8-9-2022 9:40 AM

Effective Resource Scheduling for Collaborative Computing in Edge-Assisted Internet of Things Systems

Qianqian Wang, *The University of Western Ontario*

Supervisor: Xianbin Wang, *The University of Western Ontario*
Co-Supervisor: Hongbo Zhu, *Nanjing University of Posts and Telecommunications*
A thesis submitted in partial fulfillment of the requirements for the Doctor of Philosophy degree in Electrical and Computer Engineering
© Qianqian Wang 2022

Abstract

Along with rapidly evolving communications technologies and data analytics, Internet of Things (IoT) systems interconnect billions of smart devices to gather, exchange, analyze data, and perform tasks autonomously, which poses a huge pressure on IoT devices' computing capabilities. Taking advantage of collaborative computing enabled by cloud computing and edge computing technologies, IoT devices can offload computation tasks to idle computing devices and remote servers, thus alleviating their pressure. However, scheduling resources effectively to realize collaborative computing remains a severe challenge due to diverse application objectives, limited distributed resources, and unpredictable environments. To overcome the above challenges, this thesis aims to design effective resource scheduling for collaborative computing in edge-assisted IoT systems.

First of all, horizontal collaboration amongst IoT devices is a promising way of balancing computing tasks within the device layer. However, engaging idle computing devices for sharing could be difficult as computation offloading potentially affects their local computing tasks. As an economic methodology, game theory-based methods contribute to revenue generation; thus, it is regarded as a suitable tool for addressing incentive problems. To incentivize horizontal collaborations, a hierarchical game model is first proposed in smart buildings to obtain maximum utilities for the building management systems (BMS) and idle computing devices (ICDs). The Stackelberg game model is built to analyze interactions between the BMS and ICDs, and the Cournot game model is presented to formulate internal competitions among multiple ICDs. Under the premise of the subgame perfect Nash equilibrium (SPNE), the BMS can quote the optimal pricing strategy, and ICDs can share the corresponding optimal amount of computing resources. Furthermore, to deal with unpredictability in emergency communication networks, an incomplete information-based two-tier game model is estimated. Depending on what the emergency management systems (EMS) and ICDs know, the Bayesian Nash equilibrium (BNE) is obtained under incomplete information that achieves better performances in terms of computation latency and participants' utilities. Finally, a new computational latency-based pricing scheme is designed from the perspective of the quality-of-experience (QoE) performance, where the computing offloading price varies dynamically with data processing rates. The interactive behaviors between the centralized computing sharing platform (CSP) and ICDs are modeled as the Stackelberg game, seeking out SPNE through the dynamic pricing mechanism, the computation workload selection, and the CPU frequency control. Through this scheme, the pressure of imbalanced computing capabilities in the device layer of IoT systems can be effectively relieved.

Furthermore, utilizing resources in the edge layer can effectively enhance the performance of IoT systems as edge systems generally have more powerful computing capabilities. However, due to the concurrent dynamics of application requirements, available resources, and network conditions, meeting the increasingly diverse requirements of IoT applications remains an ultimate challenge in vertical collaboration between edge systems and IoT devices. Towards this end, a device-specific QoE enhancement resource scheduling scheme is proposed through jointly optimizing communication and computation resources. Specifically, a three-layer QoE assessment model is first constructed to describe the general relationship between resource provisioning and device-specific QoE performance. Then, a two-stage resource scheduling scheme is proposed to realize simultaneous optimization of IoT devices and the edge system, where an online learning approach is designed on the edge system to schedule communication bandwidth and optimize computational rate. Based on the proposed two-stage resource scheduling scheme, IoT device-specific QoE performance can be effectively enhanced in edge-assisted IoT systems.

Keywords: Internet of things, collaborative computing, resource scheduling, game theory, reinforcement learning

Lay Summary

Along with rapidly evolving communications technologies and data analytics, Internet of Things (IoT) systems interconnect billions of smart devices for performing tasks autonomously, which poses a huge pressure on IoT devices' computing capabilities. Taking advantage of collaborative computing enabled by cloud computing and edge computing technologies, IoT devices can offload computation tasks to idle computing devices and remote servers, thus alleviating their pressure. However, scheduling resources effectively to realize collaborative computing remains a severe challenge due to diverse application objectives, limited distributed resources, and unpredictable environments. To overcome the above challenges, this thesis aims to design effective resource scheduling for collaborative computing in edge-assisted IoT systems.

To incentivize horizontal collaboration amongst IoT devices, a hierarchical game model is first proposed in smart buildings to obtain maximum utilities for the building management systems and idle computing devices (ICDs), which jointly combines the Stackelberg game and the Cournot game. Under the premise of the subgame perfect Nash equilibrium (SPNE), the BMS can quote the optimal pricing strategy, and ICDs can share the corresponding optimal amount of computing resources. Then, to deal with unpredictability in emergency communication networks, an incomplete information-based two-tier game model is estimated for analyzing the interactions between the emergency management systems (EMS) and ICDs. The Bayesian Nash equilibrium (BNE) is obtained depending on what the EMS and ICDs know. Furthermore, a new computational latency-based pricing scheme is designed from the perspective of the quality-of-experience (QoE) performance, where the computing offloading price varies dynamically with data processing rates. The interactive behaviors between the centralized computing sharing platform (CSP) and ICDs are modeled as the Stackelberg game, seeking out SPNE through the dynamic pricing mechanism, the computation workload selection, and the CPU frequency control. Finally, to meet the increasingly diverse requirements of IoT applications in vertical collaboration between edge systems and IoT devices, a device-specific QoE enhancement resource scheduling scheme is designed, where an online learning approach is proposed on the edge system to schedule communication bandwidth and computational rate simultaneously.

To my parents, husband, and daughter

Acknowledgments

I would like to express my deepest appreciation to my supervisor, Dr. Xianbin Wang, for all his guidance, patience, and support. His enlightening supervision inspired me to explore novel research areas and broadened my views in the research area. It was also his guidance and encouragement that helped me get prepare for my future career with all the professional skills. It was an incredible and rewarding journey to learn from him.

I also feel grateful to my co-supervisor, Dr. Hongbo Zhu. Thanks to his professional insights and technical guidance, I achieved exciting research findings and implemented my ideas into practice. It was my honor to work with him.

Sincere thanks to Dr. Mike Domaratzki, Dr. Hamada Ghenniwa, Dr. Longxiang Yang, Dr. Yan Zhang, and Dr. Yulong Zou for being my examination committee. I highly appreciate their precious time and constructive suggestions on my thesis and research.

Thanks to all members of our research group for the time that we spent both at work and after work. I would give my best regards to their success in both study and life. I was so lucky to meet such a big and warm research group. I would also like to extend my thanks to all my friends at UWO, who gave me strong support whenever and wherever I needed help.

As always, I feel so grateful to my parents, husband, and daughter. I highly appreciate their love and support throughout not only this degree but also in my life. They are always there for me and always back me up.

Contents

List of Figures

List of Tables

List of Abbreviations

A2C	actor-critic
A3C	advantage actor-critic
AP	access point
BMS	building management system
BNE	Bayesian Nash equilibrium
CN	core network
CPS	cyber physical system
CPU	central processing unit
CSP	computing sharing platform
DDPG	deep deterministic policy gradient
DNN	deep neural network
DPU	data transfer unit
DQN	deep Q-network
DRL	deep reinforcement learning
ECNs	emergency communication networks
EMS	emergency management system
GAE	generalized advantage estimation
ICDs	idle computing devices
IITG	incomplete information-based two-tier game
IPF	inversely proportional factors
ITU	international telecommunication union
LPU	local processing unit
MDP	Markov decision process
MEC	multi-access edge computing
MVNO	mobile virtual network operator
NE	Nash equilibrium
N-IITG	near-optimal IITG

PF	proportional factors
PPO	proximal policy optimization
QoE	quality of experience
QoE-RS	QoE enhancement resource scheduling
QoS	quality of service
RFID	radio frequency identification
RL	reinforcement learning
SAC	soft actor-critic
SPNE	subgame perfect Nash equilibrium

Chapter 1

Introduction

1.1 Overview of Edge-Assisted IoT Systems

The term "Internet of Things" (IoT) was coined by Kevin Ashton, the then executive director of the Auto-ID Center, in 1999 while his group was working on a global radio frequency identification (RFID)-based item identification system [1]. Along with rapidly evolving communications technologies and data analytics, IoT systems today are able to interconnect billions of smart devices for gathering, exchanging, analyzing data, and performing tasks autonomously [2, 3]. Such devices have been expanded to daily technical gadgets such as smartphones and wearables, smart home devices such as smart meters, as well as industrial devices like smart machines, which opens tremendous opportunities for a large number of novel applications that improve our quality of life. A new forecast from international data corporation estimates that there will be 41.6 billion connected IoT devices, or "things", generating 79.4 zettabytes of data in 2025 [4]. Due to such pervasive deployment and enlarging scale of IoT systems, increasingly complex applications put higher requirements on the computing capacity of smart devices.

Since Google proposed the concept of cloud computing in 2008 [5], cloud computing has been gradually accepted and introduced by IoT systems, breaking through the resource limitations of smart devices and providing users with high-demand applications. Cloud computing is a cost-effective model that provides abundant resources such as computing, communication, storage, and all necessary services in a simplified way: on-demand, regardless of the user's location and the type of smart device.

However, in recent years, the IoT system has put forward higher requirements for transmission bandwidth, latency, energy consumption, application performance, and reliability [6]. In conventional cloud computing, all data must be uploaded to centralized servers, and after computation, the results need to be sent back to IoT devices. This process creates tremendous pressure on the network, specifically in terms of the data transmission costs of bandwidth and resources. In this context, network performance significantly deteriorates with increasing data size on top of the cloud already being far from the users, which is unacceptable for time-sensitive IoT applications. Furthermore, most IoT devices have limited power, and heavy traffic load results in long transmission times, thus increasing power consumption costs.

To address such problem, by integrating these large amounts of idle resources distributed at the edge of networks to seamlessly provide services to users, a new computing paradigm - edge computing is proposed, which is regarded as the key technology and architectural concept for IoT systems [7, 8]. Edge computing moves the services originally located in the cloud to the proximity of users, which integrates the computing platform and the local network to provide powerful computing, storage, and communication capacities at the edge of IoT systems. Since the services provided by edge computing are closer to users, a better quality of experience (QoE) and quality of service (QoS) can be obtained by users.

Although cloud computing has difficulty meeting the high requirements of users in real-time response and low energy consumption, the edge computing paradigm itself cannot be a substitute for cloud computing because it does not have as powerful resource capacity as cloud computing. In some cases, however, the advantages of edge computing can be leveraged to offload computing services from the cloud to the edge to improve users' QoE. Accordingly, cloud computing and edge computing are complementary and mutually reinforcing. Therefore, edge-assisted IoT systems usually consist of three layers [9], i.e., device layer, edge layer, and cloud layer, as shown in Figure 1.1.

- **Device Layer**: is composed of various things, such as sensors, mobile phones, vehicles, surveillance cameras for smart cities, IoT devices for smart manufacturing, and IoT devices for smart health. This layer acts as the data source, i.e., things continuously generate and collect multiple types of data. According to things' resource capacities, the data can be processed locally or be offloaded to the edge and the cloud. In IoT system-

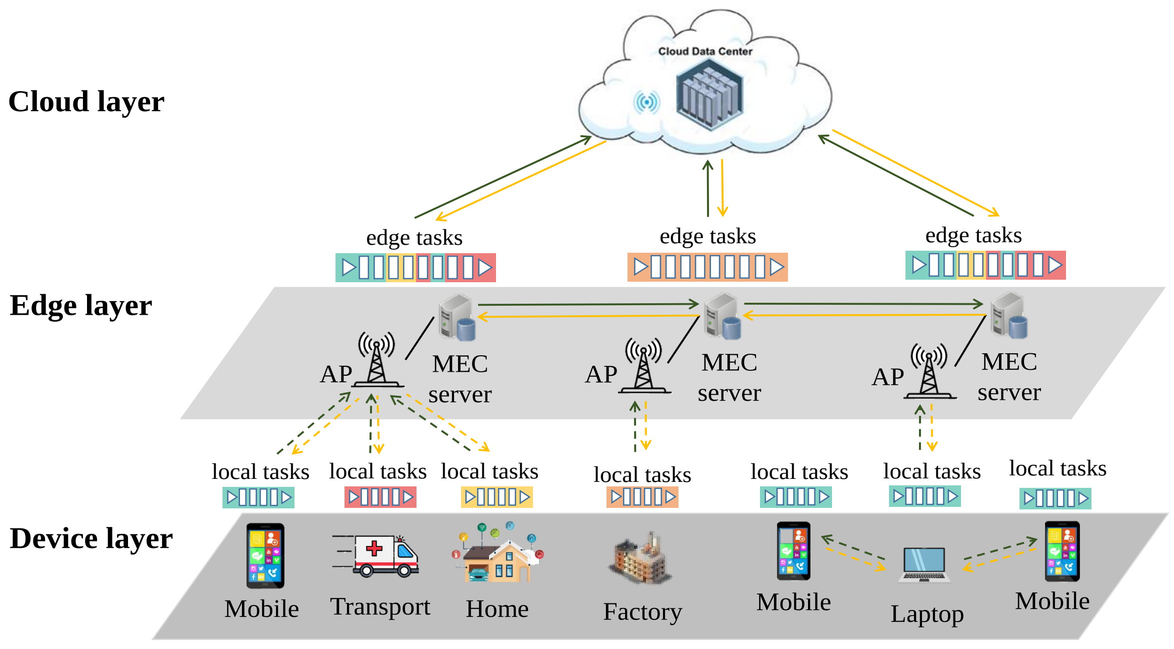


Figure 1.1: Illustration of three-layer IoT systems.

s, with the ever-increasing diversity of applications, different IoT devices usually incur specific resource demands to achieve their diverse application requirements.

- **Edge Layer**: as the core of the three-layer architecture, is an intermediate layer between the device layer and the cloud layer. From the perspective of hardware composition, the edge layer consists of various local networking and computing equipments, such as cellular base station, edge server, gateway, etc. Basically, the edge layer provides wireless access to smart devices through radio access technology and also has more powerful storage and computing capabilities than the device layer. Besides, since the edge and the cloud are complementary and mutually reinforcing, services in the cloud can be offloaded to the edge layer for load balancing and better QoE.

- **Cloud Layer**: consists of infrastructures, such as computing units, storage units, and micro data centers, connected with the edge layer through the core network (CN), i.e., backbone network. The cloud layer is undoubtedly the most powerful data processing and storaging center among the three layers. Although edge servers can process large amounts of data to reduce latency and energy consumption, the edge computing paradigm still requires the computing power and high-capacity storage infrastructure of the cloud to handle some tough tasks and global information.

By collaboratively utilizing the available computing resources in IoT devices, edge servers, and the cloud via communication technologies, IoT systems can realize the diverse application-s of IoT devices. As shown in Figure 1.1, such collaborations exist not only across different layers, i.e., vertical collaboration, but also among multiple entities in the same layer, i.e., horizontal collaboration. We elaborate on these two types of collaborations under the four specific cases as follows, i.e., device-device collaboration, device-edge collaboration, edge-edge collaboration, and device-edge-cloud collaboration.

- **Device-Device Collaboration**: The horizontal collaboration exists amongst IoT devices because of their imbalanced resources. The task generated from smart devices can be processed locally or offloaded to nearby devices with idle computing resources, which can effectively use the idle resources of the IoT device and alleviate the communication pressure of the local network.

- **Device-Edge Collaboration**: The device-edge collaboration manner involving the device layer and the edge layer is a vertical collaboration. The task generated from smart devices can be processed locally or offloaded to edge servers. Whether to offload these data depends on the device-edge collaboration strategy and smart devices' QoS and QoE requirements.

- **Device-Edge-Cloud Collaboration**: The device-edge collaboration manner has a relatively powerful capacity; however, it ignores the huge computing resources in the cloud computing center. With the ever-increasing smart devices and their resource-hungry applications, it will become increasingly difficult to rely on the resources in the edge layer alone to meet the IoT system's requirements. Therefore, it is particularly important and necessary to take full advantage of both edge computing and cloud computing and make them complementary to design a collaborative paradigm, i.e., the device-edge-cloud collaboration manner.

- **Edge-Edge Collaboration**: Generally, the edge-edge collaboration manner in edge computing does not arise in isolation. Instead, it usually comes along with the device-edge collaboration manner or the device-edge-cloud collaboration manner. Through an edge-edge collaboration manner, there is one more option for task processing.

1.2 Challenges of Resource Scheduling in Edge-Assisted IoT Systems

Taking advantage of collaborative computing enabled by cloud computing and edge computing, IoT devices can offload their computation tasks to idle IoT devices and remote servers, thus enhancing the performance of IoT applications deployed on them. However, the resources are distributed and scattered in large-scale IoT systems. It is a waste of resources if scattered ones can not be efficiently utilized by resource scheduling. For example, a recent survey indicates that the average central processing unit (CPU) utility of existing computing devices over the Internet is merely 6 to 12 percent [10]. If an efficient resource strategy is applied, they can be combined to establish an available and cost-effective computing resource pool, which helps alleviate the workloads of IoT systems and promote the distributed computing environment. Besides, with the ever-increasing diversity of applications, different IoT devices within the same IoT system inevitably incur specific resource demands to achieve their diverse application requirements. An effective resource scheduling should jointly consider their interests and improve the system utility accordingly.

Therefore, effective resource scheduling is crucial for realizing collaborative computing in edge-assisted IoT systems, but it also faces several challenges. Specifically, the critical challenges can be summarized as follows.

- **Dynamic and Diverse Objectives from Heterogeneous Applications**: Computing tasks process data generated from users, which are generally time-varying. Furthermore, the task types may vary based on heterogeneous application scenarios for diverse objectives. For example, connected and autonomous vehicles in intelligent transportation systems need to process data within several milliseconds for traffic safety; thus, low latency is their main objective. The unmanned aerial vehicles usually focus more on long battery life; thus, the objective of low energy consumption is expected during data processing. These dynamic and diverse objectives can also be referred to as performance indicators, making resource scheduling more complex.

- **Imbalanced Computing Capability amongst IoT Devices**: IoT devices are not only generators of data but also can be processors of data. Due to different computing capabilities amongst IoT devices, some struggle to cope with computing tasks, while others always remain idle. Device-device collaboration can effectively and efficiently balance the computing tasks amongst IoT devices by scavenging the enormous amount of spare computational resources. Although the concept of collaborative computing is promising, engaging idle computing devices for sharing could be difficult as they have no commitments to do so. They may expect compensation since computation offloading potentially affects local computing tasks.

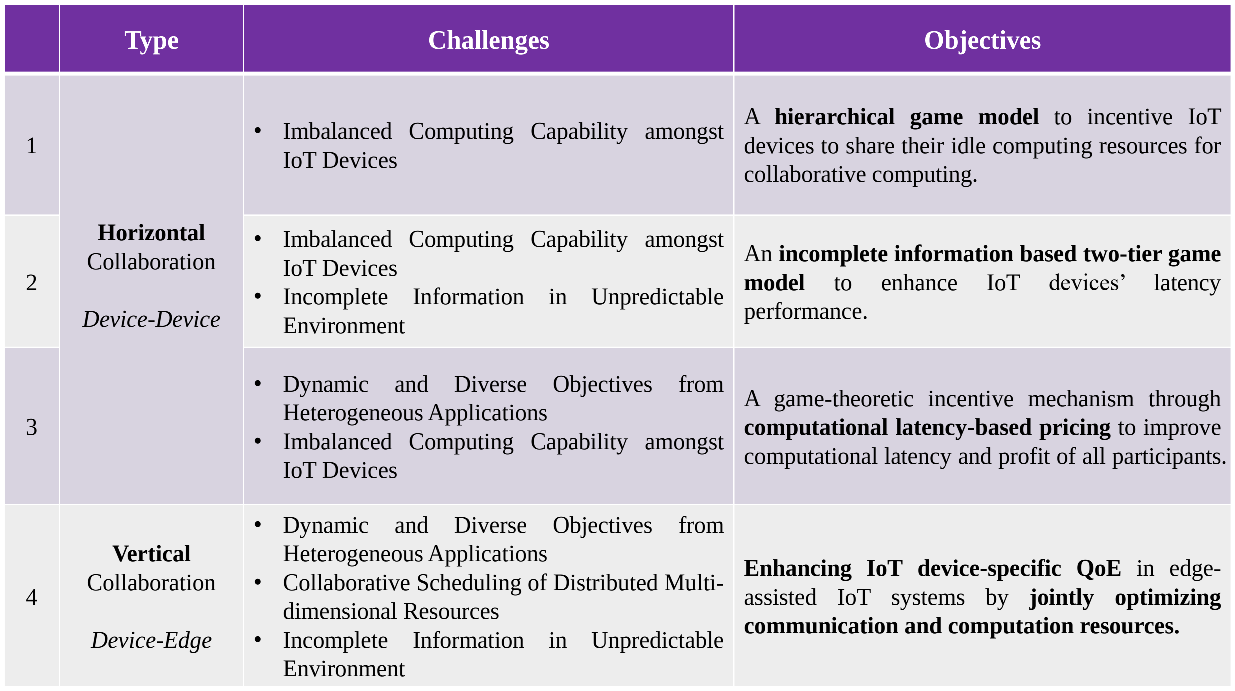
- **Collaborative Scheduling of Distributed Multi-dimensional Resources**: Various resources exist in the edge network distributively, by which the powerful serviceability is provided, and the tasks can be completed. The resources in edge networks can be categorized into three types, i.e., communication resources, storage resources (also as caching resources), and computing resources. Enhancing one performance indicator often involves multi-dimensional resources. Orchestrating these limited resources to process data better requires appropriate resource scheduling strategies. Furthermore, the ever-increasing diversity of application requirements makes resource scheduling more challenging.

- **Incomplete Information in Unpredictable Environment**: Considering that IoT systems are generally time-varying, e.g., network disconnection, channel state, and backhaul latency, real-time scheduling schemes are required to enhance the QoE or QoS of IoT devices in such an unpredictable environment. Furthermore, due to privacy and security concerns, IoT devices may be reluctant to expose their personal information, such as network connection quality and preference for energy efficiency, which is extremely common in edge-assisted IoT systems but presents a more significant challenge for resource scheduling.

Therefore, while collaborative computing significantly strengthens the serviceability of IoT systems by providing powerful computing, storage, and communication capacities, it also requires appropriate resource scheduling strategies to solve such complex problems.

1.3 Research Objectives of the Thesis

Considering the challenges mentioned above, the specific research objectives in this thesis are identified in Figure 1.2.



	Type	Challenges	Objectives
1	**Horizontal** Collaboration *Device-Device*	• Imbalanced Computing Capability amongst IoT Devices	A **hierarchical game model** to incentive IoT devices to share their idle computing resources for collaborative computing.
2		• Imbalanced Computing Capability amongst IoT Devices • Incomplete Information in Unpredictable Environment	An **incomplete information based two-tier game model** to enhance IoT devices' latency performance.
3		• Dynamic and Diverse Objectives from Heterogeneous Applications • Imbalanced Computing Capability amongst IoT Devices	A game-theoretic incentive mechanism through **computational latency-based pricing** to improve computational latency and profit of all participants.
4	**Vertical** Collaboration *Device-Edge*	• Dynamic and Diverse Objectives from Heterogeneous Applications • Collaborative Scheduling of Distributed Multi-dimensional Resources • Incomplete Information in Unpredictable Environment	**Enhancing IoT device-specific QoE** in edge-assisted IoT systems by **jointly optimizing communication and computation resources.**

Figure 1.2: Research objectives of the thesis

- **Hierarchical game model**: The horizontal collaboration amongst IoT devices is a promising way of balancing the computing tasks within the device layer. However, engaging idle computing devices for sharing could be difficult as they have no commitments to do so. They may expect compensation since computation offloading potentially affects local computing tasks. As an economic methodology, the primary and most important benefit of game theory-based approaches is revenue generation. Thus, it is regarded as a suitable tool for addressing incentive problems. In this topic, we consider a set of computing devices (*data processors, sellers*) to share their idle resources to alleviate the pressure of data processing for a resource-limit device (*data generator, buyer*). Specifically, the *data generator* dynamically determines the pricing strategy based on its demand and the availability of idle computing resources of multiple *data processors*. Then, the *data processors* share the corresponding optimal amount of computing resources. The competitions exist not only between the *data generator* and the *data processors* but also among multiple *data processors* since they belong to different owners. In this context, to encourage horizontal collaboration amongst IoT devices, a computation sharing architecture based on the hierarchical game model, combining the Stackelberg game and the Cournot game, is proposed to maximize the utilities of all participants.

- **Incomplete information based two-tier game model**: Due to privacy and security concerns, IoT devices may be reluctant to expose their personal information. Furthermore, the network environment is time-varying, such as path loss fading and computation capacities. Therefore, it is extremely difficult to acquire the complete information in advance for resource scheduling in most realistic scenarios. In this topic, we consider the same scenario as objective one, but to deal with the inevitable challenge of real-time data analysis without the complete information in device-device collaborative computing. To solve such a problem, an incomplete information-based two-tier game model is estimated, which can seek the Bayesian Nash equilibrium (BNE) under incomplete information for optimal resource scheduling in IoT systems.

- **Computational latency-based pricing**: With the ever-increasing diversity of applications in IoT systems, different devices require specific resource demands to achieve their diverse application requirements. In this topic, our design is motivated by computing scenarios with latency-sensitive tasks. Different data processing rates lead to different QoE for IoT applications; thus, a reasonable pricing strategy should consider not only the number of tasks processed but also the QoE of tasks processed. Therefore, to enable collaborative computing from the perspective of QoE performance at edge-assisted IoT systems, a computational latency-based pricing mechanism is proposed, where the unit price offered by the *buyer* varies dynamically with the data processing rates that the *seller* can provide. In this way, a higher data processing rate is encouraged by gaining more payoff to improve computational latency performance.

- **Device-Specific QoE Enhancement**: Existing studies on resource scheduling in edge computing have mainly utilized a specified QoS parameter as an optimization objective, such as energy efficiency, service latency minimization, cost efficiency, revenue maximization, etc. As an objective measure, QoS is an effective indicator for evaluating the overall network performance. However, with increasingly diverse requirements from IoT applications, the acceptability of the same QoS may be significantly distinct across heterogeneous IoT devices, which directly reduces the effectiveness of QoS-based approaches when dealing with IoT device-specific demands. Consequently, the objective of resource scheduling in edge networks has gradually changed to improve QoE rather than QoS. In this topic, to enhance IoT device-specific QoE performance, a three-layer QoE assessment model is constructed to describe the relationship between resource provisioning and device-specific QoE performance. Then, to maximize the overall QoE performance amongst IoT devices, a two-stage resource scheduling scheme is proposed to simultaneously optimize multi-dimensional resources in IoT systems.

1.4 Technical Contributions of the Thesis

The main contributions of this thesis are summarized as follows:

- To incentivize idle computing devices (ICDs) to offload computational tasks for the building management system (BMS), a hierarchical game model is proposed to obtain the maximum utility for the BMS and the ICDs, which reflect the dynamic relationships between the BMS and multiple ICDs by jointly combining the Stackelberg game and the Cournot game. To guarantee the utility of BMS and ICDs, the Stackelberg game model is built to analyze the interactions between BMS and ICDs, where the BMS, acting as a single leader, sets the appropriate offloading pricing strategy; then, the different ICDs, acting as multiple competitive followers, derive the amounts of computing resources to share. The Cournot game model is presented to formulate the internal competition among multiple ICDs by which each ICD simultaneously determines the optimal shared computing resources by considering other ICDs' strategies. Furthermore, a near-optimal algorithm is presented to achieve the subgame perfect Nash equilibrium (SPNE) of the BMS and ICDs. With this approach, the on-demand computing capacity of BMS can be effectively improved.

- To deal with the enormous challenge of real-time data analysis without the complete information in emergency communication networks, an incomplete information-based two-tier game model (IITG) is estimated, where the objective is to maximize the utilities of the emergency management systems (EMS) and ICDs. Specifically, the interactions between the EMS and ICDs are formulated as a two-tier game model, by jointly combining the Stackelberg game and the Cournot game. Through this model, the EMS can dynamically optimize its pricing mechanism, and ICDs can select the optimal computation workload accordingly. Furthermore, depending on what the EMS and ICDs know, the BNE is acquired under incomplete information, and a near-optimal IITG (N-IITG) algorithm is developed to reach the unique BNE by iterations. The N-IITG algorithm can achieve a near-optimal performance of complete information and outperforms the existing incomplete information-based methods regarding computation latency and participants' utilities.

- To enable collaborative computing from the perspective of the QoE performance, a new computational latency-based pricing scheme is designed where the computing offloading price varies dynamically with data processing rates. Then, a game-theoretic computing task allocation approach is developed among a centralized computing sharing platform (CSP) and multiple ICDs to maximize all participants' profit. The interactive behaviors between the CSP and ICDs are modeled as the Stackelberg game, seeking out SPNE through the dynamic pricing mechanism, the computation workload selection, and the CPU frequency control. By our proposed scheme, the pressure of imbalanced computing capabilities in IoT systems can be effectively relieved. Furthermore, the overall computational latency is significantly decreased, and the profit of all participants achieves maximum in collaborative computing.

- To enhance IoT device-specific QoE performance, a three-layer QoE assessment model is constructed to describe the general relationship between resource provisioning and IoT device's QoE performance, which can help resource schedulers better understand the fulfillment of different IoT devices' interests. Then, to maximize the overall QoE amongst IoT devices, a two-stage resource scheduling scheme is proposed to realize simultaneous optimization of IoT devices and the edge system. Specifically, in stage I, considering the resource-constrained nature, a distributed resource scheduling algorithm with low complexity is proposed for each IoT device to optimize its local computing processing rate; in stage II, a proximal policy optimization (PPO)-based online resource scheduling approach is designed on the edge system to optimize its bandwidth allocation and computing processing rate by interacting with multiply IoT devices without prior knowledge of their specific QoE assessment models. Based on the proposed two-stage resource scheduling scheme, IoT device-specific QoE performance can be effectively enhanced in edge-assisted systems.

1.5 Organization of the Thesis

The following details demonstrate the organization of the remaining chapters of this thesis.

A comprehensive study of resource scheduling in edge-assisted IoT systems is conducted in Chapter 2. The popular applications in IoT systems are elucidated firstly. Afterwards, the fundamentals of resource scheduling in IoT systems are introduced, including computing task offloading models, IoT resources' characteristics, and scheduling objectives. Finally, the existing methodologies and their challenges are analyzed, including game theory-based methods and reinforcement learning-based methods.

In Chapter 3, considering the collaborative computing scenario in smart buildings, a computation sharing architecture is proposed to incentivize ICDs to process computational tasks for the BMS, which combines the Stackelberg game and the Cournot game. To guarantee the utility of BMS and ICDs, the Stackelberg game model is built to analyze the interactions between BMS and ICDs. Then, the Cournot game model is presented to formulate the internal competition among multiple ICDs. Under the premise of the SPNE, the BMS can quote the optimal pricing strategy, and the ICDs can share the corresponding optimal amount of computing resources. The simulation results demonstrate that the proposed solution can effectively improve the on-demand computing capacity of the BMS.

In Chapter 4, an IITG model is estimated to deal with the enormous challenge of real-time data analysis without the complete information in emergency communication networks. IITG realizes collaborative computing by incentivizing ICDs to share computation resources, which jointly combines the Stackelberg game and the Cournot games. Depending on the given information of the EMS and the ICDs, we analyze the BNE of the EMS's pricing strategies and the ICDs' computing resource sharing strategies under incomplete information and further design the N-IITG algorithm that can iteratively convergent to the unique BNE. According to the simulation results, the proposed algorithm achieves a significant increase in computational capacity while each participant obtains the optimal profit.

In Chapter 5, a computational latency-based pricing mechanism from the perspective of the QoE performance is proposed to enable collaborative computing in edge-assisted IoT systems. A game-theoretic computing task allocation approach is developed among a centralized CSP and multiple ICDs to maximize all participants' profit. The CSP first determines the optimal task partition dynamically upon the task arrival; then, the ICDs derive the optimal central processing unit-cycle frequency correspondingly. Simulation results show that the proposed scheme can effectively relieve the pressure of unbalanced computing capabilities in IoT. Furthermore, the overall computational latency of our proposed mechanism is significantly decreased, and the profit of all participants achieves maximum in collaborative computing.

In Chapter 6, a three-layer QoE assessment model is constructed to describe the general relationship between resource provisioning and device-specific QoE performance. Then, to maximize the overall QoE performance amongst IoT devices, a two-stage resource scheduling scheme is proposed to realize simultaneous optimization of IoT devices and the edge system. Specifically, in stage I, a distributed resource scheduling algorithm with low complexity is designed for each IoT device to optimize the local computing rate by considering its resource-constrained nature; in stage II, a PPO-based online approach is proposed on the edge system to schedule communication bandwidth and optimize computational rate by interacting with multiply IoT devices without prior knowledge of their specific QoE assessment models. Extensive experiments demonstrate that our proposal outperforms the existing works from the perspective of QoE performance.

Finally, all the contributions are summarized in Chapter 7, with the identification of future research directions.

Chapter 2

Background Study on Resources Scheduling in Edge-Assisted IoT Systems

With the progressing development of information technologies, IoT systems are able to interconnect billions of smart devices that collect and exchange data amongst themselves automatically through modern communication network infrastructures [11, 12]. Without human intervention, the generated massive data demands intelligent processing, which poses a huge challenge to data processing capabilities of IoT devices. Taking advantage of collaborative computing enabled by cloud computing and edge computing technologies, IoT devices can offload their computation tasks to other idle computing devices or remote servers, thus enhancing the performance of IoT applications deployed on them. However, as mentioned in Chapter 1, realizing collaborative computing through effective resource scheduling remains a severe challenge in edge-assisted IoT systems. This chapter provides a comprehensive study of resource scheduling in IoT systems. The popular applications in IoT systems are elucidated firstly. Afterwards, the fundamentals of resource scheduling in IoT systems are introduced, including computing task offloading models, IoT resources' characteristics, and scheduling objectives. Finally, a comprehensive survey of the existing methodologies for resource scheduling is presented as well as an analysis of their challenges.

2.1 Applications in IoT systems

The ultimate goal of resource scheduling in IoT systems is to support IoT applications effectively and efficiently. Therefore, before the detailed investigations on the fundamentals of resource scheduling, the typical IoT applications are first introduced, which would help understand the objectives of resource scheduling better.

The IoT applications have experienced phenomenal growth throughout the last decades, ranging from health care to smart cities and industrial automation. Four typical IoT applications are presented here, including smart homes [13], self-driven cars [14], health care [15, 16], and industrial IoT [17, 18].

- **Smart Homes**: One of the best and the most practical applications of IoT systems, smart homes integrate different communication schemes and optimization algorithms to predict, analyze, optimize and control services, such as smart thermostats and air conditioners, speakers, smoke and motion detectors, etc., helping us make our life more comfortable and safer. Although in the same IoT system, different services still demand different requirements. For instance, the smart air conditioner is one of the schedulable loads having a predictable operating pattern that can be controlled via the management system regularly; thus, data accuracy and energy efficiency are more important. And for some non-schedulable loads, such as smoke and motion detectors, it is a critical event that requires immediate action; thus, latency will be a prime concern.

- **Self-driven Cars**: In 2009, Google launched its self-driving car project with the goal of driving more than 10 uninterrupted 100-mile routes. In 2016, self-driving technology company Waymo became a subsidiary of Alphabet, and Google's self-driving project was renamed Waymo. These cars use multiple sensors and embedded systems connected to the cloud and the Internet to generate and send data to the cloud for informed decision-making through machine learning. Since we are dealing with human lives on the road, it needs to keep passengers and those on the road safe. Therefore, reliability is essential to guarantee this security.

- **Health Care**: The IoT system has the potential to lead to many medical applications, such as remote health monitoring, fitness programs, and elderly care. Various medical devices, sensors, and diagnostic and imaging equipment can be considered as smart devices or objects that form a core part of IoT. Ease of cost-effective interaction through seamless and secure connections across individual patients, clinics, and health care organizations is an important trend. The latest health care networks powered by wireless technology are expected to support chronic diseases, early diagnosis, real-time monitoring and medical emergencies. Gateways, medical servers, and health databases play a vital role in creating health records and delivering on-demand health services to authorized stakeholders.

- **Industrial IoT**: With the introduction of IoT and Cyber-Physical Systems (CPS) concepts in industrial application scenarios, industrial automation is undergoing dramatic changes. The industrial IoT consists of interconnected sensors, instruments, and other devices connected to computerized industrial applications such as manufacturing, energy management, and more. Reliable exchange of information is a key part of the industrial IoT. Applying the ideas of CPS and IoT to the field of industrial automation has led to the definition of the concept of Industry 4.0, where 4.0 alludes to the fourth industrial revolution enabled by Internet technology to create smart products, smart production, and smart services.

According to the introduction of typical IoT applications, we can notice that diverse IoT applications usually pursue different objectives, which rely on efficient resource scheduling in IoT systems.

2.2 Fundamentals of Resource Scheduling

Resource scheduling refers to the set of methodologies that participants used to efficiently assign resources to the tasks that need to complete, and achieve the objectives of participants based on resource availability. Therefore, the main factors in resource scheduling includes tasks, resources, objectives, and methodologies, as illustrated in Figure 2.1.

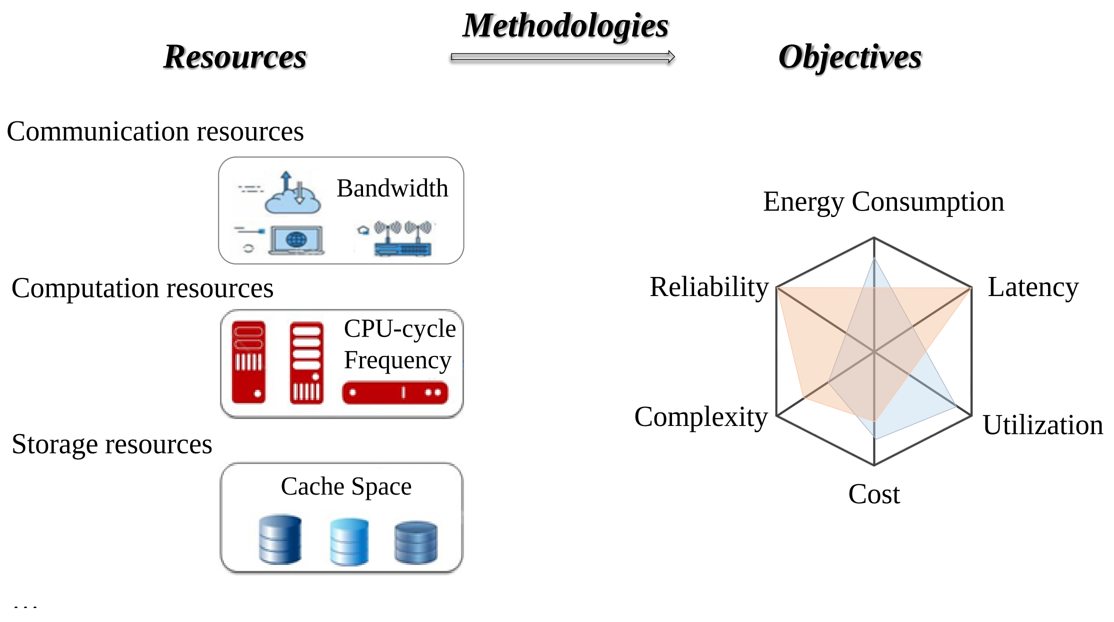


Figure 2.1: Fundamentals of resource scheduling.

2.2.1 Computing Task Offloading Models

According to the above typical IoT applications, the ultimate goal of IoT systems is to make timely and reliable decisions and provide customized services by fully utilizing data collected from IoT devices and environments. The key point is to achieve intelligent data processing for computing tasks without human interactions. This subsection introduces two computation task offloading models popularly used in existing literature in IoT systems [19], corresponding to binary and partial computation offloading, respectively.

Binary Offloading: A highly integrated or relatively simple task cannot be partitioned and has to be executed as a whole either locally at the IoT device or offloaded to remote servers, called binary offloading. In this model, each computation task is described as a 3-tuple $\{w^{\text{comm}}, w^{\text{comp}}, x\}$, where w^{comm} indicate the data amount (in byte) for transmission and w^{comp} denotes computation workload (in CPU cycle/s) to be processed. The parameter $x = \{0, 1\}$ indicates where the task will be processed; for example, $x = 0$ refers to local processing, and $x = 1$ refers to remote processing.

Partial Offloading: In practice, many IoT applications are composed of multiple components, making it possible to implement fine-grained (partial) computation offloading. Specifically, the tasks can be partitioned into two parts, i.e., one executed at the IoT device and the other offloaded for remote execution. The simplest task model for partial offloading is the data-partition model, where the task-input bits are bit-wise independent and can be arbitrarily divided into different groups and executed by different entities in IoT systems, e.g., parallel execution at the IoT devices and remote servers. In this model, each computation task can still be described as a 3-tuple $\{w^{\text{comm}}, w^{\text{comp}}, x\}$, where w^{comm} and w^{comp} represent the data amount and computation workload, respectively, similar to the binary offloading model. But unlike binary offloading model, the parameter $x \in [0, 1]$ indicates the proportion of data partition for partial offloading. When x equals to 0, it indicates that the task will be processed locally as a whole while it would be entirely offloaded to remote servers when $x = 1$.

2.2.2 Characteristics of Resources in IoT systems

Various resources exist in the IoT system, by which the powerful serviceability is provided, and the tasks can be completed. The resource involved in collaborative computing in edge-assisted IoT systems mainly includes communication resources, computing resources, and storage resources.

Communication resources in IoT systems usually represents the bandwidth and time assigned for wireless communication. Given the assigned wireless bandwidth B, the data transmission rate is denoted as tr, which can be characterized by various wireless transmission models based on Shannon's formula, e.g., $tr = B\log_2(1 + ph/w_0)$, where p is the transmission power of the data sender, and w_0 denotes the white Gaussian noise power. The channel gain h is generally affected by the path loss inverse to the distance between the data sender and receiver. Accordingly, the communication time T^{comm} can be given by $T^{\text{comm}} = w^{\text{comm}}/tr$.

Computation resources in IoT systems usually indicate the processing capability (i.e., the amount of CPU frequency in cycle/s) assigned to the process application and its processing time. When the computation task w^{comp} is performed locally, we usually assume the single-core in each IoT device and the processing capability assigned to process application w^{comp} is f. Then, the local computing time $T^{\text{comp,l}}$ equals to $(1 - x)w^{\text{comp}}/f$. On the other hand, multiple cores are assumed in remote servers, e.g., edge servers and cloud servers, and then remote computing time $T^{\text{comp,r}}$ equals to xw^{comp}/LF, where L and F are the number of cores in remote servers and the processing capability assigned to each core, respectively.

Storage resources in IoT systems are referred to as the cache size, which allocation is usually related to other types of resources. For example, caching can be a viable solution for saving network bandwidth in a large-scale IoT network by utilizing the sensing data and information stored in the cache, as long as the information value is not outdated and temporally valid.

2.2.3 Taxonomy of Objectives

Currently, the objective of resource scheduling in IoT systems can be roughly divided into two types, i.e., QoS-based objective and QoE-based objective.

QoS is the description or measurement of the overall performance of a service, such as a wireless communication network or a cloud computing service, particularly the performance achieved from the network level [20]. Several related aspects of the network service are often considered to quantitatively measure QoS, such as packet loss, bit rate, throughput, transmission delay, availability, jitter, etc.

QoE has historically emerged from QoS, which is a measure of the delight or annoyance of a customer's experiences with a service. QoS measurement is most of the time not related to a customer but to the network itself. QoE, however, is a subjective measure from the user's perspective of the overall quality of the service provided by capturing the user's aesthetic and hedonic needs. As defined by the international telecommunication union (ITU) in recommendation ITU-T P.10 in 2016 [21], QoE refers to "the degree of delight or annoyance of the user of an application or service. It results from the fulfillment of his or her expectations with respect to the utility and/or enjoyment of the application or service in the light of the user's personality and current state". QoE aims to consider every factor contributing to a user's perceived quality of a system or service, mainly includng system, human and contextual factors. The following so-called "influence factors" have been identified and classified by Reiter *et al.* [22], as shown in Figure 2.2.

Furthermore, as a measure of the end-to-end performance at the service level from the user's perspective, QoE is an important metric for designing systems and engineering processes. The

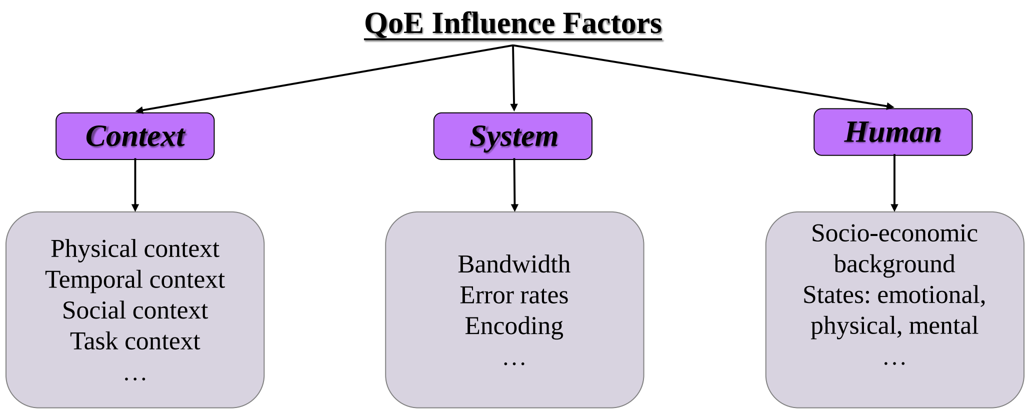


Figure 2.2: QoE influence factors.

existing common QoE measurement methods include mean opinion score, subjective quality evaluation, and objective evaluation methods. The mean opinion score is a widely used measure for assessing the quality of media signals. However, it is a limited form of QoE measurement relating to a specific media type in a controlled environment without explicitly considering user expectations. Furthermore, subjective quality evaluation requires a lot of human resources, establishing it a time-consuming process. Objective evaluation methods can provide quality results faster but require dedicated computing resources.

2.3 Existing Methodologies for Resource Scheduling and Their Challenges

2.3.1 Game Theory-based Methods

Game theory is the study of mathematical models of strategic interactions among rational agents [23, 24]. It has been widely recognized as an important tool in many fields, such as social science, systems science, and computer science. In IoT systems, devices are required to be "smart". Particularly, IoT devices can not only perform normal functions, e.g., sensing information from the surrounding environment, but also make optimal decisions without or with minimal human intervention, given their constrained resources and the dynamic of the environment for the requested IoT services. In addition, with billions of devices connecting to

IoT systems, it leads to many challenges in efficiently controlling and managing IoT devices. Consequently, new approaches with higher efficiency and more flexibility to adapt to dynamic IoT systems need to be developed. Apart from classical approaches, e.g., optimization-based approaches, economic approaches have been widely applied in IoT systems for resource scheduling [25, 26, 27, 28].

The typical definition of a normal game consists of three factors, i.e., 3-tuple $\mathcal{G} = (\mathcal{N}, \mathcal{S}, u)$.

- Players: $\mathcal{N} = \{1, 2, \ldots, N\}$ is a finite set of N, indexed by i,
- Strategies set for player i S_i: $s = (s_1, s_2, \ldots, s_N) \in S = S_1 \times S_2 \times \ldots \times S_N$ is a strategy profile,
- Utility function or Payoff function for player i: $u_i : S \to \mathbb{R}$, $u = (u_1, u_2, \ldots, u_N)$ is a profile of utility function.

Game theorists use Nash equilibrium (NE) to analyze the outcome of the strategic interaction of several players. Informally, a strategy profile can achieve a NE if no player can do better by unilaterally changing his strategy. Formally, let S_i be the set of all possible strategies for player i, where $i = 1, \ldots, N$. Let $s^* = (s_i^*, s_{-i}^*)$ be a strategy profile, a set consisting of one strategy for each player, where s_{-i}^* denotes the $N - 1$ strategies of all the players except i. Let $u_i(s_i, s_{-i}^*)$ be player i's payoff as a function of the strategies. The strategy profile s^* is a NE if

$$u_i(s_i^*, s_{-i}^*) \geq u_i(s_i, s_{-i}^*), \quad \text{for all } s_i \in S_i. \tag{2.1}$$

There are several classifications of game theory as

- *Cooperative and Non-cooperative*: A game is cooperative if the players can form binding commitments that are enforced externally, such as through contract law. If players cannot form alliances, or if all agreements need to be self-enforcing, then the game is uncooperative. Cooperative game theory provides an advanced approach because it describes only the structure, strategy, and payoffs of coalitions, whereas non-cooperative game theory also looks at how the bargaining process will affect the distribution of payoffs within each coalition. Since non-cooperative game theory is more common, cooperative games can be analyzed by means of non-cooperative game theory, provided that sufficient assumptions are made to cover all possible strategies available to players. While using a single theory may be desirable, in many cases there is insufficient information to accurately model the formal procedures available during strategic negotiation, or the resulting models are too complex to provide practical tools in the real world. In this context, cooperative game theory provides a simplified approach that allows the analysis of the entire game without making any assumptions about bargaining power.

- *Perfect / Imperfect information and complete / incomplete information*: A game is one of perfect information if all players at every step in the game know what all the other players have done previously. In practice, this can be applied to companies and consumers who have price and quality information on all available goods on the market. Incomplete information games, such as simultaneous move games, are played when the player does not know all actions that the opponent has made. Most games studied in game theory are games of incomplete information. Perfect information is often confused with complete information, which is a similar concept. Complete information requires each player to know the strategies and payoffs available to other players, but not necessarily the actions taken, whereas perfect information is knowledge of all aspects of the game and the player. However, incomplete information games can be reduced to imperfect information games by introducing "moves by nature".

- *Simultaneous and Sequential*: Simultaneous games are games in which two players move at the same time, or conversely, where the latter player is unaware of the actions of the earlier players. Sequential games, or dynamic games, are games where late players have some knowledge of early actions, which doesn't need to be perfect information about every movement of the early players; there may be little knowledge. For example, a player may know that an earlier player did not perform a particular action, but they do not know what other available actions were performed by the first player. The difference between simultaneous and sequential games is reflected in the different representations. Typically, the normal form, as a game matrix, is used to represent simultaneous games, while the extensive form, as a game tree, is used to represent sequential games. The transformation from extensive to normal form is a way that multiple extensive games correspond to the same normal form. Therefore, the concept of equilibrium for simultaneous games is not sufficient to reason about sequential games, which usually seek sub-game NE.

According to the game theory adopted in this thesis, we will briefly introduce some popular models as follows.

Stackelberg game [29] is a strategic game in which the leader player moves first, and then the follower player moves in turn. The Stackelberg model can be solved to find the SPNE or equilibria, i.e., the strategy profile that best serves each player, given the other player's strategies, and that requires every player playing in a NE in every sub-game. This model is solved by backward induction. The leader picks a quantity that maximizes its payoff, anticipating the predicted response of the follower. The follower actually observes this and chooses the expected quantity in equilibrium as a response.

We take 2-player Stackelberg game for example. To calculate the SPNE, the best response functions of the follower must first be calculated. Given s_1 from player 1, player 2 chooses s_2 at the second stage as

$$s_2^* = \arg\max_{s_2 \in S_2} u_2(s_1, s_2), \tag{2.2}$$

where the best response $s_2^*(s_1)$ of the follower is a function with regard to the strategy of the leader s_1. Now the best response function of the leader is considered in the first stage by accounting for the follower's response, which is given by

$$s_1^* = \arg\max_{s_1 \in S_1} u_1(s_1, s_2^*(s_1)). \tag{2.3}$$

Thus, the equilibrium outcome, i.e., SPNE, is that the leader plays s_1^*, and the follower plays $s_2^*(s_1)$.

Cournot game [30] involves players choosing the quantity of a homogenous product to produce independently and simultaneously, where marginal cost can be different for each player and the player's payoff is profit. Considering the cost of production is public information, the player's goal is to find their profit-maximizing quantity based on what they believe other players will produce and behave like monopolies. In this game, players want to produce in monopoly quantities, but have a strong incentive to deviate and produce more, which lowers the market-clearing price.

We still take 2-player Cournot game for example. To calculate the NE, two players need to choose their strategy simultaneously by considering the other player's strategy as

$$\begin{cases} s_1^* = \arg\max_{s_1 \in S_1} u_1(s_1, s_2) \\ s_2^* = \arg\max_{s_2 \in S_2} u_2(s_1, s_2) \end{cases}. \tag{2.4}$$

The Cournot equilibrium is reached when each player operates on their reaction function with no incentive to deviate, as they have the best response based on the other players' output.

Bayesian game [31] is a game that models the outcome of player interactions using aspects of Bayesian probability. Bayesian games are notable because they allowed the specification of the solutions to games with incomplete information for the first time in game theory. In a Bayesian game, one must specify strategy spaces, type spaces, payoff functions, and prior beliefs. Let Θ denote the set of all probability distributions on a set S. A Bayesian game is a tuple $(\mathcal{N}, \mathcal{S}, \Theta, p, u)$ where $\mathcal{N}$ and $\mathcal{S}$ indicate the set of players and strategies, respectively, similar to the normal game. The left three factors are

- Θ_i is a set of types for player i,
- $p : \Theta$ is a joint distribution of types,
- $u_i : S \times \Theta \rightarrow \mathbb{R}$ is a profile of utility function.

A BNE is defined as a strategy profile that maximizes the expected payoff for each player given their beliefs and strategies played by the other players.

In IoT systems, compared with traditional optimization, economics-based methods are effective approaches for resolving computing allocation problems since the prospect of achieving optimal revenue attracts each participant. We note that Stackelberg game-based approaches have been widely used for modeling and analyzing the computing offloading problem. In the Stackelberg game model, one player (leader) moves first, and then the others (followers) move sequentially, and each player can maximize its payoff by finding the SPNE, which aligns well with the interactions between the service providers and the service consumers in computing offloading scenarios. Furthermore, to exploit the interactions among multiple idle computing devices in the computation sharing scenario, the Cournot game model can be introduced since it is one of the most popular types of games used to model the interactions among multiple strategic generators.

However, trading between the data generators and the date processors should satisfy individual requirements with the goal of maximizing their payoff. Data processors must be incentivized to offer their computing resources, while data generators require enhancing their computational capacity to satisfy their specific objectives. The competitions exist not only between data generators and data processors but also among multiple data processors. Moreover, the existing literature is largely based on the assumption that the competitive participants have the full knowledge of the network and local constraints, such as the path loss fading and computation capacities. However, it is extremely difficult to acquire such information in advance in most realistic IoT systems due to its dynamics and uncertainty. Therefore, it is essential to design effective resource scheduling mechanisms based on game theory to address these challenges.

2.3.2 Reinforcement Learning-related Methods

In IoT systems, one of the key technical challenges is to balance the overall costs of computation and communication when making resource scheduling decisions in a dynamic environment. Reinforcement learning (RL), especially for deep reinforcement learning (DRL), is particularly suitable for solving resource scheduling problems in dynamic environments due to a number of reasons [32]. First, RL can target the optimization of long-term offloading performance. This would outperform the “one-shot” and greedy application of the approaches proposed in static environments, which may lead to strictly suboptimal results. Second, through RL, the optimal offloading policy can be learned by directly interacting with the environment without prior knowledge of the system dynamics, e.g., wireless channel or task arrival characteristics. This avoids the demand for the state transition matrix when the conventional solutions are utilized to solve the Markov decision process (MDP). Third, DRL can take full advantage of the powerful representation capability of the deep neural network (DNN). The optimal offloading policy can be adequately approximated even in complicated problems with vast state and/or action spaces. Therefore, the methodology of reinforcement learning is briefly introduced in this subsection.

First, the MDP [33] is an effective mathematical tool to model the impact of user actions in a dynamic environment and allows seeking the optimal resource scheduling decisions for achieving a particular long-term goal, which is typically defined as a 5-tuple, i.e., $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$.

- a set of environment states $\mathcal{S}$,
- a set of actions of the agent $\mathcal{A}$,
- a state transition probability matrix P that defines transition probabilities from all states s to all successor states s' due to action a, i.e., $p^a_{ss'} = \mathbb{P}[S_{t+1} = s'|S_t = s, A_t = a]$,
- a reward function R that defines the immediate reward (or expected immediate reward) received after transmitting from state s due to action a, i.e., $r^a_{ss'} = R(S_t = s, A_t = a, S_{t+1} = s')$, and $r^a_s = \mathbb{E}[r^a_{ss'}|S_t = s, A_t = a]$,
- a discount factor $\gamma \in [0, 1]$ that defines the present value of future rewards.

At each time step t, the agent observes an environment state S_t in the state space $\mathcal{S}$, and selects an action A_t from the action space $\mathcal{A}$, following the policy $\pi(a|s)$. The policy $\pi(a|s)$ is the probability of taking action A_t when observing state S_t, i.e., $\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$. Then, the environment transits to the next state S_{t+1} and emits the reward signal r_t according to the environment dynamics $p^a_{ss'}$ and the reward function $r^a_{ss'}$. This process continues indefinitely unless the agent observes a terminal state. The return G_t is the total discounted reward from time step t, defined as $G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{l=0}^{\infty} \gamma^l r_{t+l}$.

The state-value function $v_\pi(s)$ of an MDP is the expected return starting from state s, and then following policy π, i.e.,

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]. \tag{2.5}$$

The action-value function $q_\pi(s, a)$ is the expected return starting from state s, taking action a, and then following policy π, i.e.,

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]. \tag{2.6}$$

Based on the above definitions, the objective of MDP is to find the optimal policy π^*, maximizing the expectation of accumulated reward from the state s in the state space, i.e., $\pi^* = \arg\max_\pi v_\pi(s)$, or $\pi^* = \arg\max_\pi q_\pi(s, a)$, which can be be derived by value iteration or policy iteration.

However, in most practical IoT systems, the system dynamics, e.g., the state transition probability matrix P, is hard to measure or model. To this end, RL is envisioned as a promising solution to model-free sequential decision-making problems and has attracted increasing research interests [34]. Specifically, as shown in Figure 2.3, at each time step, the agent observes the state s from the environment, selects an action a, and then receives a reward r and the environment changes to s'. Therefore, the agent gathers experiences (s, a, r, s') at each time step from which it can learn. If the action taken was favorable for the given environment, it would get a positive reward; otherwise, it would be negative. The agent continues to collect the reward and update the Q-Table, aiming to maximize the expected return from each state. Here, "Q" refers to the function that the algorithm computes - the expected rewards for an action taken in a given state.

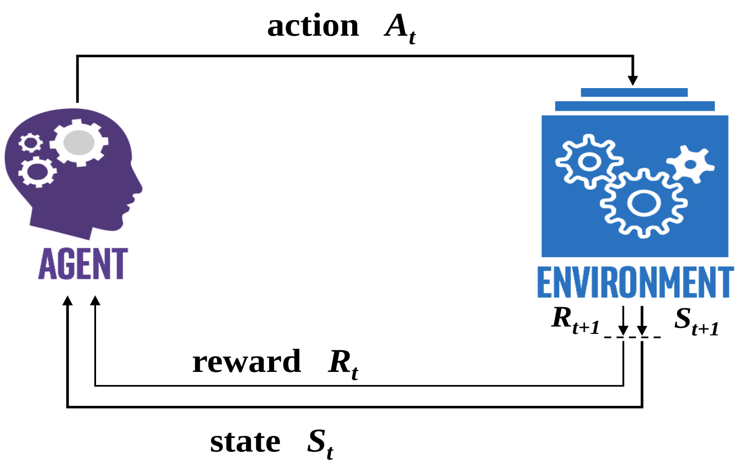


Figure 2.3: Architecture of reinforcement learning.

The typical algorithms of RL are Q-Learning and SARSA. The core of the Q-Learning algorithm is a Bellman equation as a simple value iteration update, using the weighted average of the old value and the new information:

$$Q^{\text{new}}(s,a) = Q(s,a) + \alpha(r + \gamma \max_{a \in A} Q(s',a) - Q(s,a)), \tag{2.7}$$

where α is the learning rate in the range of $(0,1]$. Here, the action used for updating Q-Table, i.e., $\max_a Q(s',a)$ might not consist with the actual action, thus Q-Learning is an off-policy learning algorithm. Different from Q-Learning, SARSA is an on-policy learning algorithm that updates the policy based on actions taken as:

$$Q^{\text{new}}(s,a) = Q(s,a) + \alpha(r + \gamma Q(s',a') - Q(s,a)), \tag{2.8}$$

where the action a' is the actual action that a SARSA agent would take.

Furthermore, to avoid trapping into a sub-optimal behaviour, RL adopts the ϵ-greedy exploration method to balance exploration and exploitation by choosing them randomly. Exploration allows an agent to improve its current knowledge about each action, hopefully leading to long-term benefits. On the other hand, exploitation chooses the greedy action to get the most reward by exploiting the agent's current action-value estimates. Therefore, when an agent explores, it gets more accurate estimates of action-values. And when it exploits, it might get more rewards. However, it cannot choose to do both simultaneously, which is also called the exploration-exploitation dilemma. Epsilon-Greedy is a simple method to balance exploration and exploitation as follows:

$$\pi(a|s) = \begin{cases} 1-\epsilon & a = \arg\max_{a \in A} Q(s,a) \\ \\ \epsilon & a \neq \arg\max_{a \in A} Q(s,a) \end{cases}, \tag{2.9}$$

where ϵ refers to the probability of choosing to explore, most of the time with a small chance of exploring.

In many practical decision-making problems, the states s of the MDP are high-dimensional and cannot be solved by traditional RL algorithms. DRL algorithms incorporate deep learning to solve such MDPs, often representing the policy $\pi(a|s)$ or other learned functions as a neural network and developing specialized algorithms that perform well in this setting. Current DRL methods can be classified into two categories: value-based methods and policy-based methods [34].

Value-based DRL methods adopt DNNs to approximate the value function. Generally, the core idea of value-based DRL methods is to minimize the difference between the value network and the real value function. A natural objective function can be written as

$$L^{\mathrm{V}}(w) = \mathbb{E}[v_\pi(s) - v(s; w)]^2, \tag{2.10}$$

where the expectation $\mathbb{E}[\cdot]$ indicates the empirical average over a finite batch of samples in an algorithm that alternates between sampling and optimization. $v(\cdot; w)$ is the value network, and w is the set of its parameters. $v_\pi(\cdot)$ represents the real value function, which is unknown but is estimated by different value-based RL methods. The popular algorithms include deep Q-network (DQN) [35, 36] and its variants [37].

On the other hand, *policy-based DRL methods* use DNNs to approximate the parameterized policy. Comparing with value-based DRL methods, policy-based DRL methods usually have better convergence properties and can learn stochastic policies. Policy-based DRL methods work by computing an estimator of the policy gradient, and the most commonly used gradient estimator has form

$$\nabla L^{\mathrm{PG}}(\theta) = \mathbb{E}[\nabla_\theta \log \pi(a|s; \theta) Q_\pi(s, a)], \tag{2.11}$$

where $\pi(a|s; \theta)$ is the policy network and $Q_\pi(s, a)$ is an estimator function. The popular algorithm is REINFORCE (Monte-Carlo Policy Gradient) [38].

In addition, actor-Critic architecture is proposed by integrating value-based algorithms and policy-based algorithms to realize stable and fast convergency [34]. The actor network decides which action should be taken, and the critic network informs the actor how good is the action and how it should adjust. The learning of the actor is based on a policy gradient approach. In comparison, critics evaluate the action produced by the actor by computing the value function. The popular deep deterministic policy gradient (DDPG) [39], soft actor-critic (SAC) [40] and proximal policy optimization (PPO) [41] algorithms are based on the actor-Critic architecture.

In IoT systems, extensive studies have been conducted for resource scheduling based on DRL. However, learning itself requires high computational resources, and it is impractical to be employed on the IoT devices side due to its limit-resource characteristic; on the other hand, the learning purely implemented on the server system side will lose the optimal behavior of IoT devices. Besides, methods instability and slow convergence brought by DRL-based algorithms is another essential issue to be solved in the resource scheduling process. Therefore, it is critical to design effective resource scheduling mechanisms based on reinforcement learning to address these challenges.

2.4 Chapter Summary

In this chapter, resource scheduling in IoT systems was thoroughly studied. The popular applications in IoT systems were elucidated firstly. Afterwards, the fundamentals of resource scheduling in IoT systems were introduced, including computing task offloading models, characteristics of IoT resources, and objectives of scheduling. Finally, the existing methodologies for resource scheduling, including game theory-based methods and reinforcement learning-based methods, were extensively surveyed and analyzed from the perspectives of their advantages and problems to be solved.

Chapter 3

Hierarchical Game-based Resource Scheduling

The horizontal collaboration amongst IoT devices is a promising way of balancing the computing tasks within the device layer. However, how to design an effective incentive mechanism to encourage computing resources sharing is a crucial problem to be addressed.

In this chapter, we consider the collaborative computing in smart buildings, which integrate IoT technologies for improving the performance of buildings and the comfort of occupants. However, the amount of data generated by IoT devices remains a challenge to the BMS in terms of intensity and complexity. Different from cloud computing and edge computing, we propose a computation sharing architecture in the device layer to incentivize ICDs (*sellers*) to offload computational tasks for the BMS (*buyer*). Considering game theory is a suitable tool for addressing incentive problems, we design a hierarchical game model consisting of a Stackelberg game and a Cournot game, to realize collaborative computing in smart buildings. This horizontal collaboration aims to achieve a dynamic increase in computational capacity for the BMS. To guarantee the utility of the BMS and ICDs, the Stackelberg game model is built to analyze the interactions between the BMS and ICDs. Then, the Cournot game model is presented to formulate the internal competition among multiple ICDs. Under the premise of SPNE, the BMS can quote the optimal pricing strategy, and ICDs can share the corresponding optimal amount of computing resources. Finally, the simulation results show that the BMS's computational capacity is enhanced on-demand, and each participant in the game obtains maximal utility.

3.1 Introduction

Currently, the adoption of IoT technologies [42, 2] into smart buildings has generated increasing interest [43]. Relying on the data generated by IoT devices, the BMS can closely monitor inside and outside conditions to promote the performance of buildings and the comfort of occupants [44, 45]. However, the amount of data remains challenging to the BMS in terms of its intensity, complexity, and efficiency.

To address this challenge, the previously used methods migrated data processing to cloud servers with extremely high computation capacity [46]; however, they are frequently unable to satisfy the latency requirement because of their physical distances. To overcome this limitation, edge computing [47, 48] has been proposed to provide computational capacity in close proximity, which can notably decrease the response time. Despite this advantage, server construction and maintenance costs must be considered, and these fixed costs cannot adjust dynamically as the computing demand fluctuates.

Toward this end, we introduce the concept of sharing economy, which is known as "access but not ownership"[49], as an alternative in the case of a large number of idle computing resources in or around smart buildings. According to the data of Geithner and McKinsey [50], the global server utilization is only 6% to 12% of their CPU. Moreover, as indicated in [10], a large number of smart devices are under-utilized in terms of their computing capability. Therefore, we propose computation sharing in smart buildings to achieve a dynamic increase in computation capacity, which exploits the enormous amount of spare computational resources in or around buildings, from smartphones, desktop PCs, web servers, or any ICDs. Computation sharing can enhance the computation capacity scalably without extra construction cost because it utilizes the computing resources from the existing idle computing devices, which is particularly useful for peak computing loads.

Although promising, it is important to note that computing resource owners have no commitments for sharing; they expect compensation when computation offloading consumes their CPU resources and potentially affects their own computing task. Hence, to enable effective computation sharing, we should design an incentive mechanism to address the following two technical challenges: *What is the optimal pricing mechanism to motivate computing devices to share their idle resources? How much data should each idle computing device offload to maximize its utility?*

In this chapter, we adopt the game-theoretic method to model and analyze the above two challenges. Apart from the traditional optimization approaches, game theory-based approaches have been widely applied in IoT systems [25, 27]. As an economic methodology, the primary and most important benefit of game theory-based approaches is the revenue generation, and thus it is regarded as a suitable tool for addressing incentive problems. Furthermore, components in computation sharing have different objectives. For example, the ICDs must be strongly incentivized to share their computing resources while the BMS requires to enhance its computational capacity with reasonable compensation. Using game theory, the problems where multiple players with contradictory objectives strategically interact with each other in a competition can be effectively resolved.

Specifically, we consider a set of computing devices (*ICDs, sellers*) to share their idle resources to alleviate the pressure of data processing for the BMS (*buyer*) as illustrated in Figure 3.1. To encourage computing sharing, the BMS dynamically determines the pricing strategy based on its demand and the availability of idle computing resources of multiple ICDs. Then, the ICDs make offloading decisions accordingly. The competitions exist not only between the BMS and ICDs but also among multiple ICDs since they belong to different owners in the computing sharing scenario. We note that the state-of-art literature mainly focuses on the competition between the service providers and consumers but ignores the competitions among multiple service providers. This motivates us to propose a two-tier game model to analyze computing sharing problems in smart buildings. The main contributions of this chapter are as follows:

- *Development of a Hierarchical Game Model*: We employ an economic incentive mechanism to encourage computing resource sharing in smart buildings, which aims to improve the computational capacity for the BMS on demand. A hierarchical game model is proposed to obtain the maximum utility for the BMS and the ICDs, which jointly combines the Stackelberg game and the Cournot game to reflect the dynamic relationships between

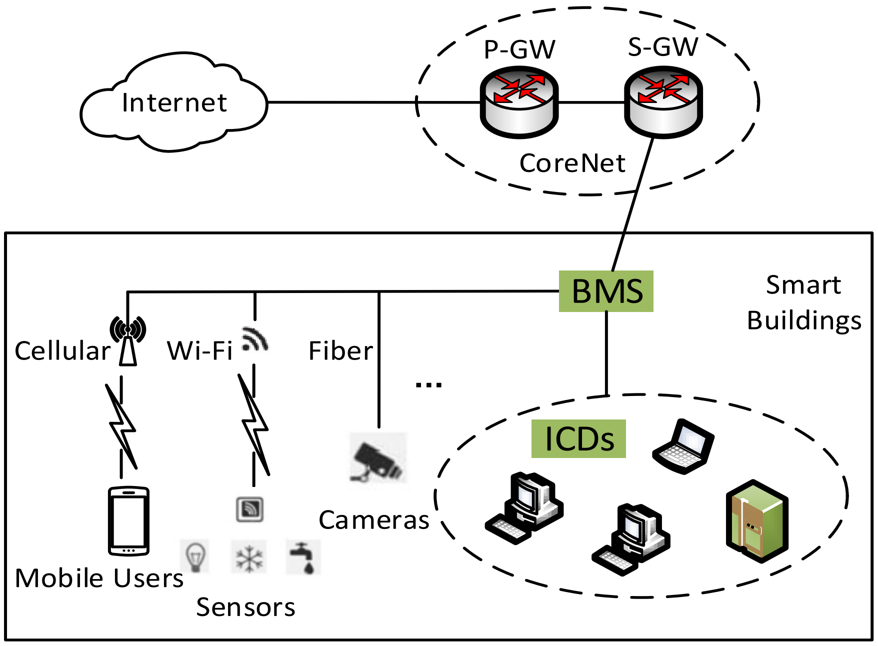


Figure 3.1: Illustration of computation sharing in buildings.

the BMS and multiple ICDs.

- *Stackelberg Game-based Analysis*: We use a Stackelberg game to model the interactive behavior of the BMS and the ICDs. To satisfy the dynamic computational demand in the building, the BMS, acting as a single leader, sets the appropriate offloading pricing strategy; then, different ICDs, acting as multiple competitive followers, derive the amounts of computing resources to share. Furthermore, we analyze the existence and uniqueness of the SPNE of the game.

- *Cournot Game-based Analysis*: We introduce the Cournot game to present the internal competition among multiple followers; i.e., each ICD's payoff depends not only on its own strategy but also on the decisions of its rivals. Given a demand-based pricing strategy, each ICD determines the optimal shared computing resources by considering other ICDs' strategies at the same time. The existence of each ICD's unique equilibrium is also proven.

- *Near-optimal Algorithm Design*: In practice, it is difficult for the participants to obtain complete information in the game. We present a near-optimal algorithm to acquire the near-optimal strategies of the BMS and the ICDs based on incomplete information, which helps to achieve SPNE by iterations. The simulation results demonstrate the correctness of the analysis, and all participants in the game obtain their near-optimal utilities.

The rest of this chapter is organized as follows. Section 3.2 provides a review of related works. Section 3.3 describes the system model and the problem formulations. In Section 3.4, we analyze the system with a two-tier game framework and find the solutions. In Section 3.4.1, we analyze the optimal amount of shared computing resources of the ICDs; in Section 3.4.2, we derive the optimal pricing strategy of the BMS; the near-optimal algorithm is proposed in Section 3.4.3. Finally, we evaluate the performance of our method in Section 3.5 and summarize the chapter in Section 3.6.

3.2 Related work

This section presents a review of the literature related to computation offloading mechanisms in IoT-based systems, and economics-based incentive methods.

The increase in IoT principles-based technical solutions incorporated in BMSs leads to high-quality data analytics[45, 51]. Various studies on cloud computing and edge computing have been performed to relieve such computation pressure[52, 53, 54]; however, these offloading methods must be well predesigned, and the server construction and maintenance costs cannot adjust as the computing demand fluctuates. In this chapter, we develop a computation sharing mechanism by applying the spare computing resources in or around smart buildings to enhance the computing capability of the BMS dynamically. The most crucial problem is establishing how to motivate the computation sharing and utilizing computing resources efficaciously.

Compared with traditional optimization, economics-based methods from the pricing dimension are effective approaches for resolving computing allocation problems in IoT systems [55, 56] since the prospect of achieving optimal revenue attracts each participant. For example, the authors of [57] proposed a joint optimization approach in IoT fog networks, where a fog node helps to offload data computing services from a data service operator to a data service subscriber. This approach was formulated as a Stackelberg game as well as a many-to-many matching game. In [58], the authors investigated the price-based computation offloading for a multiuser mobile edge computing system with a finite computation capacity. A Stackelberg game was also formulated to model the interaction between the edge cloud and users. Sladana *et al.* in [59] considered the problem of allocating wireless and computing resources to a set of autonomous wireless devices in an edge computing system, and modeled the interaction between devices and the operator as a Stackelberg game as well.

We note that Stackelberg game-based approaches have been widely used for modeling and analyzing the computing offloading problem. In the Stackelberg game model[60], one player (leader) moves first, and then the others (followers) move sequentially, and each player can maximize its payoff by finding the SPNE, which aligns well with the interactions between the service providers and the service consumers in computing offloading scenarios. This inspires us to adopt the Stackelberg game to analyze the competitions between the BMS and ICDs.

Furthermore, to exploit the interactions among multiple ICDs in the computation sharing scenario, we introduce the Cournot game model [24], considered one of the most popular types of games used to model the interactions among multiple strategic generators. For instance, to analyze competition in the mobile virtual network operator (MVNO)-oriented offloading market, a Cournot game was employed in [61] to determine the amount of cellular traffic in the retail market leased from the wholesale market among multiple MVNOs. Additionally, the Cournot game was used for modeling the uniform pricing electricity markets in day-ahead energy markets among multiple agents[62].

Therefore, in this chapter, we develop a hierarchical game model to address the strategies of computation sharing in smart building, which jointly combines the Stackelberg game and the Cournot game for modeling and analysis.

3.3 System Model and Problem Formulation

In this section, we first introduce the system model and then formulate the problem of computation sharing between the BMS and the ICDs.

3.3.1 System Model

As illustrated in Figure 3.2, we consider a computation offloading scenario with K ICDs denoted as $\mathcal{K} \triangleq \{1, 2, \ldots, K\}$, which are located in the building to support computational services for the BMS. We assume a time-slotted system, indexed by $t \in \mathcal{T} \triangleq \{0, 1, \ldots, T-1\}$. During a time slot, $Q_B(t)$ bits of data are processed for monitoring the performance of the building. As a centralized data center, the BMS determines the pricing strategy based on the computing demand and participant ICDs' computational capacity, i.e., the optimal unit price $p_I^{\text{opt}}(t)$ (per bit) abided by the ICDs in the marketplace. Accordingly, the optimal amount of computing resources $q_k^{\text{opt}}(t)$ is shared by each ICD$_k$; then, the processed data are transmitted to the BMS for use in building management. To standardize the unit of measurement, we count the amount of computation offloading or shared computing resources in bits. The bit values can also be translated into cycles by considering the processing density in cycles/bit. In this chapter, we assume that the processing densities of the ICDs are equal.

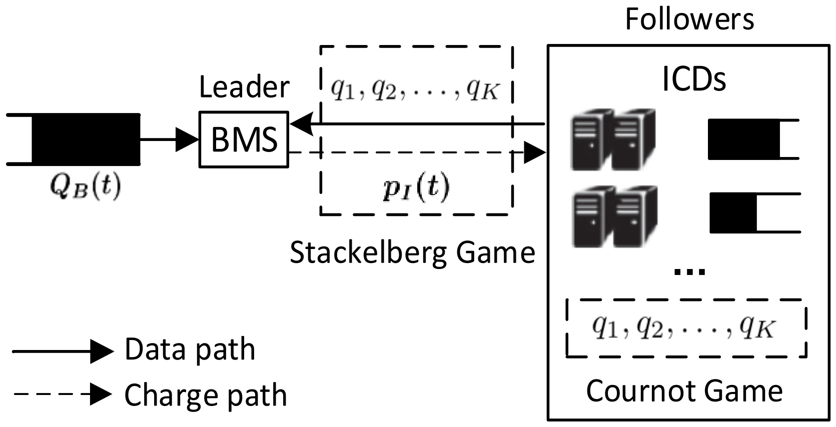


Figure 3.2: Illustration of the computation offloading architecture.

Trading between the BMS and the ICDs should satisfy individual requirements with the goal of maximizing their payoff. The ICDs must be incentivized to offer their computing resources while the BMS requires enhancing its computational capacity in order to optimize the building's management. The competitions exist not only between the BMS and the ICDs but also among multiple ICDs. Therefore, we propose a hierarchal game model to optimize resource sharing based on the above computation offloading architecture to achieve the optimal utility for each participant.

For external interaction between the BMS and the ICDs, a Stackelberg game model is applied to analyze the pricing strategy and the amount of computation offloading, where the BMS (a single leader) has the first-mover advantage to impose the optimal unit price $p_I^{\mathrm{opt}}(t)$ of computation offloading for the ICDs; then, the ICDs (multiple followers) can divide the optimal amount of computation offloading $q_k^{\mathrm{opt}}(t)$ by following the pricing strategy announced by the BMS. Both the BMS and the ICDs aim to obtain optimal utilities by maximizing their payoff.

For internal interaction among multiple ICDs, we introduce a Cournot game to model the competition. In practice, each noncooperative ICD maximizes only its own payoff by sharing more computing resources within its capacity; however, as the ICDs supply more computing resources, the BMS can damp down the unit price of computing offloading according to the theory of demand and supply. Hence, each ICD's payoff depends not only on its own strategy but also on the decisions of its rivals. These characteristics could be included in the Cournot game model. In this way, each ICD determines its optimal shared computing resources $q_k^{opt}(t)$ by considering the strategies of the other ICDs based on the Cournot game.

Combined with the Stackelberg game model and the Cournot game model, the BMS determines the computation offloading pricing strategy that changes dynamically based on its demand and the availability of idle computing resources of multiple ICDs. Then, the competing ICDs make offloading decisions based on the offloading pricing strategy as well as on their rivals' decisions. Since the internal competition among multiple ICDs fits the Cournot game model, we estimate p_I by the Cournot price function model to depict the competitive relationship. p_I, namely, the inverse demand function in the economic market, is a time-varying and decreasing function with respect to the total amount of computation offloading. Here, t is omitted since the parameters are fixed during a time slot. It is subject to $\frac{\partial p_I(t)}{\partial Q_I(t)} < 0$, where $Q_I(t) = \sum_{k \in \mathcal{K}} q_k^{\mathrm{opt}}(t)$. To simplify the problem, we estimate p_I with a universal linear model as in references [61, 62].

$$p_I(Q_I) = C - \alpha Q_I, \tag{3.1}$$

Here, C is the price function intercept that denotes the maximum unit price offered by the buyer, and α is a nonnegative coefficient that reflects the changing trend of the market price and the output, i.e., $\alpha = -\frac{\partial p_I(t)}{\partial Q_I(t)}$.

3.3.2 Problem Formulation

The hierarchical game approach is designed to enhance the computational capacity for the BMS under the premise of jointly maximizing the utility of the BMS and the ICDs. The details of the mathematical model are described as follows. We adopt the payoff to each party as its utility.

Utility of the BMS: The utility of the BMS is defined as the profit through computing offloading minus the sum of its payments to the ICDs.

$$U_{\text{BMS}}(p_I(t)) = p_B(t)Q_I(t) - p_I(t)Q_I(t), \tag{3.2}$$

Here, the unit profit $p_B(t)$ benefiting from computation offloading is a known parameter. $p_I(t)$ and $Q_I(t)$ denote the unit price of computing offloading paid by the BMS to the ICDs and the total amount of computing resources shared by the ICDs, respectively.

The BMS, acting as a single leader in the Stackelberg game, sets the optimal pricing s-trategy by predicting the strategies of the ICDs (followers). In our system model, we adopt a time-dependent scheme, i.e., all parameters are fixed during a time slot but are variable across time slots. In the following optimization analysis, we consider the deduction for a given time slot separately; thus, the time parameter t can be omitted. The optimization problem for the BMS can be formulated as follows, where $\boldsymbol{q} = \{q_1, q_2, \dots, q_K\}$ represents the computation offloading strategies of all ICDs.

$$\max_{p_I^{\text{opt}}} U_{\text{BMS}}(p_I | p_B, Q_B, \boldsymbol{q})$$

$$s.t. \quad 0 \le p_I^{\text{opt}} \le p_B \tag{3.3}$$

The BMS can determine an optimal pricing strategy p_I^{opt} given the unit profit p_B, the amount of computing task Q_B and the strategy set $\boldsymbol{q}$. The derived optimal unit price p_I^{opt} paid to the ICDs should be no more than the unit profit p_B that the BMS can derive as a benefit from computation offloading; otherwise, the utility of the BMS will be negative.

Utility of ICDs: The utility U_{ICD_k} of ICD$_k$ is defined as the price paid by the BMS minus the cost to offload $q_k(t)$ bits of data.

$$U_{\mathrm{ICD}_k}(q_k(t)) = p_I(t)q_k(t) - c_k(t)q_k(t), \tag{3.4}$$

Here, $q_k(t)$ denotes the amount of computing resources shared by ICD$_k$. $c_k(t)$ (per bit) denotes the unit cost for sharing, e.g., the energy and computing cost for offloading one bit of data. We assume that only a marginal cost exists.

The ICDs, acting as multiple followers in the Stackelberg game, deduce their optimal shared computing resources by following the pricing strategy announced by the BMS (leader). Meanwhile, each ICD's payoff does not only depend on its own strategy, but also on the decisions of its rivals, which is included in the Cournot game. The optimization problem for ICD$_k$ can be formulated as follows, where $\boldsymbol{q}_{-k}$ is the strategy set of the ICDs other than ICD$_k$.

$$\max_{q_k^{\mathrm{opt}}} U_{\mathrm{ICD}_k}(q_k | p_I, \boldsymbol{q}_{-k}), \forall k \in \mathcal{K}$$

$$s.t. \begin{cases} 0 \le q_k^{\mathrm{opt}} \le q_k^{\max} \\ \sum\limits_{k \in \mathcal{K}} q_k^{\mathrm{opt}} \le Q_B \end{cases} \tag{3.5}$$

According to the unit pricing strategy p_I, each ICD calculates its optimal amount of computation offloading q_k^{opt} given the strategies of the other ICDs $\boldsymbol{q}_{-k}$. Here, $q_k^{\max}$ denotes the maximum amount of available computing resources to be shared by ICD$_k$ during time slot t, which q_k^{opt} cannot exceed. $\sum_{k \in \mathcal{K}} q_k^{opt} \le Q_B$ denotes that the total amount of computation offloading is no more than the BMS's demand because all participants in the game are rational and thus will not share computing resources without benefit.

3.4 Game Model Analysis

In this section, we analyze the optimal strategies for the BMS and the ICDs by jointly combining a Stackelberg game and a Cournot game. The optimization problems for the BMS and the ICDs are conducted in the following two tiers. We use the derived results to prove the existence of the SPNE.

- *Tier I*: Players: the BMS (a single leader); Strategy: the unit price p_I; Utility: $U_{\text{BMS}}(p_I)$ given in (3.2).
- *Tier II*: Players: the ICDs (multiple competitive followers); Strategy: the shared computing resources $\{q_k|_{k\in\mathcal{K}}\}$; Utility: $\{U_{\text{ICD}_k}(q_k)|_{k\in\mathcal{K}}\}$ given in (3.4).

We adopt backward induction for the analysis as follows.

3.4.1 Tier II: Optimal Computing Resources Strategy

In tier II, given the leader's strategy p_I, each ICD determines the optimal amount q_k^{opt} by considering the rivals' decisions.

Lemma 3.4.1 *Jointly combining the Stackelberg game and the Cournot game, each* ICD$_k$ *determines the optimal amount of computation offloading q_k^{opt} when the strategies of the other ICDs are fixed.*

$$q_k^{\text{opt}} = \min\{q_k^*, q_k^{\max}\}. \tag{3.6}$$

Here, q_k^{max} is the maximum amount that can be shared by ICD$_k$ *during a time slot and $q_k^* = (C - c_k - \alpha\sum_{i\neq k} q_i)/2\alpha$. $\sum_{i\neq k} q_i = q_1 + q_2 + \ldots + q_{k-1} + q_{k+1} + \ldots + q_K$ denotes the total amount of computing resources shared by the ICDs other than* ICD$_k$.

Proof We substitute (3.1) into (3.4); thus, the utility of ICD$_k$ can be expressed as follows:

$$U_{\text{ICD}_k}(q_k) = -\alpha q_k^2 + \left(C - \alpha\sum_{i\neq k} q_i - c_k\right) q_k. \tag{3.7}$$

From (3.7), U_{ICD_k} is a continuous quadratic function of q_k by assuming $\sum_{i\neq k} q_i$ is fixed, and the second derivative of U_{ICD_k} with respect to q_k is $\frac{\partial^2 U_{\text{ICD}_k}}{\partial q_k^2} = -2\alpha$, where $\alpha > 0$. As $\frac{\partial^2 U_{\text{ICD}_k}}{\partial q_k^2} < 0$, U_{ICD_k} is a concave function of q_k. Therefore, we can obtain the optimal output of ICD$_k$ by setting the first derivative equal to zero.

$$q_k^* = \frac{C - c_k - \alpha\sum_{i\neq k} q_i^*}{2\alpha}. \tag{3.8}$$

In the Cournot game, the equilibrium solution $\boldsymbol{q}^* = \{q_1^*, q_2^*, \ldots, q_K^*\}$ can be solved by the above simultaneous equations as $\{q_k^* = (C - c_k - \alpha \sum_{i \neq k} q_i^*) \big/ 2\alpha \;|_{k \in \mathcal{K}}\}$, i.e., none of the ICDs can gain more utility if any of them changes its strategy. Thereafter, the optimal utility of ICD$_k$ is expressed as $U_{\text{ICD}_k}^{\text{opt}} = (C - c_k - \alpha \sum_{i \neq k} q_i^*)^2 \big/ 4\alpha$.

If the optimal amount q_k^* is larger than the upper bound $q_k^{\max}$, we should choose $q_k^{\max}$ as the optimal amount because U_{ICD_k} is a monotonically increasing function of q_k in the interval of $\left[0, q_k^*\right]$, as shown in Figure 3.3(b). This finishes the proof of Lemma 3.4.1.

The next problem to resolve is determining how to adjust the pricing strategy $p_I(Q_I)$ to maximize the utility of the BMS while guaranteeing its computational demand.

3.4.2 Tier I: Optimal Unit Price Strategy

In tier I, the BMS needs to determine the optimal unit price p_I^{opt} by predicting the strategies of the ICDs.

Theorem 3.4.2 *In the Stackelberg game model, given the unit profit p_B and the amount of computing to be performed Q_B, the BMS can determine the optimal price strategy $p_I(Q_I)$ by predicting the optimal strategies of the ICDs:*

$$p_I^{\text{opt}}(Q_I) = \left(\frac{K+1}{2} p_B - \frac{K-1}{2K} \sum c_k\right) - \frac{K p_B - \sum c_k}{2 Q_B} Q_I, \tag{3.9}$$

where $Q_I = \sum_{k \in \mathcal{K}} q_k$ denotes the computing resources shared by the ICDs in total, i.e., the enhanced computational capacity on demand.

Proof We can derive Q_I with the formula set $\{q_k^{\text{opt}} = (C - c_k - \alpha \sum_{i \neq k} q_i^{\text{opt}}) \big/ 2\alpha \;|_{k \in \mathcal{K}}\}$ as

$$Q_I = \sum_{k=1}^{K} q_k^{\text{opt}} = \frac{KC - \sum c_k}{\alpha(K+1)}. \tag{3.10}$$

Substituting (3.1) (3.10) into (3.2), the utility of the BMS can be represented as follows:

$$\begin{aligned} U_{\text{BMS}}(C, \alpha) = &- \frac{K}{\alpha(K+1)^2} \left[C - \left(\frac{K+1}{2} p_B - \frac{K-1}{2K} \sum c_k\right)\right]^2 \\ &+ \frac{1}{4\alpha K} \left(K p_B - \sum c_k\right)^2, \end{aligned} \tag{3.11}$$

where U_{BMS} is a continuous quadratic function of C when α is fixed. The second derivative of U_{BMS} with respect to C is $\frac{\partial^2 U_{\text{BMS}}}{\partial C^2} = -2K\big/\alpha(K+1)^2$. Since $K > 0$ and $\alpha > 0$, we have $\frac{\partial^2 U_{\text{BMS}}}{\partial C^2} < 0$, and U_{BMS} is a concave function of C. Furthermore, we can obtain the optimal parameter C^{opt} by setting the first derivative equal to zero as $C^{\text{opt}} = (K+1)p_B/2 - (K-1)\sum c_k/2K$; then, the optimal utility of BMS is $U_{\text{BMS}}^{\text{opt}} = (Kp_B - \sum c_k)^2\big/4\alpha K$.

The optimal utility $U_{\text{BMS}}^{\text{opt}}$ is a monotonically decreasing function with respect to α, and we can derive α^{opt} by the market equilibrium. Substituting C^{opt} into (3.10), we can obtain the total amount of idle computing resources shared by the ICDs as $Q_I = Kp_B - \sum c_k/2\alpha$. To achieve market equilibrium, the optimal parameter α^{opt} can be determined by replacing Q_I with Q_B as $\alpha^{\text{opt}} = Kp_B - \sum c_k/2Q_B$.

Finally, we substitute C^{opt} and α^{opt} into (3.1) and obtain the optimal unit price expressed as (3.9). This finishes the proof of Theorem 3.4.2.

C^{opt} and α^{opt} can be substituted into (3.8) to obtain the following theorem.

Theorem 3.4.3 *The optimal amount of computation offloading shared by* ICD$_k$ *is*

$$q_k^{\text{opt}} = \min\{q_k^*, q_k^{\max}\}, \tag{3.12}$$

where $q_k^* = (C^{\text{opt}} - c_k - \alpha^{\text{opt}} \sum_{i\neq k} q_i^*)\big/2\alpha^{\text{opt}}$.

As indicated in **Lemma 3.4.1**, when the optimal amount q_k^* of computing resources shared by ICD$_k$ is larger than the threshold, we should choose $q_k^{\max}$ as the optimal quantity. Then, the other ICDs must change their strategies to achieve the new SPNE. The adjusted demand of the BMS is represented as follows:

$$Q_B^{'} = Q_B - \sum_{i\in\mathcal{I}} q_i^{\max}, \tag{3.13}$$

and

$$p_B^{'} = \frac{p_B Q_B - p_I^{\text{opt}} \sum_{i\in\mathcal{I}} q_i^{\max}}{Q_B^{'}}, \tag{3.14}$$

where $\mathcal{I}$ denotes all ICDs that reach the threshold, and the set of the competing ICDs changes to $k \in \mathcal{K} \setminus \mathcal{I}$.

In summary, as a leader, the BMS has the information of the participant ICDs, i.e., $\{c_k|_{k\in\mathcal{K}}\}$; then, the BMS can estimate the optimal unit price $p_I^{\text{opt}}(Q_I)$ by (3.9) according to the required amount of computation offloading Q_B and the unit profit p_B. Following this pricing strategy, each ICD can decide how many idle computing resources q_k^{opt} to share according to (3.12) to achieve its maximum utility with the rivals' decisions. If any ICD reaches its maximal computing limitation, the optimal amount of shared computing resources will be recalculated. The demand of the BMS, i.e., $Q_B(t)$ and $p_B(t)$, is adjusted by (3.13) and (3.14), respectively. The BMS and the ICDs perform game rounds with the adjusted BMS's demand until this demand has been satisfied or all ICDs have reached the threshold. Unprocessed data can be offloaded to an edge server or cloud server if the BMS requires more computational capacity than the total amount the ICDs are willing to share.

In this distributed scheme, each ICD makes the decision based on the knowledge of other ICDs' strategies; however, it is difficult for the ICDs to obtain complete information on their rivals in a practical scenario since they are in a competitive relationship. In addition, the computational complexity brought by considering the rivals' strategies is $O[K^3]$, which would be considerably high if there are many ICDs involved in the computing sharing competition. To alleviate these issues, we present a near-optimal algorithm to obtain the near-optimal strategy of each ICD by iterations without rivals' information.

Next, we provide the algorithmic details of the near-optimal game mechanism.

3.4.3 Near-optimal Algorithm for Hierarchical Game Mechanism

To address the problems of incomplete information and computational complexity presented in Section 3.4.2, we relax the assumptions that the ICDs are unaware of their rivals' information and propose the near-optimal algorithm to obtain $\{q_k^{opt}|_{k\in\mathcal{K}}\}$ by iterations.

Theorem 3.4.4 *Each ICD can update its near-optimal strategy distributively by the following iterative function.*

$$q_k^*(n) = \frac{p_I^{\text{opt}} + \alpha^{\text{opt}} q_k^*(n-1) - c_k}{2\alpha^{\text{opt}}}. \tag{3.15}$$

Here, p_I^{opt} is the value of the optimal unit price $p_I^{\text{opt}}(Q_B)$, and α^{opt} reflects the impact of the ICD$_k$*'s decision on the unit price, which is a known parameter for the participant.*

Proof Based on the unit price p_I, ICD$_k$ can obtain its utility as $U_{\text{ICD}_k}(n-1) = (p_I - c_k)q_k^*(n-1)$ by sharing $q_k^*(n-1)$ idle computing resources in iteration $n-1$. In iteration n, we assume the changing amount of $q_k^*(n-1)$ as Δ, and the other optimal strategies $\boldsymbol{q}_{-k}^*(n)$ are fixed as the previous iteration; then, the corresponding utility can be represented as follows:

$$U_{\text{ICD}_k}(n) = (p_I - \alpha\Delta - c_k)\left(q_k^*(n-1) + \Delta\right) = U_{\text{ICD}_k}(n-1) + A, \tag{3.16}$$

where $A = -\alpha\Delta^2 + (p_I - \alpha q_k^*(n-1) - c_k)\Delta$. To maximize $U_{\text{ICD}_k}(n)$, the optimal Δ can be expressed as $\Delta = (p_I - \alpha q_k - c_k)/2\alpha$. Then, the updated $q_k^*(n)$ to maximize ICD$_k$'s utility should be $q_k^*(n) = q_k^*(n-1) + \Delta$ as expressed in (3.15).

We can prove the convergence by the following derivation,

$$\begin{aligned}\lim_{n\to\infty} q_k^*(n) &= \lim_{n\to\infty}\{\left[1 - \left(\frac{1}{2}\right)^n\right]\frac{p_I - c_k}{\alpha} + \left(\frac{1}{2}\right)^n q_k^{\text{init}}\}\\ &= \frac{p_I - c_k}{\alpha}.\end{aligned} \tag{3.17}$$

Substituting (3.1) into (3.8), we can obtain that the above convergent result of $q_k^*(n)$ is the same as a result calculated under complete information, which is also illustrated in Figure 3.7(a). This finishes the proof of Theorem 3.4.4.

The near-optimal algorithm of computation offloading based on incomplete information is presented as follows.

In this near-optimal algorithm, each ICD can obtain its near-optimal strategy without the decisions of other ICDs and the price structure of the BMS. The BMS calculates the optimal value of p_I^{opt} by (3.9) with the input of Q_B, p_B, c_k; then it broadcasts p_I^{opt} and α^{opt} to all ICDs. Each ICD converges to its near-optimal strategy by following the iterative function (3.15). If any ICD reaches its maximal computing limitation, the BMS and the ICDs perform game rounds with the adjusted BMS's demand (3.13), (3.14) until this demand has been satisfied or all ICDs have reached the threshold. Finally, all participants achieve the SPNE, where no one will alter its strategy if the strategies of the others remain unchanged.

Each ICD will achieve the SPNE by at least N iterations, where $N = log_2[(p_I^{\text{opt}} - c_k)\big/\alpha^{\text{opt}} - q_k^{\text{init}}] - log_2\epsilon$. This can be derived iteratively by $q_k^*(N) - q_k^*(N-1) \le \epsilon$. In this near-optimal

Algorithm 1 Near-optimal algorithm of computation offloading based on incomplete information

```
Input: $Q_B$, $p_B$, $c_k$, $q_k^{\max}$
Output: $p_I^{\text{opt}}$, $\{q_k^{\text{opt}}|_{k\in\mathcal{K}}\}$
 1: Initially, each participant ICD submits its unit cost $c_k$ to the BMS.
 2: repeat
 3:     BMS obtains $p_I^{\text{opt}}$ and $\alpha^{\text{opt}}$ and broadcasts to ICDs.
 4:     for ICD$_k$ do
 5:         set $q_k^{\text{init}}$ as a random value.
 6:         repeat
 7:             $q_k^*(n) \Leftarrow \frac{p_I^{\text{opt}}+\alpha^{\text{opt}}q_k^*(n-1)-c_k}{2\alpha^{\text{opt}}}$.
 8:         until $q_k^*(n) - q_k^*(n-1) \leq \epsilon, \forall i$
 9:         $q_k^{\text{opt}} = \min\{q_k^*, q_k^{\max}\}$
10:     if $q_k^{opt} = q_k^{\max}, \forall k \in \mathcal{K}$ then
11:         BMS adds ICD$_k$ to $\mathcal{I}$
12:         set $\mathcal{K} \triangleq \mathcal{K} \setminus \mathcal{I}$.
13:         set $Q_B(t) \Leftarrow Q_B^{'}(t)$.
14:         set $p_B(t) \Leftarrow p_B^{'}(t)$.
15: until $\sum q_k^{\text{opt}} = Q_B$ or $q_k^{\text{opt}} = q_k^{\max}|_{k\in\mathcal{K}}$
```

algorithm, the computational complexity of each ICD's convergence is $O[N]$. As indicated in 3.4.2, the game rounds would repeat K times in the worst case by considering the limitation of each ICD's computation capacity. Therefore, the total computational complexity of the BMS and the ICDs would be $O[K]$ and $O[KN]$, respectively. This finding demonstrates that the proposed near-optimal algorithm can be finished in polynomial time. Compared with the derivation based on the complete information, the computational complexity of each ICD is significantly reduced from $O[K^3]$ to $O[KN]$ when the number of competitive ICDs is high.

3.5 Simulation Results

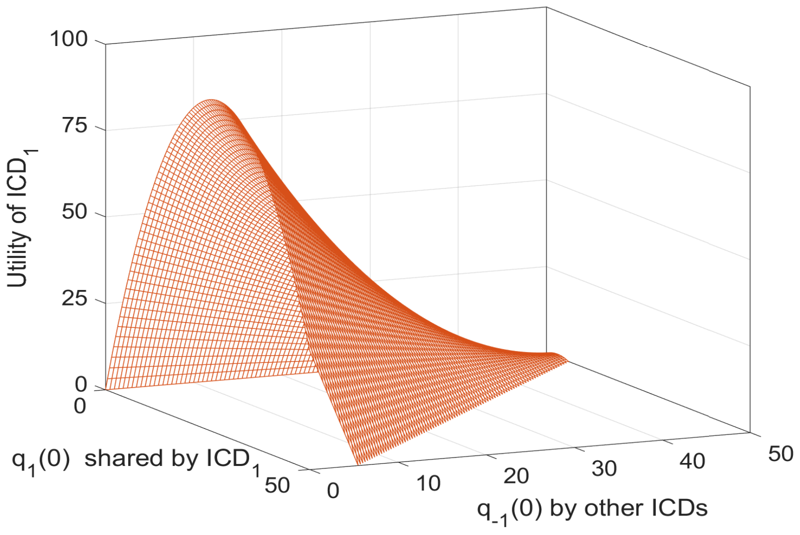
In this section, we present numerical results to evaluate the performance of computation sharing between the BMS and the ICDs in MATLAB. We first verify the optimal strategies of the BMS and the ICDs by our proposed hierarchical game model, and then illustrate the existence of SPNE and the convergence behavior of the near-optimal algorithm. Furthermore, the comparisons with existing algorithms are conducted in terms of the participants' utilities and computational latency. Finally, we demonstrate the performance of our proposed algorithm under various parameter settings. The default simulation parameters are given as follows, if not specified. The unit cost $c_k(t)$ of each ICD for sharing computing resources is a random value in [1,10] (\$/MB). Here, we set the BMS's demand $Q_B(0)$ to 50 MB and $p_B(0)$ to 15.5 per MB in time slot 0 and assume that $K = 10$; i.e., 10 ICDs are competing to share computing resources. The corresponding $q_k^{\max}$ is set as a random nonnegative value, for example, we set $q_k^{\max}(0)$ as [10, 20, 20, 15, 10, 10, 10, 20, 20, 20] MB as default.

3.5.1 Optimal Strategy Analysis

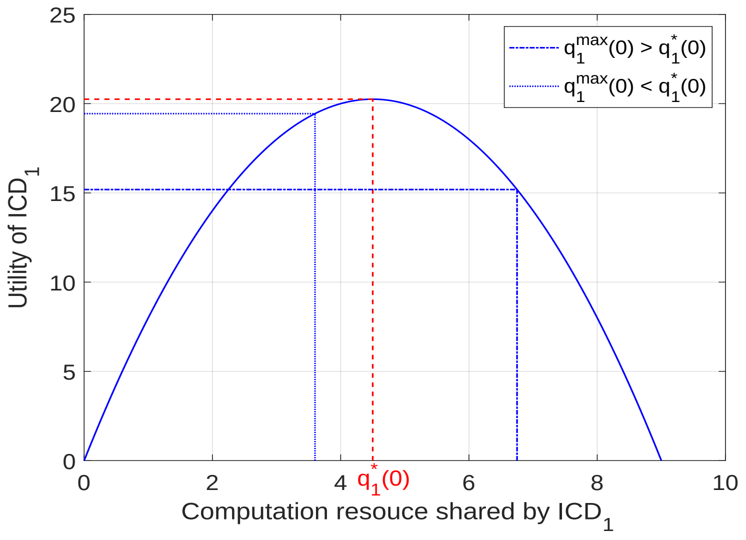
We conduct simulations to illustrate the strategy optimization of the BMS and the ICDs, i.e., p_I^{opt} and q^{opt}.

As shown in Figure 3.3, we evaluate the optimal shared computing resources of ICD_1 by maximizing its utility. We obtain the optimal unit price $p_I^{\text{opt}}(Q_I(t))$ by (3.9). When the total amount $\sum_{i\neq 1} q_i^{\text{opt}}(t)$ of computing resources shared by the other ICDs increases, as shown in subgraph (a), the optimal utility of ICD_1 generally decreases due to the competition. Furthermore, when $\sum_{i\neq 1} q_i^{\text{opt}}(0)$ is fixed, as presented in subgraph (b), the utility of ICD_1 is a concave function versus $q_1(0)$, and $q_1^{\text{opt}}(0)$ is the optimal amount of computing resources shared by ICD_1 to maximize its payoff. When the optimal amount $q_1^{\text{opt}}(0)$ is larger than the threshold $q_1^{\max}(0)$, we should choose $q_1^{\max}(0)$ as the optimal strategy; otherwise, $q_1^{\text{opt}}(0)$ is the best choice.

Figure 3.4 and Figure 3.5 illustrate the utility of the BMS as the parameter $\alpha(0)$ and $C(0)$ changes, respectively. Figure 3.4 shows the process for deriving the optimal parameter $\alpha^{\text{opt}}(0)$ when $C^{\text{opt}}(0)$ is fixed. From subgraph (a), the utility of the BMS decreases monotonically when $\alpha(0) \geq \alpha^{\text{opt}}(0)$, and the BMS can gain the maximum profit when $\alpha(0) \in (0, \alpha^{\text{opt}}(0)]$. Compared with subgraph (b), we find that the utility of the BMS does not increase even when the ICDs share more computing resources than the BMS's demand when $\alpha(0) \in (0, \alpha^{\text{opt}}(0)]$. Therefore, we can conclude that $\alpha^{\text{opt}}(0)$, achieving the market equilibrium, is the optimal parameter to maximize the utility of the BMS. In Figure 3.5, the parameter $\alpha(0)$ is set as the fixed $\alpha^{\text{opt}}(0)$ by (3.9). We observe that the utility of the BMS is a concave function versus $C(0)$ and that $C^{\text{opt}}(0)$ is the optimal parameter of the unit price $p_I(0)$ that maximizes the utility of the BMS.

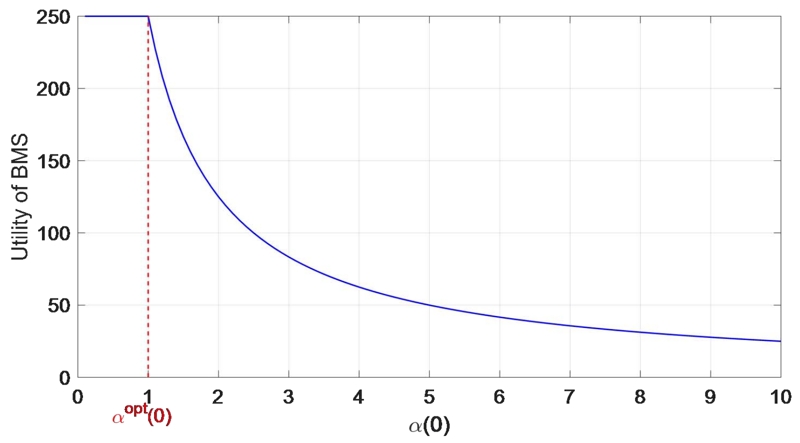


(a) ICD$_1$ and other ICDs

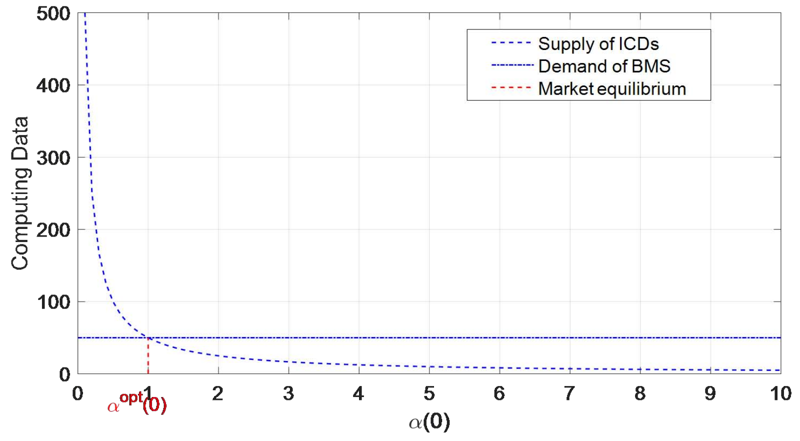


(b) ICD$_1$

Figure 3.3: Utility of ICD$_1$ versus the amount of computing resources shared by ICD$_1$ and the other ICDs.



(a) Utility of the BMS



(b) Market Equilibrium

Figure 3.4: Utility of BMS and the market equilibrium versus $\alpha(0)$.

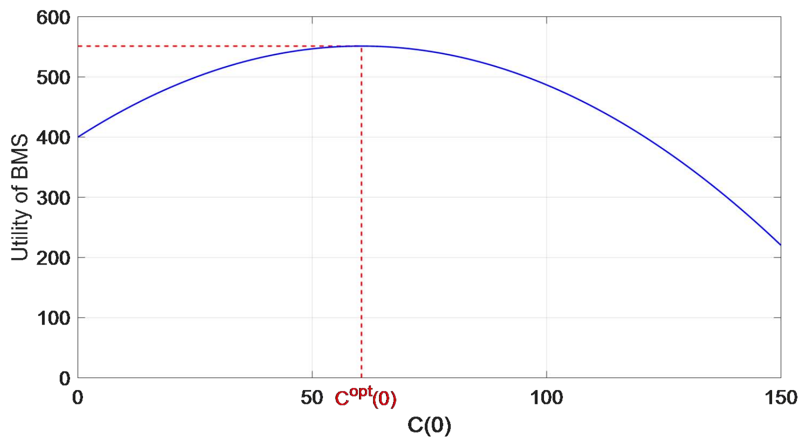
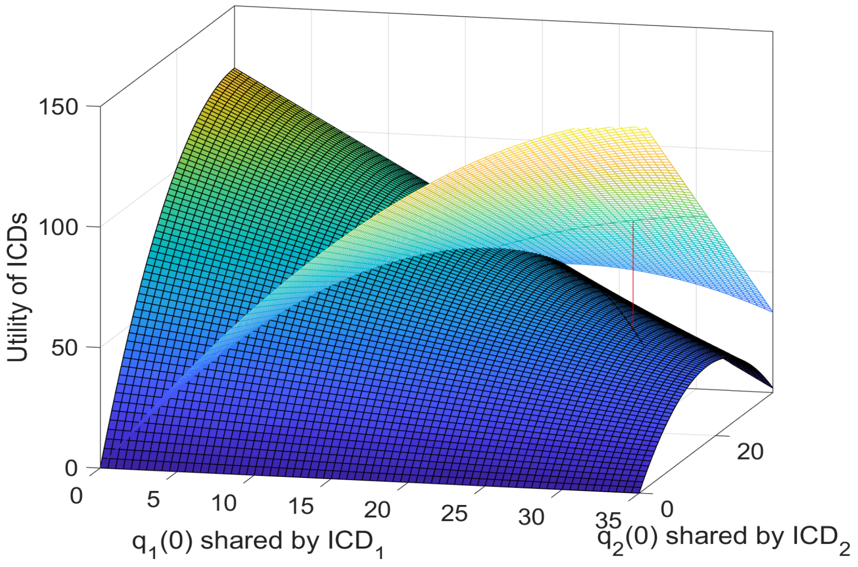


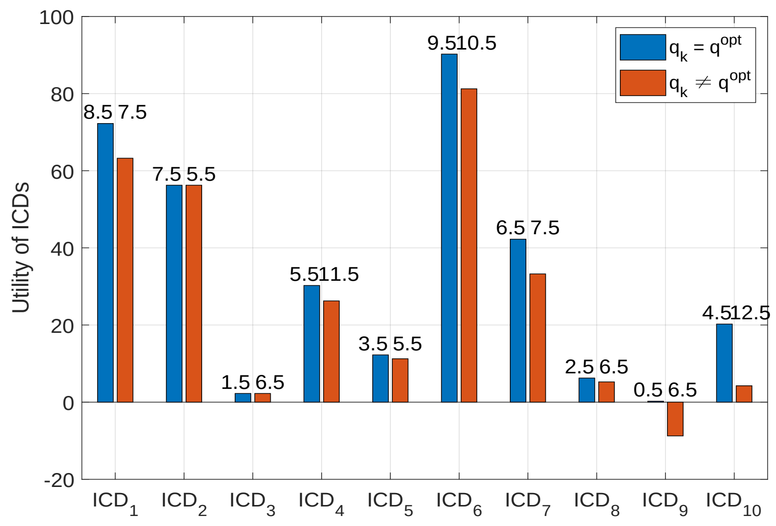
Figure 3.5: Utility of BMS versus $C(0)$.

3.5.2 SPNE Analysis

In this section, we first show the existence of SPNE and then illustrate the convergence behavior to SPNE of the near-optimal algorithm.



(a) 2 ICDs



(b) 10 ICDs

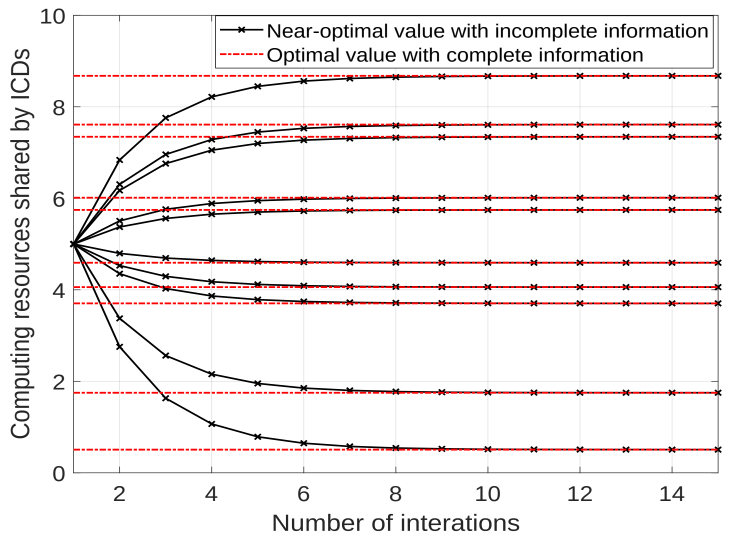
Figure 3.6: Utility of ICDs under SPNE.

Figure 3.6 investigates how the ICDs achieve the SPNE. First, we assume that there are 2 ICDs, as shown in subgraph (a). The maximum utility of each ICD occurs when the other's strategy is fixed so that the projection of the intersecting curves of the two maximum utility lines in the XY axis is the equilibrium of ICD_1 and ICD_2. Subgraph (b) describes achieving the SPNE with 10 ICDs in terms of the utility of the ICDs. The curve is the optimal strategy of each ICD derived by our proposed algorithm. Then, we adjust the shared computing amount randomly and notice that the payoff will only decrease if $q_k \neq q^{opt}$ regardless of how the strategy is adjusted. For example, ICD_1 decreases its shared computing resource from 8.5 MB to 7.5 MB, and its utility also decreases from 72.25 to 63.25; however, the utility of ICD_4 drops from 30.25 to 26.25 even it offloads 6 MB more data processing load. This proves that all ICDs achieve the SPNE by our proposed algorithm, where neither ICD can obtain a greater payoff if the other does not change its strategy.

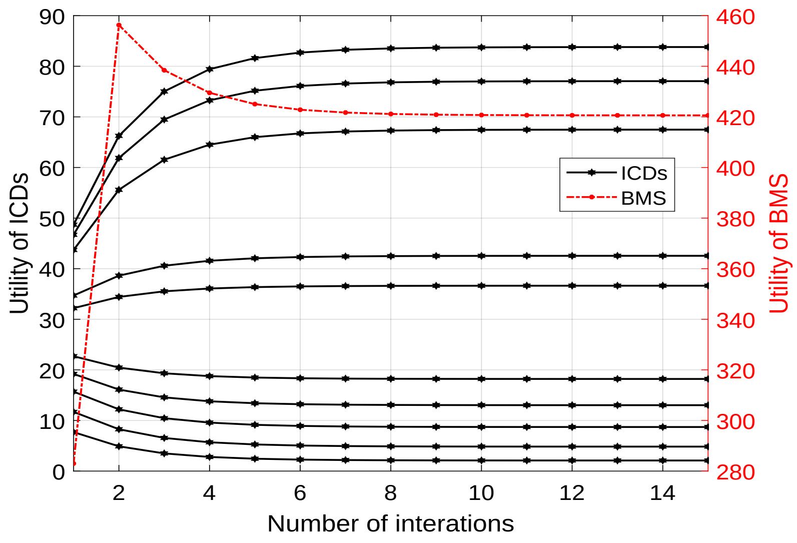
Next, we simulate the convergence iterations with our proposed near-optimal algorithm, as shown in Figure 3.7. The upper limit ϵ is set as 10^{-6} MB, each ICD sets its initial strategy as the average value, i.e., $50/10$ MB, and other parameters are set as default values. From subgraph (a), the near-optimal amount of computing resources shared by each ICD quickly converges to the optimal value calculated by (3.12). This indicates that our proposed near-optimal algorithm can help the ICDs obtain their optimal strategies even when each ICD is unaware of its rivals' information. From subgraph (b), the utilities of the BMS and the ICDs converge to the SPNE, where no one will alter its strategy if the strategies of the others remain unchanged. We note that the utility of BMS reaches its highest value in Iteration 2; however, it still drops to a stable small value in the following iterations. This occurs because the BMS and the ICDs keep their strategies unchanged only when SPNE is implemented. Otherwise, their benefits will be affected by other participants even though they do not change their own strategies.

3.5.3 Algorithm Comparison

We first compare the performance of the proposed algorithm in Figure 3.8 with three baselines in terms of the utility of the BMS and the ICDs; i.e., (a) offloading equally: all ICDs offload the same amount of computing resource for data processing; i.e., Q_B/K; (b) offloading randomly: each ICD randomly chooses its offloading decision; and (c) equal to $q_k^{\max}$: all ICDs share their maximum idle computing resource, i.e., $q_k^{\max}$. We note that the maximum utility of the BMS is achieved when $q_k(0)$ equals $q_k^{\max}(0)$; however, some ICDs' payoffs are negative, such as ICD_3



(a)



(b)

Figure 3.7: Convergence iterations with the near-optimal algorithm.

and ICD$_4$, because the BMS can provide a very low unit price, even lower than ICDs' unit cost, when the ICDs are willing to share considerably more computing resources. By contrast, the utilities of several ICDs are maximum, but the BMS's profit is negative when data are offloaded to each ICD randomly. These two strategies are not recommended because of the negative payoffs. When offloading data equally, i.e., $\{q_k(0) = 5\text{MB}|_{k\in\mathcal{K}}\}$, some of the ICDs, such as ICD$_2$, receive a greater payoff (marked as *) by adjusting their shared computing resource amounts even when the others' strategies are fixed, which means that this strategy does not achieve the equilibrium. Meanwhile, the amount of computation offloading derived by our proposed algorithm is optimal, and the payoff will only decrease regardless of how the strategy is adjusted, as shown in Figure 3.6(b). Therefore, this curve represents the optimal strategies of the BMS and the ICDs under the SPNE.

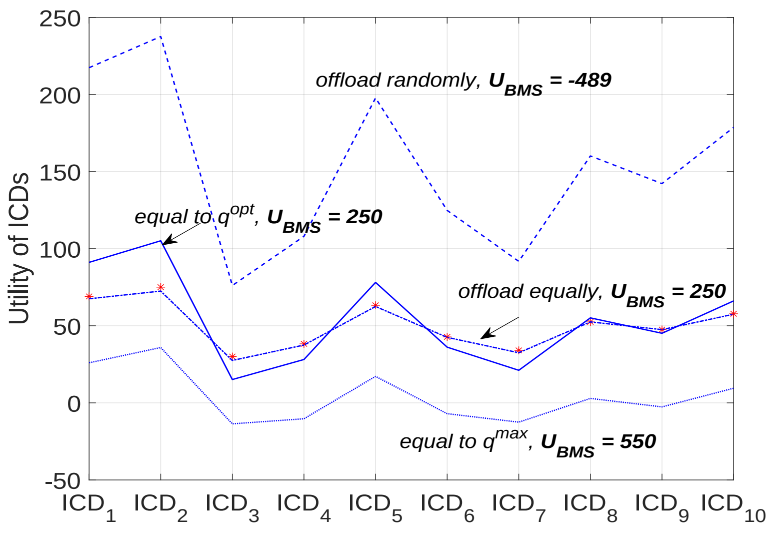


Figure 3.8: Algorithm comparison in terms of utilities of BMS and ICDs.

Next, we compare the performance of the proposed computation sharing algorithm with the edge offloading algorithm [58] in terms of computing latency. We use the same simulation setups for edge computing as [58]. According to circuit theory [63], the unit cost of executing the computation task is proportional to the square of CPU frequency, and we obtain each ICD's CPU frequency accordingly. The computing latency is defined as the maximum value of each ICD's data processing time, i.e., $max\{q_k/f_k\}$. We can find that our proposed algorithm can significantly reduce the computing latency with the increasing of the number of participant ICDs.

As shown in Figure 3.9, our proposed computing sharing algorithm can attain better performance than edge computing when there are sufficient ICDs (more than 50). Consequently, it provides a novel solution to alleviate the computing pressure in smart buildings, especially for peak computing loads.

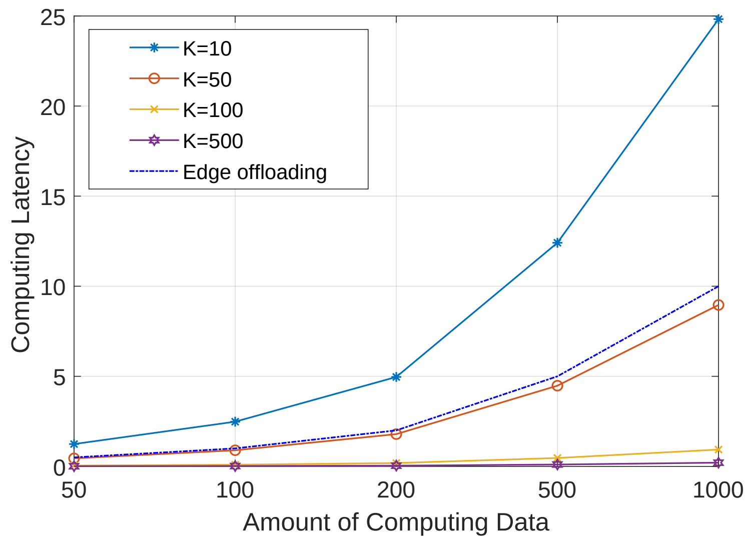


Figure 3.9: Algorithm comparison in terms of computing latency.

3.5.4 Impacts of Parameters

In this section, we consider the utilities of the BMS and the ICDs when the demand of the BMS $Q_B(t)$ and the threshold of computing resources shared by each ICD $q_k^{\max}(t)$ change in different time slots $t \in [0, 3]$. $Q_B(t)$ is set as [100, 200, 300, 300] MB in each time slot. To simulate the impact of the maximum amount of idle computing resource, we set $q_k^{\max}$ as [40, 40, 40, 40, 40, 50, 40, 50, 50, 40] MB during the time slot 0 to 2, and we change it to [40, 20, 40, 30, 50, 40, 40, 40, 40, 40] MB in time slot 3. In Figure 3.10, the utilities of the BMS and the ICDs increase as the number of computing resources shared by each ICD increases. During time slot 3, the deduced amounts $q_2^*(3)$ and $q_4^*(3)$ exceed the upper threshold $q_2^{\max}(3)$ and $q_4^{\max}(3)$, respectively. ICD$_2$ and ICD$_4$ adjust their strategies by using (3.13) and (3.14) to recalculate their optimal outputs. We can see that the adjusted utilities of the ICDs increase, except those of ICD$_2$ and ICD$_4$ because of their limited computational capacities.

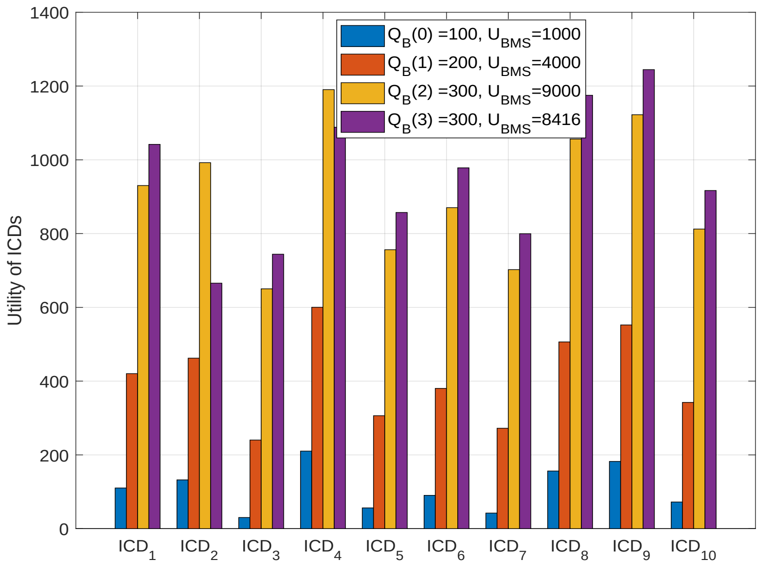


Figure 3.10: Utilities of BMS and ICDs versus the demand of BMS.

Table 3.1: Market equilibrium between the demand of BMS and the supply of ICDs

	Optimal amount of computing resources shared by each ICD										
Q_B	q_1^{opt}	q_2^{opt}	q_3^{opt}	q_4^{opt}	q_5^{opt}	q_6^{opt}	q_7^{opt}	q_8^{opt}	q_9^{opt}	q_{10}^{vopt}	$\sum_{k\in\mathcal{K}} q_k^{\text{opt}}$
100	11/2	23/2	27/2	21/2	17/2	13/2	15/2	19/2	25/2	29/2	100
200	41/2	43/2	31/2	49/2	35/2	39/2	33/2	45/2	47/2	37/2	200
300	61/2	63/2	51/2	69/2	55/2	59/2	53/2	65/2	67/2	57/2	300
300	65/2	20	55/2	30	59/2	63/2	57/2	69/2	71/2	61/2	300

In Table 3.1, we list the optimal amount of computing resources shared by each ICD of Figure 3.10, where the 1st row refers to $t = 0$, the 2nd row refers to $t = 1$, the 3rd row refers to $t = 2$, and the 4th row refers to $t = 3$. As shown in Table 3.1, our proposed model achieves market equilibrium because the total amount of computing resources $\sum_{k\in\mathcal{K}} q_k^{\text{opt}}(t)$ shared by each ICD is equal to the demand of the BMS $Q_B(t)$ when the BMS and ICDs reach the SPNE.

Next, we investigate the performance of the BMS and the ICDs when the number of participant ICDs varies. As Figure 3.11 shows, the utility of each ICD decreases substantially as K increases from 5 to 1000. This result occurs because more participants lead to increased competition and a lower workload, which reduces the earnings for each ICD. Under the condition of constant demand, the algorithm realizes the steady utility of the BMS when the number of participant ICDs changes, which contributes to market stability. Furthermore, as shown in Figure 3.9, computing latency can be significantly reduced as the number of participants increases, which helps to achieve lower latency.

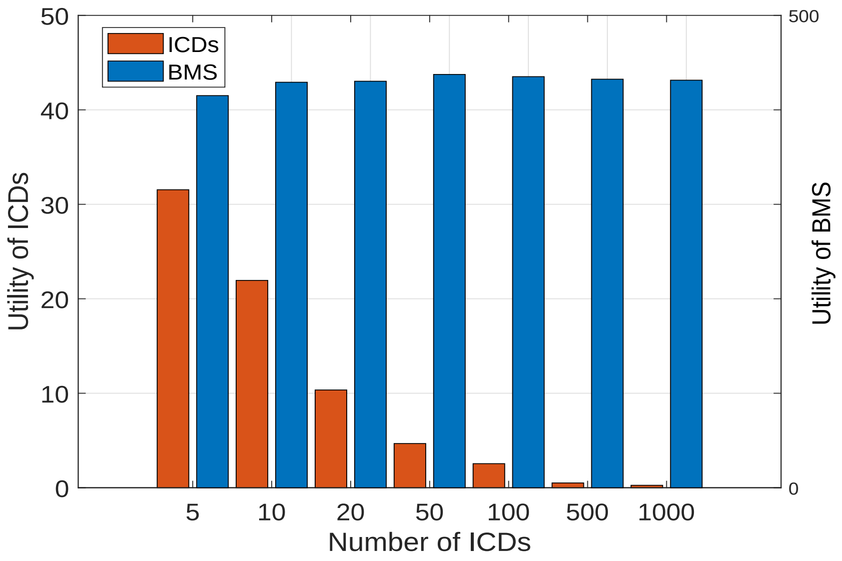


Figure 3.11: Utilities of BMS and ICDs versus the number of participant ICDs.

3.6 Chapter Summary

This chapter proposed a hierarchical game model to incentivize computation sharing in smart buildings. We used a Stackelberg game to model the interaction between the BMS and the ICDs, in which the BMS first provided the optimal unit price to the ICDs; then, the ICDs derived the amounts of computing resources to share. Moreover, the ICDs' decisions were also affected by their rivals' strategies, which were formulated by a Cournot game. We analyzed and obtained the business relationship between the BMS's demand for data to be processed and the ICDs' sharing supply under the premise of ensuring that the BMS and ICDs reach the SPNE. According to the simulation results, the computational capacity was enhanced in the building, and each party obtained the optimal profit by our proposed scheme.

Chapter 4

Hierarchical Game-based Resource Scheduling under Incomplete Information

The hierarchical game model proposed in Chapter 3 can effectively incentivize the horizontal collaboration in the device layer. However, it requires the complete information of all participants during resource scheduling, which is not practical in some IoT scenarios. This chapter further explores the incentive mechanism for collaborative computing without complete information in emergency communication networks (ECNs).

Due to disasters' urgent and unpredictable nature, the EMS faces an enormous challenge of real-time data analysis without the complete information from ECNs. In this chapter, we propose an IITG model to realize collaborative computing at the edge of ECNs, incentivizing ICDs to share their computation resources by maximizing the utilities of EMS and ICDs without the prior knowledge of participants. Furthermore, we develop a N-IITG algorithm to seek the unique BNE, where the EMS can dynamically optimize its pricing mechanism, and the ICDs can select the optimal computation workload accordingly. Simulation results reveal that N-IITG outperforms the existing incomplete information-based methods in terms of computation latency and participants' utilities.

4.1 Introduction

To reduce the harmful effects of unpredictable disasters, the EMS requires timely analysis of the massive amount of data for providing effective disaster relief strategies [64]. This stringent requirement brings a significant challenge to real-time data computation in ECNs. Moreover, due to the highly uncertain nature of disasters, it is extremely difficult to obtain complete information of ECNs in advance [65]. Therefore, how to design an effective data processing mechanism based on incomplete information becomes essential for ECNs.

To relieve the computation pressure in ECNs, some researchers propose the collaborative computing at the edge of networks, by utilizing the spare computational resources from s-martphones, desktop PCs, or other computational devices in proximity to achieve a dynamic increase in computational capacity [10, 66]. It has a similar architecture of collaborative computing in Chapter 3. While the concept of collaborative computing is promising, engaging ICDs for sharing could be difficult as they have no commitments to do so. They may expect compensation since computation offloading potentially affects local computing tasks. Therefore, successful exploitation of collaborative computing at the edge of ECNs requires a careful design of the incentive mechanism. The game theory-based approach is one of the suitable tools to model and analyze the incentive mechanism [67]. Nevertheless, the existing mechanisms in the literature have overlooked several critical issues. First, recent literature is largely based on the assumption that the competitive participants have the full knowledge of the network and local constraints [68, 61], such as the path loss fading and computation capacities. However, it is extremely difficult to acquire such information of ECNs in advance in most realistic scenarios. Furthermore, the self-interested participants are non-cooperative in ECNs, and thus the competitions, in this case, exist not only between the EMS (service consumer) and the ICDs (service provider), but also among the ICDs. Although the previous game-theoretic studies can cope with the competition between the service provider and the service consumer [68, 69, 70] or inside service providers [61, 71], few consider the above competitive relationships simultaneously.

In this chapter, we propose an IITG model to incentivize collaborative computing at the edge of ECNs, where the objective is to maximize the utilities of the EMS and the ICDs through optimizing the pricing strategy and the computing resources allocation simultaneously. Specifically, the interactions between the EMS and the ICDs are formulated as a two-tier game model, which jointly combines the Stackelberg game and the Cournot game. Through this model, the EMS can dynamically optimize its pricing mechanism, and the ICDs can select the optimal computation workload accordingly. Furthermore, depending on what the EMS and the ICDs know, we seek the BNE under incomplete information, and a N-IITG algorithm is developed to reach the unique BNE by iterations. Simulation results demonstrate that our design achieves a near-optimal performance of complete information and outperforms the existing incomplete information based methods in terms of computation latency and participants utilities.

The rest of this chapter is organized as follows. Section 4.2 describes the system model, especially for incomplete information model. In Section 4.3, we analyze the system with a two-tier game framework and yield the unique BNE. Finally, we evaluate the performance of our method in Section 4.4 and summarize the chapter in Section 4.5.

4.2 System Model

We consider a collaborative computing scenario illustrated in Figure 4.1. There are I ICDs, denoted as $\mathcal{I} \triangleq \{1, 2, \ldots, I\}$, locating at the edge of ECNs to offer computational services for the EMS. During a time slot, Q_{EMS} bits of data are processed for emergency management, and we assume P_{EMS} as the unit profit that the EMS benefits from the data analysis. As a centralized datacenter, the EMS needs to determine the optimal unit price p^{opt} (per bit) to incentivize collaborative computing. Then, the optimal amount of computing resources q_i^{opt} is provided by each ICD$_i$. Once the collaboration is set up, the processed data will be transmitted back to the EMS for emergency management, and each ICD gets its compensation for sharing its computation resources.

To incentivize the collaborative computing from the ICDs and satisfy the computation requirement from the EMS, the objective of this chapter is to jointly optimize the pricing strategy p^{opt} and the computing resources allocation $\{q_i^{\text{opt}}|_{i\in\mathcal{I}}\}$ to maximize the utilities of the EMS and the ICDs. We adopt the payoff to each participant as its utility. The mathematical model is described in detail as follows.

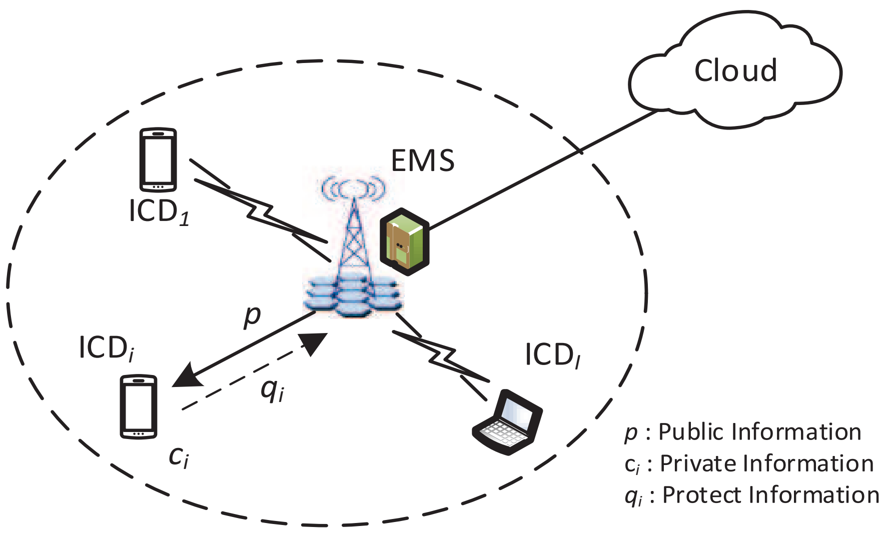


Figure 4.1: Illustration of the collaborative computing architecture.

4.2.1 Utility Model

The utility of the EMS is defined as the profit through computing offloading minus the sum of its payments to the ICDs, which can be given by

$$U_{\text{EMS}}(p) = (P_{\text{EMS}} - p)\min\{Q_{\text{EMS}}, Q_{\text{ICD}}\}, \tag{4.1}$$

where the unit profit P_{EMS} benefiting from the data analysis is a known parameter, which may vary with the importance of the emergency management task. The price strategy p denotes the unit price of computing offloading offered by the EMS. The total amount of the shared computing resources paid by the EMS is the minimum value of Q_{EMS} and Q_{ICD}, where Q_{ICD} is the total amount of computing resources shared by the ICDs. We assume rationality in this chapter that the EMS will not buy the computing resources beyond its requirement, and the ICDs will not share computing resources without benefit.

The utility U_i of ICD$_i$ is defined as the reward paid by the EMS minus the cost to accomplish q_i bits of data processing, which can be expressed as

$$U_i(q_i) = (p - c_i)q_i, \tag{4.2}$$

where q_i denotes the number of computing resources shared by ICD$_i$. The unit cost c_i (per bit) for sharing denotes the transmission energy and computing cost for offloading one bit of data, which dynamically varies with CPU performance, path loss fading, and other network environments [66].

4.2.2 Incomplete Information Model

Since ECNs have the characteristic of high uncertainty, it is impractical to get complete information in advance during collaborative computing. Similar to other incomplete information scenarios [69, 70, 71], the EMS may know each ICD's characteristics or not, such as collaboration costs c_i. Thus, we develop an incomplete information model depending on whether the information is known by others.

- *Public Information*: Known by all participants.
- *Protect Information*: Known by part of participants.
- *Private Information*: Only known by participant itself.

Then, we can classify the parameters in (4.1) and (4.2) into three categories as listed in Table 4.1.

Table 4.1: Incomplete information model

Information Category	**Parameter**
Public Information	p
Protect Information	q_i
Private Information	$Q_{\text{EMS}}, P_{\text{EMS}}, c_i$

Considering the practical situation, we assume that the unit price p is common knowledge of all participants, including the EMS and the ICDs. The offloading strategy q_i is the protect information which can be obtained by the EMS and each ICD. The private information c_i is the actual unit cost known to each ICD but is not known by EMS or other ICDs. Also, the computing task parameters Q_{EMS} and P_{EMS} are the private information of the EMS.

By exploring (4.1) and (4.2), we notice that the EMS and the ICDs have different objectives. The EMS expects the ICDs to accomplish a more massive task with a lower unit price, and the ICDs are the opposite. Meanwhile, each non-cooperative ICD competes to acquire a higher payoff by sharing a larger amount of computing resources; however, with more computing resources supplied by ICD, the EMS can damp down the unit price of computing offloading according to the theory of demand and supply. Hence, the competitions exist not only between the EMS and the ICDs but also among multiple ICDs. This motivates us to propose an incomplete information based two-tier game (IITG) model to tackle the joint optimization problem of the EMS and the ICDs.

4.3 Two-tier Game Analysis under Incomplete Information

In this section, we analyze the optimal strategies of the EMS and the ICDs by jointing the Stackelberg game and the Cournot game under incomplete information. The optimization problems for the EMS and the ICDs are conducted by the following two tiers. We use the derived results to prove the existence of BNE.

- *Tier I*: Players: the EMS (a single leader); Strategy: the unit price p; Utility: $U_{\text{EMS}}(p)$ given in (4.1).
- *Tier II*: Players: the ICDs (multiple competitive followers); Strategy: the shared computing resources $\{q_i|_{i\in\mathcal{I}}\}$; Utility: $\{U_i(q_i)|_{i\in\mathcal{I}}\}$ given in (4.2).

We first model the interaction between the EMS and the ICDs as a Stackelberg game, i.e., the pricing strategy p and the amount of computation offloading q_i. In our proposed collaborative computing model, the EMS (a single leader) has the first-mover advantage to imposes the optimal unit price p^{opt} of computation offloading for the ICDs. Then, the ICDs (multiple followers) can divide the optimal amount of computation offloading q_i^{opt} by following the pricing strategy announced by the EMS. Both the EMS and the ICDs aim to obtain the optimal utilities by maximizing their payoff. These characteristics align well with the Stackelberg game model. The objective function of the EMS is

$$\max_{p^{\text{opt}}} U_{\text{EMS}}(p|P_{\text{EMS}}, Q_{\text{EMS}}, \boldsymbol{q})$$

$$s.t. \quad 0 \le p^{\text{opt}} \le P_{\text{EMS}}, \tag{4.3}$$

where $\boldsymbol{q} = \{q_1, q_2, \ldots, q_i, \ldots, q_I\}$ represents the computation offloading strategies of all ICDs. The derived optimal unit price p^{opt} paid to the ICDs should be no more than the unit profit P_{EMS} that the EMS can benefit from computation offloading; otherwise, the utility of the EMS will be negative.

Then, the ICDs, acting as multiple followers in the Stackelberg game, deduce their optimal shared computing resources by following the pricing strategy announced by the EMS (leader). Moreover, the internal competitions among multiple ICDs are modeled as a Cournot game, which is one of the most popular types of games used to model the interactions among multiple strategic generators. Since each ICD's payoff depends not only on its own strategy but also on the decisions of its rivals, the optimization problem for ICD$_i$ can be expressed as

$$\max_{q_i^{\text{opt}}} U_i(q_i | p, \boldsymbol{q}_{-i}), \forall i \in \mathcal{I}$$

$$s.t. \quad \sum_{i \in \mathcal{I}} q_i^{\text{opt}} \le Q_{\text{EMS}}, \tag{4.4}$$

where $\boldsymbol{q}_{-i}$ is the strategy set of the ICDs other than ICD$_i$. $\sum_{i \in \mathcal{I}} q_i^{\text{opt}} \le Q_{\text{EMS}}$ denotes that the total amount of computation offloading is no more than the EMS's demand because all participants in the game are rational and thus will not share computing resources without benefit.

Furthermore, since the internal competition among multiple ICDs fits the Cournot game model, and thus we estimate p by the Cournot price function model to depict the competitive relationship. The unit price p is a decreasing function with respect to the total amount of computation offloading. It is subject to $\frac{\partial p}{\partial Q_{\text{ICD}}} < 0$, where $Q_{\text{ICD}} = \sum_{i \in \mathcal{I}} q_i$. To simplify the problem, we estimate p with a universal linear model as in references [61, 71], i.e.,

$$p = \max \left\{ 0, p_{\max} - \alpha \sum_{i \in \mathcal{I}} q_i \right\}, \tag{4.5}$$

where $p_{\max}$ is the price function intercept that denotes the maximum unit price offered by the EMS, and α is a nonnegative coefficient that reflects the change trend of the unit price and the total amount of shared computing resources, i.e., $\alpha = -\frac{\partial p}{\partial Q_{\text{ICD}}}$.

Next, we adopt backward induction [68, 69, 70] for IITG analysis as follows.

4.3.1 Tier II: Optimal Computing Offloading Strategy

In this tier, we optimize the computing offloading strategy for each ICD given the EMS's (leader) strategy.

Theorem 4.3.1 *Each* ICD$_i$ *(follower) determines the optimal amount of computation offloading q_i^{opt} given the unit price strategy p and α from the EMS (leader), given by*

$$q_i^{\text{opt}} = \frac{p - c_i}{\alpha}, \tag{4.6}$$

where c_i is the private information known by each ICD itself.

Proof We substitute (4.5) into (4.2); thus, the utility of ICD$_i$ can be given by

$$U_i(q_i) = -\alpha q_i^2 + \left(p_{\max} - \alpha \sum_{k \neq i} \widehat{q_k} - c_i\right) q_i. \tag{4.7}$$

Under incomplete information model, the rivals' decision $\{q_k|_{k\neq i}\}$ is the protect information to ICD$_i$, and thus each ICD only can estimate the rivals' decision as $\widehat{q_k}$. From (7), U_i is a continuous quadratic function of q_i by assuming $\sum_{k\neq i} \hat{q}_k$ is fixed, and the second derivative of U_i with respect to q_i is $\frac{\partial^2 U_i}{\partial q_i^2} = -2\alpha$, where $\alpha > 0$. As $\frac{\partial^2 U_i}{\partial q_i^2} < 0$, U_i is a concave function of q_i. Therefore, by setting the first derivative equal to zero, the optimal output of ICD$_i$ satisfies $q_i^{\text{opt}} = (p_{\max} - c_i - \alpha \sum_{k\neq i} \widehat{q_k^{\text{opt}}}) \big/ 2\alpha$. As p is known, we can substitute (4.5) into the above equation, and obtain q_i^{opt} expressed as (4.6). This finishes the proof of Theorem 4.3.1.

The next problem is to adjust the pricing strategy p to maximize the utility of the EMS while guaranteeing its computational demand.

4.3.2 Tier I: Optimal Unit Price Strategy

Under incomplete information, the EMS has a rough estimation on each ICDs' cost information. We consider that the unit cost of ICD$_i$ is a variable following certain distributions.

Let us denote $c_{i,n}$ as the unit cost of ICD$_i$ with probability $\gamma_{i,n}$. The probability $\gamma_{i,n}$ satisfies $\sum_{n\in\mathcal{N}_i}\gamma_{i,n} = 1$, where $n \in \mathcal{N}_i \triangleq \{1, 2, \ldots, N_i\}$ indicates there are N_i kinds of possible values for unit cost c_i.

Theorem 4.3.2 *Given the unit profit P_{EMS} and the computing workload Q_{EMS}, the EMS can determine the optimal price strategy $p(Q_{\text{ICD}})$ by the the estimation of the unit cost of the ICDs, which can be given by*

$$p^{\text{opt}}(Q_{\text{ICD}}) = \left(\frac{I+1}{2}P_{\text{EMS}} - \frac{I-1}{2I}\sum_{i\in\mathcal{I}}\mathbb{E}_{n\in\mathcal{N}_i}[c_{i,n}]\right) - \frac{IP_{\text{EMS}} - \sum_{i\in\mathcal{I}}\mathbb{E}_{n\in\mathcal{N}_i}[c_{i,n}]}{2Q_{\text{EMS}}}Q_{\text{ICD}}, \tag{4.8}$$

where $\mathbb{E}_{n\in\mathcal{N}_i}[c_{i,n}]$ denotes the estimation of the unit cost of ICD$_i$ *and $Q_{\text{ICD}} = \sum_{i\in\mathcal{I}} q_i$ denotes the enhanced computational capacity for emergency management on demand.*

Proof From (4.6) indicated in Theorem 4.3.1., the optimal amount of shared computing resources from ICD$_i$ satisfies $q_{i,n}^{\text{opt}} = (p - c_{i,n})/\alpha$ when its actual unit cost is $c_{i,n}$. From the perspective of the EMS, it can determine the optimal strategy $q_i^{\text{opt}} = q_{i,n}^{\text{opt}}$ with the probability $\gamma_{i,n}$. Thus the expected amount of shared computing resources can be expressed as

$$\begin{aligned}\mathbb{E}[Q_{\text{ICD}}] &= \sum_{i\in\mathcal{I}}\mathbb{E}_{n\in\mathcal{N}_i}[q_{i,n}^{\text{opt}}] \\ &= \sum_{i\in\mathcal{I}}\sum_{n\in\mathcal{N}_i}\frac{p - c_{i,n}}{\alpha}\gamma_{i,n} \\ &= \frac{Ip - \sum_{i\in\mathcal{I}}\sum_{n\in\mathcal{N}_i} c_{i,n}\gamma_{i,n}}{\alpha},\end{aligned} \tag{4.9}$$

where $\sum_{n\in\mathcal{N}_i} c_{i,n}\gamma_{i,n}$ is the expected unit cost of ICD$_i$, which can be denoted as $\mathbb{E}_{n\in\mathcal{N}_i}[c_{i,n}]$.

Substituting (4.5) into (4.9), and the total expected amount of shared computing resources is $\mathbb{E}[Q_{\text{ICD}}] = (Ip_{\max} - \sum_{i\in\mathcal{I}}\mathbb{E}_{n\in\mathcal{N}_i}[c_{i,n}])/\alpha$. The expected utility of the EMS can be represented as (4.10) by substituting the above equation into (4.1).

$$
\begin{aligned}
\mathbb{E}\left[U_{\text{EMS}}(p_{\max},\alpha)\right] = &- \frac{I}{\alpha(I+1)^2}\left[p_{\max} - \left(\frac{I+1}{2}P_{\text{EMS}} - \frac{I-1}{2I}\sum_{i\in\mathcal{I}}\mathbb{E}_{n\in\mathcal{N}_i}[c_{i,n}]\right)\right]^2 \\
&+ \frac{1}{4\alpha I}\left(IP_{\text{EMS}} - \sum_{i\in\mathcal{I}}\mathbb{E}_{n\in\mathcal{N}_i}[c_{i,n}]\right)^2 .
\end{aligned}
\tag{4.10}
$$

When α is fixed, $\mathbb{E}\left[U_{\text{EMS}}\right]$ is a continuous quadratic function of $p_{\max}$. Similar to the proof of Theorem 1., we can obtain the optimal parameter $p_{\max}^{\text{opt}}$ by setting the first derivative equal to zero as

$$
p_{\max}^{\text{opt}} = \frac{(I+1)P_{\text{EMS}}}{2} - \frac{(I-1)\sum_{i\in\mathcal{I}}\mathbb{E}_{n\in\mathcal{N}_i}[c_{i,n}]}{2I}. \tag{4.11}
$$

From (4.10), we can obtain that $\mathbb{E}\left[U_{\text{EMS}}\right]$ is a monotonically decreasing function with respect to α. Substituting $p_{\max}^{\text{opt}}$ back to $\mathbb{E}[Q_{\text{ICD}}]$, and it can be rewritten as

$$
\mathbb{E}[Q_{\text{ICD}}] = \frac{(IP_{\text{EMS}} - \sum_{i\in\mathcal{I}}\mathbb{E}_{n\in\mathcal{N}_i}[c_{i,n}])}{2\alpha}. \tag{4.12}
$$

To satisfy the computation requirement from the EMS, the optimal parameter α^{opt} can be determined by replacing $\mathbb{E}[Q_{\text{ICD}}]$ with Q_{EMS} as

$$
\alpha^{\text{opt}} = \frac{(IP_{\text{EMS}} - \sum_{i\in\mathcal{I}}\mathbb{E}_{n\in\mathcal{N}_i}[c_{i,n}])}{2Q_{\text{EMS}}}. \tag{4.13}
$$

Finally, we substitute $p_{\max}^{\text{opt}}$ and α^{opt} into (4.5) and obtain the optimal unit price expressed as (4.8). This finishes the proof of Theorem 4.3.2.

In this way, the EMS and the ICDs can yield BNE $\{p^{\text{opt}}, \{q_{i,n}^{\text{opt}}; c_{i,n}\}_{n\in\mathcal{N}_i, i\in\mathcal{I}}\}$. Since there are N_i kinds of possible values for ICD$_i$, BNE is not unique. To address this problem, we design the N-IITG algorithm to iteratively get the unique near-optimal strategies of the EMS and the ICDs based on the IITG model. Next, we provide the details of the near-optimal algorithm.

4.3.3 N-IITG Algorithm

To address the uniqueness problem of incomplete information, we propose the N-IITG algorithm to yield the unique BNE of the EMS and the ICDs.

Algorithm 2 N-IITG Algorithm: Seeking the unique BNE

```
Input: $Q_{\text{EMS}}$, $P_{\text{EMS}}$, $\{c_{i,n}, \gamma_{i,n}\}_{n\in\mathcal{N}_i, i\in\mathcal{I}}$, $\epsilon$, $T$
Output: $p^{\text{opt}}$, $\{q_i^{\text{opt}}|_{i\in\mathcal{I}}\}$
1: Initially, the EMS estimates of the unit cost by $\hat{c}_i^{\,0} \Leftarrow \mathbb{E}_{n\in\mathcal{N}_i}[c_{i,n}]$.
2: initialize $t \Leftarrow 1$.
3: repeat
4:     EMS obtains $p^{\text{opt}}$ and $\alpha^{\text{opt}}$ by (4.8).
5:     for ICD$_i$ do
6:         $q_i^{\text{opt}} \Leftarrow \frac{p^{\text{opt}}-c_{i,n}}{\alpha^{\text{opt}}}$ by (4.6).
7:     update $\hat{c}_i^{\,t} \Leftarrow p^{\text{opt}} - \alpha^{\text{opt}} q_i^{\text{opt}}, \forall i \in \mathcal{I}$
8:     set $t \Leftarrow t + 1$.
9: until $t \geq T$ or $|\hat{c}_i^{\,t} - \hat{c}_i^{\,t-1}| \leq \epsilon, \forall i \in \mathcal{I}$
```

Different from the complete information scenario, the EMS and the ICDs need to find out the unique optimal strategies by iterations from step 3 to step 9. Under incomplete information model, the EMS only has the estimation of the unit cost of ICD$_i$, and it initializes the estimated unit cost $\hat{c}_i$ in expectation and deduces the optimal unit price p^{opt} by (4.8). Then, each ICD determines its particular optimal strategy q_i^{opt} by (4.6) based on the actual unit cost. In each iteration, the EMS can update the estimated unit cost $\hat{c}_i$ by the observed actions q_i^{opt} from the ICDs. Then, the EMS and the ICDs recalculate their optimal strategies until the unit cost of each ICD converges to a stable value, or the number of iterations exceeds its maximum threshold. Finally, all participants achieve the unique BNE where no one will alter its strategy if the strategies of the others remain unchanged.

Next, we discuss the time complexity of the N-IITG algorithm. As shown in Algorithm 1, the time complexity of unit cost estimation is $\mathcal{O}[I]$. All participants can determine their strategies by the closed-form solution, i.e., $\mathcal{O}[1]$. Therefore, the overall time complexity is $\mathcal{O}[NI]$, where N is the iteration count required to converge. This result demonstrates that the N-IITG algorithm can be finished in polynomial time.

4.4 Simulation Results

In this section, we present numerical results to evaluate the performance of the N-IITG algorithm. Parameter settings are given first. The EMS's demand Q_{EMS} is set as to 50 MB and P_{EMS} to 20 per MB. We consider that there are 10 ICDs in proximity having idle computing resources, i.e., $I = 10$. The unit cost c_i for sharing computing resources is private information, i.e., each ICD knows well its actual unit cost c_i, while the EMS and the other ICDs only have a rough estimation based on the probability distribution, and then c_i is estimated as $\hat{c}_i = \mathbb{E}_{n\in\mathcal{N}_i}[c_{i,n}]$.

Here, we take Bernoulli distribution for instance, and divide the unit cost into two types for simplicity, i.e., $c_i \triangleq \{c_i^L, c_i^H\}$, and the probability is $\Gamma_i \triangleq \{\gamma_i^L, \gamma_i^H\}$, where $\gamma_i^L + \gamma_i^H = 1$. The low unit cost c_i^L denotes low local computing intensity and transmission power, and c_i^H is the exact opposite case. According to these settings, we assume that the unit cost c_i is uniformly and randomly distributed over [1,10] ($/MB).

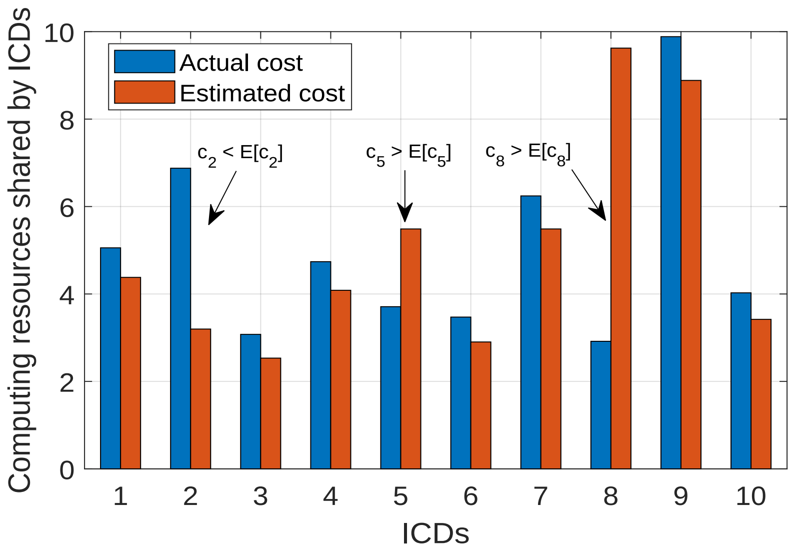
According to Theorem 4.3.2., the EMS determines its optimal unit price p^{opt} based on the estimation of the unit cost, i.e.,

$$p^{\text{opt}}(Q_{\text{ICD}}) = \left(\frac{I+1}{2}P_{\text{EMS}} - \frac{I-1}{2I}\sum_{i\in\mathcal{I}}\left(c_i^L\gamma_i^L + c_i^H\gamma_i^H\right)\right) - \frac{IP_{\text{EMS}} - \sum_{i\in\mathcal{I}}\left(c_i^L\gamma_i^L + c_i^H\gamma_i^H\right)}{2Q_{\text{EMS}}}Q_{\text{ICD}}. \tag{4.14}$$

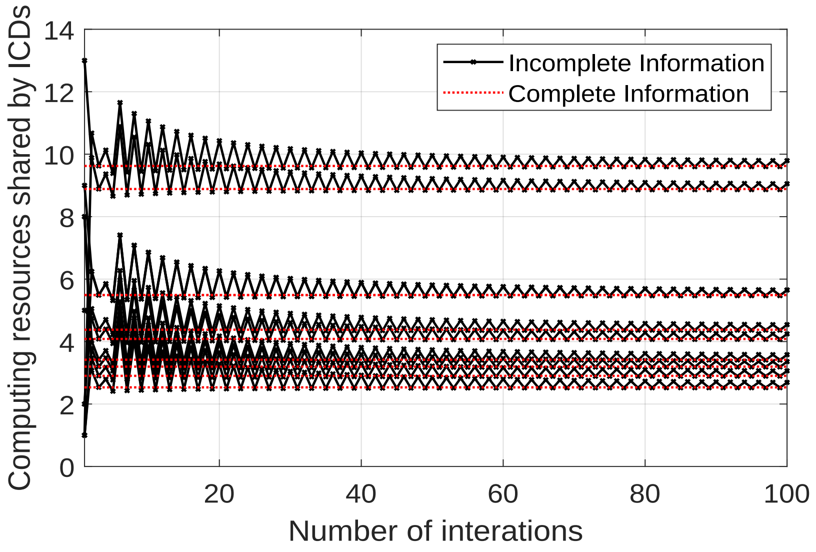
Then, according to Theorem 4.3.1., ICD$_i$ determines its optimal amount of shared computing resources q_i^{opt} according its actual unit cost, which can be given by

$$q_i^{\text{opt}} = \begin{cases} q_i^{L\text{opt}} = \dfrac{p^{\text{opt}} - c_i^L}{\alpha}, & c_i = c_i^L \\ q_i^{H\text{opt}} = \dfrac{p^{\text{opt}} - c_i^H}{\alpha}, & c_i = c_i^H. \end{cases} \tag{4.15}$$

As shown in Figure 4.2, we investigate the shared computing resources of each ICD under incomplete information and compare it with the complete information scenario. Here, the result of complete information can be derived by c_i instead of $\hat{c}_i$. To analyze the difference of actual unit cost and estimated unit cost, we only set c_2, c_5 and c_8 as unknown parameters, which should be estimated by the expectation. As illustrated in Figure 4.2 (a), the ICDs are willing to offload more workload than the EMS's prediction if actual unit cost is smaller than estimated unit cost, which makes the participants unable to reach equilibrium after one iteration. Next, we simulate the convergence iterations with our proposed N-IITG algorithm. The upper limit ϵ is set as 10^{-2}. From Figure 4.2 (b), the computing resources shared by each ICD converges to the value under complete information, and the total amount of the shared computing resources satisfy the EMS's computation requirement. This result indicates that the N-IITG algorithm can achieve a near-optimal performance of complete information under the condition of incomplete information.



(a) One iterations

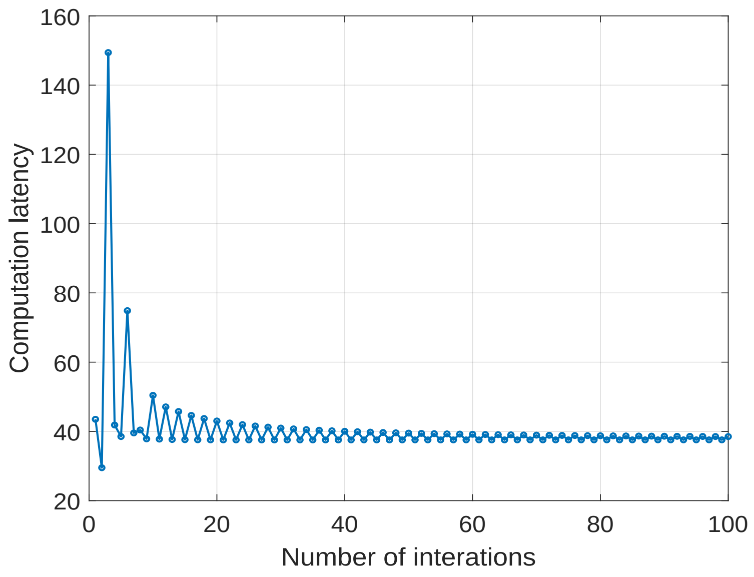


(b) Multiple iterations

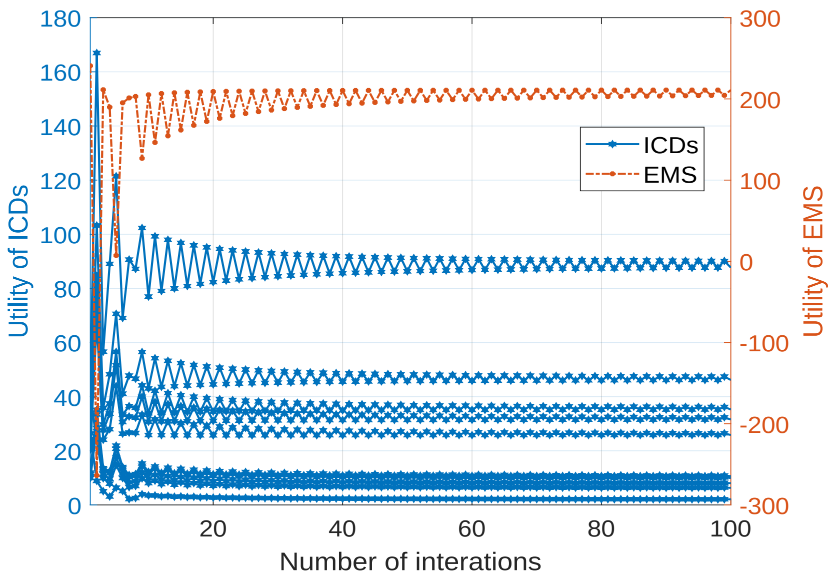
Figure 4.2: Shared computing resources versus c_i and $\hat{c}_i$.

In Figure 4.3, we evaluate the convergency performance of the EMS and the ICDs from the perspective of computing latency and their utilities. Since the unit cost c_i is related to each ICD's local computing intensity, we assume the processing rate varies inversely with c_i. The computing latency is defined as the maximum value of all ICDs' processing time. Figure

4.3 (a) shows the computing latency can be significantly decreased and fluctuate to a stable value with the algorithm iterations. Meanwhile, when achieving the unique BNE by the N-IITG algorithm, each participant obtains the maximal profit under equilibrium in collaborative computing, as shown in Figure 4.3 (b).



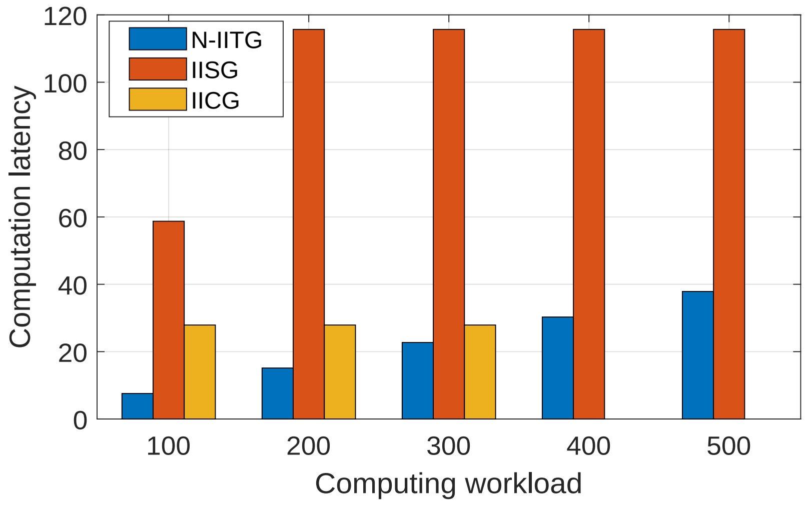
(a) Computation latency



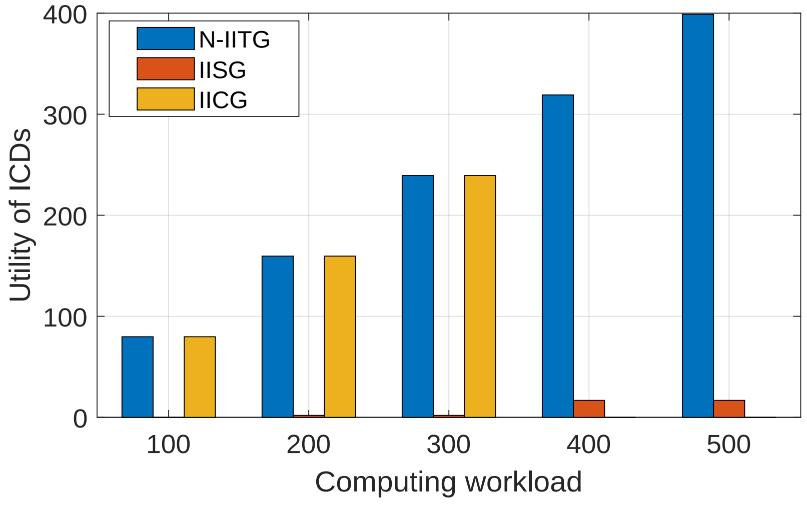
(b) Utility of participants

Figure 4.3: Convergence iterations with N-IITG algorithm.

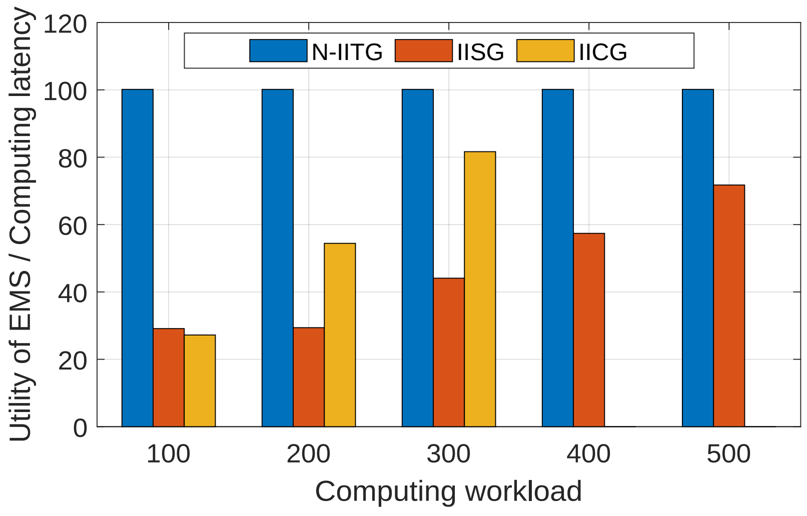
In Figure 4.4, we compare the performance of the N-IITG algorithm with two algorithms in terms of the computation latency and the participants' utilities, i.e., incomplete information



(a)



(b)



(c)

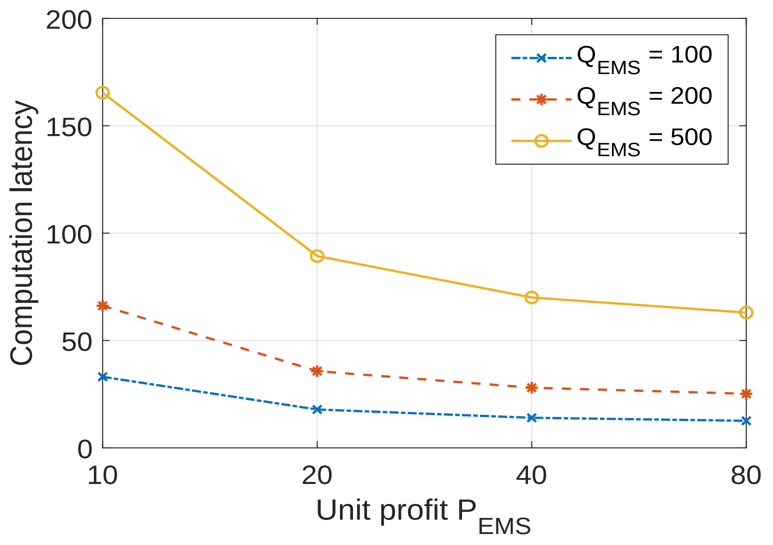
Figure 4.4: Algorithm comparison versus computing workload Q_{EMS}.

based Stackelberg game algorithm (IISG) [70], and incomplete information based Cournot game algorithm (IICG) [71]. Considering that the unit cost is proportional to the square of processing rate [66], we define the computation latency as $max\{q_i \Big/ \sqrt{c_i}\ |_{i\in\mathcal{I}}\}$. Figure 4.4(a) and Figure 4.4(b) demonstrate that our proposed N-IITG algorithm can achieve a lower computation latency and a higher average utility of ICDs under different computing workloads. Since IICG cannot dynamically adjust its pricing strategy, the computing task fails to be processed when it increases to 400 MB or above. Therefore, the computation latency of IICG is infinite, and the utilities are zero in Figure 4.4. Furthermore, apart from the utility, the computation latency is also important to the performance of the EMS in ECNs. Therefore, we compare the performance of the EMS by the ratio of the above two items, and the N-IITG algorithm outperforms the other two algorithms as well. That is because our proposed N-IITG algorithm considers not only the interaction between the EMS and the ICDs but also the internal competitions among the ICDs. In this way, the EMS can dynamically adapt its pricing strategy to inspire the competitive ICDs for sharing sufficient computing resources, which contributes to relieve the computation pressure in ECNs. Each ICD involved in collaborative computing can maximize its utility by selecting the optimal computation workload.

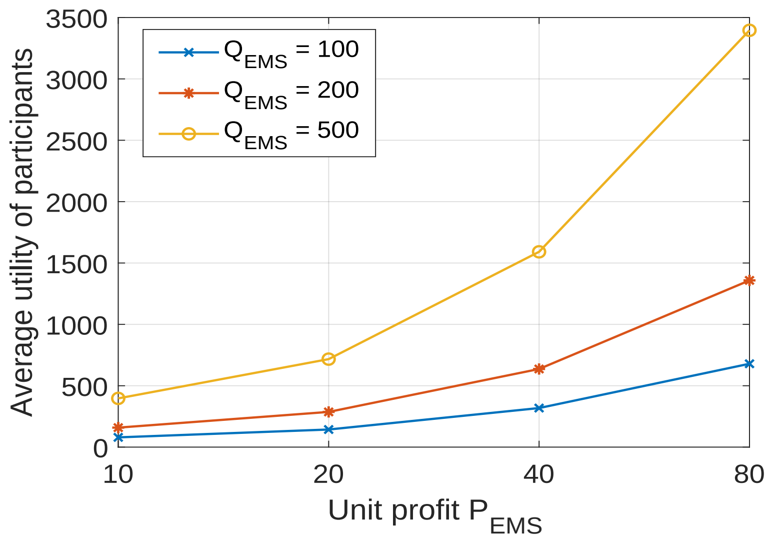
The unit profit P_{EMS} benefiting from data analysis is a known parameter, which may vary with the importance of the emergency management task. To better evaluate the effect of the parameter P_{EMS}, we conduct the experiments in terms of computing latency and participants' utilities under different values as shown in Figure 4.5. We can obtain that the computation latency decrease with an increase of P_{EMS}, and the participants can obtain higher utilities. This finding indicates that when the emergency management task is urgent, the unit profit P_{EMS} should be set higher in order to get a faster data processing.

4.5 Chapter Summary

This chapter proposed an IITG model to incentivize collaborative computing in ECNs, which jointly combined the Stackelberg game and the Cournot game. Depending on the given information of the EMS and the ICDs, we analyzed the BNE of the EMS and the ICDs under incomplete information, and further designed the N-IITG algorithm that can iteratively conver-



(a)



(b)

Figure 4.5: Computational latency and utilities versus unit profit P_{EMS}.

gent to the unique BNE. According to the simulation results, the proposed scheme achieved a significant increase in computational capacity while each participant obtained the optimal profit.

Chapter 5

Computational Latency Pricing-based Resource Scheduling

As discussed in the previous two chapters, utilizing the idle computing resources from the distributed IoT devices can sustainably increase the computational capacity and thereby effectively alleviate the pressure on resource-constrained devices. Considering the ever-increasing diversity of applications in IoT systems, a reasonable incentive mechanism should consider not only the number of tasks processed but also the performance they have achieved.

This chapter is motivated by computing scenarios with latency-critical tasks that a computational latency-based resource scheduling mechanism is proposed to enable collaborative computing amongst IoT devices from the perspective of QoE performance. Specifically, we consider the collaborative computing system where a user offloads the computation-intensive and latency-sensitive tasks to multiple ICDs by a CSP. A computational latency-based pricing mechanism is first proposed from the perspective of the QoE performance, where the computing offloading price varies dynamically with the data processing rates; then, a game-theoretic computing task allocation approach is developed among the CSP and multiple ICDs to maximize all participants' profit. The CSP first determines the optimal task partition dynamically upon the tasks' arrival; then, the ICDs derive the optimal central processing unit-cycle frequency correspondingly. Simulation results demonstrate that the overall computational latency of our proposed mechanism is significantly decreased, and the profit of all participants is maximum in collaborative computing.

5.1 Introduction

By interconnecting devices, machines, and industrial processes, the IoT technology can support many vertical applications, including factory automation, smart grids, and intelligent transport systems [72, 73]. Frequently, such applications are large-scale and latency-sensitive [74], which bring a huge burden on real-time data computation. Since many IoT devices, e.g., sensors and actuators, are characterized by limited computational resources, migrating data processing from resource-constrained IoT devices to the platforms with extremely high computational capability [75, 76, 77] is considered as a sustainable approach for relieving such a pressure.

With the emerging of the IoT devices equipped with multi-core processors, ranging from smartphones to laptop computers, researchers have proposed to realize collaborative computing [78] by encouraging these computing devices to share their idle computational resources. A recent survey indicates that the average CPU utility of existing computing devices over the Internet is merely 6 to 12 percent [10]. Scavenging the enormous amount of spare computational resources over IoT can provide a new distributed and dynamic computing service platform according to demand, which is similar to Uber for transportation services and Airbnb for hospitality.

Successful exploitation of collaborative computing requires a careful design of the cooperative mechanism. The related studies devoted to this area can be roughly categorized into two groups: those focusing on the task allocation based on the tradeoff between CPU power consumption and computational latency by optimization-based approaches[79] and those focusing on the profit maximization by the economic and pricing based methods[25]. Our study here belongs to the second category. Since computing devices have no commitments for collaboration, an efficient incentive scheme is critical to encourage the sharing of idle computational resources. The primary benefit of the economic methodology is the revenue generation[25], and thus it is regarded as a suitable tool for addressing incentive problems.

Specifically, we consider a computational offloading market through collaborative computing when the end-user (*buyer*) cannot complete the computational-intensive and latency-sensitive tasks in time. In this market, the tasks can be offloaded to multiple ICDs (*seller*) by virtue of the CSP (*intermediary*), such as a wireless access point. To enable collaborative computing sustainably, we will address the following two technical and economic challenges:

1) How to formulate the pricing mechanism in the resource sharing market? Compared with the flat-rate pricing, smart pricing[80] is preferred in practical applications. Our design is motivated by computing scenarios with latency-critical tasks. Different data processing rates lead to different QoE for IoT applications [81]. Hence, a reasonable pricing strategy should consider not only the number of tasks processed but also the QoE of tasks processed. However, we notice that state-of-the-art literature mainly focuses on the processing time or the computational capacity but does not consider the affection of QoE, which inspires us to propose an efficient pricing mechanism by comprehensively considering the QoE of collaborative computing. Here we take latency as a metric of QoE for consideration. To reduce the computing response time, we propose a computational latency-based pricing mechanism where the unit price offered by the *buyer* varies dynamically with the data processing rates that the *seller* can provide. A higher data processing rate is encouraged by gaining more payoff to improve computational latency performance.

2) How to allocate computation tasks by jointly maximizing the profit of all participants? We adopt the Stackelberg game model [60] to study the interactions between the CSP and the ICDs. In the Stackelberg game model, one player (leader) moves first, and then the others (followers) move sequentially. Each player can maximize its payoff by finding the SPNE, which aligns well with the interactions between the service providers and the service consumers in computing task allocation scenarios. In this work, the CSP has the first-mover advantage to determine the pricing strategy and the task partitions upon the tasks' arrival; then, the ICDs provide its computational resources correspondingly. Both of the CSP and the ICDs aim to maximize their payoff. These characteristics fit the Stackelberg game model. Thus, the CSP acts as a single leader, the ICDs function as multiple followers, and all participants in this game model find out SPNE with the goal of maximizing their payoff.

In summary, we propose a game-theoretic incentive mechanism for collaborative computing through computational latency-based pricing. The main contributions of this chapter are as follows:

- *Computational latency-based pricing*: A new computational latency-based pricing mechanism is proposed in reflecting the QoE performance in collaborative computing. We analyze the economic challenges in collaborative computing and design the computational latency-based pricing scheme where the computing offloading price varies dynamically with the data processing rates. The effect of latency on the pricing helps achieve the overall computational latency optimization.

- *Game-theoretic task scheduling*: We develop a game-theoretic incentive mechanism to encourage computing resource sharing. The interactive behaviors between the CSP and the ICDs are modeled as the Stackelberg game, where the objective is to obtain the maximum utilities for the CSP and the ICDs by seeking out SPNE through the dynamic pricing mechanism, the computation workload selection, and the CPU frequency control.

- *Near-optimal algorithm*: In practice, the data processing rate is restricted by power consumption and CPU capacity. For the constraint case in which we cannot get the closed-form solution directly, we present a near-optimal algorithm to find out the near-optimal strategies of the CSP and the ICDs. We also analyze the existence and uniqueness of SPNE by simulation.

The rest of this chapter is organized as follows. Section 5.2 provides a review of related works. Section 5.3 describes the system model and introduces the computational latency-based pricing scheme. In Section 5.4, we analyze the system with the Stackelberg game model, prove the existence of SPNE, and develop a near-optimal algorithm that can achieve SPNE. Finally, we evaluate our method's performance in Section 5.5 and summarize the chapter in Section 5.6.

5.2 Related Work

This section presents a review of the literature related to the pricing mechanisms and the task scheduling methods for collaborative computing in IoT-based systems.

In the literature, the Stackelberg game-based approaches have been widely used for modeling and analyzing the computing offloading problems [26]. For example, the authors of [57] proposed a joint optimization approach in IoT fog networks, where a fog node helps to offload data computing services from a data service operator to a data service subscriber. This approach was formulated as a Stackelberg game as well as a many-to-many matching game. By considering the competing characteristics of multi-tenant environments in cloud computing, [82] proposed a cloud resource allocation model based on an imperfect information Stackelberg game using a hidden Markov model in a cloud computing environment. To encourage IoT devices to share their unused resources, a Stackelberg game was formulated in [83] to decide the price that can be offered to mobile devices for application execution and the amount of execution unit that each mobile device is willing to provide. We notice that the Stackelberg game model aligns well with the interactions between the service providers and the service consumers in computing offloading scenarios.

Besides the computing workload selection, CPU-cycle frequency is another important factor affecting task scheduling [66]. The authors in [84] proposed an optimization framework of offloading from a single mobile device (MD) to multiple edge devices by considering fixed CPU frequency and elastic CPU frequency, respectively. It aimed to minimize both total tasks' execution latency and the MD's energy consumption by jointly optimizing the task allocation decision and the MD's CPU frequency. Moreover, the authors developed a distributed energy-efficient dynamic offloading and resource scheduling algorithm in [85], which consists of three sub algorithms of the computation offloading selection, the clock frequency control, and the transmission power allocation. These designs inspire us to apply the dynamic CPU-cycle frequency technology for task scheduling through game-theory analysis.

Furthermore, the pricing of the shared computational resource is a critical factor for collaborative computing since developing an appropriate pricing model will not only gain higher profits but also provide decisions for computation task scheduling [86]. Amazon Elastic Compute Cloud (Amazon EC2) [87] is a successful case that provides various pricing for computing capacity in the cloud. There are three ways to pay for Amazon EC2 instances: on-demand instances, reserved instances, and spot instances. Similarly, Microsoft Azure [88] has the pay-as-you-go pricing and the reserved pricing, and Google cloud [89] provides the resource-based pricing as well as sustained use discounts and committed use discounts. We notice that the existing pricing mechanisms are based on the usage of the time or the capacity but do not consider the affection of QoE, such as the latency. Moreover, researchers have proposed pricing-based mechanisms for users and offloading service providers in [90, 91]. To date, previous studies considered the latency as an impact factor of the profit, but not be reflected in the pricing mechanism directly. With the rapid growth of latency-sensitive IoT applications, such as factory automation, it is necessary to explore the pricing mechanism in collaborative computing from the perspective of the latency performance.

In summary, although collaborative computing can theoretically alleviate the computation pressure of the resource-limited devices, this method is still challenging to the practical application from two aspects, i.e., pricing and task scheduling. Thus, it is essential to design an effective incentive mechanism to address these two challenges.

5.3 System Model

We consider a computing offloading scenario managed by the CSP in the IoT system, as illustrated in Figure 5.1. There are a set of resource-constrained end-users and I ICDs, which are denoted as $\mathcal{I} \triangleq \{1, 2, \ldots, I\}$.

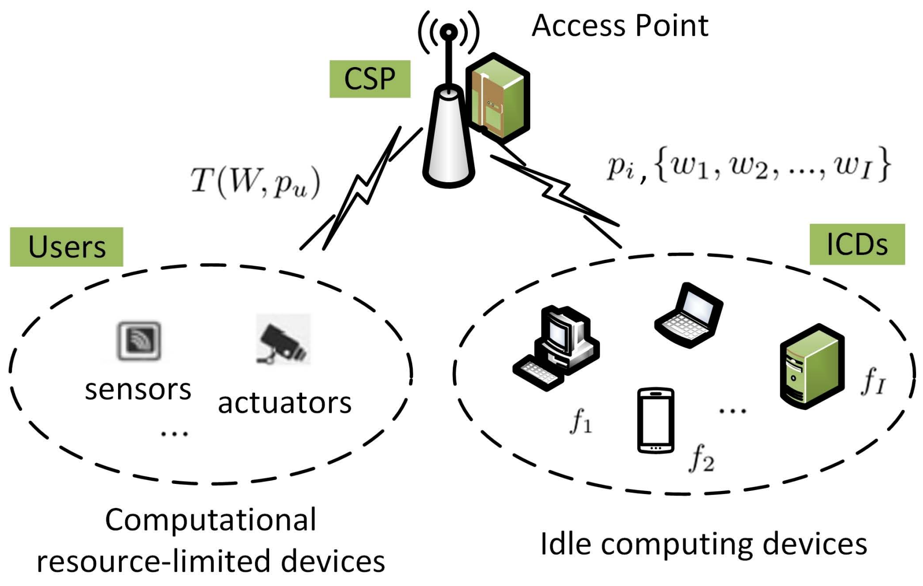


Figure 5.1: Illustration of computing offloading in collaborative computing architecture.

A quasi-static model is considered in this chapter where the environment remains unchanged during a computation offloading time slot. During a computing offloading time slot, there is W (in CPU cycles) computation workload to be processed from an end-user. Here, we assume that the simple task model for partial offloading[66], which is the data-partition model where the input computation tasks can be arbitrarily divided into bit-wise and executed by different ICDs independently. The unit price that the end-user would like to pay for this computing task is p_u (\$), which dynamically changes according to the computing response time of the CSP. In this way, the computation task is represented as $T(W, p_u)$.

The computation task T is offloaded to the CSP for collaborative computing. Upon the arrival of T, the CSP decides the optimal pricing strategy p_i^{opt} (\$) and the corresponding task partitions, i.e., the optimal workload w_i^{opt} (cycle) offloading to each ICD$_i$; then, the ICDs find out the optimal CPU-cycle frequency f_i^{opt} (GHz) distributively based on p_i^{opt} and w_i^{opt}. Notice that all participants are selfish and only follow the strategic behaviors that maximize their own utilities.

Based on the above collaborative model, we aim at jointly optimizing the pricing strategy p_i, the task partition w_i and each ICD's CPU-cycle frequency f_i, in order to minimize the overall computational latency t under the premise of maximizing the utilities of the CSP and the ICDs. Here, the overall computational latency t can be expressed as $t = \max\{w_i/f_i \,|_{i\in\mathcal{I}}\}$. We adopt the payoff to each participant as its utility. The mathematical model is described as follows.

5.3.1 Utility of CSP

As an intermediary, the utility of the CSP is defined as the revenue received from the end-user minus the sum of its payment to the ICDs, which can be expressed as

$$U_{\text{CSP}}(\{p_i, w_i|_{i\in\mathcal{I}}\}) = p_u W - \sum p_i w_i, \tag{5.1}$$

where the unit price p_u charged the end-user for computation offloading is a parameter related to the computing response time of the CSP, and p_i and w_i denote the unit price paid by the CSP to the ICDs and the workload offloaded to each ICD, respectively. Here, p_i is also a varying parameter which is related to the execution time of the ICD$_i$.

Given the computation task $T(W, p_u)$, the CSP, acting as the leader in the Stackelberg game, sets the optimal pricing and the task partition strategy by predicting the strategies of the ICDs (followers). The optimization problem for the CSP can be formulated as

$$\max_{\{p_i^{\text{opt}}, w_i^{\text{opt}}|_{i\in\mathcal{I}}\}} U_{\text{CSP}}(\{p_i, w_i|_{i\in\mathcal{I}}\}|W, p_u, \boldsymbol{f})$$

$$s.t. \begin{cases} 0 \leq p_i \leq p_u \\ \sum_{i=1}^{I} w_i = W \end{cases}, \tag{5.2}$$

where $\boldsymbol{f}$ represents the ICDs' (follower) strategies, i.e., the CPU-cycle frequency for processing computation tasks, which is fed back to the CSP (leader).

5.3.2 Utility of ICDs

We take ICD$_i$ as an example, and the others can derive out the same strategy. The utility of ICD$_i$ is defined as the income paid by the CSP minus the cost to execute the workload w_i, given by

$$U_{\text{ICD}_i}(f_i) = p_i w_i - p_e k_i w_i f_i^2, \tag{5.3}$$

where $p_e k_i w_i f_i^2$ denotes the cost of the CPU power consumption of ICD$_i$ for executing the computation tasks. According to the circuit theory[63], the CPU power consumption can be divided into several factors, including the dynamic, short-circuit, and leakage power consumption, where the dynamic power consumption dominates the others. In particular, the dynamic power consumption can be represented as $k_i w_i f_i^2$, where k_i is a constant related to the hardware architecture. In order to unify the unit, we convert the power consumption into the corresponding price as $p_e k_i w_i f_i^2$, where p_e is the unit price of the CPU power consumption.

The ICDs, acting as multiple followers in the Stackelberg game, deduce their optimal CPU-cycle frequency by following the CSP's strategy (leader). The optimization problem for ICD$_i$ can be formulated as

$$\max_{f_i^{\text{opt}}} U_{\text{ICD}_i}(f_i|p_i, w_i), \forall i \in \mathcal{I}$$

$$s.t. \begin{cases} 0 \leq f_i \leq f_i^{\max} \\ k_i w_i f_i^2 \leq E_i^{\max} \end{cases}. \tag{5.4}$$

According to the dynamic pricing p_i and the task partition w_i, each ICD calculates its optimal CPU-cycle frequency f_i^{opt} to maximize its utility. f_i cannot exceed each ICD's limitation of computational capacity $f_i^{\max}$ and the CPU power consumption $E_i^{\max}$.

5.3.3 Computational Latency-based Pricing

As indicated in [92], the latency is an important influencing factor of QoE, especially for real-time applications. When the ICDs are in the proximity of the CSP in local regions, the transmission time can be omitted [93, 94], especially for the computational-intensive tasks. Therefore, we mainly focus on the overall computing response time and propose a computational latency-based pricing mechanism, in which both of the unit price p_u and p_i are varying dynamically according to the computational latency. The user's QoE is usually decreasing with the increase of the response time. To reflect this relationship, we construct the computational latency-based pricing model between the unit price and the response time, as shown in Figure 5.2.

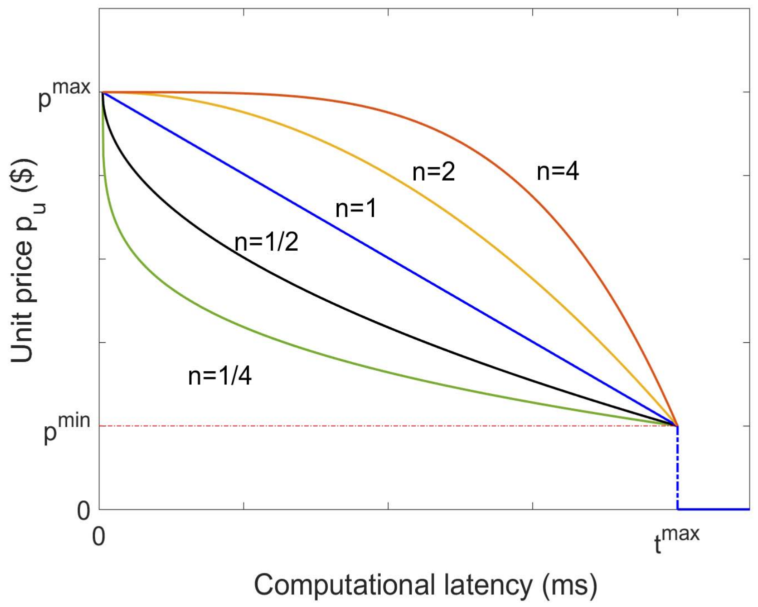


Figure 5.2: Computational latency-based pricing p_u.

When the computational latency exceeds its upper bound $t^{\max}$, the benefit of the CSP will be set as 0. During the time period $[0, t^{\max}]$, the unit price p_u reduces from $p^{\max}$ to $p^{\min}$ with regard to the computing response time t. Without loss of generality, we use the following formulation to model the decreasing relationship between the unit price and the response time, i.e.,

$$p_u = p(t) = \begin{cases} p^{\max} - (p^{\max} - p^{\min})(\frac{t}{t^{\max}})^n & t \in (0, t^{\max}] \\ 0 & t \in (t^{\max}, \infty), \end{cases} \tag{5.5}$$

where $t = \max\{t_i|_{i\in\mathcal{I}}\}$, t_i indicates each ICD$_i$'s computing response time, and n denotes the tendency to change. Specifically, we can divide the changes between p_u and t into three forms: decreasing linearly ($n = 1$), slow-fast ($n > 1$) and fast-slow ($n < 1$).

As an intermediary, the CSP needs to guarantee its own payoff. In this chapter, we consider that the CSP receives commissions proportionally from the fees paid by the end-user, as illustrated in Figure 5.3.

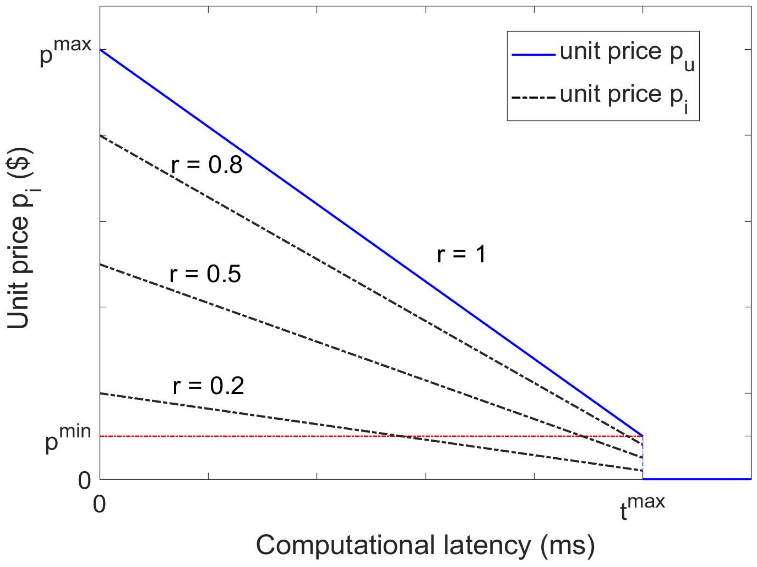


Figure 5.3: Computational latency-based pricing p_i.

We take the linear relationship as an example, i.e., $n = 1$. The rate of the commission is set as r, and thus the unit price p_i paid to the ICDs can be expressed as

$$p_i = rp(t_i)$$

$$s.t. \quad r \leq 1. \tag{5.6}$$

Since all participants in the game are rational, the unit price p_i offered by the CSP would not exceed the end-user's payment p_u, i.e., $r \leq 1$.

As shown in Figure 5.2, reducing the overall computational latency can enhance the QoE performance so that the end-user is willing to pay a higher unit price p_u. In order to obtain better payoff, the CSP inspires the ICDs to speed up their processing rates by changing p_i with the ICDs' execution time, as shown in Figure 5.3. Increasing r will encourage the ICDs to provide a higher data proceeding rate, which helps to reduce the overall computational latency; however, this increase leads to a decrease in the revenue of the CSP. Therefore, given the unit price p_u from the end-user, the CSP needs to determine an optimal commission rate r with the goal of maximizing its utility.

5.4 Game Model Analysis

We model the interactions between the CSP (leader) and the ICDs (followers) as the Stackelberg game which contains two stages. In each stage, the players determine the strategies to maximize their utilities.

- *Stage I*: Players: CSP (leader); Strategy: commission rate r, task partition $\{w_i|_{i\in\mathcal{I}}\}$; Utility: $U_{\text{CSP}}(r, \{w_i|_{i\in\mathcal{I}}\})$ given in (5.1).
- *Stage II*: Players: ICDs (followers); Strategy: CPU-cycle frequency f_i; Utility: $U_{\text{ICD}_i}(f_i)$ given in (5.3).

Our goal is to determine SPNE of the Stackelberg game, where neither the CSP or the ICDs have incentives to deviate unilaterally. We will use backward induction to obtain the corresponding SPNE.

5.4.1 Stage II: Optimal Strategy of ICDs

In Stage II, following the leader's strategy, each ICD can find out the optimal CPU-cycle frequency f_i^{opt} in order to maximize its utility.

Theorem 5.4.1 *Given the workload w_i and the unit price p_i, the optimal CPU-cycle frequency*

f_i^{opt} can be expressed as

$$f_i^{\mathrm{opt}} = \min\{\max\{\frac{w_i}{t^{\max}}, f_i^*\}, f_i^{\max}, f_i^{\mathrm{E_{max}}}\}, \tag{5.7}$$

where $w_i/t^{\max}$ denotes the computational latency constraint. f_i^* is the optimal value to maximize U_{ICD_i}, i.e.,

$$f_i^* = \sqrt[n+2]{\frac{nr(p^{\max} - p^{\min})w_i^n}{2p_e(t^{\max})^n k_i}}. \tag{5.8}$$

$f_i^{\max}$ and $f_i^{\mathrm{E_{max}}}$ are the limitations of the ICD$_i$*'s computational capacity from the perspective of the frequency and the power consumption, respectively. $f_i^{\mathrm{E_{max}}}$ can be calculated by*

$$f_i^{\mathrm{E_{max}}} = \sqrt{\frac{E_i^{\max}}{k_i w_i}}, \tag{5.9}$$

where $E_i^{\max}$ indicates the maximum CPU power consumption afforded by ICD$_i$.

Proof We substitute (5.5), (5.6) into (5.3); thus, the utility of ICD$_i$ can be expressed as

$$U_{\mathrm{ICD}_i}(f_i) = \begin{cases} rp^{\max}w_i - r\frac{p^{\max} - p^{\min}}{(t^{\max})^n} w_i^{n+1} f_i^{-n} - p_e k_i w_i f_i^2 & \frac{w_i}{f_i} \in (0, t^{\max}] \\ 0 & \frac{w_i}{f_i} \in (t^{\max}, \infty) \end{cases}. \tag{5.10}$$

Since U_{ICD_i} equals to 0 when $f_i < w_i/t^{\max}$, $w_i/t^{\max}$ is the lower bound of f_i^{opt}. In the interval of $[w_i/t^{\max}, \infty]$, U_{ICD_i} is a continuous quadratic function of f_i, and the second derivative of U_{ICD_i} with respect to f_i is $\frac{\partial^2 U_{\mathrm{ICD}_i}}{\partial f_i^2} = -rn(n+1)(p^{\max} - p^{\min})w_i^{n+1} \big/ (t^{\max})^n f_i^{n+2} - 2p_e k_i w_i$. As $\frac{\partial^2 U_{\mathrm{ICD_i}}}{\partial f_i^2} < 0$, U_{ICD_i} is a concave function of f_i. Consequently, we can obtain the optimal CPU-cycle frequency of ICD$_i$ by setting the first derivative of U_{ICD_i} with respect to f_i equal to zero, as expressed in (5.8).

In addition, f_i^{opt} is constrained by the computational capacity, i.e., $f_i^{\mathrm{opt}} \leq \min\{f_i^{\max}, f_i^{\mathrm{E_{max}}}\}$, where $f_i^{\mathrm{E_{max}}}$ can be derived out by $k_i w_i f_i^2 \leq E_i^{\max}$.

Thus, we can obtain the lower bound of the optimal CPU-cycle frequency as $w_i/t^{\max}$ and the upper bound as $\min\{f_i^{\max}, f_i^{\mathrm{E_{max}}}\}$. We need to consider four cases:

- Case 1: If the lower bound $w_i/t^{\max}$ is larger than the upper bound $\min\{f_i^{\max}, f_i^{\mathrm{E_{max}}}\}$, there will be no optimal solution, and thus ICD$_i$ will not take part in the collaborative computing.

- Case 2: If f_i^* given in (5.8) is in the interval of the lower bound and the upper bound, f_i^{opt} should be set as f_i^*.
- Case 3: If f_i^* given in (5.8) is smaller than the lower bound $w_i/t^{\max}$, U_{ICD_i} will be a monotonically decreasing function of f_i. That is because $\frac{\partial U_{\text{ICD}_i}}{\partial f_i}$ is negative when $f_i \in [w_i/t^{\max}, \min\{f_i^{\max}, f_i^{\text{E}_{\max}}\}]$. Thus, to maximize ICD$_i$'s utility, f_i^{opt} should be set as the lower bound $w_i/t^{\max}$.
- Case 4: If f_i^* given in (5.8) is larger than the upper bound $\min\{f_i^{\max}, f_i^{\text{E}_{\max}}\}$, U_{ICD_i} will be a monotonically increasing function of f_i. That is because $\frac{\partial U_{\text{ICD}_i}}{\partial f_i}$ is positive when $f_i \in [w_i/t^{\max}, \min\{f_i^{\max}, f_i^{\text{E}_{\max}}\}]$. Thus, to maximize ICD$_i$'s utility, f_i^{opt} should be set as the upper bound $\min\{f_i^{\max}, f_i^{\text{E}_{\max}}\}$.

Therefore, given the unit price p_i and the task partition w_i, we can obtain the optimal CPU-cycle frequency f_i^{opt} by (5.7). This finishes the proof of Theorem 5.4.1.

5.4.2 Stage I: Optimal Strategy of CSP

Now we analyze the CSP's strategy in Stage I. The CSP, as a leader, determines the optimal commission rate r and the optimal task partition set $\{w_i^{opt}|_{i\in\mathcal{I}}\}$ simultaneously to maximize its utility.

5.4.2.1 Optimal Task Partition

First, we analyze the optimization problem of the task partition for the CSP.

Theorem 5.4.2 *By considering the optimal strategies of the ICDs $\{f_i^{\text{opt}}|_{i\in\mathcal{I}}\}$, the CSP can determine the optimal task partition set $\{w_i^{\text{opt}}|_{i\in\mathcal{I}}\}$ as*

$$w_i^{\text{opt}} = W\frac{f_i^{\text{opt}}}{\sum_i^I f_i^{\text{opt}}}, \forall i \in \mathcal{I}, \tag{5.11}$$

and we can get the following representation of the optimal task partition set $\{w_i^{\text{init}}|_{i\in\mathcal{I}}\}$ as the initial values when we do not consider the limitations of f_i^{opt}, i.e.,

$$w_i^{\text{init}} = W\frac{\sqrt{k_i^{-1}}}{K}, \forall i \in \mathcal{I}, \tag{5.12}$$

where $K = \sum_{i=1}^{I}\sqrt{k_i^{-1}}$.

Proof We substitute (5.5), (5.6) into (5.1); thus, the utility of the CSP can be expressed as

$$U_{\text{CSP}}(\{w_i|_{i\in\mathcal{I}}\}) = p(t)W - r\sum_{i=1}^{I} p(t_i)w_i, \tag{5.13}$$

where $t = \max\{t_i|_{i\in\mathcal{I}}\}$.

Since the unit price $p(t)$ is a decreasing function regarding to the computing response time t, i.e., $p(t)$ is not more than $p(t_i)$ as $t = \max\{t_i|_{i\in\mathcal{I}}\} \geq t_i$. Then, by assuming the commission rate r is fixed, we can obtain the derivation as

$$p(t)W - r\sum_{i=1}^{I} p(t_i)w_i \leq p(t)W - rp(t)\sum_{i=1}^{I} w_i = (1-r)p(t)W \tag{5.14}$$

The condition for maximizing (5.14) exists only when the execution time is equal among all ICDs, i.e.,

$$t = \frac{w_i}{f_i} = \frac{w_j}{f_j}, \forall i, j \in \mathcal{I}. \tag{5.15}$$

In this way, the computation task W should be partitioned according to the ratio of f_i^{opt} as (5.11). An ICD with higher CPU-cycle frequency will be assigned with more workload to reduce the overall computational latency, and thereby the CSP can receive a higher reward.

Substituting (5.8) into (5.15), we can get the expression as

$$w_i^2 k_i = w_j^2 k_j, \forall i, j \in \mathcal{I}. \tag{5.16}$$

Since $\sum_{i=1}^{I} w_i = W$, we can obtain the closed-form solution of the initial task partition set $\{w_i^{\text{init}}|_{i\in\mathcal{I}}\}$ as expressed in (5.12). This finishes the proof of Theorem 5.4.2.

5.4.2.2 Optimal Commission Rate

Next, we analyze the optimization problem of the commission rate for the CSP.

Theorem 5.4.3 *Given the computation task T, the CSP can determine the optimal commission rate r^{opt} as*

$$r^{\text{opt}} = \min\{\max\{r^{\min}, r^*\}, 1\}, \tag{5.17}$$

where r^* is the optimal value to maximize the CSP's utility which satisfies

$$\frac{2}{n+2} r^{-\frac{n}{n+2}} + \frac{n}{n+2} r^{-\frac{2n+2}{n+2}} = A, \tag{5.18}$$

where $A = p^{\max} \Big/ \sqrt[n+2]{(p^{\max} - p^{\min})^2 [\frac{2p_e W^2}{nK^2 (t^{\max})^2}]^n}$. $r^{\min}$ is the lower bound indicates the minimum commission rate corresponding to the overall computational latency limitation, given by

$$r^{\min} = \frac{2p_e W^2}{nK^2 (p^{\max} - p^{\min})(t^{\max})^2}. \tag{5.19}$$

Proof We substitute (5.8), (5.11) into (5.15); thus, the overall computational latency can be derived out as

$$t = \sqrt[n+2]{\frac{2p_e (t^{\max})^n W^2}{nr(p^{\max} - p^{\min})K^2}}. \tag{5.20}$$

Since the overall computational latency t is not more than its upper bound $t^{\max}$, we can obtain the lower bound of the commission rate as (5.19).

Next, we substitute (5.20) into (5.14); thus, the utility of CSP can be expressed as

$$U_{\text{CSP}}(r) = \begin{cases} W(1-r)[p^{\max} - \sqrt[n+2]{(p^{\max} - p^{\min})^2 (\frac{2p_e W^2}{nrK^2 (t^{\max})^2})^n}] & r \in [r^{\min}, 1] \\ 0 & \text{otherwise} \end{cases}. \tag{5.21}$$

From (5.21), U_{CSP} is a continuous quadratic function of r in the interval of $[r^{\min}, 1]$. Since the second derivative $\frac{\partial^2 U_{\text{CSP}}}{\partial r^2} = -nWp^{\max}(2r^{\frac{-2n-2}{n+2}} + r^{\frac{-3n-4}{n+2}}) \big/ A(n+2)^2 < 0$, U_{CSP} is a concave function of r. Consequently, we can obtain the optimal commission rate r^* by setting the first derivative of U_{CSP} with respect to r equal to zero, as expressed in (5.18). Then, we can get the optimal unit price p_i^{opt} by substituting the commission rate r^{opt} into (5.6). This finishes the proof of Theorem 5.4.3.

The optimal task partition set $\{w_i^{opt}|_{i\in\mathcal{I}}\}$ and the optimal unit price p_i^{opt} are substituted back to (5.7) to get the final optimal CPU-cycle frequency f_i^{opt} at the end of the Stackelberg game iteration.

To consider the limitation of CPU-cycle frequency, the optimal f_i^{opt} should satisfy the constraints indicated in (5.4), i.e., f_i cannot exceed each ICD's limitation of computational capacity $f_i^{\max}$ and the CPU power consumption $E_i^{\max}$. In addition, f_i should satisfy the latency requirement, i.e., $f_i \geq w_i/t^{\max}$. Otherwise, it cannot obtain any benefit under the computational latency-based pricing. If any ICD exceeds its CPU-cycle frequency limitation, we cannot substitute (5.8) into (5.15) and get the closed-form solution (5.16) directly. To tackle this problem, we design a near-optimal algorithm for task scheduling in the following subsection.

5.4.3 Near-optimal Algorithm

By considering the CPU-cycle frequency constraints in practical scenarios, we develop a near-optimal algorithm for task scheduling in collaborative computing. Next, we provide the near-optimal algorithm in detail.

Upon the arrival of the computation task T, i.e., the computation workload W and the unit price p_u that the end-user would like to pay for this computing task, the CSP, acting as a single leader, can estimate the optimal commission rate r^{opt} and the optimal unit price p_i^{opt} by (5.17) and (5.6), respectively. Then, it obtains the initial task partition set $\{w_i^{\mathrm{init}}|_{i\in\mathcal{I}}\}$ by (5.12) for each ICD with the goal of maximizing its payoff. Based on the unit price p_i and the workload w_i, the ICDs, acting as multiple followers, can calculate their corresponding CPU-cycle frequency $\{f_i^*|_{i\in\mathcal{I}}\}$ by (5.8) to achieve their maximum utilities. If any ICD reaches its limitation as indicated in (5.7), f_i^{opt} is set as the lower/upper bound, and then the optimal task partition and the optimal CPU-cycle frequency can be recalculated by step 11 and step 12. To minimize the overall computational latency and maximize the utility of the CSP, the computational latency difference between each ICD should be close to zero, i.e., no more than ϵ. The iterations of (5.11) and (5.8) help the game solution converge to a near-optimal

Algorithm 3 A near-optimal algorithm for task scheduling in collaborative computing

```
Input: $W$, $p_u$, $p_e$, $\{k_i|_{i\in\mathcal{I}}\}$, $\{f_i^{\max}|_{i\in\mathcal{I}}\}$, $\{E_i^{\max}|_{i\in\mathcal{I}}\}$, $\epsilon$
Output: $p_i^{\text{opt}}$, $\{w_i^{\text{opt}}|_{i\in\mathcal{I}}\}$, $\{f_i^{\text{opt}}|_{i\in\mathcal{I}}\}$
 1: Initially, end-user sends the computing task to CSP, i.e., $W$ and $p_u$; each participant ICD
    reports $k_i$ to CSP.
 2: CSP set $r^{\text{opt}}$ by (5.17) and broadcast $p_i^{\text{opt}}$ by (5.6) to ICDs.
 3: CSP calculates $\{w_i^{\text{init}}|_{i\in\mathcal{I}}\}$ by (5.12), and each ICD gets $f_i^*$ by (5.8).
 4: repeat
 5:     if $f_i^* < \frac{w_i}{t^{\max}}$ then
 6:         $f_i^{\text{opt}} \leftarrow \frac{w_i}{t^{\max}}$
 7:     else if $f_i^* > \min\{f_i^{\max}, f_i^{\text{E}_{\max}}\}$ then
 8:         $f_i^{\text{opt}} \leftarrow \min\{f_i^{\max}, f_i^{\text{E}_{\max}}\}$
 9:     else
10:         $f_i^{\text{opt}} \leftarrow f_i^*$
11:     calculate $\{w_i^{\text{opt}}|_{i\in\mathcal{I}}\}$ by (5.11)
12:     calculate $\{f_i^*|_{i\in\mathcal{I}}\}$ by (5.8)
13: until $|w_i^{\text{opt}}/f_i^{\text{opt}} - w_j^{\text{opt}}/f_j^{\text{opt}}| \le \epsilon, \forall i, j \in \mathcal{I}$
```

equilibrium, which will be shown in Figure 5.10 in Section 5. Finally, all participants achieve SPNE, where no one will alter its strategy if the strategies of the others remain unchanged.

5.5 Simulation Results

In this section, we present numerical results to evaluate the proposed collaborative computing approach in MATLAB from different aspects, including the computational latency performance and the utility of the CSP and the ICDs. We further compare our proposed algorithm with the existing algorithms proposed in [83] and [84], respectively. Parameter settings are given first.

5.5.1 Parameter Settings

The simulation parameters are set as follows unless specified otherwise. In the simulation, the CSP owns 20 ICDs that would like to offer idle computing resources for collaborative computing. We assume that the energy unit price is set as $p_e = 0.039$ \$ [91] and the hardware-related coefficient k_i randomly distributed from 1 to 16. We simulate the overall workload W varies from 50 Megacycles to 1,500 Megacycles, and the latency limitation $t^{\max}$ increases from 50ms to 100ms correspondingly. For ease of analysis, $p^{\min}$ is set as 0 \$ and $p^{\max}$ is fixed as 2

\$. The maximum value of CPU-cycle frequency and CPU power consumption of ICDs are set as same value $f_i^{\max} = 2.5$ GHz[95].

5.5.2 Numerical Results

We first simulate the impact of the tendency to change n in Equation (5.5) on the overall computational latency in Figure 5.4.

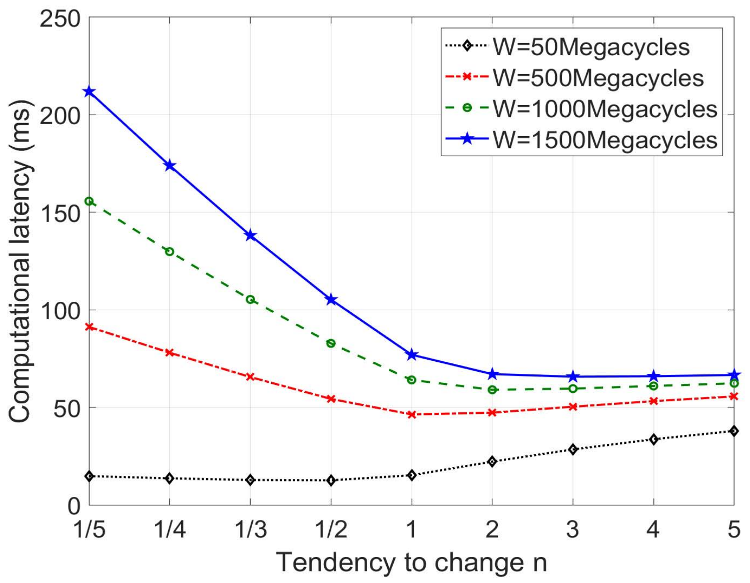
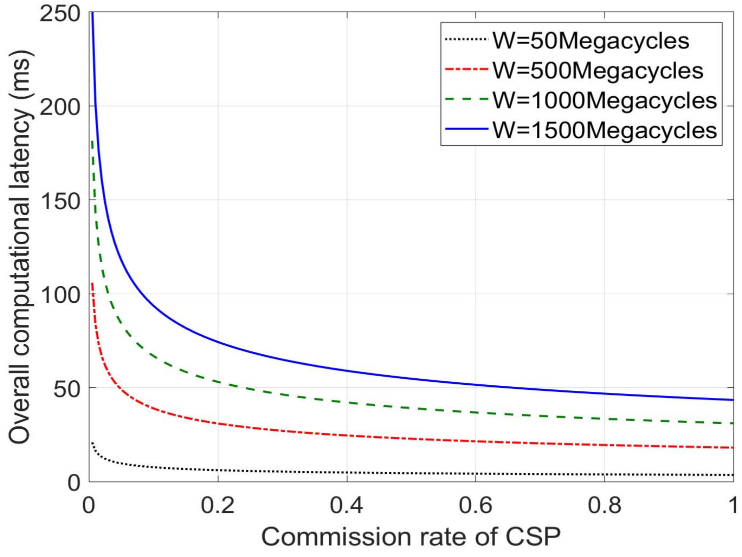


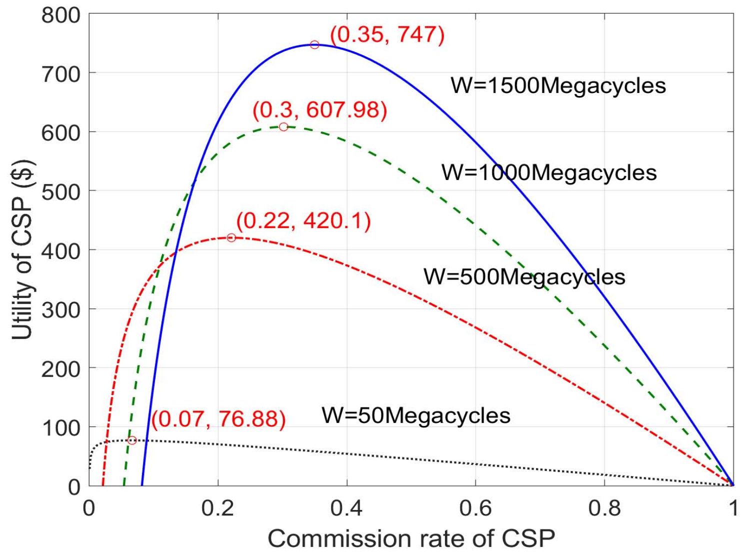
Figure 5.4: Overall computational latency versus tendency to change n.

When n is more than 1, the overall computational latency is generally less than the cases of $n < 1$. That is because the end-user offers a larger unit price p_u with the slow-fast change ($n > 1$) than the fast-slow change ($n < 1$), as shown in Figure 5.2. Thus, the end-user can choose the proper tendency of change to balance its cost and latency requirement. Furthermore, by comparing different workload curves in Figure 5.4, we can observe that the computational time decreases mainly as n increases from 0 to 1; however, it varies slightly as n increases from 1 to larger values. In other words, we can gain a good latency performance when the tendency to change is equal to 1, i.e., the linear relationship between the unit price and the computational latency. Hence, we will set $n = 1$ in the following performance simulations of the CSP and the ICDs.

In Figure 5.5(a), we plot the overall computational latency against the commission rate for



(a)



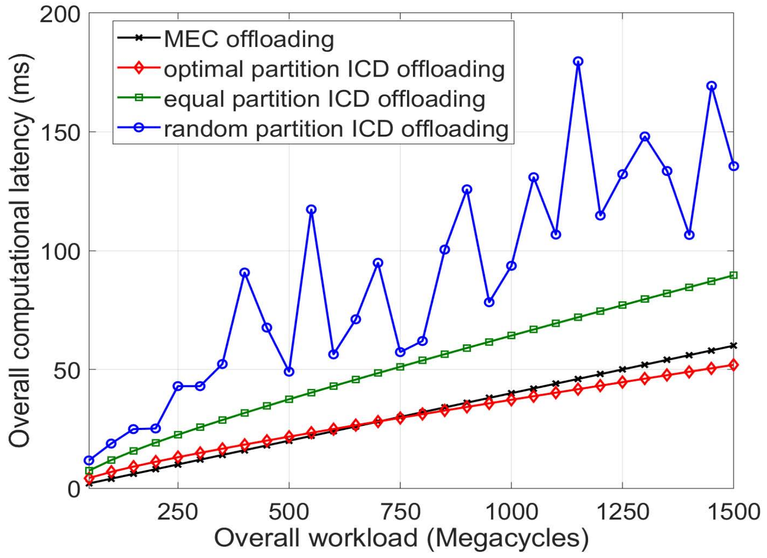
(b)

Figure 5.5: Overall latency and utility of CSP versus commission rate r.

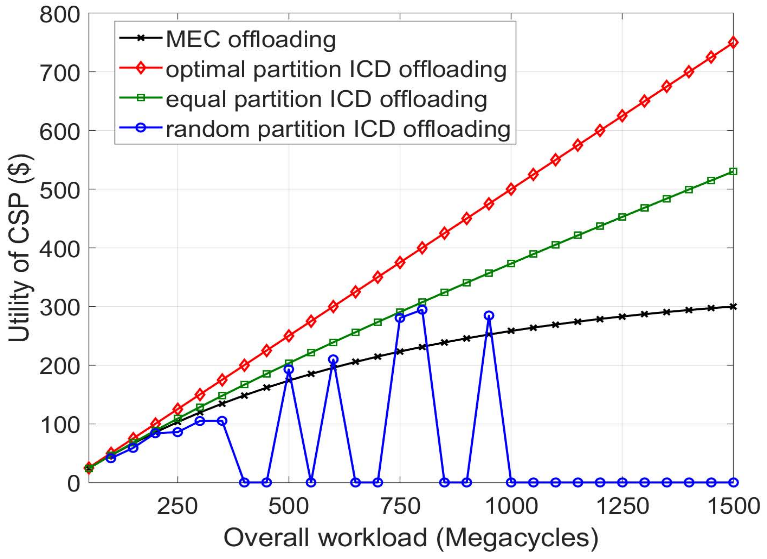
the various workload. The general tendency is the larger the commission rate is, the less the computational latency will be. It implies that the higher unit price paid by the CSP to the ICDs can achieve better latency performance. Figure 5.5(b) illustrates how to choose the optimal commission rate under the different workload. We can see that the utility of the CSP is a concave function versus the commission rate. As shown in Figure 5.5(a), a higher commission rate leads to lower computational latency, and thus the CSP gains higher payoff from the end-user; however, a higher commission rate also increases its cost paid to the ICDs. Therefore, the CSP should choose the optimal commission rate to maximize its payoff.

As shown in Figure 5.6, we compare the overall execution latency and the utility of the CSP of our proposed approach with the three different task partition mechanisms under various workload, i.e., (a) MEC offloading: the tasks are executed on a powerful MEC server as a whole; (b) equal partition ICD offloading: partitioning tasks equally by ICD offloading. i.e., W/K; (c) random partition ICD offloading: partitioning tasks randomly by ICD offloading. Here, the CPU-cycle frequency of the MEC server is set as 25 GHz. When the total amount of workload W arises, as shown in subgraph (a), the response time represents an increasing trend for all cases. Compared with the other two ICD offloading mechanisms, our proposed approach has a notable improvement, where the latency decreases to that of the half of equal task partition and that of one-third of random task partition. When the workload is less than 750 Megacycles, offloading to MEC can obtain lower computational latency. Nevertheless, the end-user can get better latency performance when the workload increases to 750 Megacycles or above, and it achieves by 13.5 percent improvement when the workload is 1,500 Megacycles. Thus, it is recommended to offload a vast amount of computation-intensive and latency-sensitive tasks by our proposed approach. From subgraph (b), we can obtain that the CSP's utility of our proposed approach is consistently the maximum compared with that of others, which is improved by 41.5 percent for the CSP compared with MEC offloading.

Figure 5.7 illustrates the latency and the utility of a certain ICD by comparing our proposed approach with the fixed CPU-cycle frequency mechanism. As shown in subgraph (a), the optimal CPU-cycle frequency mechanism can get better performance when the computing workload exceeds 75 Megacycles, and the latency decreases by 17.4 percent when the workload is 135 Megacycle. While as shown in subgraph (b), the utility of the ICD of our proposed

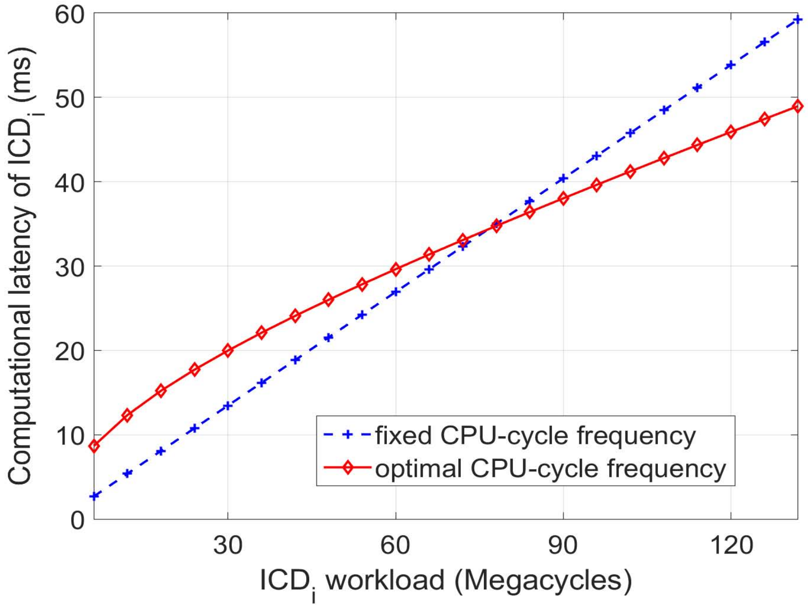


(a)

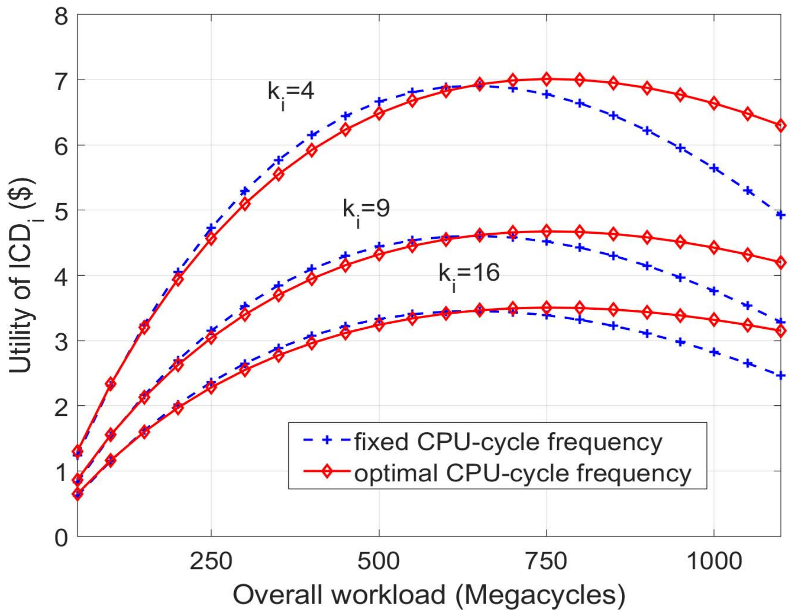


(b)

Figure 5.6: Overall latency and utility of CSP versus workload W.



(a)



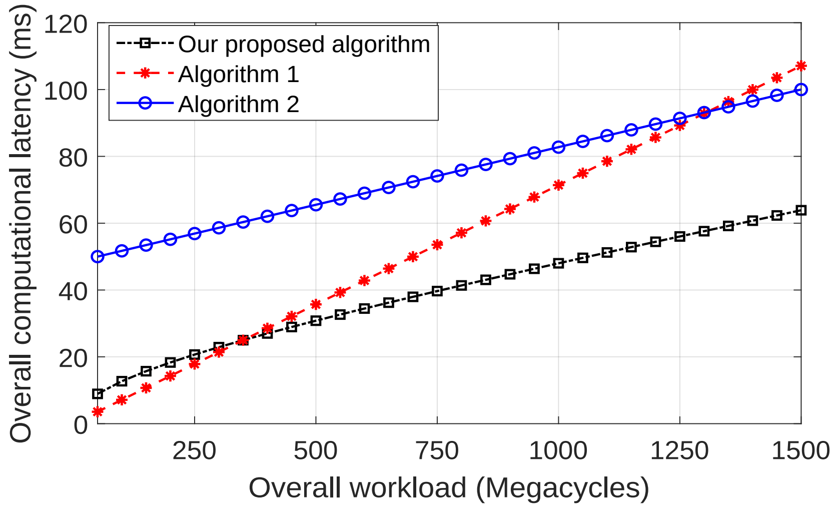
(b)

Figure 5.7: Latency and Utility of ICD$_i$ versus workload W.

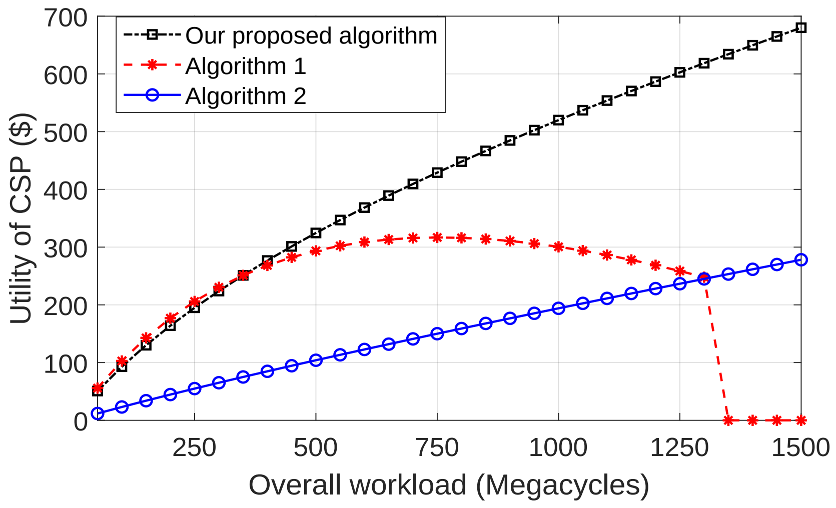
approach is slightly less than the fixed approach when the overall computing workload is lower than 600 Megacycles. When the workload rises from 600 Megacycles, this utility is significantly superior to the fix approach, and it can be improved by 27.9 percent. This is because the optimal f_i^{opt} comprehensively considers the tradeoff between latency and power consumption.

Furthermore, we compare the performance of the proposed computation sharing algorithm in Figure 5.8 with two algorithms in terms of the computing latency and all participants' utilities: Algorithm 1 [83]: fixed CPU-cycle frequency, and Algorithm 2 [84]: elastic CPU-cycle frequency. Both of these two algorithms do not consider the QoE in pricing. The computing latency is defined as the maximum value of each ICD's data processing time, i.e., $max\{q_k/f_k\}$. Motivated by the latency-based pricing, we can find that our proposed algorithm can significantly reduce the computing latency with the increasing of the computational workload and achieves by 40.4 and 36.1 percent improvement compared with Algorithm 1 and Algorithm 2, respectively. This is because Algorithm 1 adopts the fixed CPU-cycle frequency, and the computational latency grows linearly with the increase of the computing workload. Without a latency-based pricing incentive, Algorithm 2 adjusts its CPU-cycle frequency to a minimum value satisfying the latency requirement, which minimizes the energy consumption but leads to high processing time. Next, we compare the proposed computation sharing algorithm with two algorithms in terms of the utilities of the CSP (in subgraph (b)) and the ICDs (in subgraph (c)). We can find that all participants obtain the maximum profit in collaborative computing by our proposed algorithm, especially for the computational-intensive tasks. Inspired by our proposed latency-based pricing, the ICDs are willing to improve its processing rate, which not only enhances the overall computational latency performance but also achieves better benefits for all participants.

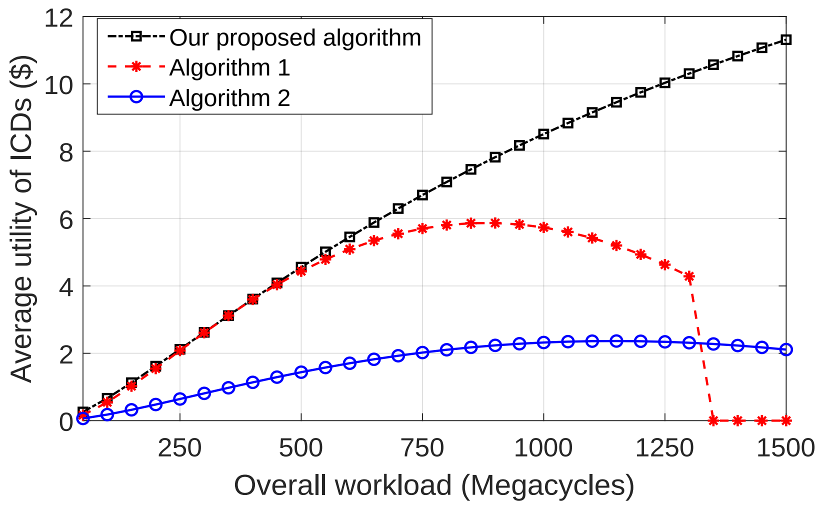
By considering the limitation of f_i, we simulate the convergence iterations with our proposed near-optimal algorithm in Figure 5.9 and Figure 5.10. The upper limit of time difference ϵ is set as 0.0001 ms. From Figure 5.9, the response time converges to a value larger than that when the CPU-cycle frequency of each ICD is not limited. That is due to the reduction of the whole computational capacity brought by the limitation of f_i. Figure 5.10 further reveals how the CSP and the ICDs achieve SPNE by our proposed near-optimal algorithm. In each iteration, the CSP updates its computing task partition based on the previous feedback of the ICDs,



(a)



(b)



(c)

Figure 5.8: Algorithm comparison versus workload W.

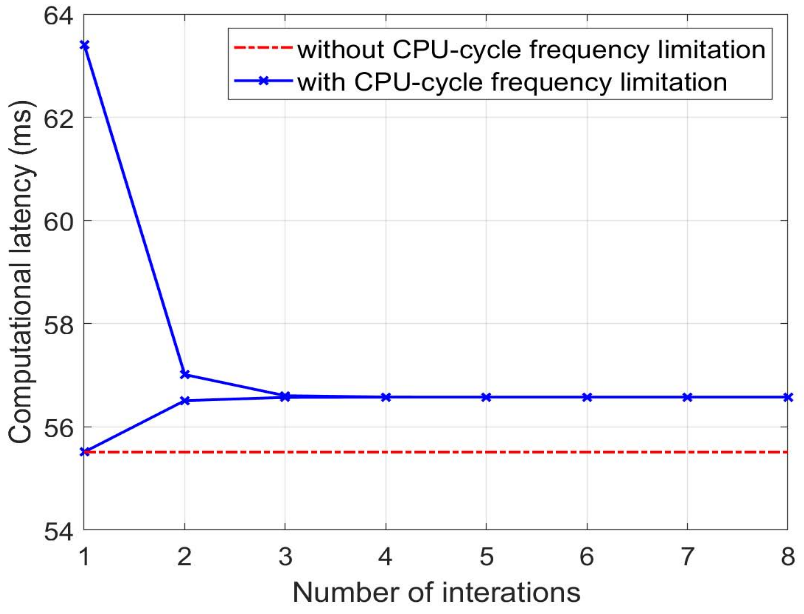


Figure 5.9: Overall computational latency convergence iterations.

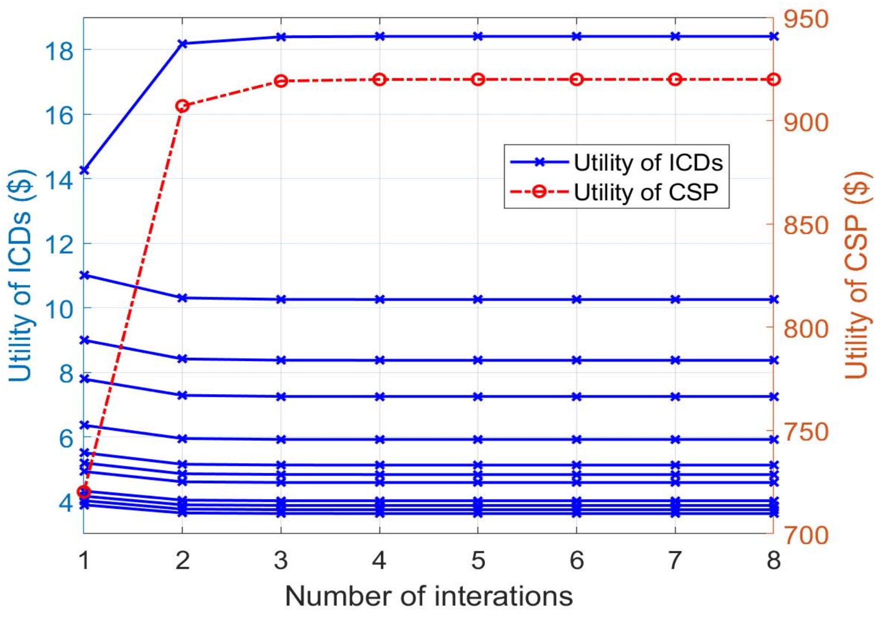


Figure 5.10: Utilities of CSP and ICDs convergence iterations.

and each ICD adjusts its best strategy accordingly until the computational latency of each ICD converges to an equal value. Then, no one will alter its strategy if the strategies of the others remain unchanged.

5.6 Chapter Summary

This chapter proposed a computational latency-based resource scheduling mechanism to enable collaborative computing amongst IoT devices from the perspective of QoE performance. Specifically, we modeled a computational latency-based pricing scheme and utilized the Stackelberg game to analyze the interactions between the CSP and the ICDs, in which the CSP first determined the optimal unit price p_i^{opt} and the optimal task partition strategy $\{w_i^{\text{opt}}|_{i\in\mathcal{I}}\}$ according to the computation tasks from the end-user; then, the ICDs derived out the optimal CPU-cycle frequency f_i^{opt} correspondingly. According to the simulation results, the overall computational latency was significantly decreased for the computational-intensive tasks compared with the existing schemes. All participants obtained the maximum profit by our proposed game-theoretic task partition mechanism.

Chapter 6

Joint Communication and Computation Resource Scheduling for QoE Enhancement

According to resource scheduling methods proposed in the previous three chapters, IoT devices can realize horizontal collaboration through effective incentive mechanisms. This chapter will further explore the vertical collaborative computing between the edge level and the device layer.

As the core of three-tier edge-assisted IoT systems, edge servers have more powerful computing capabilities than the device layer. Utilizing the resources in the edge layer can effectively enhance IoT devices' QoE performance. However, due to the concurrent dynamics of application requirements, available resources, and network conditions, meeting the increasingly diverse requirements of IoT applications remains an ultimate challenge to effective resource scheduling in this vertical collaboration. Existing studies have mainly explored the resource scheduling problem with a specified QoS as an optimization objective, which may lose effectiveness when dealing with the diverse requirements across heterogeneous IoT devices. Towards this end, we focus on enhancing IoT device-specific QoE performance through jointly optimizing communication and computation resources in this chapter. First, a three-layer QoE assessment model is constructed to describe the general relationship between resource provisioning and device-specific QoE performance. Then, to maximize the overall QoE performance amongst IoT devices, a two-stage resource scheduling scheme is proposed to realize simultaneous optimization of IoT devices and the edge system. Specifically, in stage I, a distributed resource scheduling algorithm with low complexity is designed for each IoT device to optimize the local computing rate by considering its resource-constrained nature; in stage II, a PPO-based online approach is proposed on the edge system to schedule communication bandwidth and optimize computational rate by interacting with multiply IoT devices without prior knowledge of their specific QoE assessment models. Finally, extensive experiments demonstrate that our proposal outperforms the existing works from the perspective of QoE performance.

6.1 Introduction

With the rapid development of wireless communication technologies, further assisted by edge computing, the IoT applications have experienced phenomenal growth throughout the last decades, ranging from health care to smart cities and industrial automation [73]. Within these edge-assisted IoT systems, an IoT device can take advantage of high bandwidth in local communication networks and nearby resource-rich edge servers to offload computation, thus enhancing the performance of IoT applications deployed on them [66, 96].

However, with the ever-increasing diversity of applications, different IoT devices within the same edge network inevitably incur specific resource demands to achieve their diverse application requirements, leading to the ultimate challenge of resource scheduling in the edge-assisted IoT system. The core challenge lies in how to optimize communication and computation resources collaboratively in such a dynamic system for device-specific task handling, which could be affected by many factors, specifically including diverse requirements of IoT applications, dynamics of local network conditions, and edge server resource utilization [9]. Given these aggregated impacts, effective joint communication and computation resource scheduling to satisfy device-specific demands has become the key indicator of edge-assisted IoT systems.

Existing studies on resource scheduling in edge computing have mainly utilized a specified QoS parameter as optimization objective, such as energy efficiency [97, 98], service latency minimization [99, 100], cost efficiency [101], revenue maximization [102], etc. As an objective measure, QoS is an effective indicator for evaluating the overall network performance.

However, with increasingly diverse requirements from IoT applications, the acceptability of the same QoS may be significantly distinct across heterogeneous IoT devices, which directly reduces the effectiveness of QoS-based approaches when dealing with IoT device-specific demands.

Consequently, the objective of resource scheduling in edge networks has gradually changed to improve QoE rather than QoS. As defined by the International Telecommunication Union, QoE generally represents "The degree of delight or annoyance of the user of an application or service" [21]. That said, QoE is not determined by network metrics alone but also by the acceptability of IoT devices to these metrics, which makes it an effective indicator for evaluating performance from the perspective of IoT devices. Therefore, establishing a suitable QoE assessment model becomes a key point for resource scheduling in IoT systems. Currently, several QoE-based resource scheduling have been proposed given the particular application scenarios, which assess the IoT devices' performance through relating QoS to QoE by the predefined functions, e.g., exponential function [103], linear function [104, 105, 106], or other scenario-tailored functions [107, 108]. However, with more diverse applications enabled by edge-assisted IoT systems, new QoE assessment models that can manifest greater applicability in general cases need further extensive investigations.

Furthermore, realizing effective resource scheduling in edge-assisted IoT systems is not trivial. It requires joint optimization of communication and computation resources dynamically available at local IoT devices and edge systems, which is a complicated problem affected by many factors. Moreover, obtaining prior knowledge of the system dynamics, e.g., local network conditions or application characteristics, is often difficult in practice. To solve such a problem, extensive studies have been conducted from the perspective of intelligent operations, mainly through DRL [109, 110]. Different from conventional optimization approaches, the DRL-based methods can learn the best application task offloading policy and optimize the resource allocation strategies in a trial-and-error manner by interacting with the unknown environment to achieve a long-term optimization [111, 112, 113, 114, 115, 116]. Although the DRL-based solutions have great potential in dealing with resource scheduling problems, the involved learning requires high computational resources, and adopting learning on IoT devices is impractical due to their resource-constrained nature. On the other hand, implementing the learning purely on the edge system may lose the optimal behavior of IoT devices. Consequently, the architecture of DRL-based resource scheduling should be carefully designed in edge-assisted IoT systems. Furthermore, instability and slow convergence of the DRL-based algorithms is another critical issue to be addressed during the resource scheduling process.

Motivated by these issues mentioned above, we propose a two-stage resource scheduling scheme to enhance IoT device-specific QoE performance through jointly optimizing communication and computation resources in edge-assisted IoT systems. Specifically, we first construct a three-layer QoE assessment model describing the general relationship between resource provisioning and IoT devices' performance, which can help resource schedulers better understand the fulfillment of different IoT devices' interests. Based on the proposed QoE assessment model, the device-specific QoE enhancement resource scheduling problem is designed as the optimization objective in this chapter. Then, to maximize the overall QoE performance amongst IoT devices, a two-stage resource scheduling scheme is proposed to realize simultaneous optimization of IoT devices and the edge system. In stage I, considering the resource-constrained nature of IoT devices, we develop a distributed resource scheduling algorithm with low complexity, where each IoT device can obtain its optimal local computing processing rate based on its own QoE requirement. If the IoT device cannot achieve its requirement even with the optimal strategy, the application will be offloaded to the edge system for remote processing and further turn to stage II. In stage II, without prior knowledge of QoE assessment models from different IoT devices, a DRL-based online approach is designed on the edge system to jointly optimize the communication and computation resources in a long-term way, where the edge system continuously optimizes its bandwidth allocation and computing processing rate by interacting with multiply IoT devices until the best policy is found. The proposed online approach is based on the PPO algorithm [117], which can help achieve fast convergence and improve learning stability in stage II. The main contributions of this chapter are summarized as follows:

- *General model for QoE assessment*: A three-layer QoE assessment model is constructed to describe the general relationship between resource provisioning and IoT device-specific QoE performance, which can help resource schedulers better understand the fulfillment of different IoT devices' interests.

- *Two-stage resource scheduling scheme*: To maximize the overall QoE performance amongst IoT devices, a two-stage resource scheduling scheme is proposed to realize simultaneous optimization of IoT devices and the edge system. Specifically, in stage I, considering the resource-constrained nature, a distributed resource scheduling algorithm with low complexity is proposed for each IoT device to optimize its local computing processing rate; in stage II, a PPO-based online resource scheduling approach is designed on the edge system to optimize its bandwidth allocation and computing processing rate by interacting with multiply IoT devices without prior knowledge of their specific QoE assessment models.

- *Simulation result*: Extensive experiments are conducted to evaluate the performance of our proposal. The results show that our proposed two-stage resource scheduling scheme outperforms the existing works from the perspective of QoE performance.

The rest of this chapter is organized as follows. Section 6.2 provides a review of related works. In Section 6.3, we describe the system model, especially for the QoE assessment model. Section 6.4 analyzes the device-specific QoE enhancement resource scheduling problem and proposes a two-stage resource scheduling scheme. Finally, we evaluate our proposal's performance in Section 6.5 and summarize the chapter in Section 6.6.

6.2 Related Work

With the rapid proliferation of IoT applications, effective resource scheduling in the edge computing environment has gradually been a widely focused research problem.

So far, existing works have investigated the edge resource scheduling problem with various objectives. For example, to minimize the energy consumption, the authors in [97] and [98] optimized the computation and communication resources allocation by leveraging the Lagrange duality method and stochastic optimization methods, respectively. Wang *et al.* [99] jointly optimized the offloading decision and the computation resource allocation to minimize the average task duration, and Ma *et al.* [100] investigated cooperative service caching and workload scheduling in mobile edge computing, aiming at minimizing the service time. [101] explored the cost-efficient method to match the IoT services to appropriate fog nodes while guaranteeing minimal delay for IoT services and efficient resource utilization on fog nodes. To maximize all participants' utilities, an incomplete information-based two-tier game algorithm was proposed in [102] for edge resource allocation. These works explored the resource scheduling problem with a specified QoS optimization objective, such as energy efficiency [97, 98], service latency minimization [99, 100], cost efficient [101], and revenue maximization [102]. However, QoS is a metric to evaluate the overall network performance instead of IoT device-specific performance, which causes the QoS-based approaches may lose their effectiveness when dealing with diverse IoT device-specific demands [118].

Therefore, improving QoE in edge computing instead of QoS attracts widespread attention. Being perceived as subjective, QoE is not determined by network metrics alone but also by the acceptability of IoT devices to these metrics, which helps service providers understand how to improve their services from the perspective of IoT devices [119, 120]. Currently, several types of research have been conducted on QoE-based resource scheduling. For example, considering the heterogeneous impact of delays on users' QoE, a QoE-aware service-enhancement method from an orthometric perspective was designed in [103]. The authors in [104] proposed a new QoE model to study computation offloading, which is influenced by service latency, energy consumption, and task success rate. An edge-assisted crowd cast framework was explored for the sheer amount of viewing data towards intelligent decisions for personalized QoE demands [105]. He *et al.* [106] proposed a novel QoE model restricted by the energy consumption in task offloading services of edge-enabled Internet of vehicles. The authors in [107] studied the edge resource allocation problem across multiple service requests with the objective of overall QoE maximization. The solution that performs multi-client joint QoE optimization for adaptive video streaming during bottleneck bandwidth sharing was presented in [108]. However, the existing studies mainly evaluated the QoE performance based on a pre-defined function, e.g., logarithmic, exponential, and other scenario-tailored functions. With more diverse applications enabled by edge-assisted IoT systems, new QoE assessment models that can manifest greater applicability in general cases need further extensive investigations.

Furthermore, realizing effective resource scheduling in edge-assisted IoT systems is not trivial. It needs to collaboratively optimize communication and computation resources from

IoT devices and the edge system in a dynamic environment without prior knowledge. To solve such a problem, extensive studies have been conducted based on DRL-based approaches, which are classified into two categories: value-based methods and policy-based methods [34].

Value-based DRL methods adopt DNN to approximate the value function by minimizing the difference between the value network and the real value function. In [111], authors adopted the Q-learning based method and Deep Q-learning method to obtain the optimal policies of computation offloading and resource allocation. A deep Q-learning based online offloading algorithm was designed in [112] to adapt task offloading decisions and wireless resource allocations optimally. The authors in [113] proposed an improved deep Q-network algorithm to minimize the long-term weighted sum of the average completion time and the average number of requested resources. Unlike value-based DRL methods, policy-based DRL methods use DNNs to approximate the parameterized policy directly by the policy gradient. Thus, policy-based DRL methods usually have faster convergence and are more suitable for the large-scale action space, but they always suffer from convergence stability. Zhan *et al.* [114] designed a decentralized algorithm for computation offloading based actor-critic framework. The implement of [115] and [116] were based on PPO algorithm. However, these works still explored the resource scheduling problem with a specified QoS optimization objective. To enhance QoE, [104] and [106] proposed improved algorithms based on DDPG algorithm for optimal local strategies learning. [105] provided a novel scheduling framework that considered viewers' personalized QoE through an advantage actor-critic (A3C) algorithm from the edge side. Although the above solutions have demonstrated great potential in dealing with resource scheduling problems, learning itself requires high computational resources, and the architecture implemented in edge-assisted IoT systems should be carefully designed by considering the resource characteristics. Besides, methods instability and slow convergence brought by DRL-based algorithms is another essential issue to be addressed in the resource scheduling process.

Motivated by these issues mentioned above, we propose a two-stage resource scheduling scheme to enhance IoT device-specific QoE performance through jointly optimizing communication and computation resources in edge-assisted IoT systems. The detailed system model and corresponding analysis are introduced in the following sections.

6.3 System Model and Problem Formulation

This chapter considers a typical application scenario of an edge-assisted IoT system, as shown in Figure 6.1. The edge system consists of one access point (AP) integrated with a multi-access edge Computing (MEC) server and severs N IoT devices denoted as $\mathcal{N} \triangleq \{1, 2, \dots, N\}$. Here, the IoT devices with limited resources act as sources of the computation tasks with different QoE requirements. At the same time, the edge system is characterized as the service provider by a higher computation capability and has access to virtually unlimited energy. Binary offloading is assumed here so that the applications generated by IoT devices can be either processed locally or offloaded to the MEC server through the AP as one single task. When receiving the applications from multiply IoT devices, the edge system can process the application by itself or offload them to the remote cloud server with sufficient computing resources. In this chapter, our analysis concentrates on the interactions between the edge system and multiple served IoT devices.

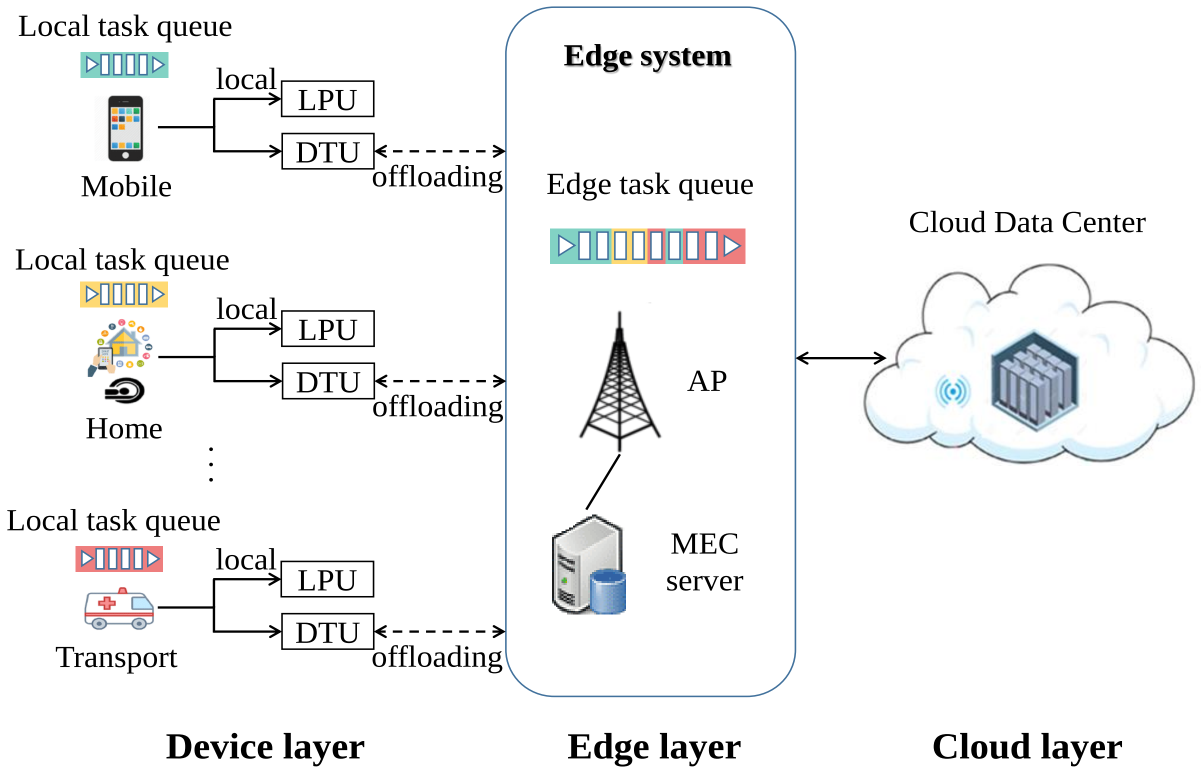


Figure 6.1: Illustration of an edge-assisted IoT system.

The computing architecture of an IoT device consists of a task queue, a local processing unit (LPU), and a data transfer unit (DTU). We assume that applications generated by the same IoT device have identical properties, while applications from various IoT devices have different characteristics. Each IoT device schedules the computation resource based on its available information, including the task queue state, the local computing state, and the QoE requirement, accordingly making the offloading strategy. An application assigned for local computing is processed on the LPU with the optimal processing rate. Otherwise, the application would be offloaded to the edge system for remotely proceeding via the DTU. Afterward, the edge system receives the requests from different IoT devices, executes the task queue by available communication resources from the AP and computation resources from the MEC server, and finally sends the computation result back to IoT devices.

The whole system operates in a slotted fashion: the smallest time interval for resource scheduling is deemed a unit time slot. At the beginning of each time slot, the resource schedulers, including IoT devices and the edge system, monitor their states and then schedule computation and/or communication resources during the entire time slot. The time when the schedulers start to process their task queue is taken as the initial time slot 0, and T indicates the period the schedulers consider for performance evaluation.

The mathematical models, including the task model, the QoE assessment model, and the computing model, are described in detail as follows.

6.3.1 Task Model

We first model the IoT device's workload as a Poisson process with rate λ_n, indicating the expected number of applications arriving in the IoT device's task queue in each time slot. During a time period T, the IoT device n has a computation task queue $C_n = \{C_{n,1}, \dots, C_{n,i}, \dots, C_{n,I_n}\}$ where $\mathbb{E}(I_n) = \lambda_n T$. Each computation task is described as a 3-tuple $C_{n,i}(t^{\mathrm{a}}_{n,i}, w^{\mathrm{comm}}_{n,i}, w^{\mathrm{comp}}_{n,i})$, where $t^{\mathrm{a}}_{n,i}$ is the task's arrival time, $w^{\mathrm{comm}}_{n,i}$ and $w^{\mathrm{comp}}_{n,i}$ indicate the data amount (in byte) for transmission and computation workload (in CPU cycle/s) to be processed, respectively. We consider tasks without stringent execution priority, and they are sorted in the queue by their generated time.

All tasks waiting in the edge task queue are generated by diverse IoT devices with different

QoE requirements. The matrix $\mathbf{Q}_n$ denotes the constraint, the QoE function and the importance of each QoE factor for the IoT device n as illustrated in Table 6.1. Each line stands for one QoE factor, where *Constraint* column indicates the minimum or maximum threshold of a QoE factor, *QoE function* reflects the relationship between the QoE factor to its QoE performance, and $\vec{\alpha}_n = [\alpha_{n,1}, \alpha_{n,2}, \dots, \alpha_{n,K}]$ represents the importance of each QoE factor that satisfies $0 \leq \alpha_{n,k} \leq 1$ and $\sum_{k=1}^{K} \alpha_{n,k} = 1$.

Table 6.1: Illustration of QoE requirement

QoE factor	**Constraint**	**QoE function**	**Importance**
Latency	$T_n^{\max}$	QoE_n^T	α_n^T
Energy	$E_n^{\max}$	QoE_n^E	α_n^E
Reliability	$R_n^{\min}$	QoE_n^R	α_n^R
…	…	…	…

6.3.2 QoE Assessment Model

As introduced in Section I, QoE generally represents the overall acceptability of services subjectively perceived by IoT devices, which can help resource schedulers understand what may be wrong with their services and how to improve them. Therefore, a proper QoE assessment model is exactly anticipated to better optimize the utilization of limited resources to improve the overall IoT devices' satisfaction.

In this section, a generic QoE assessment model is designed to better evaluate IoT devices' satisfaction with resource scheduling in the edge-assisted IoT system, as Figure 6.2 shows.

This model consists of three layers, i.e., resource layer, QoE factor (QoS) layer, and QoE layer. The objective mapping from resources $\vec{re} \in \mathbb{R}^M$ to the QoE factors $\vec{sq} \in \mathbb{R}^K$ can be given by $\vec{sq} = SQ(\vec{re})$, i.e.,

$$\begin{cases} sq_1 = SQ_1(re_1, re_2, ..., re_M) \\ sq_2 = SQ_2(re_1, re_2, ..., re_M) \\ ... \\ sq_K = SQ_K(re_1, re_2, ..., re_M) \end{cases}, \tag{6.1}$$

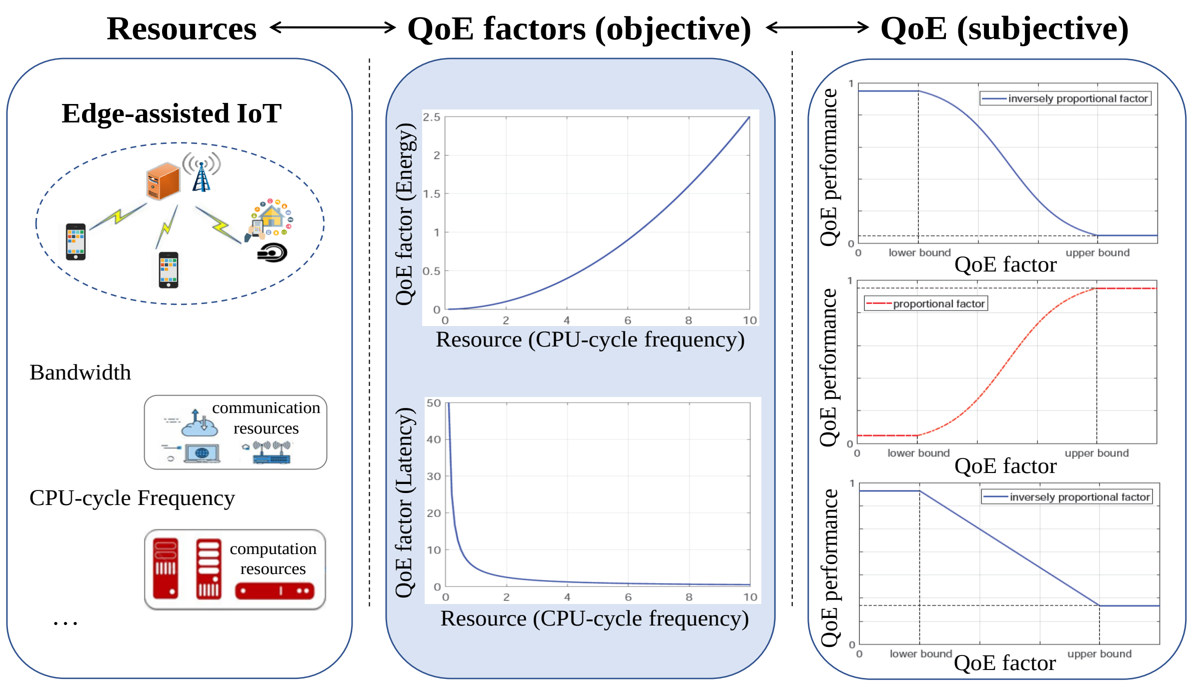


Figure 6.2: Illustration of the three-layer QoE assessment model.

where the resources $\vec{r}$ can refer to communication resources as bandwidth, computation resources as CPU-cycle frequency, etc. The QoE factors $\vec{sq}$ indicate computation latency, energy consumption, reliability, etc. The functions $SQ : \mathbb{R}^M \to \mathbb{R}^K$ describe the relationship between resources and QoE factors. Furthermore, since QoE is the overall acceptability of services by IoT devices, we adopt scalarization to map QoE factors $\vec{sq} \in \mathbb{R}^K$ to a scalar value qoe, which can be expressed as

$$qoe = [\alpha_1, \alpha_2, ..., \alpha_K] \begin{bmatrix} QoE_1(sq_1) \\ QoE_2(sq_2) \\ ... \\ QoE_K(sq_K) \end{bmatrix}, \tag{6.2}$$

where α_k indicates the importance of the QoE factor k. In general, there are two types of QoE factors, i.e., proportional factor (QoE-PF) and inversely proportional factor (QoE-IPF). For QoE-PF, e.g., the reliability, a higher value generally improves the QoE performance. From the lower bound point to the upper bound point, the QoE performance keeps a steady increase along with the factor's improvement, and remains virtually unchanged close to the highest

QoE performance when the factor reaches the upper bound point. Conversely, for QoE-IPF, e.g., the latency, the QoE performance decreases gradually when the factor grows from the lower bound point to the upper bound point. To match with our aforementioned QoE and QoE-factor correlation, we assume that the QoE function is continuously derivable in its scope $QoE_k \in [0, 1]$ and satisfies the variation trend as

$$\begin{cases} \frac{\mathrm{d}QoE_k}{\mathrm{d}sq_k} > 0 & \text{QoE} - \text{PF} \\ \frac{\mathrm{d}QoE_k}{\mathrm{d}sq_k} < 0 & \text{QoE} - \text{IPF} \end{cases}. \tag{6.3}$$

This assumption has no prior restriction on QoE function types, and can also be applied to the above mentioned QoE models in [103, 104, 105, 106, 107].

In this chapter, we consider the scenario with 2-dimensional QoE factors, i.e., latency and energy, and therefore these two factors will be introduced in the following computing models. Note that the proposed model and analysis can be extended to 3-dimensional or more QoE factors through scalarization directly.

6.3.3 Local Computing Model

When the computation task $C_{n,i}$ is performed locally, we assume the single core in each IoT device and the processing capability (i.e., the amount of CPU frequency in cycle/s) assigned to process application $C_{n,i}$ is $f_{n,i}$. Then, the total local computing latency $T_{n,i}^{\text{comp,l}}$ consists of two parts, i.e., the wait time $t_{n,i}^{\text{w,l}}$ and the computing time $t_{n,i}^{\text{comp,l}}$. The wait time $t_{n,i}^{\text{w,l}}$ in the task queue of IoT device n can be calculated as $\max\{0, t_{n,i-1}^{\text{f,l}} - t_{n,i}^{\text{a}}\}$, where $t_{n,i-1}^{\text{f,l}}$ indicates the finish time of previous task. The computing time $t_{n,i}^{\text{comp,l}}$ equals to $w_{n,i}^{\text{comp}} \big/ f_{n,i}$. Then, the total local latency can be represented as

$$T_{n,i}^{\text{comp,l}} = \max\{0, t_{n,i-1}^{\text{f,l}} - t_{n,i}^{\text{a}}\} + \frac{w_{n,i}^{\text{comp}}}{f_{n,i}}. \tag{6.4}$$

According to the circuit theory [63], the total local energy consumption $E_{n,i}^{\text{comp,l}}$ for local computing can be formulated as

$$E_{n,i}^{\text{comp,l}} = k_n w_{n,i}^{\text{comp}} f_{n,i}^2, \tag{6.5}$$

where k_n is a coefficient reflecting the relationship between computation capability and energy consumption at the IoT device side.

6.3.4 Edge Computing Model

When the IoT device n cannot process the application $C_{n,i}$ with its QoE requirement, $C_{n,i}$ will be added to the task queue in the edge system for further proceeding. In general, the computation results of such tasks have sufficiently smaller data sizes compared with those of the input, and thus the size and transmission time of the output data of all tasks are considered to be negligible throughout this chapter. Then, the processing in the edge system mainly consists of two parts: 1) *communication part* that refers to receiving the computing task by the AP through wireless communication links, and 2) *computation part* that refers to processing the computing task by the MEC server.

For the communication part, the data transmission rate is denoted as $tr_{n,i}$, which can be characterized by various wireless transmission models based on Shannon's formula. In this chapter, when data is offloaded from the IoT device to the AP over the assigned wireless bandwidth $B_{n,i}$, the transmission rate is expressed as $tr_{n,i} = B_{n,i} \log_2(1 + \frac{p_{n,i}h_n}{w_0})$, where $p_{n,i}$ is the transmission power of the IoT device, and w_0 denotes the white Gaussian noise power. The channel gain h_n is generally affected by the path loss inverse to the distance from the IoT device to the AP.

Accordingly, the communication latency $T_{n,i}^{\text{comm,e}}$ for uplink transmission can be given by

$$T_{n,i}^{\text{comm,e}} = \max\{0, t_{n,i-1}^{\text{f,comm}} - t_{n,i}^{\text{a,comm}}\} + \frac{w_{n,i}^{\text{comm}}}{tr_{n,i}}, \tag{6.6}$$

where the first part of (6.6) is the wait time $t_{n,i}^{\text{w,comm}}$ in the transmission queue of AP. The time $t_{n,i-1}^{\text{f,comm}}$ and $t_{n,i}^{\text{a,comm}}$ indicate the finish time of previous task and the arrival time of current task, respectively. The second part is the communication time $t_{n,i}^{\text{comm,e}}$. Furthermore, the total energy consumption for communication $E_{n,i}^{\text{comm,e}}$ can be defined as

$$E_{n,i}^{\text{comm,e}} = p_{n,i} \frac{w_{n,i}^{\text{comm}}}{tr_{n,i}}. \tag{6.7}$$

After the application $C_{n,i}$ is offloaded to the AP, the MEC server will process the task for the computation part. We assume there are multiply cores equipped in the MEC server. The processing capability assigned to process application $C_{n,i}$ in L-core of MEC server is $F_{n,i}$. Then, the computation latency for processing $C_{n,i}$ in the MEC server $T_{n,i}^{\text{comp,e}}$ can be represented as

$$T_{n,i}^{\text{comp,e}} = \max\{0, t_{n,i-1}^{\text{f,comp}} - t_{n,i}^{\text{a,comp}}\} + \frac{w_{n,i}^{\text{comp}}}{F_{n,i}}. \tag{6.8}$$

Similar to the local computing, the first part of (6.8) is the wait time $t_{n,i}^{\text{w,comp}}$ in the processing queue of MEC server, and the second part is the computing time $t_{n,i}^{\text{comp,e}}$. The corresponding energy $E_{n,i}^{\text{comp,e}}$ consumed in MEC server side can be given by

$$E_{n,i}^{\text{comp,e}} = K_n w_{n,i}^{\text{comp}} F_{n,i}^2, \tag{6.9}$$

where K_n is a coefficient reflecting the relationship between computation capability and energy consumption at the MEC server side.

In summary, the total latency in MEC computing model consists of the communication latency $T_{n,i}^{\text{comm,e}}$ and computation latency $T_{n,i}^{\text{comp,e}}$. The energy consumption $E_{n,i}^{\text{comm,e}}$ and $E_{n,i}^{\text{comp,e}}$ occur in the IoT device and the MEC server should be considered in each side, respectively.

6.3.5 Problem Formulation

In this chapter, our optimization objective is to maximize the overall QoE performance across heterogeneous IoT devices through jointly scheduling communication and computation resources from IoT devices and the edge system.

Based on the above system model, the QoE performance of application $C_{n,i}$ can be represented as

$$qoe_{n,i} = \begin{cases} qoe_{n,i}^{l} & qoe_{n,i}^{l} > qoe_{n}^{\text{threshold}} \\ qoe_{n,i}^{e} & \text{otherwise} \end{cases}, \tag{6.10}$$

where $qoe_{n}^{\text{threshold}}$ indicates the QoE requirement of IoT device n for application $C_{n,i}$. If local computing can satisfy this requirement, the IoT device will conduct local computing; otherwise, the application will be offloaded to edge system for further processing. We take 2-dimensional QoE factors in this chapter, and thus the QoE importance can be indicated by one parameter α_n. Then, the QoE gained by local computing is expressed as

$$\begin{aligned} qoe^{l}_{n,i} &= \alpha_n QoE^T(T^{\text{comp,l}}_{n,i}) + (1-\alpha_n) QoE^E(E^{\text{comp,l}}_{n,i}) \\ &= \alpha_n QoE^T(t^{\text{w,l}}_{n,i} + \frac{w^{\text{comp}}_{n,i}}{f_{n,i}}) + (1-\alpha_n) QoE^E(k_n w^{\text{comp}}_{n,i} f^2_{n,i}) \end{aligned}, \quad (6.11)$$

where QoE^T and QoE^E are monotonically decreasing regarding to QoE factors $T^{\text{comp,l}}_{n,i}$ and $E^{\text{comp,l}}_{n,i}$, respectively. Both of $T^{\text{comp,l}}_{n,i}$ and $E^{\text{comp,l}}_{n,i}$ are affected by IoT device's computation resource, i.e., CPU frequency $f_{n,i}$. When the application is offloaded to the edge system for further processing, its QoE performance can be formulated as

$$\begin{aligned} qoe^{e}_{n,i} &= \alpha_n QoE^T(T^{\text{comm,e}}_{n,i} + T^{\text{comp,e}}_{n,i}) + (1-\alpha_n) QoE^E(E^{\text{comm,e}}_{n,i}) \\ &= \alpha_n QoE^T(t^{\text{w}}_{n,i} + \frac{w^{\text{comm}}_{n,i}}{B_{n,i}\log_2(1+\frac{p_{n,i}|h|^2}{w_0})} + \frac{w^{\text{comp}}_{n,i}}{F_{n,i}}) \\ &+ (1-\alpha_n) QoE^E(\frac{p_{n,i} w^{\text{comm}}_{n,i}}{B_{n,i}\log_2(1+\frac{p_{n,i}|h|^2}{w_0})}) \end{aligned}, \quad (6.12)$$

where the wait time $t^{\text{w}}_{n,i}$ includes the communication wait time $t^{\text{w,comm}}_{n,i}$ and computation wait time $t^{\text{w,comp}}_{n,i}$. Since QoE is used to evaluate the IoT device's satisfaction, we only consider the energy consumption $E^{\text{comm,e}}_{n,i}$ on IoT device's side. Both of $T^{\text{comm,e}}_{n,i}$ and $E^{\text{comm,e}}_{n,i}$ are affected by edge system's communication resource, i.e., bandwidth $B_{n,i}$, and $T^{\text{comp,e}}_{n,i}$ is also related to edge system's computation resource, i.e., CPU frequency $F_{n,i}$.

Consequently, the device-specific QoE enhancement resource scheduling (QoE-RS) problem is expressed as

$$\mathbf{P(1)}: \max_{f_{n,i}, B_{n,i}, F_{n,i}} \sum_{n=1}^{N}\sum_{i=1}^{I} qoe_{n,i}(C_{n,i}, \mathbf{Q}_n)$$

$$s.t. \begin{cases} 0 \le f_{n,i} \le f^{\max}_n \\ \sum_{n=1}^{N} B_{n,i} \le B^{\max} \\ 0 \le F_{n,i} \le F^{\max} \end{cases}. \quad (6.13)$$

Solving the QoE-RS problem requires optimizing the strategies of IoT devices and the edge system simultaneously, which leads to several technical challenges. First, from the perspective of the IoT device, it demands resource scheduling with low complexity due to its limited computation resource. Then, the edge system is conceived as a bin with finite available computation and communication resources, and our objective is to schedule these resources for the overall QoE enhancement amongst different IoT devices. Considering the stochastic computation tasks, the dynamic communication environment, and the unknown QoE assessment model, it is hard for the edge system to solve resource scheduling problems with traditional optimization solutions.

6.4 QoE-RS Optimization Solution

According to the above analysis, the solution to the QoE-RS problem (P1) is highly nontrivial since it needs to optimize resource allocation variables $f_{n,i}$, $B_{n,i}$, and $F_{n,i}$ simultaneously to achieve the overall QoE maximization. To solve this problem, we separate the QoE-RS problem into two stages.

In the first stage, we aim to devise a local policy for each IoT device that generates an optimal computation resource scheduling strategy $f_{n,i}$ based on QoE formulation and accordingly determine the offloading policy. Then, each IoT device can optimize its strategy distributively, and the QoE-RS problem of the first stage reduces to a convex problem (P2) as follows.

$$\mathbf{P(2)}: \quad \max_{f_{n,i}} \sum_{i=1}^{I} qoe_{n,i}(C_{n,i}, \mathbf{Q}_n)$$

$$s.t. \ 0 \leq f_{n,i} \leq f_n^{\max} \tag{6.14}$$

In the second stage, we interest in joint optimize communication resource $B_{n,i}$, and computation resource $F_{n,i}$ of the edge system based on the IoT devices' tasks information, and the subproblem (P3) can be given by

$$\mathbf{P(3)}: \quad \max_{B_{n,i},F_{n,i}} \sum_{n=1}^{N} \sum_{i=1}^{I} qoe_{n,i}(C_{n,i})$$

$$s.t. \begin{cases} \sum_{n=1}^{N} B_{n,i} \leq B^{\max} \\ 0 \leq F_{n,i} \leq F^{\max} \end{cases}. \tag{6.15}$$

The major difficulty of P(3) lying in the edge's QoE-RS problem is the uncertainty in the practical scenario, specifically including stochastic computation tasks, the dynamic communication environment, and the unknown QoE assessment model. Traditional optimization algorithms require iteratively adjusting the resource scheduling with static information, which is fundamentally infeasible for real-time optimization problems with uncertainty. To tackle such an issue, we adopt the PPO algorithm, a popular DRL approach, to propose a novel online resource scheduling algorithm that can achieve long-term optimization without prior information.

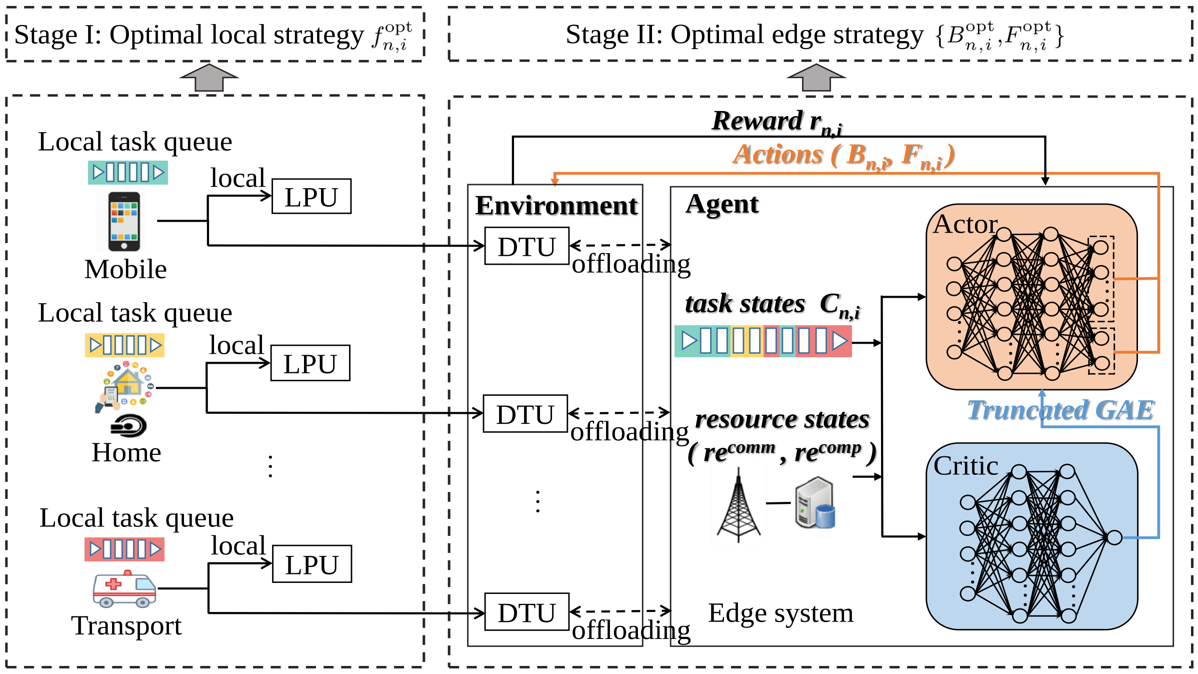

Stage I: Optimal local strategy $f_{n,i}^{\text{opt}}$
Stage II: Optimal edge strategy $\{B_{n,i}^{\text{opt}}, F_{n,i}^{\text{opt}}\}$
Local task queue
local
LPU
Mobile
Home
Transport
DTU
offloading
Environment
Agent
Reward $r_{n,i}$
Actions ($B_{n,i}$, $F_{n,i}$)
Actor
Critic
task states $C_{n,i}$
resource states (re^{comm}, re^{comp})
Truncated GAE
Edge system

Figure 6.3: Illustration of the two-stage QoE-RS optimization.

Accordingly, problem (P1) can be decomposed into two subproblems, namely, IoT devices' QoE-RS problem (P2) and edge's QoE-RS problem (P3), which can be solved by our proposed two-stage resource scheduling scheme as illustrated in Figure 6.3.

6.4.1 Stage I: IoT Devices' QoE-RS Optimization

In this section, we present a distributed algorithm for optimally solving problem (P2) on IoT devices.

Lemma 6.4.1 *Given the task $C_{n,i}$ including computing workload $w_{n,i}^{\mathrm{comp}}$ and its QoE requirement Q_n, the lower bound of optimal CPU-cycle frequency is $f_{n,i}^{\mathrm{low}} = w_{n,i}^{\mathrm{comp}} \big/ T_n^{\max}$, and the upper bound is $f_{n,i}^{\mathrm{up}} = \min\{f_{n,i}^{\max}, f_{n,i}^{\mathrm{E_{max}}}\}$, where $f_{n,i}^{\mathrm{E_{max}}} = \sqrt{E_n^{\max} \big/ k_n w_{n,i}^{\mathrm{comp}}}$. The solution of optimal CPU-cycle frequency exists only when $f_{n,i}^{\mathrm{low}} <= f_{n,i}^{\mathrm{up}}$.*

Here, the lower bound $f_{n,i}^{\mathrm{low}}$ is devised by the application latency constraint, i.e., $w_{n,i}^{\mathrm{comp}} \big/ f_{n,i} <= T_n^{\max}$. The upper bound $f_{n,i}^{\mathrm{up}}$ is limited by the IoT device's computational capacity from the perspective of the CPU frequency as $f_{n,i} <= f_{n,i}^{\max}$, and the power consumption as $k_n w_{n,i}^{\mathrm{comp}} f_{n,i}^2 <= E_n^{\max}$.

Theorem 6.4.2 *According to Lemma 6.4.1, CPU frequency $f_{n,i}$ has its closed interval $[f_{n,i}^{\mathrm{low}}, f_{n,i}^{\mathrm{up}}]$. As definition in (6.4) and (6.5), $T_{n,i}^{\mathrm{comp,l}}$ and $E_{n,i}^{\mathrm{comp,l}}$ are continuous regarding to CPU frequency $f_{n,i}$ on its closed interval. Furthermore, based on the definition of qoe^l in (6.11), the bounded functions QoE^T and QoE^E are continuous regarding to $T_{n,i}^{\mathrm{comp,l}}$ and $E_{n,i}^{\mathrm{comp,l}}$, respectively. Then, $qoe_{n,i}^l$ is continuous regarding to CPU frequency $f_{n,i}$. According to extreme value theorem, the real-valued function $qoe_{n,i}^l(f_{n,i})$ is continuous on the closed interval $[f_{n,i}^{\mathrm{low}}, f_{n,i}^{\mathrm{up}}]$, and thus $qoe_{n,i}^l$ must attain a maximum at least once. That is, there must exist a optimal resource scheduling strategy that CPU-cycle frequency $f_{n,i}^{\mathrm{opt}}$ in $[f_{n,i}^{\mathrm{low}}, f_{n,i}^{\mathrm{up}}]$ such that:*

$$qoe_{n,i}^l(f_{n,i}^{\mathrm{opt}}) >= qoe_{n,i}^l(f_{n,i}),\ \forall f_{n,i} \in [f_{n,i}^{\mathrm{low}}, f_{n,i}^{\mathrm{up}}]. \tag{6.16}$$

Theorem 6.4.2 proves the existence of the optimal resource scheduling strategy, i.e., CPU-cycle frequency $f_{n,i}^{\mathrm{opt}}$, to maximize the QoE performance of IoT devices. Below we will give out the process to obtain this optimal strategy.

Theorem 6.4.3 *When $f_{n,i}^{\mathrm{low}}$ is no more than $f_{n,i}^{\mathrm{up}}$, the optimal CPU-cycle frequency $f_{n,i}^{\mathrm{opt}}$ can be* *calculated as follows given the task information $C_{n,i}$ and its QoE requirement Q_n.*

$$f^{\text{opt}}_{n,i} = \begin{cases} f^{\text{low}}_{n,i} & \text{if } \alpha_n = 0 \\ \underset{f \in \{f^*_{n,i}, f^{\text{low}}_{n,i}, f^{\text{up}}_{n,i}\}}{\arg\max} \; qoe^l_{n,i}(f_{n,i}) & \text{if } 0 < \alpha_n < 1 \\ f^{\text{up}}_{n,i} & \text{otherwise} \end{cases}, \tag{6.17}$$

and $f^*_{n,i}$ can be derived by

$$f^*_{n,i} = \underbrace{\left[\alpha_n \frac{\mathrm{d}QoE^T}{\mathrm{d}T^{\text{comp,l}}_{n,i}}, (1-\alpha_n)\frac{\mathrm{d}QoE^E}{\mathrm{d}E^{\text{comp,l}}_{n,i}} \right]}_{\text{QoE}\to\text{QoE factor}} \underbrace{\begin{bmatrix} -(\sum_{j=0}^{\lambda_n T - i} \beta^j) w^{\text{comp}}_{n,i} {f^*_{n,i}}^{-2} \\ 2k_n w^{\text{comp}}_{n,i} f^*_{n,i} \end{bmatrix}}_{\text{QoE factor}\to\text{Resource}}, \tag{6.18}$$

where

$$\beta = \begin{cases} 0 & \text{if } f^*_{n,i} \geq w^{\text{comp}}_{n,i}\lambda_n \\ \in (0,1] & \text{if } f^*_{n,i} < w^{\text{comp}}_{n,i}\lambda_n \end{cases}.$$

Proof When the IoT device n conducts the local computing, it needs to obtain the optimal policy given its QoE assessment model. The analysis will be done in three cases as $\alpha_n = 0$, $\alpha_n = 1$ and $0 < \alpha_n < 1$.

First, suppose that $\alpha_n = 0$, the QoE gained by the local computing can be rewritten as

$$qoe^l_{n,i}(f_{n,i}) = QoE^E(k_n w^{\text{comp}}_{n,i} f^2_{n,i}), \tag{6.19}$$

which means that the QoE performance is only affected by the energy consumption. We can observer that $\frac{\mathrm{d}QoE^E}{\mathrm{d}E^{\text{comp,l}}_{n,i}} < 0$ and $\frac{\mathrm{d}E^{\text{comp,l}}_{n,i}}{\mathrm{d}f_{n,i}} = 2k_n w^{\text{comp}}_{n,i} f_{n,i} > 0$, and then $qoe^l_{n,i}(f_{n,i})$ is a monotonically decreasing function of $f_{n,i}$. Thus, to maximize $qoe^l_{n,i}$, $f^{\text{opt}}_{n,i}$ should be set as the lower bound $f^{\text{low}}_{n,i}$.

Second, suppose that $\alpha_n = 1$, the QoE gained by the local computing can be reformulated as

$$qoe^l_{n,i}(f_{n,i}) = QoE^T(t^{\text{w,l}}_{n,i} + \frac{w^{\text{comp}}_{n,i}}{f_{n,i}}), \tag{6.20}$$

which means that the QoE performance is only affected by the computing latency. Since $\frac{\mathrm{d}QoE^T}{\mathrm{d}T^{\text{comp,l}}_{n,i}} < 0$ and $\frac{\mathrm{d}T^{\text{comp,l}}_{n,i}}{\mathrm{d}f_{n,i}} = -w^{\text{comp}}_{n,i} f^{-2}_{n,i} < 0$, $qoe^l_{n,i}(f_{n,i})$ is a monotonically increasing function of $f_{n,i}$. Thus, to maximize $qoe^{l}_{n,i}$, $f^{\text{opt}}_{n,i}$ should be set as the upper bound $f^{\text{up}}_{n,i}$.

Finally, we will discuss the case when α_n is in the range of $(0, 1)$. According to extreme value theorem, we first calculate critical values of $qoe^{l}_{n,i}(f_{n,i})$ over the interval $(f^{\text{low}}_{n,i}, f^{\text{up}}_{n,i})$ by setting the first derivation equals to 0 as

$$\underbrace{\left[\alpha_n \frac{\mathrm{d}QoE^{T}}{\mathrm{d}T^{\text{comp,l}}_{n,i}}, (1-\alpha_n)\frac{\mathrm{d}QoE^{E}}{\mathrm{d}E^{\text{comp,l}}_{n,i}}\right]}_{\text{QoE}\rightarrow\text{QoE factor}} \underbrace{\begin{bmatrix} \dfrac{\mathrm{d}T^{\text{comp,l}}_{n,i}}{\mathrm{d}f_{n,i}} \\ \dfrac{\mathrm{d}E^{\text{comp,l}}_{n,i}}{\mathrm{d}f_{n,i}} \end{bmatrix}}_{\text{QoE factor}\rightarrow\text{Resource}} = 0. \tag{6.21}$$

Then, evaluating critical numbers and bound points can find the optimal strategy for IoT devices' computation resource scheduling. We can notice that the first part reflects the relationship between QoE and QoE factor, and the second one is related to the QoE factor to resources.

Here, the hypothesis ignoring the wait time is when the IoT device can finish the current computing task before the next one arrives, i.e., $f^{*}_{n,i}$ should satisfy $w^{\text{comp}}_{n,i}\big/f^{*}_{n,i}t <= t^{\text{a}}_{n,i+1} - t^{\text{a}}_{n,i}$. As mentioned in Section 6.3.1, the task generates stochastically which follows a Poisson process with rate λ_n, and then we can obtain the expected interval of task arriving as $1/\lambda_n$. Therefore, $\frac{\mathrm{d}T^{\text{comp,l}}_{n,i}}{\mathrm{d}f_{n,i}}$ equals to $-w^{\text{comp}}_{n,i} f^{-2}_{n,i}$ if $f^{*}_{n,i} >= w^{\text{comp}}_{n,i}\lambda_n$. When $f^{*}_{n,i} < w^{\text{comp}}_{n,i}\lambda_n$, the wait time $t^{\text{w,l}}_{n,i}$ should be considered by the expectation value as $w^{\text{comp}}_{n,i-1}\big/f^{*}_{n,i-1} - 1/\lambda_n$. Then, the computation time in a long-term by considering the impact on subsequent tasks is reformulated as $T^{\text{comp,l}}_{n,i} = \frac{w^{\text{comp}}_{n,i}}{f_{n,i}} + \sum_{j=0}^{\lambda_n T-i-1} \beta^{j}(\frac{w^{\text{comp}}_{n,i}}{f_{n,i}} - \frac{1}{\lambda_n})$, where $\beta \in (0, 1]$ is the adaptive discount factor of the impact on subsequent tasks. In this case, $\frac{\mathrm{d}T^{\text{comp,l}}_{n,i}}{\mathrm{d}f_{n,i}}$ can be updated by $-(\sum_{j=0}^{\lambda_n T-i}\beta^{j})w^{\text{comp}}_{n,i} f^{-2}_{n,i}$. Finally, we can get the solution as (6.18), by which the IoT device can choose the optimal strategy distributively to achieve an approximate long-term optimization.

According to (6.18), the IoT Device's QoE-RS problem has been transferred into seeking the solution to a monadic equation when $0 < \alpha_n < 1$, which can be solved within polynomial time.

Since the difference to find optimal strategy among IoT devices is related to the first part of (6.21), we take the linear QoE assessment model as an example to further explain the solving process when $0 < \alpha_n < 1$. In this case, the QoE gained in local computing can be expressed explicitly as

$$\begin{aligned} qoe^l_{n,i}(f_{n,i}) =& \alpha_n(1 - \frac{\frac{w^{\text{comp}}_{n,i}}{f_{n,i}} + \sum_{j=0}^{\lambda_n T-i-1} \beta^j(\frac{w^{\text{comp}}_{n,i}}{f_{n,i}} - \frac{1}{\lambda_n})}{T^{\max}_{n,i}}) \\ &+ (1-\alpha_n)(1 - \frac{k_n w^{\text{comp}}_{n,i} f^2_{n,i}}{E^{\max}_{n,i}}) \end{aligned}. \tag{6.22}$$

In the interval of $[f^{\text{low}}_{n,i}, f^{\text{up}}_{n,i}]$, $qoe^l_{n,i}$ is a continuous quadratic function of $f_{n,i}$. Taking first derivative of $qoe^l_{n,i}$ with respect to f_i, we get

$$\frac{\mathrm{d}qoe^l_{n,i}}{\mathrm{d}f_{n,i}} = \left[-\frac{\alpha_n}{T^{\max}_{n,i}}, -\frac{(1-\alpha_n)}{E^{\max}_{n,i}}\right] \begin{bmatrix} -(\sum_{j=0}^{\lambda_n T-i} \beta^j) w^{\text{comp}}_{n,i} f^{-2}_{n,i} \\ 2k_n w^{\text{comp}}_{n,i} f_{n,i} \end{bmatrix}. \tag{6.23}$$

If there exists $f_{n,i}$ that makes the first derivation equals to 0, it is the optimal strategy to maximize $qoe^l_{n,i}$. Consequently, we can obtain the optimal CPU-cycle frequency $f^*_{n,i}$ as

$$f^*_{n,i} = \sqrt[3]{(\frac{\sum_{j=0}^{\lambda_n T-i} \beta^j)\alpha_n E^{\max}_{n,i}}{2k_n(1-\alpha_n)T^{\max}_{n,i}}}. \tag{6.24}$$

Setting $\beta = 0$, the optimal strategy in (24) is simplified as

$$f^*_{n,i} = \sqrt[3]{\frac{\alpha_n E^{\max}_{n,i}}{2k_n(1-\alpha_n)T^{\max}_{n,i}}}. \tag{6.25}$$

If $f^*_{n,i}$ deviated by (6.25) is less than $w^{\text{comp}}_{n,i}\lambda_n$, the discount factor β will be adjusted to a value between $(0, 1]$ in order to achieve a long-term optimization by considering the impact on subsequent tasks.

Furthermore, as indicated in Theorem 6.4.2, we need to further discuss the above critical numbers with bound points in the following three cases:

- Case 1: when $f^*_{n,i}$ is less than $f^{\text{low}}_{n,i}$, $qoe^l_{n,i}$ will be a monotonically decreasing function of $f_{n,i}$ since $\frac{\mathrm{d}qoe^l_{n,i}}{\mathrm{d}f_{n,i}}$ is negative when $f_{n,i} \in [f^{\text{low}}_{n,i}, f^{\text{up}}_{n,i}]$. Thus, to maximize $qoe^l_{n,i}$, f^{opt}_i should be set as the lower bound $f^{\text{low}}_{n,i}$, i.e., $f^{\text{opt}}_{n,i} = \max\{f^{\text{low}}_{n,i}, f^*_{n,i}\}$.

- Case 2: when $f^*_{n,i}$ is larger than $f^{\text{up}}_{n,i}$, $qoe^l_{n,i}$ will be a monotonically increasing function of

$f_{n,i}$ since $\frac{\mathrm{d}qoe^{l}_{n,i}}{\mathrm{d}f_{n,i}}$ is positive when $f_{n,i} \in [f^{\mathrm{low}}_{n,i}, f^{\mathrm{up}}_{n,i}]$. Thus, to maximize $qoe^{l}_{n,i}$, f^{opt}_{i} should be set as the upper bound $f^{\mathrm{up}}_{n,i}$, i.e., $f^{\mathrm{opt}}_{n,i} = \min\{f^{*}_{n,i}, f^{\mathrm{up}}_{n,i}\}$.

- Case 3: If $f^{*}_{n,i}$ given in (6.18) falls into the range of $[f^{\mathrm{low}}_{n,i}, f^{\mathrm{up}}_{n,i}]$, f^{opt}_{i} should be set as f^{*}_{i}.

Substituting f^{*}_{i} into (6.17), we can get the optimal strategy of local computing as f^{opt}_{i} with which the IoT device can reach its maximum $qoe^{l}_{n,i}$ by (6.11). If the optimal strategy still cannot satisfy the QoE requirement, the IoT device will offload the task to the edge system for remote processing. The detailed algorithm for IoT devices' QoE-RS optimization is shown in Algorithm 4.

Algorithm 4 Stage I: Distribute Algorithm for IoT Devices' QoE-RS Optimization

```
1:  Initialize computation resources of IoT devices
2:  Initialize task generation by IoT devices
3:  for $n = 1, 2, \ldots, N$ do
4:      for $i = 1, 2, \ldots, I$ do
5:          IoT device $n$ processes computation task $C_{n,i}$
6:          obtain $f^{\mathrm{low}}_{n,i}$ and $f^{\mathrm{up}}_{n,i}$ by Lemma 6.4.1
7:          if $\alpha_n = 0$ then
8:              $f^{\mathrm{opt}}_{n,i} \leftarrow f^{\mathrm{low}}_{n,i}$
9:          else if $\alpha_n = 1$ then
10:             $f^{\mathrm{opt}}_{n,i} \leftarrow f^{\mathrm{up}}_{n,i}$
11:         else
12:             obtain $f^{*}_{n,i}$ by (6.18)
13:             $f^{\mathrm{opt}}_{n,i} \leftarrow \arg\max_{f \in \{f^{*}_{n,i}, f^{\mathrm{low}}_{n,i}, f^{\mathrm{up}}_{n,i}\}} qoe^{l}_{n,i}(f_{n,i})$
14:         obtain $f^{\mathrm{opt}}_{n,i}$ and corresponding $qoe^{l}_{n,i}$ by (6.11)
15:         if $qoe^{l}_{n,i} \geq qoe^{\mathrm{threshold}}_{n}$ then
16:             process $C_{n,i}$ locally
17:         else
18:             offload $C_{n,i}$ to the task queue of edge system
```

6.4.2 Stage II: Edge's QoE-RS Optimization

In the edge system, due to massive applications generated by IoT devices with different QoE requirements, our objective is not only to complete more computation tasks in the edge system, but also to ensure device-specific QoE requirements in the long-term maximization. To achieve these goals, we adopt DRL that can optimize strategies in a trial-and-error manner by interacting with the unknown environment to achieve a long-term optimization.

First, we formulate the problem as a Markov decision process (MDP), defined as a 5-tuple, i.e., $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$, including state space $\mathcal{S}$, action space $\mathcal{A}$, state transition function P, reward function R, and discount factor γ. The detailed conceptions about MDP model can be found in Section 2.3.2. In the edge's QoE-RS problem, the set of IoT devices is considered as the environment, and the edge system plays the role of the agent, which continually performs decisions to interact with the IoT devices. The information of the computation tasks and the available resources are adopted as the state. The action refers to scheduling the communication and computation resources in the edge system. The objective of DRL in this chapter is to find the optimal policy π^* , to maximize the expectation of accumulated QoE reward from any state in the state space, which is coincide with the optimization objective (6.15). The details of the above elements are introduced in the following subsections.

State: The state $s \in \mathcal{S}$ describes the information of the computation tasks generated by the IoT device n as well as resources available in the edge system, which can be given as

$$\mathcal{S} = \{s | s = (n, t_{n,i}^{\mathrm{a}}, w_{n,i}^{\mathrm{comm}}, w_{n,i}^{\mathrm{comp}}, re^{\mathrm{comm}}, re^{\mathrm{comp}})\}, \tag{6.26}$$

where the tuple $\{t_n^a, w_n^{\mathrm{comm}}, w_n^{\mathrm{comp}}\}$ denotes the information of computation tasks offloaded from IoT device n to the edge system for processing, and re^{comm} and re^{comp} indicate the communication and computation resources that are available for offloading and computing, respectively. Here, the communication resources re^{comm} is referred to the percentage of bandwidth that avaiable for allocation. Considering that it is impractical to divide the bandwidth into an infinitely small channel, we assume that the total bandwidth is divided into M channels, and then re^{comm} equals to $\{m/M|_{m=0,1,\cdots,M}\}$. The computation resources re^{comp} is referred to the ready time of the MEC server for computing. As mentioned above, MEC server has multiply core for processing, and then re^{comp} should be in range of $[0, T]$ and satisfies $re^{\mathrm{comp}} = \min\{re_l^{\mathrm{comp}}|_{l \in L}\}$.

Action: The action $a \in \mathcal{A}$ serves two resource allocations, i.e., the bandwidth as communication resources and the CPU-cycle frequency as computation resources. Accordingly, the action a can be described as

$$\mathcal{A} = \{a | a = (B_{n,i}, F_{n,i})\}. \tag{6.27}$$

Since the total bandwidth is divided into M channels, the bandwidth of each channel is

$B^{\max}/M$. Then, the action of $B_{n,i}$ is a discrete value that its effective range is from 0 to $re^{\text{comm}}B^{\max}$. The multiply cores of the MEC server can process the computing task independently, and thus the action of $F_{n,i}$ of each core has the same limitation as $F_{n,i} \in [0, F^{\max}]$. When the action equals to 0, the task is discarded due to the unavailable resource or other reasons, and it can be forwarded to the remote cloud server for processing, which is beyond the scope of our problem.

Reward: A reward signal is designed to coincide with the objective of our optimization problem. In this chapter, our objective is defined as maximizing the accumulated QoE of all computation tasks from different IoT devices in a time period as expressed in (15). By decomposing the objective into small pieces at each decision epoch, the immediate reward r_t can be constructed as a function of state s_t and action a_t, which is the QoE achieved for processing the application $C_{n,i}$ at time step t, given by

$$r_t = \begin{cases} 1 + qoe^e_{n,i} & qoe^e_{n,i} > qoe^{\text{threshold}}_n \\ -1 & \text{otherwise} \end{cases}. \tag{6.28}$$

If the computation task is completed within the QoE requirement, 1 is offered as the reward; otherwise, -1 is offered as the penalty. $qoe^e_{n,i}$ is the additive reward to evaluate the achieved QoE performance. Furthermore, considering the energy consumed by the task processing in the MEC server, we add $cost^{\text{comp}}$ as the ratio of actual energy consumption to the one at maximum processing rate, i.e., $(F_{n,i}/F^{\max})^2$, then, the immediate reward is finally rewritten as

$$r'_t = r_t - cost^{\text{comp}}(F_{n,i}). \tag{6.29}$$

To maximize the long-term utility of the edge system, the accumulated reward G_t is given by $G_t = \sum_{t=0}^{T} \gamma^t r'_t$.

PPO-based Solution: The route to success in reinforcement learning is not as obvious since the algorithms usually suffer from the stability and rate of convergency, and they also require substantial effort in tuning in order to get good results. In this chapter, we adopt the PPO algorithm, one of the popular DRL approaches based on Actor-Critic architecture that integrates value-based algorithms and policy-based algorithms to realize stable and fast convergency. The detailed conceptions about Actor-Critic architecture can be found in Section 2.3.2. Furthermore, PPO tries to compute an update at each step that minimizes the cost function while ensuring the deviation from the previous policy is relatively small, which further improves its learning performance while making it much simpler to implement and tune. This motivates us to optimize the online strategy in the edge system with the PPO algorithm. The objective function of PPO is formed as

$$L_t^{\mathrm{CLIP}}(\theta) = \mathbb{E}[\min(pr_t(\theta)\hat{A}_t, \mathrm{clip}(pr_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)], \tag{6.30}$$

where the hyper-parameter ε is usually set as 0.1 or 0.2. The probability ratio $pr_t(\theta)$ is the ratio between the new and old policy, given by

$$pr_t(\theta) = \frac{\pi(a_t|s_t;\theta)}{\pi(a_t|s_t;\theta_{\mathrm{old}})}, \tag{6.31}$$

where $\pi(a_t|s_t;\theta)$ represents the approximated policy function by Actor network parameter θ. The generalized advantage estimation (GAE) is proposed to make a compromise between variance and bias. The GAE estimator is written as

$$\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} = \sum_{l=0}^{T-t-1} (\gamma\lambda)^l \delta_{t+l}, \tag{6.32}$$

where λ is used to adjust the bias-variance tradeoff. The value δ_t is calculated by $r_t+\gamma v(s_{t+1};w)-v(s_t;w)$, where $v(s_t;w)$ represents the approximated value function with Critic network parameter w.

The optimal action can be obtained when the objective function converges after the iterations. The details of the proposed online algorithm based on PPO are presented in Algorithm 5.

In summary, upon the application task generation, each IoT device distributively optimizes its computation resource, i.e., CPU frequency $f_{n,i}$ based on (6.18) and make an offloading decision based on QoE requirement. If the optimal QoE can satisfy its requirement, the application will be proceeded locally and thus hold $qoe_{n,i}^{l}$ that is the gained QoE by local computing. Otherwise, the application will be offloaded to the edge system for remote processing, thereby

Algorithm 5 Stage II: PPO-based Online Algorithm for Edge's QoE-RS Optimization

```
1: Initialize computation and communication resource state of edge system
2: Initialize task state generated by IoT devices
3: Initialize Actor and Critic network randomly and obtain $\pi_\theta$, i.e., the policy of edge system
4: Initialize sampling policy $\pi_{\theta_{\text{old}}}$ with $\theta_{\text{old}} \leftarrow \theta$
5: /* Sampling (exploring) with $\pi_{\theta_{\text{old}}}$ */
6: for iteration = 1, 2, ... do
7:     for $t = 1, 2, \ldots$ do
8:         Sample a whole episode in the environment with $\pi_{\theta_{\text{old}}}$, and store the trajectory in the replay buffer $B$
9:         Compute advantage estimates $\hat{A}_t^{\text{GAE}(\gamma,\lambda)}$ according to (6.32)
10:    /* Optimizing $\pi_{\theta_{\text{old}}}$ */
11:    for epoch = 1, 2, ... do
12:        Update the Actor network based on the objective function (6.30), i.e., $\theta \leftarrow \arg\max_\theta L^{\text{CLIP}}(\theta)$, by Adam using the sampled data from $B$ and $A$ for one epoch
13:        Update the Critic network to minimize the squared-error loss, i.e., $w \leftarrow \arg\min_w \mathbb{E}[\delta_n(w)]^2$, by Adam using the sampled data from $B$ for one epoch
14:    Synchronise the sampling policy with $\theta_{\text{old}} \leftarrow \theta$
```

having $qoe^e_{n,i}$ which indicates the gained QoE by edge computing. In the edge system, the online PPO-based algorithm jointly optimizes communication and computation resources, i.e., bandwidth $B_{n,i}$ and CPU frequency $F_{n,i}$ to enhance the QoE requirement of IoT devices. Finally, the proceeded applications will get back to IoT devices.

6.5 Simulation

In this section, extensive simulation experiments are conducted to evaluate the proposed two-stage resource scheduling scheme for both local computing and edge computing from the perspective of QoE performance. Our algorithm and network architecture are implemented using Pytorch.

6.5.1 Simulation Setup

We consider the QoE-RS problem over a period of 100 seconds (T = 100). Each IoT device contains a single processing core, while the MEC server contains L independent processing cores (i.e., virtual machines). Each IoT device's random arrival tasks follow the Poisson dis-

Table 6.2: Simulation parameters of system

Parameter	**Value**
Length of time step	100 sec
Max end user's CPU frequency	10 MHz
Max MEC's CPU frequency	100 MHz
Number of MEC's cores	4
Max AP's bandwidth	100 Mbps
Number of AP's channels	10
Number of application generated	[1,5] /sec
Size of data amount	[1,20] MB
Computation intensity	[1,10] cycles/Byte

Table 6.3: Training hyper-parameters

Parameter	**Value**
Size of hidden layers	2
Size of neural in each hidden layer	128
Optimization method	Adam
Actor learning rate	10^{-4}
Critic learning rate	10^{-3}
Clipping rate	0.2
Discount factor	0.99

tribution with an arrival rate of $\lambda_n \in [1, 5]$, and the computation intensity of tasks is randomly sampled from $[1, 10]$. The main simulation parameters of the system are listed in Table 6.2.

For the PPO-based online algorithm for edge computing, we adopt the same multi-layer perception (MLP) architecture (2 fully connected hidden layers of 128 units) with hyperbolic *relu* activation function to construct the deep neural network. For the output layer of Actor-network, we adopt *softmax* activation function for discrete action of communication resource and *sigmoid* activation function for continuous action of computation resource. The main training hyper-parameters are listed in Table 6.3.

6.5.2 Performance of IoT Device's Strategy

We evaluate the performance of our proposed distributed algorithm for local computing through comparisons against three representative benchmark approaches, including two state-of-the-art approaches (i.e., LS, and ES), and a randomized baseline approach (RS):

- Short-term Optimal algorithm (S-Opt): This optimal QoE-RS approach achieves short-term QoE maximization and acquires optimal strategy when setting discount factor $\beta = 0$.
- Long-term Optimal algorithm (L-Opt): This optimal QoE-RS approach achieves long-term QoE maximization and acquires optimal strategy by adjusting discount factor $\beta \in (0, 1]$.
- Latency-based strategy (LS) : The optimization objective of the latency-based strategy is to achieve service latency minimization, which is consistent with the state-of-the-art methods proposed in [99, 100].
- Energy-based strategy (ES) : The optimization objective of the energy-based strategy is to achieve energy minimization, which is consistent with the state-of-the-art methods proposed in [97, 98].
- Random strategy (Rand): This approach schedules IoT services' strategies following a random fashion, where each IoT device randomly selects the CPU-cycle frequency $f_{n,i}$ to proceed with application $C_{n,i}$.

We first simulate the QoE performance of a single task under different QoE importance. The performance of the five algorithms is shown in Figure 6.4. We can observe that the proposed short-term optimal algorithm outperforms other competing methods as α varies. For the long-term optimal algorithm, it adjusts the optimal policy according to (18), resulting in a lower QoE performance for a single task. When α increases to 0.3, the task can be completed with the optimal policy before the next task is generated so that the long-term optimal policy is the same as the short-term optimal policy. LS focuses on latency minimization, and thus its QoE performance has the same trend as α increases. Conversely, the QoE performance of ES that aims to minimize energy consumption decreases as α increases. This result demonstrates that our proposed optimal strategy maintains excellent QoE performance under various QoE requirements when only a single task is considered.

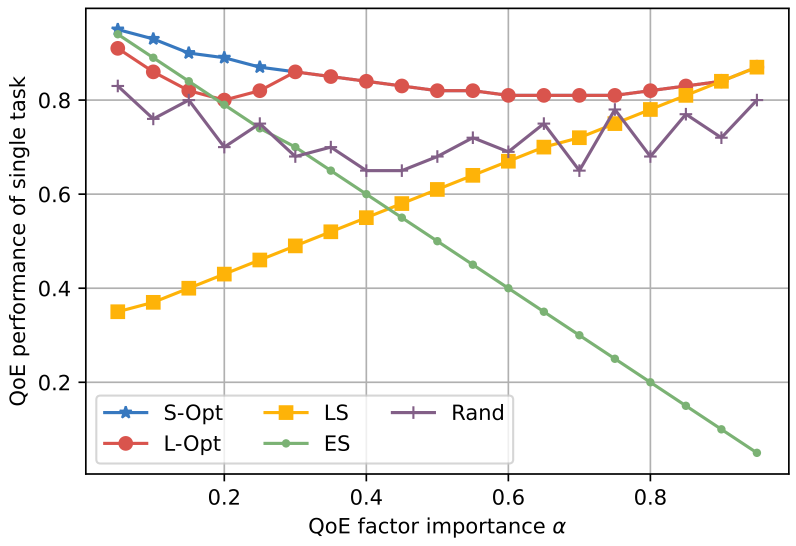


Figure 6.4: QoE performance of local CPU-cycle frequency with single task

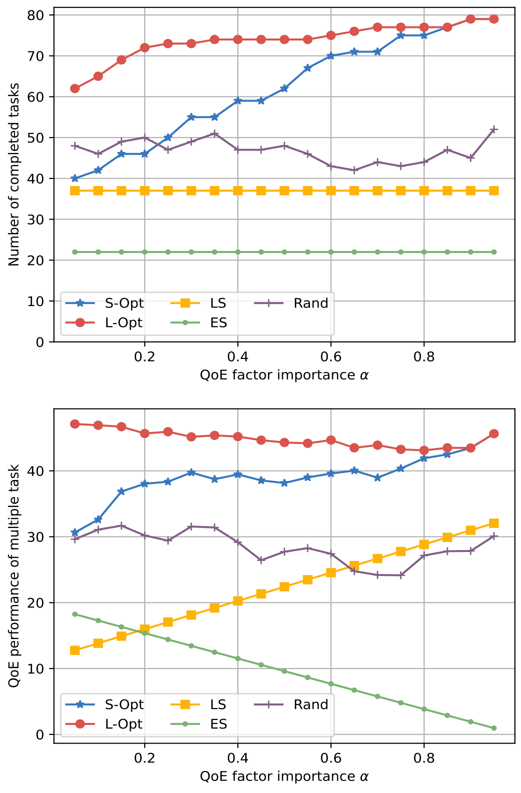
Next, we compare the total number of applications completed and the overall QoE performance of multiple applications under various arrival rates and workload conditions. As illustrated in Figure 6.5, in the multi-task scenario, the completed application and cumulative QoE performance of our proposed long-term optimization algorithm remain high compared with other methods, indicating that the long-term optimal algorithm can achieve better performance in multi-task scenarios. For the short-term optimal algorithm, the optimal CPU-cycle frequency goes up with the importance α, so that there is a gradual improvement when α increases from 0 to 1. The overall QoE performance drops significantly when the average workload increases from 7.5 Megacycles to 30 Megacycles, i.e., Figure 6.5(a) and Figure 6.5(b), respectively. This is because IoT devices cannot meet their QoE requirements when the workload increases dramatically, even by the optimal strategy. Then, compared to Figure 6.5(a) and Figure 6.5(c), the workload remains the same, the average arrival rate increases from 1/s to 4/s, and the overall QoE performance improves accordingly; however, the average QoE performance reduces. That is because the workload of each task does not change, but the generation interval becomes smaller, which also affects the QoE performance. When computing tasks increase, QoE performance will suffer no matter of workload or arriving interval, and thus more and more tasks will be offloaded to the edge system for further processing.

6.5.3 Performance of Edge's Strategy

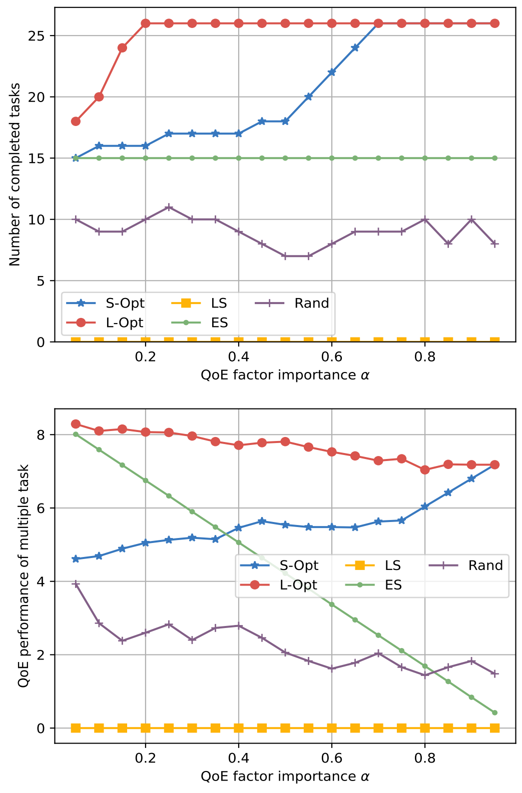
Now, we evaluate the performance of the proposed PPO-based online solution for the QoE-SR problem in the edge system through comparisons with the following four baseline algorithms.

- Joint Communication and Computation scheduling strategy (Comm-Comp): This is a PPO-based online QoE-RS approach that achieves the long-term QoE maximization and acquires optimal communication and computation resource scheduling strategy by Algorithm 2.
- Communication scheduling strategy (Comm): This approach is to achieve the long-term QoE maximization by optimizing communication resource scheduling, which is consistent with the state-of-the-art methods proposed in [112].
- Computation scheduling strategy (Comp): This approach is to achieve the long-term QoE maximization by optimizing computation resource scheduling, which is consistent with the state-of-the-art methods proposed in [113].
- Maximum strategy (Max): The edge proceeds the current application with its maximum available communication and computation resources, regardless of the subsequent applications.
- Random strategy (Rand): This approach schedules edge's strategies in a random fashion. According to a random rank, the edge system randomly selects the bandwidth $B_{n,i}$ and CPU-cycle frequency $F_{n,i}$ to proceed application $C_{n,i}$.

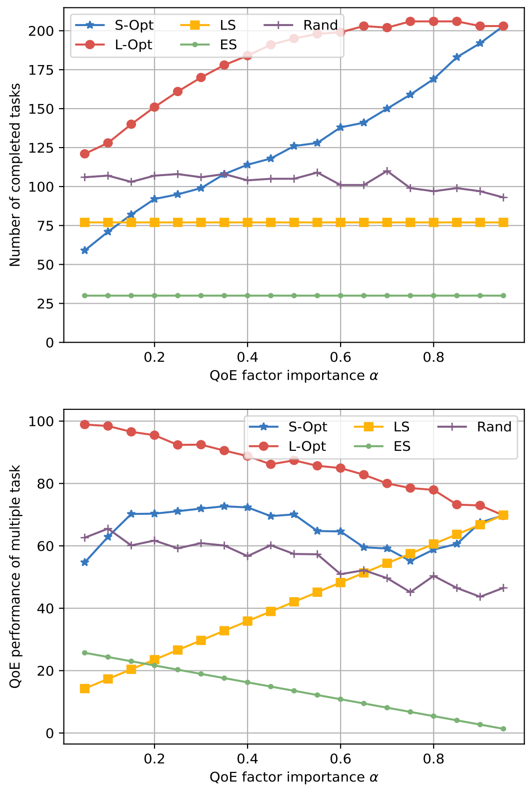
Convergence Performance: We first validate the convergence performance of the proposed PPO-based online algorithm by training and testing in the experiment environment. To evaluate the impact of different QoE assessment models, the importance α is set to 0, 0.5, and 1, respectively. The network parameters are listed in Table II. Training is performed for 5000



(a) $\lambda_n = 1/\mathrm{s}$, $\widehat{w_n^{\mathrm{comp}}} = 7.5$ Megacyclyes



(b) $\lambda_n = 1/\mathrm{s}$, $\widehat{w_n^{\mathrm{comp}}} = 30$ Megacyclyes



(c) $\lambda_n = 4/\mathrm{s}$, $\widehat{w_n^{\mathrm{comp}}} = 7.5$ Megacyclyes

Figure 6.5: QoE performance of local CPU-cycle frequency with multiply tasks

epochs. After every five training epochs, testing is performed in the same environment, and the corresponding accumulated test rewards are recorded as solid lines. The average training results for every five epochs are recorded with dashed lines. Figure 6.6 shows the training and testing curves. It concludes that all scenarios converge faster (before 2000 training epochs) and have a stable convergency. Moreover, compared to training and testing curves, the trained network has good generalization ability and can also work pretty well on the testing data.

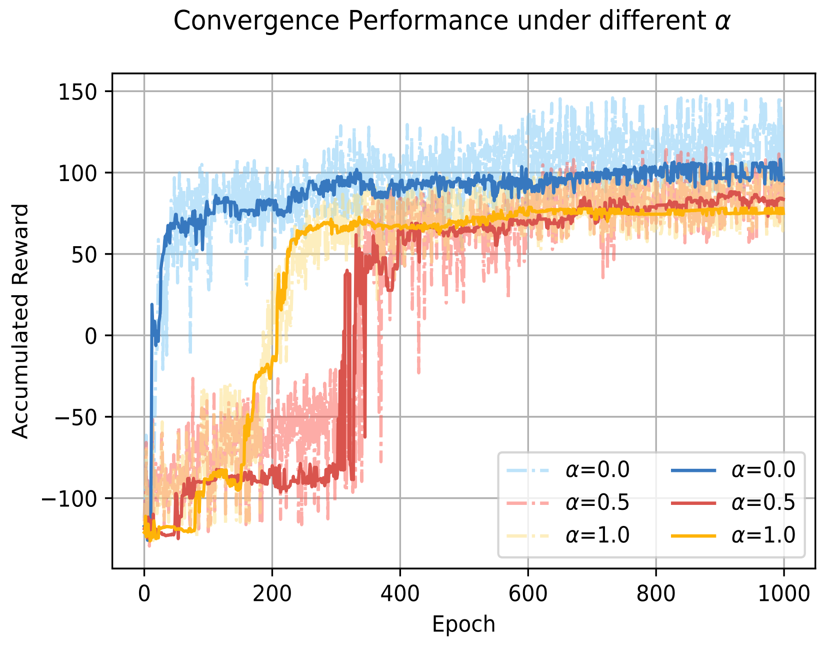
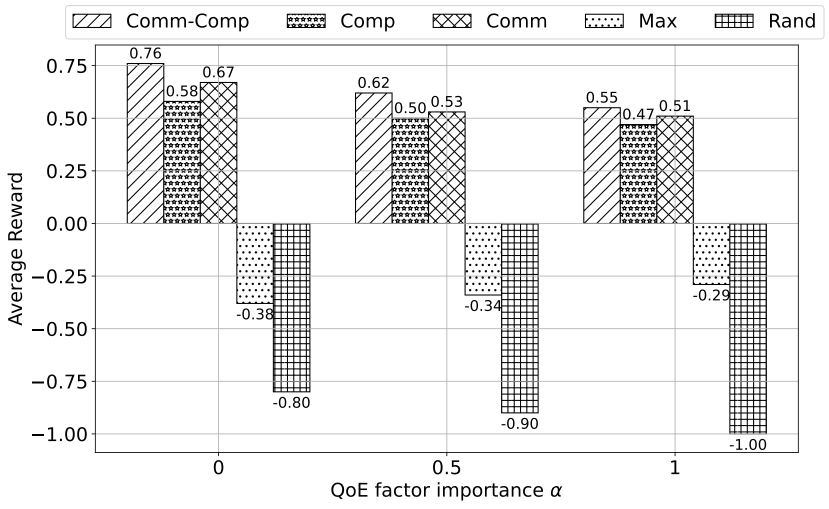
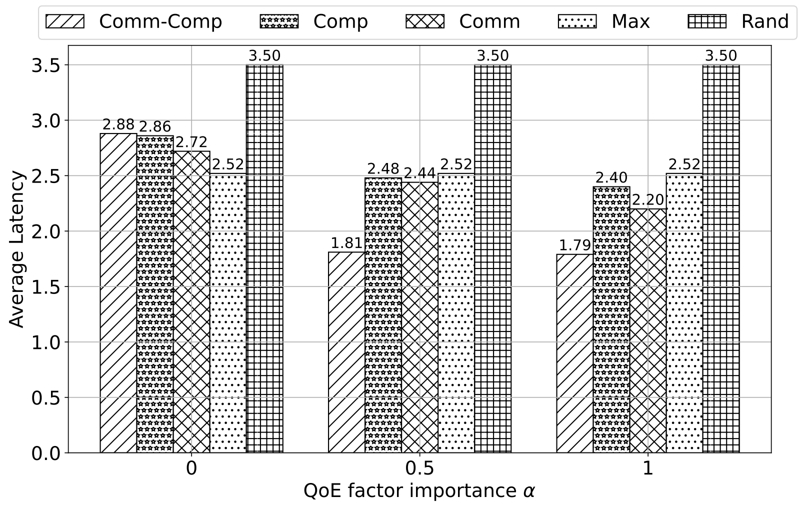


Figure 6.6: Convergence of PPO-based online algorithm

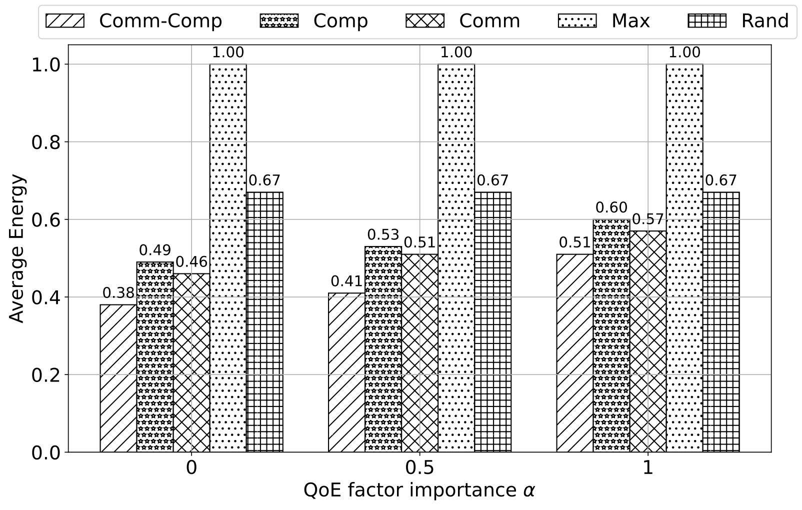
QoE Performance: Figure 6.7 illustrates the average reward, latency, and energy consumption for executing tasks with different QoE requirements on the edge system. Due to their inflexible behaviors, we can obtain that the Max strategy and the Rand strategy always keep the same latency and energy consumption and have a lower QoE performance. Compared to these two strategies, Max strategy has an upward trend with the increase of α, while Rand strategy has an opposite trend. That is because the scheduling objective is more concerned with energy consumption when α equals 0, and turns to latency when α grows to 1. Obviously, our proposed PPO-based online algorithm, which jointly optimizes the communication and computation resources, outperforms communication only optimal strategy or computation only optimal strategy. It can consistently achieve the best QoE performance regardless of the change of α when the agent does not have prior knowledge of the environment dynamics.



(a) Average reward



(b) Average latency



(c) Average energy consumption

Figure 6.7: QoE performance of PPO-based online algorithm

6.6 Chapter Summary

This chapter focused on enhancing device-specific QoE performance in edge-assisted IoT systems by optimizing communication and computing resources jointly. First, a three-layer QoE assessment model was constructed to describe the general relationship between resource provisioning and device-specific QoE performance. Then, to maximize the overall QoE performance amongst IoT devices, a two-stage resource scheduling scheme was proposed to realize simultaneous optimization of IoT devices and the edge system. Specifically, in stage I, a distributed resource scheduling algorithm with low complexity was designed for each IoT device to optimize the local computing rate by considering its resource-constrained nature; in stage II, a PPO-based online approach was proposed on the edge system to schedule communication bandwidth and optimize computational rate by interacting with multiply IoT devices without prior knowledge of their specific QoE assessment models. Finally, extensive experiments demonstrated that our proposal outperforms the existing works from the perspective of QoE performance.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Taking advantage of collaborative computing enabled by cloud computing and edge computing technologies, IoT devices can offload their computation tasks to other idle computing devices or remote servers, thus effectively relieving their computing capability pressure and enhancing the performance of IoT systems. However, realizing collaborative computing through effective resource scheduling remains a severe challenge in edge-assisted IoT systems. The related issues addressed in the thesis are fallen into the following two major aspects: incentive mechanism for horizontal collaboration amongst IoT devices and multi-dimensional resource management for vertical collaboration between the edge server and IoT devices in the edge-assisted IoT systems. More specifically,

- **Incentive mechanism for horizontal collaboration**: IoT devices are not only generators of data but also can be processors of data. Due to different computing capabilities amongst IoT devices, some struggle to cope with computing tasks, while others always remain idle. The horizontal collaboration amongst IoT devices can effectively and efficiently balance the computing tasks in the device layer by scavenging the enormous amount of spare computational resources. Although the concept of collaborative computing is promising, engaging idle computing devices for sharing could be difficult as they have no commitments to do so. They may expect compensation since computation offloading potentially affects local computing tasks. Thus, designing an efficient incentive mechanism for computational resource sharing is essential.

- **Multi-dimensional resource management for vertical collaboration**: As the core of three-tier edge-assisted IoT systems, edge servers have more powerful computing capabilities than the device layer. Utilizing the distributed resources in the edge layer can effectively enhance QoE and QoS performance. However, improving one performance indicator often involves multi-dimensional resources, such as communication resources and computing resources. Orchestrating the multi-dimensional resources to process those data requires appropriate resource scheduling strategies. Furthermore, with the ever-increasing diversity of application requirements, different IoT devices within the same edge network inevitably incur specific resource demands to achieve their diverse application requirements, making the resource scheduling process more complex.

To overcome the issues mentioned above, a number of resource scheduling schemes for collaborative computing in edge-assisted IoT systems have been developed. The contributions that have been made in this thesis and the conclusions drawn from these contributions are summarized as follows:

In chapter 3, considering the collaborative computing scenario in smart buildings, a computation sharing architecture has been proposed to incentivize ICDs to offload computational tasks for the BMS, which combines the Stackelberg game and the Cournot game. To guarantee the utility of BMS and ICDs, the Stackelberg game model has been built to analyze the interactions between BMS and ICDs. Then, the Cournot game model has been presented to formulate the internal competition among multiple ICDs. Under the premise of the subgame perfect Nash equilibrium, the BMS can quote the optimal pricing strategy, and the ICDs can share the corresponding optimal amount of computing resources. The simulation results have demonstrated that the proposed solution can effectively improve the on-demand computing capacity of BMS.

In chapter 4, an incomplete information-based two-tier game model has been estimated to deal with the enormous challenge of real-time data analysis without the complete information in emergency communication networks. IITG can realize collaborative computing by incentivizing ICDs to share computation resources, which jointly combines the Stackelberg game and the Cournot game. Depending on the given information of the EMS and the ICDs, we have analyzed the BNE of the EMS's pricing strategies and the ICDs' computing resource sharing strategies under incomplete information, and further designed the N-IITG algorithm that can iteratively convergent to the unique PBNE. According to the simulation results, the proposed algorithm has achieved a significant increase in computational capacity while each participant obtains the optimal profit.

In chapter 5, a computational latency-based pricing mechanism from the perspective of the QoE performance has been proposed to enable collaborative computing in edge-assisted IoT systems. A game-theoretic computing task allocation approach has been developed among a centralized CSP and multiple ICDs to maximize all participants' profit. The CSP first determines the optimal task partition dynamically upon the task arrival; then, the ICDs derive the optimal central processing unit-cycle frequency correspondingly. Simulation results have shown that the proposed scheme can effectively relieve the pressure of unbalanced computing capabilities in IoT. The overall computational latency of our proposed mechanism has been significantly decreased, and the profit of all participants has achieved the maximum in collaborative computing.

In chapter 6, a three-layer QoE assessment model has been constructed to describe the general relationship between resource provisioning and device-specific QoE performance. Then, to maximize the overall QoE performance amongst IoT devices, a two-stage resource scheduling scheme has been proposed to realize simultaneous optimization of IoT devices and the edge system. Specifically, in stage I, a distributed resource scheduling algorithm with low complexity has been designed for each IoT device to optimize the local computing rate by considering its resource-constrained nature; in stage II, a PPO-based online approach has been proposed for the edge system to schedule communication bandwidth and optimize computational rate by interacting with multiply IoT devices without prior knowledge of their specific QoE assessment models. Finally, extensive experiments have demonstrated that our proposal outperforms the existing works from the perspective of QoE performance.

7.2 Future Work

The technical issues on resource scheduling for horizontal collaboration amongst IoT devices and vertical collaboration between the edge server and IoT devices have been addressed in the thesis, where several incentive and intelligent mechanisms are adopted to improve the QoS and QoE performance from different perspectives. Many other challenges are still required to be investigated and addressed to enhance the performance of the entire edge-assisted IoT systems. The future research directions are identified and summarized in this section, including resource orchestration with cloud computing, data-driven resource scheduling, and privacy and security concerned with resource scheduling. Details of these topics are given as follows:

- **Resource orchestration with cloud computing**: The device-edge collaboration manner has a relatively powerful capacity; however, it ignores the huge computing resources in the cloud computing center. With the ever-increasing smart devices and their resource-hungry applications, it will become increasingly difficult to rely on the resources in the edge layer alone to meet the service requirements of smart devices. Therefore, it is particularly important and necessary to take full advantage of both edge computing and cloud computing and make them complementary to design a collaborative paradigm, i.e., the device-edge-cloud collaboration manner. For example, considering the physical distance, edge servers can focus on the latency-sensitive computing tasks and transfer the computation-intensive and non-urgent tasks to cloud servers. Furthermore, the edge-edge collaboration manner in edge computing does not arise in isolation. Instead, it usually comes along with the device-edge-cloud collaboration manner. As the upper layer of edge servers, the cloud layer can act as a central resource manager to jointly optimize distributed resources in edge servers. However, realizing resource orchestration with the assistance of cloud computing is nontrivial due to many factors. Primarily, due to the heterogeneity, computational servers between remote cloud and local edge nodes significantly differ in terms of their capacity, speed, response time, and energy consumption. Moreover, considering the mobility in edge-edge collaborative computing, computing tasks and resources continuously change between IoT devices, making resource orchestration more complex. Therefore, efficiently utilizing the multi-layer resources in IoT systems becomes crucial to improving the performance of IoT applications.

- **Data-driven resource scheduling**: The ultimate goal of IoT systems is to make decisions automatically through gathering, exchanging, and analyzing data from environments. As the foundation of making decisions, data is crucial to IoT systems, and thus it is essential to be aware of data characteristics during resource scheduling. Currently, most existing studies treat the collected data equally during resource scheduling but ignore the characteristics of the data itself, such as data generation source, data generation time, and data content. For example, the new arrival data generally has more valid information than historical data, or the data generated in an emergency has higher priority than the data in a normal state. Taking such data characteristics into account can improve performance during resource scheduling. Therefore, how to model data characteristics and enable data-driven resource scheduling needs to be considered in future work.

- **Privacy and security concerned with resource scheduling**: Along with more and more data shared by IoT devices, the possibility of having sensitive information exposed is getting extremely high. During the transmission process from IoT devices to remote servers, a series of unexpected leakages of sensitive data, e.g., user location, personal picture, etc., often inevitably incur. For example, suppose a user generates one request exceeding its computation capability through collaborative computing. In that case, privacy disclosure is inevitable as we cannot assume all the involved IoT devices and remote servers could be trusted. Thus, privacy preservation is necessary to prevent privacy leakage from data transition among multiple entities in IoT systems. On the other hand, excessive privacy protection may lead to inefficient utilization of the distributed and scattered resources in IoT systems, thus degrading the performance of collaborative computing. All problems mentioned above pose a great challenge to achieving an optimal trade-off between privacy protection and resource scheduling, which can be further investigated.

Bibliography

[1] Kevin Ashton. That internet of things thing. `https://www.rfidjournal.com/that-internet-of-things-thing`, 1999.

[2] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE communications surveys & tutorials*, 17(4):2347–2376, 2015.

[3] Jie Lin, Wei Yu, Nan Zhang, Xinyu Yang, Hanlin Zhang, and Wei Zhao. A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet of Things Journal*, 4(5):1125–1142, 2017.

[4] Worldwide global datasphere iot device and data forecast, 2019-2023. Technical Report IDC # 146187419, IDC, April 2020.

[5] Introducing google app engine + our new blog. `http://googleappengine.blogspot.com/2008/04/introducing-google-app-engine-our-new.html`, 2008.

[6] Maria Stoyanova, Yannis Nikoloudakis, Spyridon Panagiotakis, Evangelos Pallis, and Evangelos K. Markakis. A survey on the internet of things (iot) forensics: Challenges, approaches, and open issues. *IEEE Communications Surveys Tutorials*, 22(2):1191–1221, 2020.

[7] Mahadev Satyanarayanan. The emergence of edge computing. *Computer*, 50(1):30–39, 2017.

[8] Nasir Abbas, Yan Zhang, Amir Taherkordi, and Tor Skeie. Mobile edge computing: A survey. *IEEE Internet of Things Journal*, 5(1):450–465, 2018.

[9] Quyuan Luo, Shihong Hu, Changle Li, Guanghui Li, and Weisong Shi. Resource scheduling in edge computing: A survey. *IEEE Communications Surveys & Tutorials*, 23(4):2131–2165, 2021.

[10] Dianlei Xu, Yong Li, Xinlei Chen, Jianbo Li, Pan Hui, Sheng Chen, and Jon Crowcroft. A survey of opportunistic offloading. *IEEE Communications Surveys & Tutorials*, 20(3):2198–2236, 2018.

[11] Mamta Agiwal, Abhishek Roy, and Navrati Saxena. Next generation 5g wireless networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 18(3):1617–1655, 2016.

[12] Dinh C. Nguyen, Ming Ding, Pubudu N. Pathirana, Aruna Seneviratne, Jun Li, Dusit Niyato, Octavia Dobre, and H. Vincent Poor. 6g internet of things: A comprehensive survey. *IEEE Internet of Things Journal*, 9(1):359–383, 2022.

[13] Adam Zielonka, Andrzej Sikora, Marcin Woźniak, Wei Wei, Qiao Ke, and Zongwen Bai. Intelligent internet of things system for smart home optimal convection. *IEEE Transactions on Industrial Informatics*, 17(6):4308–4317, 2021.

[14] Jun Zhang and Khaled B. Letaief. Mobile edge intelligence and computing for the internet of vehicles. *Proceedings of the IEEE*, 108(2):246–261, 2020.

[15] S. M. Riazul Islam, Daehan Kwak, MD. Humaun Kabir, Mahmud Hossain, and Kyung-Sup Kwak. The internet of things for health care: A comprehensive survey. *IEEE Access*, 3:678–708, 2015.

[16] Yazdan Ahmad Qadri, Ali Nauman, Yousaf Bin Zikria, Athanasios V. Vasilakos, and Sung Won Kim. The future of healthcare internet of things: A survey of emerging technologies. *IEEE Communications Surveys Tutorials*, 22(2):1121–1167, 2020.

[17] Martin Wollschlaeger, Thilo Sauter, and Juergen Jasperneite. The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0. *IEEE Industrial Electronics Magazine*, 11(1):17–27, 2017.

[18] Wenliang Mao, Zhiwei Zhao, Zheng Chang, Geyong Min, and Weifeng Gao. Energy-efficient industrial internet of things: Overview and open issues. *IEEE Transactions on Industrial Informatics*, 17(11):7225–7237, 2021.

[19] Pavel Mach and Zdenek Becvar. Mobile edge computing: A survey on architecture and computation offloading. *IEEE Communications Surveys Tutorials*, 19(3):1628–1656, 2017.

[20] ITUT Recommendation. E. 800: Terms and definitions related to quality of service and network performance including dependability. *ITU-T August*, 1994, 1994.

[21] ITUT Rec. P. 10: Vocabulary for performance and quality of service, amendment 2: New definitions for inclusion in recommendation itu-t p. 10/g. 100. *Int. Telecomm. Union, Geneva*, page 10, 2008.

[22] Ulrich Reiter, Kjell Brunnström, Katrien De Moor, Mohamed-Chaker Larabi, Manuela Pereira, Antonio Pinheiro, Junyong You, and Andrej Zgank. Factors influencing quality of experience. In *Quality of experience*, pages 55–72. Springer, 2014.

[23] Roger B Myerson. *Game theory: analysis of conflict*. Harvard university press, 1997.

[24] Hal R Varian. *Intermediate microeconomics: a modern approach: ninth international student edition*. WW Norton & Company, 2014.

[25] Nguyen Cong Luong, Dinh Thai Hoang, Ping Wang, Dusit Niyato, Dong In Kim, and Zhu Han. Data collection and wireless communication in internet of things (iot) using economic analysis and pricing models: A survey. *IEEE Communications Surveys & Tutorials*, 18(4):2546–2590, 2016.

[26] Nguyen Cong Luong, Ping Wang, Dusit Niyato, Yonggang Wen, and Zhu Han. Resource management in cloud networking using economic analysis and pricing models: A survey. *IEEE Communications Surveys Tutorials*, 19(2):954–1001, 2017.

[27] José Moura and David Hutchison. Game theory for multi-access edge computing: Survey, use cases, and future trends. *IEEE Communications Surveys & Tutorials*, 21(1):260–288, 2019.

[28] Chuanxiu Chi, Yingjie Wang, Xiangrong Tong, Madhuri Siddula, and Zhipeng Cai. Game theory in internet of things: A survey. *IEEE Internet of Things Journal*, pages 1–1, 2021.

[29] Marwaan Simaan and Jose B Cruz. On the stackelberg strategy in nonzero-sum games. *Journal of Optimization Theory and Applications*, 11(5):533–555, 1973.

[30] Irving Fisher. Cournot and mathematical economics. *The Quarterly Journal of Economics*, 12(2):119–138, 1898.

[31] John C Harsanyi. Games with incomplete information played by bayesian players, i–iii part i. the basic model. *Management science*, 14(3):159–182, 1967.

[32] Nguyen Cong Luong, Dinh Thai Hoang, Shimin Gong, Dusit Niyato, Ping Wang, Ying-Chang Liang, and Dong In Kim. Applications of deep reinforcement learning in communications and networking: A survey. *IEEE Communications Surveys Tutorials*, 21(4):3133–3174, 2019.

[33] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

[34] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[35] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[36] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[37] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[38] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[39] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[40] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[41] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[42] Towards a definition of the internet of things (iot). Technical Report `http://iot.ieee.org/definition.html`, IEEE, 2015.

[43] Muhammad Rizwan Bashir and Asif Qumer Gill. Iot enabled smart buildings: A systematic review. In *2017 Intelligent Systems Conference (IntelliSys)*, pages 151–159. IEEE, 2017.

[44] Daniel Minoli, Kazem Sohraby, and Benedict Occhiogrosso. Iot considerations, requirements, and architectures for smart buildingsłenergy optimization and next-generation building management systems. *IEEE Internet of Things Journal*, 4(1):269–283, 2017.

[45] Brian Ramprasad, Jenn McArthur, Marios Fokaefs, Cornel Barna, Mark Damm, and Marin Litoiu. Leveraging existing sensor networks as iot devices for smart buildings. In *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, pages 452–457. IEEE, 2018.

[46] Pouya Jamborsalamati, Edstan Fernandez, MJ Hossain, and FHM Rafi. Design and implementation of a cloud-based iot platform for data acquisition and device supply management in smart buildings. In *2017 Australasian Universities Power Engineering Conference (AUPEC)*, pages 1–6. IEEE, 2017.

[47] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646, 2016.

[48] Joy Dutta and Sarbani Roy. Iot-fog-cloud based architecture for smart city: Prototype of a smart building. In *2017 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence*, pages 237–242. IEEE, 2017.

[49] Juho Hamari, Mimmi Sjöklint, and Antti Ukkonen. The sharing economy: Why people participate in collaborative consumption. *Journal of the association for information science and technology*, 67(9):2047–2059, 2016.

[50] Chengzhi Lu, Kejiang Ye, Guoyao Xu, Cheng-Zhong Xu, and Tongxin Bai. Imbalance in the cloud: An analysis on alibaba cluster trace. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 2884–2892. IEEE, 2017.

[51] Jianli Pan, Raj Jain, Subharthi Paul, Tam Vu, Abusayeed Saifullah, and Mo Sha. An internet of things framework for smart energy in buildings: designs, prototype, and experiments. *IEEE internet of things journal*, 2(6):527–537, 2015.

[52] Francisco-Javier Ferrández-Pastor, Higinio Mora, Antonio Jimeno-Morenilla, and Bruno Volckaert. Deployment of iot edge and fog computing technologies to develop smart building services. *Sustainability*, 10(11):3832, 2018.

[53] Minghui Min, Liang Xiao, Ye Chen, Peng Cheng, Di Wu, and Weihua Zhuang. Learning-based computation offloading for iot devices with energy harvesting. *IEEE Transactions on Vehicular Technology*, 68(2):1930–1941, 2019.

[54] Yixue Hao, Yiming Miao, Long Hu, M Shamim Hossain, Ghulam Muhammad, and Syed Umar Amin. Smart-edge-cocaco: Ai-enabled smart edge with joint computation, caching, and communication in heterogeneous iot. *IEEE Network*, 33(2):58–64, 2019.

[55] Liang Xiao, Tianhua Chen, Caixia Xie, Huaiyu Dai, and H Vincent Poor. Mobile crowdsensing games in vehicular networks. *IEEE Transactions on Vehicular Technology*, 67(2):1535–1545, 2018.

[56] Mingmei Li, Tony QS Quek, and Costas Courcoubetis. Mobile data offloading with uniform pricing and overlaps. *IEEE Transactions on Mobile Computing*, 18(2):348–361, 2019.

[57] Huaqing Zhang, Yong Xiao, Shengrong Bu, Dusit Niyato, F Richard Yu, and Zhu Han. Computing resource allocation in three-tier iot fog networks: A joint optimization approach combining stackelberg game and matching. *IEEE Internet of Things Journal*, 4(5):1204–1215, 2017.

[58] Mengyu Liu and Yuan Liu. Price-based distributed offloading for mobile-edge computing with computation capacity constraints. *IEEE Wireless Communications Letters*, 7(3):420–423, 2018.

[59] Sla̅dana Jošilo and György Dán. Wireless and computing resource allocation for selfish computation offloading in edge computing. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 2467–2475. IEEE, 2019.

[60] Xiuli He, Ashutosh Prasad, Suresh P Sethi, and Genaro J Gutierrez. A survey of stackelberg differential game models in supply and marketing channels. *Journal of Systems Science and Systems Engineering*, 16(4):385–413, 2007.

[61] Fei Sun, Fen Hou, Haibo Zhou, Bo Liu, Jiacheng Chen, and Lin Gui. Equilibriums in the mobile-virtual-network-operator-oriented data offloading. *IEEE Transactions on Vehicular Technology*, 67(2):1622–1634, 2018.

[62] Hamed Kebriaei, Ashkan Rahimi-Kian, and Majid Nili Ahmadabadi. Model-based and learning-based decision making in incomplete information cournot games: a state estimation approach. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(4):713–718, 2015.

[63] Thomas D Burd and Robert W Brodersen. Processor design for portable systems. *Journal of VLSI signal processing systems for signal, image and video technology*, 13(2):203–221, 1996.

[64] Robin R Murphy. Emergency informatics: Using computing to improve disaster management. *Computer*, 49(5):19–27, 2016.

[65] Evangelos K Markakis, Ilias Politis, Asimakis Lykourgiotis, Yacine Rebahi, George Mastorakis, Constandinos X Mavromoustakis, and Evangelos Pallis. Efficient next generation emergency communications over multi-access edge computing. *IEEE Communications Magazine*, 55(11):92–97, 2017.

[66] Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B Letaief. A survey on mobile edge computing: The communication perspective. *IEEE communications surveys & tutorials*, 19(4):2322–2358, 2017.

[67] Nguyen Cong Luong, Ping Wang, Dusit Niyato, Ying-Chang Liang, Zhu Han, and Fen Hou. Applications of economic and pricing models for resource management in 5g wireless networks: A survey. *IEEE Communications Surveys & Tutorials*, 21(4):3298–3339, 2019.

[68] He Fang, Li Xu, and Xianbin Wang. Coordinated multiple-relays based physical-layer security improvement: A single-leader multiple-followers stackelberg game scheme. *IEEE transactions on information forensics and security*, 13(1):197–209, 2018.

[69] Lingjie Duan, Takeshi Kubo, Kohei Sugiyama, Jianwei Huang, Teruyuki Hasegawa, and Jean Walrand. Motivating smartphone collaboration in data acquisition and distributed computing. *IEEE Transactions on Mobile Computing*, 13(10):2320–2333, 2014.

[70] Jiangtian Nie, Jun Luo, Zehui Xiong, Dusit Niyato, and Ping Wang. A stackelberg game approach toward socially-aware incentive mechanisms for mobile crowdsensing. *IEEE Transactions on Wireless Communications*, 18(1):724–738, 2019.

[71] Chuanyin Li, Jiandong Li, Yuzhou Li, and Zhu Han. Pricing game with complete or incomplete information about spectrum inventories for mobile virtual network operators. *IEEE Transactions on Vehicular Technology*, 68(11):11118–11131, 2019.

[72] Godfrey Anuga Akpakwu, Bruno J. Silva, Gerhard P. Hancke, and Adnan M. Abu-Mahfouz. A survey on 5g networks for the internet of things: Communication technologies and challenges. *IEEE Access*, 6:3619–3647, 2018.

[73] Lalit Chettri and Rabindranath Bera. A comprehensive survey on internet of things (iot) toward 5g wireless systems. *IEEE Internet of Things Journal*, 7(1):16–32, 2020.

[74] Philipp Schulz, Maximilian Matthe, Henrik Klessig, Meryem Simsek, Gerhard Fettweis, Junaid Ansari, Shehzad Ali Ashraf, Bjoern Almeroth, Jens Voigt, Ines Riedel, et al. Latency critical iot applications in 5g: Perspective on the design of radio interface and network architecture. *IEEE Communications Magazine*, 55(2):70–78, 2017.

[75] Mec in 5g network. Technical Report `https://www.etsi.org/technologies-clusters/white-papers-and-brochures/etsi-white-papers`, ETSI, June 2018.

[76] Sukhpal Singh Gill and Rajkumar Buyya. A taxonomy and future directions for sustainable cloud computing: 360 degree view. *ACM Computing Surveys (CSUR)*, 51(5):1–33, 2018.

[77] Sambit Kumar Mishra, Deepak Puthal, Bibhudatta Sahoo, Prem Prakash Jayaraman, Song Jun, Albert Y Zomaya, and Rajiv Ranjan. Energy-efficient vm-placement in cloud data center. *Sustainable computing: informatics and systems*, 20:48–55, 2018.

[78] Changsheng You and Kaibin Huang. Exploiting non-causal cpu-state information for energy-efficient mobile cooperative computing. *IEEE Transactions on Wireless Communications*, 17(6):4104–4117, 2018.

[79] Zhengguo Sheng, Chinmaya Mahapatra, Victor C. M. Leung, Min Chen, and Pratap Kumar Sahu. Energy efficient cooperative computing in mobile wireless sensor networks. *IEEE Transactions on Cloud Computing*, 6(1):114–126, 2018.

[80] Soumya Sen, Carlee Joe-Wong, Sangtae Ha, and Mung Chiang. A survey of smart data pricing: Past proposals, current plans, and future trends. *Acm computing surveys (csur)*, 46(2):1–37, 2013.

[81] Mohammad Aazam, Khaled A Harras, and Sherali Zeadally. Fog computing for 5g tactile industrial internet of things: Qoe-aware resource allocation model. *IEEE Transactions on Industrial Informatics*, 15(5):3085–3092, 2019.

[82] Wei Wei, Xunli Fan, Houbing Song, Xiumei Fan, and Jiachen Yang. Imperfect information dynamic stackelberg game based resource allocation using hidden markov for cloud computing. *IEEE transactions on services computing*, 11(1):78–89, 2018.

[83] Xiumin Wang, Xiaoming Chen, Weiwei Wu, Ning An, and Lusheng Wang. Cooperative application execution in mobile cloud computing: A stackelberg game approach. *IEEE Communications Letters*, 20(5):946–949, 2016.

[84] Thinh Quang Dinh, Jianhua Tang, Quang Duy La, and Tony QS Quek. Offloading in mobile edge computing: Task allocation and computational frequency scaling. *IEEE Transactions on Communications*, 65(8):3571–3584, 2017.

[85] Songtao Guo, Jiadi Liu, Yuanyuan Yang, Bin Xiao, and Zhetao Li. Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing. *IEEE Transactions on Mobile Computing*, 18(2):319–333, 2019.

[86] Dusit Niyato, Dinh Thai Hoang, Nguyen Cong Luong, Ping Wang, Dong In Kim, and Zhu Han. Smart data pricing models for the internet of things: a bundling strategy approach. *IEEE Network*, 30(2):18–25, 2016.

[87] Amazon ec2 pricing. `https://aws.amazon.com/ec2/pricing/`, 02.

[88] Microsoft azure windows virtual machines pricing. `https://azure.microsoft.com/en-ca/pricing/details/virtual-machines/windows/`, 02.

[89] Google compute engine pricing. `https://cloud.google.com/compute/pricing`, 02.

[90] Yeongjin Kim, Jeongho Kwak, and Song Chong. Dual-side optimization for cost-delay tradeoff in mobile edge computing. *IEEE Transactions on Vehicular Technology*, 67(2):1765–1781, 2018.

[91] Hamed Shah-Mansouri, Vincent WS Wong, and Robert Schober. Joint optimal pricing and task scheduling in mobile cloud computing systems. *IEEE Transactions on Wireless Communications*, 16(8):5218–5232, 2017.

[92] Ying Wang, Peilong Li, Lei Jiao, Zhou Su, Nan Cheng, Xuemin Sherman Shen, and Ping Zhang. A data-driven architecture for personalized qoe management in 5g wireless networks. *IEEE Wireless Communications*, 24(1):102–110, 2017.

[93] Chenmeng Wang, Chengchao Liang, F Richard Yu, Qianbin Chen, and Lun Tang. Computation offloading and resource allocation in wireless cellular networks with mobile edge computing. *IEEE Transactions on Wireless Communications*, 16(8):4924–4938, 2017.

[94] Zhaolong Ning, Peiran Dong, Xiangjie Kong, and Feng Xia. A cooperative partial computation offloading scheme for mobile edge computing enabled internet of things. *IEEE Internet of Things Journal*, 6(3):4804–4814, 2019.

[95] Jeongho Kwak, Okyoung Choi, Song Chong, and Prasant Mohapatra. Processor-network speed scaling for energy–delay tradeoff in smartphone applications. *IEEE/ACM Transactions on Networking*, 24(3):1647–1660, 2016.

[96] Yongmin Zhang, Xiaolong Lan, Ju Ren, and Lin Cai. Efficient computing resource sharing for mobile edge-cloud computing networks. *IEEE/ACM Transactions on Networking*, 28(3):1227–1240, 2020.

[97] Xiaowen Cao, Feng Wang, Jie Xu, Rui Zhang, and Shuguang Cui. Joint computation and communication cooperation for energy-efficient mobile edge computing. *IEEE Internet of Things Journal*, 6(3):4188–4200, 2019.

[98] Ying Chen, Ning Zhang, Yongchao Zhang, Xin Chen, Wen Wu, and Xuemin Sherman Shen. Toffee: Task offloading and frequency scaling for energy efficiency of mobile devices in mobile edge computing. *IEEE Transactions on Cloud Computing*, 9(4):1634–1644, 2021.

[99] Yue Wang, Xiaofeng Tao, Xuefei Zhang, Ping Zhang, and Y Thomas Hou. Cooperative task offloading in three-tier mobile computing networks: An admm framework. *IEEE Transactions on Vehicular Technology*, 68(3):2763–2776, 2019.

[100] Xiao Ma, Ao Zhou, Shan Zhang, and Shangguang Wang. Cooperative service caching and workload scheduling in mobile edge computing. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 2076–2085. IEEE, 2020.

[101] Sarhad Arisdakessian, Omar Abdel Wahab, Azzam Mourad, Hadi Otrok, and Nadjia Kara. Fogmatch: an intelligent multi-criteria iot-fog scheduling approach using game theory. *IEEE/ACM Transactions on Networking*, 28(4):1779–1789, 2020.

[102] Qianqian Wang, Yongxu Zhu, and Xianbin Wang. Incomplete information based collaborative computing in emergency communication networks. *IEEE Communications Letters*, 24(9):2038–2042, 2020.

[103] Junxu Xia, Geyao Cheng, Deke Guo, and Xiaolei Zhou. A qoe-aware service-enhancement strategy for edge artificial intelligence applications. *IEEE Internet of Things Journal*, 7(10):9494–9506, 2020.

[104] Haodong Lu, Xiaoming He, Miao Du, Xiukai Ruan, Yanfei Sun, and Kun Wang. Edge qoe: Computation offloading with deep reinforcement learning for internet of things. *IEEE Internet of Things Journal*, 7(10):9255–9265, 2020.

[105] Fangxin Wang, Cong Zhang, Feng Wang, Jiangchuan Liu, Yifei Zhu, Haitian Pang, and Lifeng Sun. Deepcast: Towards personalized qoe for edge-assisted crowdcast with deep reinforcement learning. *IEEE/ACM Transactions on Networking*, 28(3):1255–1268, 2020.

[106] Xiaoming He, Haodong Lu, Miao Du, Yingchi Mao, and Kun Wang. Qoe-based task offloading with deep reinforcement learning in edge-enabled internet of vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 22(4):2252–2261, 2021.

[107] Songyuan Li, Jiwei Huang, Jia Hu, and Bo Cheng. Qoe-deer: A qoe-aware decentralized resource allocation scheme for edge computing. *IEEE Transactions on Cognitive Communications and Networking*, pages 1–1, 2021.

[108] Xiaoteng Ma, Qing Li, Yong Jiang, Gabriel-Miro Muntean, and Longhao Zou. Learning-based joint qoe optimization for adaptive video streaming based on smart edge. *IEEE Transactions on Network and Service Management*, pages 1–1, 2022.

[109] Yaqiong Liu, Mugen Peng, Guochu Shou, Yudong Chen, and Siyu Chen. Toward edge intelligence: multiaccess edge computing for 5g and internet of things. *IEEE Internet of Things Journal*, 7(8):6722–6747, 2020.

[110] Fatima Hussain, Syed Ali Hassan, Rasheed Hussain, and Ekram Hossain. Machine learning for resource management in cellular and iot networks: Potentials, current solutions, and open challenges. *IEEE Communications Surveys & Tutorials*, 22(2):1251–1275, 2020.

[111] Yi Liu, Huimin Yu, Shengli Xie, and Yan Zhang. Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks. *IEEE Transactions on Vehicular Technology*, 68(11):11158–11168, 2019.

[112] Liang Huang, Suzhi Bi, and Ying-Jun Angela Zhang. Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks. *IEEE Transactions on Mobile Computing*, 19(11):2581–2593, 2020.

[113] Xiong Xiong, Kan Zheng, Lei Lei, and Lu Hou. Resource allocation based on deep reinforcement learning in iot edge computing. *IEEE Journal on Selected Areas in Communications*, 38(6):1133–1146, 2020.

[114] Yufeng Zhan, Song Guo, Peng Li, and Jiang Zhang. A deep reinforcement learning based offloading game in edge computing. *IEEE Transactions on Computers*, 69(6):883–893, 2020.

[115] Jiangliang Jin and Yunjian Xu. Optimal policy characterization enhanced proximal policy optimization for multitask scheduling in cloud computing. *IEEE Internet of Things Journal*, 9(9):6418–6433, 2022.

[116] Wenhan Zhan, Chunbo Luo, Jin Wang, Chao Wang, Geyong Min, Hancong Duan, and Qingxin Zhu. Deep-reinforcement-learning-based offloading scheduling for vehicular edge computing. *IEEE Internet of Things Journal*, 7(6):5449–5465, 2020.

[117] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[118] Abdurauf Khamosh, Mohammad Anwer Anwer, Nasratullah Nasrat, Javid Hamdard, Gawhar Shah Gawhari, and Abdul Rahim Ahmadi. Impact of network qos factors on qoe of iot services. In *2020 - 5th International Conference on Information Technology (InCIT)*, pages 61–65, 2020.

[119] James Nightingale, Pablo Salva-Garcia, Jose M. Alcaraz Calero, and Qi Wang. 5g-qoe: Qoe modelling for ultra-hd video streaming in 5g networks. *IEEE Transactions on Broadcasting*, 64(2):621–634, 2018.

[120] Mohammad Aazam and Khaled A. Harras. Mapping qoe with resource estimation in iot. In *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, pages 464–467, 2019.

Curriculum Vitae

Name:	Qianqian Wang
Post-Secondary Education and Degrees:	2020 - present, Ph.D. Electrical and Computer Engineering The University of Western Ontario London, Ontario, Canada
	2016 - 2020, Ph.D. Communication and Information System Nanjing University of Posts and Telecommunications Nanjing, China
	2004 - 2007, M.Eng. Computer and its Application Nanjing University of Posts and Telecommunications Nanjing, China
	2000 - 2004, B.Eng. Computer Science and Technology Nanjing University of Posts and Telecommunications Nanjing, China
Related Working Experience	Lecture Jinling Institute of Technology Nanjing, China 2013 - present
	Software Engineer Motorola (China) Electronics Ltd Nanjing Branch Nanjing, China 2007 - 2012

Publications:

Journal:

[1] Q. Wang, H. Zhao, Q. Wang, H. Cao, G. S. Aujla, and H. Zhu, "Enabling secure wireless multimedia resource pricing using consortium blockchains," in *Future Generation Computer Systems*, vol. 110, pp. 696-707, September 2019.

[2] Q. Wang, Q. Wang, S. Jin, H. Zhu and X. Wang, "A hierarchical game model for computation sharing in smart buildings," in *China Communications*, vol. 17, no. 3, pp. 188-204, March 2020.

[3] Q. Wang, Y. Zhu and X. Wang, "Incomplete Information Based Collaborative Computing in Emergency Communication Networks," in *IEEE Communications Letters*, vol. 24, no. 9, pp. 2038-2042, September 2020.

[4] Q. Wang, Q. Wang, H. Zhu and X. Wang, "Enabling Collaborative Computing Sustainably Through Computational Latency-Based Pricing," in *IEEE Transactions on Sustainable Computing*, vol. 5, no. 4, pp. 541-551, October - December 2020.

[5] Q. Wang, L. Chen, Q. Wang, H. Zhu and X. Wang, "Anomaly-Aware Network Traffic Estimation via Outlier-Robust Tensor Completion," in *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2677-2689, December 2020.

[6] Q. Wang, Q. Wang, H. Zhu and X. Wang, "Device-Specific QoE Enhancement through Joint Communication and Computation Resource Scheduling in Edge-Assisted IoT Systems," (submitted to *IEEE/ACM Transactions on Networking*).

[7] Q. Wang, L. Chen, H. Zhu and X. Wang, "Collaborative Network Traffic Estimation with Anomaly Detection via Noise-immune Low-rank Tensor Completion," (submitted to *IEEE/ACM Transactions on Networking*).