2020.

5. **Wei Yang Bryan Lim**, Jer Shyuan Ng, Zehui Xiong, Dusit Niyato, Song Guo, Cyril Leung, and Chunyan Miao, "Dynamic resource allocation for hierarchical federated learning", in *Proceedings of the 16th International Conference on Mobility, Sensing and Networking (MSN)*, Tokyo, Japan, December 2020.

6. Jer Shyuan Ng, **Wei Yang Bryan Lim**, Zehui Xiong, Sahil Garg, Dusit Niyato, and Cyril Leung, "An incentive mechanism for resource allocation in

coded distributed computing", in *Proceedings of IEEE International Conference on Big Data Science and Engineering (BigDataSE)*, Shenyang, China, August 2021.

7. Wei Chong Ng, **Wei Yang Bryan Lim**, Jer Shyuan Ng, Suttinee Sawadsitang, Zehui Xiong, and Dusit Niyato, "Optimal stochastic coded computation offloading in unmanned aerial vehicles network", in *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, Madrid, Spain, December 2021.

8. Jer Shyuan Ng, **Wei Yang Bryan Lim**, Zehui Xiong, Sahil Garg, Yang Zhang, Dusit Niyato, and Cyril Leung, "A double auction mechanism for coded distributed computing in smart vehicles", in *Proceedings of the 18th International Conference on Wireless Networks and Mobile Systems (WINSYS)*, Virtual Conference, July 2021.

9. Jer Shyuan Ng, **Wei Yang Bryan Lim**, Sahil Garg, Zehui Xiong, Dusit Niyato, Mohsen Guizani, and Cyril Leung, "Collaborative coded computation offloading: An all-pay auction approach", in *Proceedings of IEEE International Conference on Communications (ICC)*, Montreal, Canada, June 2021.

10. Zehui Xiong, **Wei Yang Bryan Lim**, Jiawen Kang, Dusit Niyato, Ping Wang, and Chunyan Miao, "Incentive mechanism design for mobile data rewards using multi-dimensional contract", in *IEEE Wireless Communications and Networking Conference (WCNC)*, Seoul, Korea, April 2020.

11. Jer Shyuan Ng, **Wei Yang Bryan Lim**, Hong-Ning Dai, Zehui Xiong, Jianqiang Huang, Dusit Niyato, Xian-Sheng Hua, Cyril Leung, Chunyan Miao, "Communication-efficient federated learning in UAV-enabled IoV: A joint auction-coalition formation approach", *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, Taipei, December 2020.

12. Yi Liu, Neeraj Kumar, Zehui Xiong, **Wei Yang Bryan Lim**, Jiawen Kang, and Dusit Niyato, "Communication-efficient federated learning for anomaly detection in industrial Internet of Things", in *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, Taipei, December 2020.

13. Wenqi Yang, Yang Zhang, **Wei Yang Bryan Lim**, Zehui Xiong, Yutao Jiao, and Jiangming Jin, "Privacy is not free: Energy-aware federated learning for mobile and edge intelligence", in *Proceedings of the 12th International Conference on Wireless Communications and Signal Processing (WCSP)*, Nanjing, China, October 2020.

Demonstration Track Papers

1. Hongchao Jiang, **Wei Yang Bryan Lim**, Jer Shyuan Ng, Harold Teng, Han Yu, Zehui Xiong, Dusit Niyato, and Chunyan Miao, "AI-empowered decision support for COVID-19 social distancing", in *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI-21)*, demo track, Virtual Conference, Feburary 2021.

2. Harold Teng, Hongchao Jiang, Xuan Rong Zane Ho, **Wei Yang Bryan Lim**, Jer Shyuan Ng, Han Yu, Zehui Xiong, Dusit Niyato, and Chunyan Miao, "Predictive analytics for COVID-19 social distancing", in *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, demo track, Montreal, Canada, August 2021.

3. Austine Yapp, Nicholas Koh, Yan Ting Lai, Jiawen Kang, Xuandi Li, Jer Shyuan Ng, Hongchao Jiang, **Wei Yang Bryan Lim**, Zehui Xiong, and Dusit Niyato, "Communication-efficient and scalable decentralized federated edge learning", in *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, demo track, Montreal, Canada, August 2021.

Bibliography

[1] D. Xu, T. Li, Y. Li, X. Su, S. Tarkoma, and P. Hui, "A survey on edge intelligence," *arXiv preprint arXiv:2003.12172*, 2020.

[2] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

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

[4] S. Niknam, H. S. Dhillon, and J. H. Reed, "Federated learning for wireless communications: Motivation, opportunities, and challenges," *IEEE Communications Magazine*, vol. 58, no. 6, pp. 46–51, 2020.

[5] O. Simeone, "A very brief introduction to machine learning with applications to communication systems," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 4, pp. 648–664, 2018.

[6] I. Stojmenovic, S. Wen, X. Huang, and H. Luan, "An overview of fog computing and its security issues," *Concurrency and Computation: Practice and Experience*, vol. 28, no. 10, pp. 2991–3005, 2016.

[7] M. Gerla, E.-K. Lee, G. Pau, and U. Lee, "Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds," in *2014 IEEE world forum on internet of things (WF-IoT)*. IEEE, 2014, pp. 241–246.

[8] I. Kadota, E. Uysal-Biyikoglu, R. Singh, and E. Modiano, "Minimizing the age of information in broadcast wireless networks," in *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2016, pp. 844–851.

[9] A. Abeshu and N. Chilamkurti, "Deep learning: the frontier for distributed attack detection in fog-to-things computing," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 169–175, 2018.

[10] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "Dïot: A federated self-learning anomaly detection system for

iot," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 756–767.

[11] D. Preuveneers, V. Rimmer, I. Tsingenopoulos, J. Spooren, W. Joosen, and E. Ilie-Zudor, "Chained anomaly detection models for federated learning: An intrusion detection case study," *Applied Sciences*, vol. 8, no. 12, p. 2663, 2018.

[12] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning," *arXiv preprint arXiv:1809.07857*, 2018.

[13] 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.

[14] 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)*. IEEE, 2018, pp. 1–6.

[15] Y. Qian, L. Hu, J. Chen, X. Guan, M. M. Hassan, and A. Alelaiwi, "Privacy-aware service placement for mobile edge computing via federated learning," *Information Sciences*, vol. 505, pp. 562–570, 2019.

[16] M. Chen, O. Semiari, W. Saad, X. Liu, and C. Yin, "Federated echo state learning for minimizing breaks in presence in wireless virtual reality networks," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 177–191, 2019.

[17] K. Hamidouche, A. T. Z. Kasgari, W. Saad, M. Bennis, and M. Debbah, "Collaborative artificial intelligence (ai) for user-cell association in ultra-dense cellular systems," in *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2018, pp. 1–6.

[18] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Distributed federated learning for ultra-reliable low-latency vehicular communications," *IEEE Transactions on Communications*, vol. 68, no. 2, pp. 1146–1159, 2019.

[19] Y. Saputra, H. Dinh, D. Nguyen, E. Dutkiewicz, M. Mueck, and S. Srikanteswara, in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[20] P. Bolton, M. Dewatripont *et al.*, *Contract theory*. MIT press, 2005.

[21] R. Pryss, M. Reichert, J. Herrmann, B. Langguth, and W. Schlee, "Mobile crowd sensing in clinical and psychological trials–a case study," in *2015 IEEE 28th International Symposium on Computer-Based Medical Systems*. IEEE, 2015, pp. 23–24.

[22] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges," *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32–39, 2011.

[23] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[24] D. Oletic and V. Bilas, "Design of sensor node for air quality crowdsensing," in *2015 IEEE Sensors Applications Symposium (SAS)*. IEEE, 2015, pp. 1–5.

[25] Y. Jing, B. Guo, Z. Wang, V. O. Li, J. C. Lam, and Z. Yu, "Crowdtracker: Optimized urban moving object tracking using mobile crowd sensing," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3452–3463, 2017.

[26] H.-J. Hong, C.-L. Fan, Y.-C. Lin, and C.-H. Hsu, "Optimizing cloud-based video crowdsensing," *IEEE Internet of Things Journal*, vol. 3, no. 3, pp. 299–313, 2016.

[27] W. He, G. Yan, and L. Da Xu, "Developing vehicular data cloud services in the iot environment," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1587–1595, 2014.

[28] P. Li, J. Li, Z. Huang, T. Li, C.-Z. Gao, S.-M. Yiu, and K. Chen, "Multi-key privacy-preserving deep learning in cloud computing," *Future Generation Computer Systems*, vol. 74, pp. 76–85, 2017.

[29] B. Custers, A. Sears, F. Dechesne, I. Georgieva, T. Tani, and S. van der Hof, *EU Personal Data Protection in Policy and Practice*. Springer, 2019.

[30] B. M. Gaff, H. E. Sussman, and J. Geetter, "Privacy and big data," *Computer*, vol. 47, no. 6, pp. 7–9, 2014.

[31] 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.

[32] G. Ananthanarayanan, P. Bahl, P. Bodík, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha, "Real-time video analytics: The killer app for edge computing," *Computer*, vol. 50, no. 10, pp. 58–67, 2017.

[33] H. Li, K. Ota, and M. Dong, "Learning iot in edge: Deep learning for the internet of things with edge computing," *IEEE Network*, vol. 32, no. 1, pp. 96–101, 2018.

[34] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869–904, 2020.

[35] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[36] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2015.

[37] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[38] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 308–318.

[39] F. Cicirelli, A. Guerrieri, G. Spezzano, A. Vinci, O. Briante, A. Iera, and G. Ruggeri, "Edge computing and social internet of things for large-scale smart environments development," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2557–2571, 2017.

[40] A. Hard, K. Rao, R. Mathews, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[41] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, "Federated learning of predictive models from federated electronic health records," *International journal of medical informatics*, vol. 112, pp. 59–67, 2018.

[42] K. Powell, "Nvidia clara federated learning to deliver ai to hospitals while protecting patient data," *https://blogs.nvidia.com/blog/2019/12/01/clara-federated-learning/*, 2019.

[43] D. Verma, S. Julier, and G. Cirincione, "Federated ai for building ai solutions across multiple agencies," *arXiv preprint arXiv:1809.10036*, 2018.

[44] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Communications Surveys & Tutorials*, 2019.

[45] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys & Tutorials*, 2019.

[46] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, p. 12, 2019.

[47] 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.

[48] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.

[49] L. Cui, S. Yang, F. Chen, Z. Ming, N. Lu, and J. Qin, "A survey on application of machine learning for internet of things," *International Journal of Machine Learning and Cybernetics*, vol. 9, no. 8, pp. 1399–1417, 2018.

[50] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Networks and Applications*, vol. 18, no. 1, pp. 129–140, 2013.

[51] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2017.

[52] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[53] J. Yao, T. Han, and N. Ansari, "On mobile edge caching," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2525–2553, 2019.

[54] 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.

[55] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[56] J. A. Cornell, *Classical and modern regression with applications*. Taylor & Francis, 1987.

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

[58] T. Lin, S. U. Stich, K. K. Patel, and M. Jaggi, "Don't use large mini-batches, use local sgd," *arXiv preprint arXiv:1808.07217*, 2018.

[59] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[60] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[61] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International journal of computer vision*, vol. 40, no. 2, pp. 99–121, 2000.

[62] M. Duan, "Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications," *arXiv preprint arXiv:1907.01132*, 2019.

[63] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding data augmentation for classification: when to warp?" in *2016 international conference on digital image computing: techniques and applications (DICTA)*. IEEE, 2016, pp. 1–6.

[64] J. M. Joyce, "Kullback-leibler divergence," *International encyclopedia of statistical science*, pp. 720–722, 2011.

[65] M. Jaggi, V. Smith, M. Takác, J. Terhorst, S. Krishnan, T. Hofmann, and M. I. Jordan, "Communication-efficient distributed dual coordinate ascent," in *Advances in neural information processing systems*, 2014, pp. 3068–3076.

[66] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4424–4434.

[67] J. C. Bezdek and R. J. Hathaway, "Convergence of alternating optimization," *Neural, Parallel & Scientific Computations*, vol. 11, no. 4, pp. 351–368, 2003.

[68] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," *arXiv preprint arXiv:1912.00818*, 2019.

[69] J. Ren, X. Shen, Z. Lin, R. Mech, and D. J. Foran, "Personalized image aesthetics," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 638–647.

[70] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, 2019.

[71] L. Huang, Y. Yin, Z. Fu, S. Zhang, H. Deng, and D. Liu, "Loadaboost: Loss-based adaboost federated machine learning on medical data," *arXiv preprint arXiv:1811.12629*, 2018.

[72] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, 2016, pp. 265–283.

[73] T. Ryffel, A. Trask, M. Dahl, B. Wagner, J. Mancuso, D. Rueckert, and J. Passerat-Palmbach, "A generic framework for privacy preserving deep learning," *arXiv preprint arXiv:1811.04017*, 2018.

[74] S. Caldas, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," *arXiv preprint arXiv:1812.01097*, 2018.

[75] A. Go, R. Bhayani, and L. Huang, "Sentiment140," *Site Functionality, 2013c. URL http://help. sentiment140. com/site-functionality. Abruf am*, vol. 20, 2016.

[76] F. Zheng, K. Li, J. Tian, X. Xiang *et al.*, "A vertical federated learning method for interpretable scorecard and its application in credit scoring," *arXiv preprint arXiv:2009.06218*, 2020.

[77] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[78] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[79] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[80] W. Luping, W. Wei, and L. Bo, "Cmfl: Mitigating communication overhead for federated learning," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 954–964.

[81] H. Wang, S. Sievert, S. Liu, Z. Charles, D. Papailiopoulos, and S. Wright, "Atomo: Communication-efficient learning via atomic sparsification," in *Advances in Neural Information Processing Systems*, 2018, pp. 9850–9861.

[82] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified sgd with memory," in *Advances in Neural Information Processing Systems*, 2018, pp. 4447–4458.

[83] S. Caldas, J. Konečny, H. B. McMahan, and A. Talwalkar, "Expanding the reach of federated learning by reducing client resource requirements," *arXiv preprint arXiv:1812.07210*, 2018.

[84] Z. Tao and Q. Li, "esgd: Communication efficient distributed deep learning on the edge," in {*USENIX*} *Workshop on Hot Topics in Edge Computing (HotEdge 18)*, 2018.

[85] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[86] Y. Liu, Y. Kang, X. Zhang, L. Li, Y. Cheng, T. Chen, M. Hong, and Q. Yang, "A communication efficient vertical federated learning framework," *arXiv preprint arXiv:1912.11187*, 2019.

[87] X. Yao, C. Huang, and L. Sun, "Two-stream federated learning: Reduce the communication costs," in *2018 IEEE Visual Communications and Image Processing (VCIP)*. IEEE, 2018, pp. 1–4.

[88] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*. JMLR. org, 2015, pp. 97–105.

[89] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer feature learning with joint distribution adaptation," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 2200–2207.

[90] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[91] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Edge-assisted hierarchical federated learning with non-iid data," *arXiv preprint arXiv:1905.06641*, 2019.

[92] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "Emnist: an extension of mnist to handwritten letters," *arXiv preprint arXiv:1702.05373*, 2017.

[93] N. Strom, "Scalable distributed dnn training using commodity gpu cloud computing," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[94] J. Chen, X. Pan, R. Monga, S. Bengio, and R. Jozefowicz, "Revisiting distributed synchronous sgd," *arXiv preprint arXiv:1604.00981*, 2016.

[95] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," *arXiv preprint arXiv:1712.01887*, 2017.

[96] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu, "Gaia: Geo-distributed machine learning approaching {LAN} speeds," in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, 2017, pp. 629–647.

[97] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones." in *Esann*, 2013.

[98] M. Buscema and S. Terzi, "Semeion handwritten digit data set," *Center for Machine Learning and Intelligent Systems, California, USA*, 2009.

[99] M. R. Sprague, A. Jalalirad, M. Scavuzzo, C. Capota, M. Neun, L. Do, and M. Kopp, "Asynchronous federated learning for geospatial applications," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2018, pp. 21–28.

[100] M. I. Jordan, J. D. Lee, and Y. Yang, "Communication-efficient distributed statistical inference," *Journal of the American Statistical Association*, vol. 114, no. 526, pp. 668–681, 2019.

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

[102] M. Sviridenko, "A note on maximizing a submodular set function subject to a knapsack constraint," *Operations Research Letters*, vol. 32, no. 1, pp. 41–43, 2004.

[103] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 4615–4625.

[104] N. Yoshida, T. Nishio, M. Morikura, K. Yamamoto, and R. Yonetani, "Hybrid-fl: Cooperative learning mechanism using non-iid data in wireless networks," *arXiv preprint arXiv:1905.07210*, 2019.

[105] T. T. Anh, N. C. Luong, D. Niyato, D. I. Kim, and L.-C. Wang, "Efficient training management for mobile crowd-machine learning: A deep reinforcement learning approach," *arXiv preprint arXiv:1812.03633*, 2018.

[106] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[107] M. J. Neely, E. Modiano, and C.-P. Li, "Fairness and optimal stochastic control for heterogeneous networks," *IEEE/ACM Transactions On Networking*, vol. 16, no. 2, pp. 396–409, 2008.

[108] S. Corbett-Davies and S. Goel, "The measure and mismeasure of fairness: A critical review of fair machine learning," *arXiv preprint arXiv:1808.00023*, 2018.

[109] T. Li, M. Sanjabi, and V. Smith, "Fair resource allocation in federated learning," *arXiv preprint arXiv:1905.10497*, 2019.

[110] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Convergence time optimization for federated learning over wireless networks," *arXiv preprint arXiv:2001.07845*, 2020.

[111] D. López-Pérez, A. Valcarce, G. De La Roche, and J. Zhang, "Ofdma femtocells: A roadmap on interference avoidance," *IEEE Communications Magazine*, vol. 47, no. 9, pp. 41–48, 2009.

[112] G. Zhu, Y. Wang, and K. Huang, "Low-latency broadband analog aggregation for federated edge learning," *arXiv preprint arXiv:1812.11494*, 2018.

[113] B. Nazer and M. Gastpar, "Compute-and-forward: Harnessing interference through structured codes," *IEEE Transactions on Information Theory*, vol. 57, no. 10, pp. 6463–6486, 2011.

[114] M. M. Amiri and D. Gunduz, "Federated learning over wireless fading channels," *arXiv preprint arXiv:1907.09769*, 2019.

[115] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *arXiv preprint arXiv:1812.11750*, 2018.

[116] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," *arXiv preprint arXiv:1609.04836*, 2016.

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

[118] P. D. Tao *et al.*, "The dc (difference of convex functions) programming and dca revisited with dc models of real world nonconvex optimization problems," *Annals of operations research*, vol. 133, no. 1-4, pp. 23–46, 2005.

[119] Z.-Q. Luo, N. D. Sidiropoulos, P. Tseng, and S. Zhang, "Approximation bounds for quadratic optimization with homogeneous quadratic constraints," *SIAM Journal on optimization*, vol. 18, no. 1, pp. 1–28, 2007.

[120] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," *arXiv preprint arXiv:1903.03934*, 2019.

[121] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[122] S. Feng, D. Niyato, P. Wang, D. I. Kim, and Y.-C. Liang, "Joint service pricing and cooperative relay communication for federated learning," *arXiv preprint arXiv:1811.12082*, 2018.

[123] M. J. Osborne *et al.*, *An introduction to game theory*. Oxford university press New York, 2004, vol. 3, no. 3.

[124] Y. Sarikaya and O. Ercetin, "Motivating workers in federated learning: A stackelberg game perspective," *arXiv preprint arXiv:1908.03092*, 2019.

[125] L. U. Khan, N. H. Tran, S. R. Pandey, W. Saad, Z. Han, M. N. Nguyen, and C. S. Hong, "Federated learning for edge networks: Resource optimization and incentive mechanism," *arXiv preprint arXiv:1911.05642*, 2019.

[126] Y. Zhan, P. Li, Z. Qu, D. Zeng, and S. Guo, "A learning-based incentive mechanism for federated learning," *IEEE Internet of Things Journal*, 2020.

[127] 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," *arXiv preprint arXiv:1905.07479*, 2019.

[128] R. Jurca and B. Faltings, "An incentive compatible reputation mechanism," in *EEE International Conference on E-Commerce, 2003. CEC 2003.* IEEE, 2003, pp. 285–292.

[129] R. Dennis and G. Owen, "Rep on the block: A next generation reputation system based on the blockchain," in *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*. IEEE, 2015, pp. 131–138.

[130] H. T. Nguyen, N. C. Luong, J. Zhao, C. Yuen, and D. Niyato, "Resource allocation in mobility-aware federated learning networks: A deep reinforcement learning approach," *arXiv preprint arXiv:1910.09172*, 2019.

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

[132] J. Su, D. Yi, B. Su, Z. Mi, C. Liu, X. Hu, X. Xu, L. Guo, and W.-H. Chen, "Aerial visual perception in smart farming: Field study of wheat yellow rust monitoring," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 3, pp. 2242–2249, 2020.

[133] L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[134] J. Hopkins and P. Hawking, "Big data analytics and iot in logistics: A case study," *The International Journal of Logistics Management*, 2018.

[135] L. Lipper, P. Thornton, B. M. Campbell, T. Baedeker, A. Braimoh, M. Bwalya, P. Caron, A. Cattaneo, D. Garrity, K. Henry *et al.*, "Climate-smart agriculture for food security," *Nature climate change*, vol. 4, no. 12, pp. 1068–1072, 2014.

[136] Y. Zhu, J. Song, and F. Dong, "Applications of wireless sensor network in the agriculture environment monitoring," *Procedia Engineering*, vol. 16, pp. 608–614, 2011.

[137] D. I. Patrício and R. Rieder, "Computer vision and artificial intelligence in precision agriculture for grain crops: A systematic review," *Computers and electronics in agriculture*, vol. 153, pp. 69–81, 2018.

[138] P. Boniecki, H. Piekarska-Boniecka, K. Świerczyński, K. Koszela, M. Zaborowicz, and J. Przybył, "Detection of the granary weevil based on x-ray images of damaged wheat kernels," *Journal of Stored Products Research*, vol. 56, pp. 38–42, 2014.

[139] A. Antonacci, F. Arduini, D. Moscone, G. Palleschi, and V. Scognamiglio, "Nanostructured (bio) sensors for smart agriculture," *TrAC Trends in Analytical Chemistry*, vol. 98, pp. 95–103, 2018.

[140] S. Zhang, J. Li, H. Luo, J. Gao, L. Zhao, and X. S. Shen, "Towards fresh and low-latency content delivery in vehicular networks: An edge caching aspect," in *2018 10th International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE, 2018, pp. 1–6.

[141] F. Gabry, V. Bioglio, and I. Land, "On energy-efficient edge caching in heterogeneous networks," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3288–3298, 2016.

[142] M. Zeng, Y. Li, K. Zhang, M. Waqas, and D. Jin, "Incentive mechanism design for computation offloading in heterogeneous fog computing: A contract-based approach," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–6.

[143] D. Wu, J. Yan, H. Wang, D. Wu, and R. Wang, "Social attribute aware incentive mechanism for device-to-device video distribution," *IEEE Transactions on Multimedia*, vol. 19, no. 8, pp. 1908–1920, 2017.

[144] Y. Zhang, L. Song, W. Saad, Z. Dawy, and Z. Han, "Contract-based incentive mechanisms for device-to-device communications in cellular networks," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 10, pp. 2144–2155, 2015.

[145] L. Gao, X. Wang, Y. Xu, and Q. Zhang, "Spectrum trading in cognitive radio networks: A contract-theoretic modeling approach," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 4, pp. 843–855, 2011.

[146] S. Diamond and S. Boyd, "Cvxpy: A python-embedded modeling language for convex optimization," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2909–2913, 2016.

[147] W. Y. B. Lim, S. Garg, Z. Xiong, D. Niyato, C. Leung, C. Miao, and M. Guizani, "Dynamic contract design for federated learning in smart healthcare applications," *IEEE Internet of Things Journal*, 2020.

[148] F. Li and Y. Wang, "Routing in vehicular ad hoc networks: A survey," *IEEE Vehicular Technology Magazine*, vol. 2, no. 2, pp. 12–22, 2007.

[149] H. Hartenstein and L. Laberteaux, "A tutorial survey on vehicular ad hoc networks," *IEEE Communications Magazine*, vol. 46, no. 6, pp. 164–171, 2008.

[150] W. Xu, H. Zhou, N. Cheng, F. Lyu, W. Shi, J. Chen, and X. Shen, "Internet of vehicles in big data era," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 1, pp. 19–35, 2017.

[151] J. Wan, J. Liu, Z. Shao, A. V. Vasilakos, M. Imran, and K. Zhou, "Mobile crowd sensing for traffic prediction in internet of vehicles," *Sensors*, vol. 16, no. 1, p. 88, 2016.

[152] F. Yang, S. Wang, J. Li, Z. Liu, and Q. Sun, "An overview of internet of vehicles," *China communications*, vol. 11, no. 10, pp. 1–15, 2014.

[153] J. Wang, C. Jiang, Z. Han, Y. Ren, and L. Hanzo, "Internet of vehicles: Sensing-aided transportation information collection and diffusion," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 3813–3825, 2018.

[154] P. M. Kumar, G. Manogaran, R. Sundarasekar, N. Chilamkurti, R. Varatharajan *et al.*, "Ant colony optimization algorithm with internet of vehicles for intelligent traffic control system," *Computer Networks*, vol. 144, pp. 154–162, 2018.

[155] M. Florian, S. Finster, and I. Baumgart, "Privacy-preserving cooperative route planning," *IEEE Internet of Things Journal*, vol. 1, no. 6, pp. 590–599, 2014.

[156] L.-M. Ang, K. P. Seng, G. K. Ijemaru, and A. M. Zungeru, "Deployment of iov for smart cities: applications, architecture, and challenges," *IEEE Access*, vol. 7, pp. 6473–6492, 2018.

[157] H. Zhou, H. Kong, L. Wei, D. Creighton, and S. Nahavandi, "Efficient road detection and tracking for unmanned aerial vehicle," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 297–309, 2014.

[158] H. Zhou, L. Wei, M. Fielding, D. Creighton, S. Deshpande, and S. Nahavandi, "Car park occupancy analysis using uav images," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2017, pp. 3261–3265.

[159] M. Elloumi, R. Dhaou, B. Escrig, H. Idoudi, and L. A. Saidane, "Monitoring road traffic with a uav-based system," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2018, pp. 1–6.

[160] B. Coifman, M. McCord, R. G. Mishalani, M. Iswalt, and Y. Ji, "Roadway traffic monitoring from an unmanned aerial vehicle," in *IEE Proceedings-Intelligent Transport Systems*, vol. 153, no. 1. IET, 2006, pp. 11–20.

[161] R. Ke, Z. Li, J. Tang, Z. Pan, and Y. Wang, "Real-time traffic flow parameter estimation from uav video based on ensemble classifier and optical flow," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 1, pp. 54–64, 2018.

[162] H. Binol, E. Bulut, K. Akkaya, and I. Guvenc, "Time optimal multi-uav path planning for gathering its data from roadside units," in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*. IEEE, 2018, pp. 1–5.

[163] L. Zhang, Z. Zhao, Q. Wu, H. Zhao, H. Xu, and X. Wu, "Energy-aware dynamic resource allocation in uav assisted mobile edge computing over social internet of vehicles," *IEEE Access*, vol. 6, pp. 56 700–56 715, 2018.

[164] O. Bekkouche, T. Taleb, and M. Bagaa, "Uavs traffic control based on multi-access edge computing," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–6.

[165] M. Gharibi, R. Boutaba, and S. L. Waslander, "Internet of drones," *IEEE Access*, vol. 4, pp. 1148–1162, 2016.

[166] A. Koubâa and B. Qureshi, "Dronetrack: Cloud-based real-time object tracking using unmanned aerial vehicles over the internet," *IEEE Access*, vol. 6, pp. 13 810–13 824, 2018.

[167] A. Koubâa, B. Qureshi, M.-F. Sriti, Y. Javed, and E. Tovar, "A service-oriented cloud-based management system for the internet-of-drones," in *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2017, pp. 329–335.

[168] L. Gupta, R. Jain, and G. Vaszkun, "Survey of important issues in uav communication networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1123–1152, 2015.

[169] Y. Zeng and R. Zhang, "Energy-efficient uav communication with trajectory optimization," *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3747–3760, 2017.

[170] L. E. Dubins and D. A. Freedman, "Machiavelli and the gale-shapley algorithm," *The American Mathematical Monthly*, vol. 88, no. 7, pp. 485–494, 1981.

[171] Z. Zhou, J. Feng, B. Gu, B. Ai, S. Mumtaz, J. Rodriguez, and M. Guizani, "When mobile crowd sensing meets uav: Energy-efficient task assignment and route planning," *IEEE Transactions on Communications*, vol. 66, no. 11, pp. 5526–5538, 2018.

[172] G. Karypis and V. Kumar, "Multilevel graph partitioning schemes," in *ICPP (3)*, 1995, pp. 113–122.

[173] Y. Zeng, J. Xu, and R. Zhang, "Energy minimization for wireless communication with rotary-wing uav," *IEEE Transactions on Wireless Communications*, vol. 18, no. 4, pp. 2329–2345, 2019.

[174] Q. Zhang, W. Saad, M. Bennis, X. Lu, M. Debbah, and W. Zuo, "Predictive deployment of uav base stations in wireless networks: Machine learning meets contract theory," *arXiv preprint arXiv:1811.01149*, 2018.

[175] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[176] Q. Liu, S. Huang, J. Opadere, and T. Han, "An edge network orchestrator for mobile augmented reality," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 756–764.

[177] Z. Wang, L. Gao, and J. Huang, "Multi-cap optimization for wireless data plans with time flexibility," *IEEE Transactions on Mobile Computing*, vol. 19, no. 9, pp. 2145–2159, 2019.

[178] A. E. Roth and M. Sotomayor, "Two-sided matching," *Handbook of game theory with economic applications*, vol. 1, pp. 485–541, 1992.

[179] Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz, and J. Rodriguez, "Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3113–3125, 2019.

[180] G. O'Malley, "Algorithmic aspects of stable matching problems," Ph.D. dissertation, University of Glasgow, 2007.

[181] S. Sawadsitang, D. Niyato, P.-S. Tan, and P. Wang, "Joint ground and aerial package delivery services: A stochastic optimization approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 6, pp. 2241–2254, 2018.

[182] Y. Guo, S. Gu, Q. Zhang, N. Zhang, and W. Xiang, "A coded distributed computing framework for task offloading from multi-uav to edge servers," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2021, pp. 1–6.

[183] S. Prakash, S. Dhakal, M. R. Akdeniz, Y. Yona, S. Talwar, S. Avestimehr, and N. Himayat, "Coded computing for low-latency federated learning over wireless edge networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 233–250, 2020.

[184] W. Y. B. Lim, J. S. Ng, Z. Xiong, D. Niyato, C. Leung, C. Miao, and Q. Yang, "Incentive mechanism design for resource sharing in collaborative edge learning," *arXiv preprint arXiv:2006.00511*, 2020.

[185] M. S. H. Abad, E. Ozfatura, D. Gunduz, and O. Ercetin, "Hierarchical federated learning across heterogeneous cellular networks," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 8866–8870.

[186] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Wireless communications for collaborative federated learning," *IEEE Communications Magazine*, vol. 58, no. 12, pp. 48–54, 2020.

[187] J. W. Weibull, *Evolutionary game theory*. MIT press, 1997.

[188] E. Peltonen, M. Bennis, M. Capobianco, M. Debbah, A. Ding, F. Gil-Castiñeira, M. Jurmu, T. Karvonen, M. Kelanti, A. Kliks *et al.*, "6g white paper on edge intelligence," *arXiv preprint arXiv:2004.14850*, 2020.

[189] M. Xie, H. Yin, H. Wang, F. Xu, W. Chen, and S. Wang, "Learning graph-based poi embedding for location-based recommendation," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, 2016, pp. 15–24.

[190] M. Handy, M. Haase, and D. Timmermann, "Low energy adaptive clustering hierarchy with deterministic cluster-head selection," in *4th international workshop on mobile and wireless communications network.* IEEE, 2002, pp. 368–372.

[191] H. Junping, J. Yuhui, and D. Liang, "A time-based cluster-head selection algorithm for leach," in *2008 IEEE Symposium on Computers and Communications.* IEEE, 2008, pp. 1172–1176.

[192] R. Ferdous, V. Muthukkumarasamy, and E. Sithirasenan, "Trust-based cluster head selection algorithm for mobile ad hoc networks," in *2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications.* IEEE, 2011, pp. 589–596.

[193] G. Zhang, K. Yang, and H.-H. Chen, "Socially aware cluster formation and radio resource allocation in d2d networks," *IEEE Wireless Communications*, vol. 23, no. 4, pp. 68–73, 2016.

[194] K. Zhu, E. Hossain, and D. Niyato, "Pricing, spectrum sharing, and service selection in two-tier small cell networks: A hierarchical dynamic game approach," *IEEE Transactions on Mobile Computing*, vol. 13, no. 8, pp. 1843–1856, 2013.

[195] X. Gong, L. Duan, X. Chen, and J. Zhang, "When social network effect meets congestion effect in wireless networks: Data usage equilibrium and optimal pricing," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 2, pp. 449–462, 2017.

[196] W. Y. B. Lim, Z. Xiong, C. Miao, D. Niyato, Q. Yang, C. Leung, and H. V. Poor, "Hierarchical incentive mechanism design for federated machine learning in mobile networks," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9575–9588, 2020.

[197] W. Y. B. Lim, J. Huang, Z. Xiong, J. Kang, D. Niyato, X.-S. Hua, C. Leung, and C. Miao, "Towards federated learning in uav-enabled internet of vehicles: A multi-dimensional contract-matching approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 5140–5154, 2021.

[198] D. Niyato and E. Hossain, "Dynamics of network selection in heterogeneous wireless networks: An evolutionary game approach," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 4, 2008.

[199] J. Hofbauer and K. Sigmund, "Evolutionary game dynamics," *Bulletin of the American mathematical society*, vol. 40, no. 4, pp. 479–519, 2003.

[200] X. Gao, S. Feng, D. Niyato, P. Wang, K. Yang, and Y.-C. Liang, "Dynamic access point and service selection in backscatter-assisted rf-powered cognitive networks," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8270–8283, 2019.

[201] J. Engwerda, *LQ dynamic optimization and differential games*. John Wiley & Sons, 2005.

[202] S. Sastry, "Lyapunov stability theory," in *Nonlinear Systems*. Springer, 1999, pp. 182–234.

[203] 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)*. IEEE, 2019, pp. 874–879.

[204] R. B. Myerson, "Optimal auction design," *Mathematics of operations research*, vol. 6, no. 1, pp. 58–73, 1981.

[205] P. Dütting, Z. Feng, H. Narasimhan, D. C. Parkes, and S. S. Ravindranath, "Optimal Auctions through Deep Learning," *arXiv preprint arXiv:1706.03459*, 2017.

[206] N. C. Luong, Z. Xiong, P. Wang, and D. Niyato, "Optimal auction for edge computing resource management in mobile blockchain networks: A deep learning approach," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–6.

[207] K. Zhu, D. Niyato, and P. Wang, "Optimal bandwidth allocation with dynamic service selection in heterogeneous wireless networks," in *2010 IEEE Global Telecommunications Conference Globecom 2010*. IEEE, 2010, pp. 1–5.

[208] M. Lee, S. Hosseinalipour, C. G. Brinton, G. Yu, and H. Dai, "A fast graph neural network-based method for winner determination in multi-unit combinatorial auctions," *IEEE Transactions on Cloud Computing*, 2020.

[209] J. Kang, Z. Xiong, C. Jiang, Y. Liu, S. Guo, Y. Zhang, D. Niyato, C. Leung, and C. Miao, "Scalable and communication-efficient decentralized federated edge learning with multi-blockchain framework," in *International Conference on Blockchain and Trustworthy Systems*. Springer, 2020, pp. 152–165.

[210] G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici, "Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers," *International Journal of Security*, vol. 10, no. 3, pp. 137–150, 2015.

[211] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 1322–1333.

[212] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 601–618.

[213] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *arXiv preprint arXiv:1712.07557*, 2017.

[214] L. Jiang, R. Tan, X. Lou, and G. Lin, "On lightweight privacy-preserving collaborative learning for internet-of-things objects," in *Proceedings of the International Conference on Internet of Things Design and Implementation*, 2019, pp. 70–81.

[215] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. ACM, 2015, pp. 1310–1321.

[216] B. Hitaj, G. Ateniese, and F. Pérez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 603–618.

[217] Z. Gu, H. Jamjoom, D. Su, H. Huang, J. Zhang, T. Ma, D. Pendarakis, and I. Molloy, "Reaching data confidentiality and model accountability on the caltrain," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2019, pp. 336–348.

[218] A. Albaseer, B. S. Ciftler, M. Abdallah, and A. Al-Fuqaha, "Exploiting unlabeled data in smart cities using federated learning," *arXiv preprint arXiv:2001.04030*, 2020.

[219] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," *arXiv preprint arXiv:1812.00564*, 2018.

[220] A. Singh, P. Vepakomma, O. Gupta, and R. Raskar, "Detailed comparison of communication efficiency of split learning and federated learning," *arXiv preprint arXiv:1909.09145*, 2019.

[221] I. I. Eliazar and I. M. Sokolov, "Measuring statistical heterogeneity: The pietra index," *Physica A: Statistical Mechanics and its Applications*, vol. 389, no. 1, pp. 117–125, 2010.

[222] J.-S. Leu, T.-H. Chiang, M.-C. Yu, and K.-W. Su, "Energy efficient clustering scheme for prolonging the lifetime of wireless sensor network with isolated nodes," *IEEE Communications Letters*, vol. 19, no. 2, pp. 259–262, 2014.