UNIVERSITY OF TECHNOLOGY SYDNEY
Faculty of Engineering and Information Technology

Mobile Edge Computing for Future Internet-of-Things

by

Chenshan Ren

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

Doctor of Philosophy

Sydney, Australia

2020

Certificate of Authorship/Originality

I, Chenshan Ren, declare that this thesis, is submitted in fulfilment of the requirements for the award of doctor of philosophy, in the Faculty of Engineering and Information Technology at the University of Technology Sydney. This thesis is wholly my own work unless otherwise reference or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis. I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of the requirements for a degree except as fully acknowledged within the text. This thesis is the result of a research candidature jointly delivered with Beijing University of Posts and Telecommunications as part of a Collaborative Doctoral Research Degree. This research is supported by the Australian Government Research Training Program.

Signature:

Date: 02/07/2020

ABSTRACT

Mobile Edge Computing for Future Internet-of-Things

by

Chenshan Ren

Integrating sensors, the Internet, and wireless systems, Internet-of-Things (IoT) provides a new paradigm of ubiquitous connectivity and pervasive intelligence. The key enabling technology underlying IoT is mobile edge computing (MEC), which is anticipated to realize and reap the promising benefits of IoT applications by placing various cloud resources, such as computing and storage resources closer to smart devices and objects. Challenges of designing efficient and scalable MEC platforms for future IoT arise from the physical limitations of computing and battery resources of IoT devices, heterogeneity of computing and wireless communication capabilities of IoT networks, large volume of data arrivals and massive number connections, and large-scale data storage and delivery across the edge network. To address these challenges, this thesis proposes four efficient and scalable task offloading and cooperative caching approaches are proposed.

Firstly, for the multi-user single-cell MEC scenario, the base station (BS) can only have outdated knowledge of IoT device channel conditions due to the time-varying nature of practical wireless channels. To this end, a hybrid learning approach is proposed to optimize the real-time local processing and predictive computation offloading decisions in a distributed manner.

Secondly, for the multi-user multi-cell MEC scenario, an energy-efficient resource management approach is developed based on distributed online learning to tackle the heterogeneity of computing and wireless transmission capabilities of edge servers and IoT devices. The proposed approach optimizes the decisions on task offloading, processing, and result delivery between edge servers and IoT devices to minimize the time-average energy consumption of MEC.

Thirdly, for the computing resource allocation under large-scale network, a distributed online collaborative computing approach is proposed based on Lyapunov optimization for data analysis in IoT application to minimize the time-average energy consumption of network.

Finally, for the storage resource allocation under large-scale network, a distributed IoT data delivery approach based on online learning is proposed for caching application in mobile applications. A new profitable cooperative region is established for every IoT data request admitted at an edge server, to avoid invalid request dispatching.

Acknowledgements

First, I would like to thank my principal supervisor, Prof. Ren Ping Liu, for his guidance, encouragement, enthusiasm, and technical comments through all years of my research. I also would like to express my deep gratitude to my co-supervisors, Dr. Wei Ni and Prof. Eryk Dutkiewicz, for their valuable advice and guidance of my research.

Special thanks to Beijing University of Posts and Telecommunications (BUP-T), Beijing, China, and University of Technology Sydney (UTS) for providing a scholarship and other financial support for my study.

To my family, thank dad and mom for your love and encouragement to help me complete this study. I have been incredibly fortunate to have love and support from such family. Finally, to my husband, thank you for encouraging, enduring and inspiring me through the years. Love you always.

Chenshan Ren
Sydney, Australia, 2020.

List of Publications

Journal Papers

J-1 . **C. Ren, X. Lyu, W. Ni, H. Tian, R. P. Liu.** *"Distributed Online Learning of Fog Computing under Non-uniform Device Cardinality" in* ***IEEE Internet of Things Journal,*** vol. 6, no. 1, pp. 1147-1159, Feb. 2019.

J-1 . **C. Ren, X. Lyu, W. Ni, H. Tian, R. P. Liu.** *"Profitable Cooperative Region for Distributed Oline Edge Caching" in* ***IEEE Transactions on Communications,*** vol. 67, no. 7, pp. 4696-4078, Jul. 2019.

J-3 . **X. Lyu, C. Ren, W. Ni, H. Tian, R. P. Liu, Y. Jay Guo.** *"Multi-timescale Decentralized Online Orchestration of Software-Defined Networks" in* ***IEEE Journal on Selected Areas in Communications,*** vol. 36, no. 12, pp. 2716-2730, Dec. 2018.

J-4 . **X. Lyu, C. Ren, W. Ni, H. Tian, R. P. Liu.** *"Distributed Optimization of Collaborative Regions in Large-Scale Inhomogeneous Fog Computing" in* ***IEEE Journal on Selected Areas in Communications,*** vol. 36, no. 3, pp. 574-586, March 2018.

J-5 . **X. Lyu, C. Ren, W. Ni, H. Tian, R. P. Liu, E. Dutkiewicz.** *"Optimal Online Data Partitioning for Geo-Distributed Machine Learning in Edge of Wireless Networks" in* ***IEEE Journal on Selected Areas in Communications,*** DOI: 10.1109/JSAC.2019.2934002.

Contents

List of Figures

Abbreviation

Internet of Things - IoT

Mobile Edge Computing - MEC

Mobile Edge Computation Offloading - MECO

Quality of service - QoS

Small Cell Management entity - SCM

Mobile Micro Clouds - MMC

Fast Moving Personal Cloud - MobiScud

Follow Me Cloud - FMC

Radio Access Network - RAN

Online Convex Optimization- OCO

Stochastic Gradient Descent- SGD

Mixed integer programming - MIP

Dynamic programming- DP

Time-division multiple access - TDMA

Quality of experience - QoE

Base station - BS

Long-term evolution - LTE

Mixed integer non-linear programming - MINLP

KarushKuhnTucker - KKT

Cumulative distribution function - CDF

Integer programming - IP

Linear programming - LP

Independent and identical distributed - i.i.d.

Round robin	RR
Proportional fair	PF
Device-to-Device	D2D
Precision time protocol	PTP
First-in-first-out	FIFO
Left-hand-side	LHS
Right-hand-side	RHS
Neighbor discovery protocol	NDP
Head-of-line	HOL

Chapter 1

Introduction

1.1 Background

1.1.1 Future Internet-of-Things (IoT)

In the past decades, the Internet has evolved from peer-to-peer networking to world-wide-web, and mobile-Internet to the Internet-of-Things (IoT) [1]. The idea of IoT is originated from Radio Frequency Identification (RFID) tags, each of which has a distinct address to communicate with other devices/tags and connect to the Internet [5]. With the rapid development of wireless technologies, Near Field communications (NFC), Machine to Machine (M2M) communications, and Vehicular to Vehicular (V2V) communications are able to facilitate the implementation of IoT devices [5].

IoT can bridge the gap between the Internet and traditional telecommunication networks, enable smart objects to connect to the Internet, and realize ubiquitous and pervasive intelligence. By 2020, the number of IoT devices will exceed 30 billion [6], and up to 45% of the Internet traffic are generated from such large number of IoT devices [7]. The applications of future IoT can be listed as follows.

- *Healthcare.* Wearable low-power IoT/medical sensors can monitor health-related data and keep records. The e-health applications (e.g., remote surgeries) would require ultra-low latency and uninterrupted communication links. The collaborations among surgeons can be conducted in different geographical locations [1, 8–12].

- *Autonomous driving.* The concept of (semi-)autonomous driving is based on Vehicle to Everything (V2X), converging the technologies of Vehicle to Vehicle (V2V), vehicle to infrastructure (V2I), vehicle to device, vehicle to pedestrian, vehicle to home, and vehicle to grid. The application of autonomous driving requires high-reliable and mission-critical computing services [13].

- *Industry manufacturing.* In the industrial IoT (IIoT) application, the data generated by the IIoT devices (including sensors, actuators, machines and robots) can be processed and analysed by exploiting recent progress on artificial intelligence to enable smart manufacturing. The typical features of the IIoT services, (e.g., smart factory) include large-scale computation tasks, ultra-low latency, high reliability, and heterogeneous accessibility requirements.

- *Smart city.* The data from the large number of sensors across the city can be collected and analysed to facilitate the application of smart city. With the increasingly densely deployed IoT devices and the increasingly large size of networks, the data from the smart city applications can be in large volume.

1.1.2 Mobile Edge Computing MEC

According to European Telecommunications Standardization Institute (ETSI), the definition of MEC is to " *provide IT and cloud computing capabilities within the Radio Access Network (RAN) in close proximity to mobile subscribers* [14]." The features of Mobile Edge Computing (MEC) can be summarized as follows.

- **Low latency**: The MEC server is located at the edge of the network, which can significantly reduce the distance between the server and the terminal device. As a result, the delays of transmitting the data through the core network to the data centers, as typically done in Mobile Cloud Computing (MCC), can be eliminated, to provide low-latency and high-reliability services (e.g., autonomous driving).

- **High energy efficiency**: By offloading the computation-intensive applications from the mobile devices to the edge servers, MEC can reduce the energy consumption of local execution and prolong the battery life of the devices. According to [15], the battery life time of running AR applications via MEC can be increased by 30-50%.

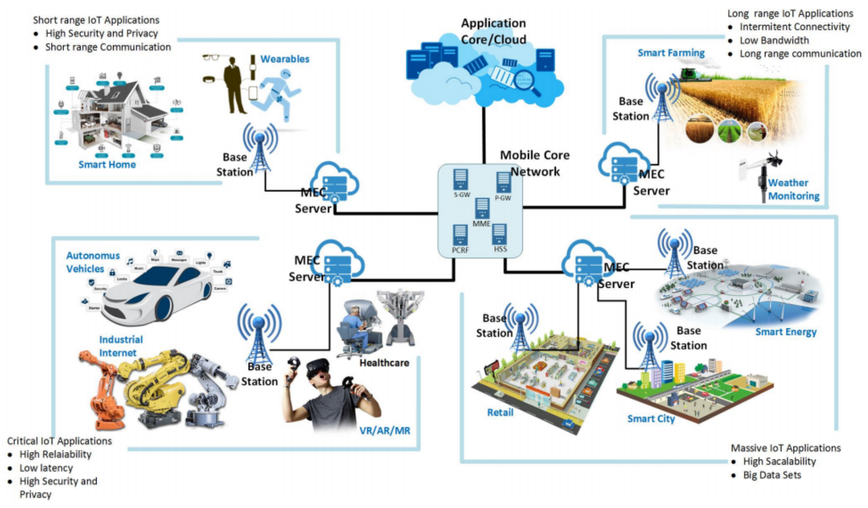


Figure 1.1 : The application scenarios of MEC for future IoT [1].

- **Scalability**: Different from the data center networks, where the servers are located in the same geo-location and can be controlled in a centralized manner, in MEC, the servers are geo-distributed and typically organized in a decentralized manner. This increases the flexibility and scalability of MEC platform. The distributed architecture can also enable the management of a large number of devices, and provide low-latency computing services.

In a nutshell, MEC is a distributed computing platform by placing computing, networking, and storage capabilities inside the devices in edge networks, and supporting low-latency, high-reliability, and high-connectivity applications [8–11, 16–19].

1.1.3 MEC for Future IoT

By placing various cloud resources (e.g., for computing and storage) close to smart devices, the advantages of MEC are highly in line with the features of IoT services [8–11]. As a result, MEC is anticipated to realize the promising benefits of IoT. Fig. 1.1 illustrates the typical application scenarios of MEC for future IoT. In many practical applications of smart city (i.e., smart metering, smart heating, cleaning services, smart energy, and smart farming), the geo-distributed IoT devices would generate large volumes of data to be processed at the edge servers and the cloud, thereby providing reactive learning from the big data. These applications are typically delay-tolerant. For example, the typical applications of smart energy, the delay can be 1s to several hours[1]. MEC can provide pervasive computing services to meet the requirements of low-latency, high-reliability, and high-connectivity applications for future IoT. By exploiting data processing at the point of capture, MEC has the potential to relieve network congestions pertaining to IoT services and increase network capacity of computing and storage.

MEC can help offload resource-hungry tasks from resource-restrained cordless IoT devices to MEC servers consisting of base stations (BSs) and gateways through wireless links [16–18]. MEC can guarantee the quality of services for computation-intensive and latency-critical applications, such as imaging processing, and augmented reality.

MEC is promising to orchestrate resources of wired network for handling the IoT data and network services with efficiency and agility [19–21]. The data can be pre-processed by the MEC servers, and the pre-processed results will be delivered to the data center for big data analytics to provide guidelines for some IoT services, such as smart city, and intelligent transportation systems. The pre-processing procedure can reduce the congestion of the backhual, and relieve the burden of data center for data analysis.

The unique features of future IoT can be summarized as follows.

- **Physical limitations of computing and battery resources of future IoT devices.** In future networks, the IoT devices and sensors can be increasingly densely deployed to achieve accurate and broadly environment monitoring. Consider the simple IoT devices (e.g., sensors). The physical sizes of such devices are restrained, limiting the battery sizes and computing resources that can be installed inside the devices.

- **Heterogeneity of computing and wireless capabilities of future IoT networks.** With the development of communication technologies, a large number of cordless or mobile IoT devices are connected to the Internet through different radio interfaces, e.g., 3GPP LTE/LTE-advanced and WiFi. The future IoT will generate prevalent heterogeneity for networks, such as the heterogeneity in data, and data collection, as well as distinct data control policies/regulations on typical consortium IoT platforms [8].

- **Large volume of data arrivals and massive connections in future IoT networks.** The growing demands for ultra-low latency, massive connectivity, and high reliability for large numbers of IoT services have yielded other critical issues, such as the limited connections (i.e., connection capacity, bandwidth, or the number of simultaneously affordable connections) between mobile edge cloud and smart devices/objects [8–11, 22]. Wireless IoT devices operate in shared wireless media, which have unique characteristics of unpredictable and time-varying channels, and can become increasingly congested with a growing number of IoT devices.

- **Distributed large-scale data storage and delivery across the edge network in mobile applications.** Hundreds of thousands mobile devices need to be connected to the Internet, and collect data for specific mobile applications. Given the sheer size of mobile devices, the real-time data transmission through core network could barely be feasible and would lead to network congestions and delay around centralized platforms. Moreover, given the limited storage and geo-distribution of MEC servers, the centralized caching mechanism of IoT data is impractical.

1.2 Challenges of MEC for Future IoT

Fig. 1.2 illustrates the applications of MEC for future IoT, where a large number of IoT devices with wireless interfaces to the edge cloud generate big data destined for the data centers (or sinks). This thesis is motivated to address the aforementioned four features of future IoT.

1.2.1 Scenarios of MEC for Future IoT

We can see in Fig. 1.2 that there are two types of IoT services and applications for future IoT. The first type of services and applications in the network are latency-

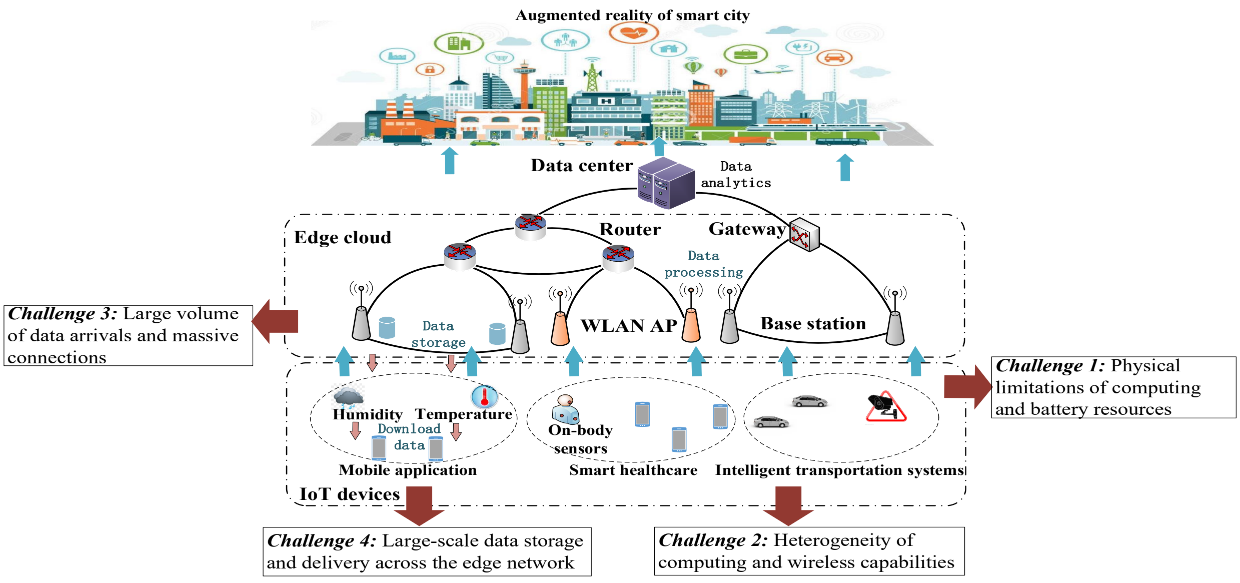

Augmented reality of smart city
Data center
Data analytics
Edge cloud
Router
Gateway
Data processing
Data storage
WLAN AP
Base station
Humidity
Temperature
Download data
Mobile application
On-body sensors
Smart healthcare
Intelligent transportation systems
IoT devices
Challenge 3: Large volume of data arrivals and massive connections
Challenge 1: Physical limitations of computing and battery resources
Challenge 4: Large-scale data storage and delivery across the edge network
Challenge 2: Heterogeneity of computing and wireless capabilities

Figure 1.2 : The application of MEC in IoT networks.

sensitive tasks, such as face recognition, and ultra-high-definition video streaming. Such tasks can be supported by some edge servers in close proximity to IoT devices, such as base stations, access points, and cloudlet. We consider two scenarios in this case to tackle the features of future IoT mentioned in Section 1.1.1

- **Multi-user single-cell task scheduling**. Given the feature of physical limitations of computing and battery resources of future IoT devices, task scheduling is studied to minimize the total energy consumption of IoT devices.

- **Multi-user multi-cell cooperative computing**. Given the feature of heterogeneity of computing and wireless capabilities of future IoT networks, cooperative computing is studied as the extension of first scenario.

The other type of tasks is big data analytics (e.g., smart city), where the large size of data cannot be supported in the MEC servers and need to be processed across edge cloud and data centers. Then, large-scale network scenario is taken into consideration. Ubiquitously deployed edge network nodes, such as switches, routers, and gateways in different subnetworks, can all play a role of MEC servers and collaborate with one another, leveraging the embedded computing and storage capabilities of the devices and helping process, and cache IoT data for corresponding services and applications.

The unified network is of particular value to many appealing, bandwidth-demanding IoT services, and mobile applications. For example, extensively deployed surveillance cameras, and cameras installed on mobile vehicles, can collect and upload real-time images or videos for regenerating augmented reality of environments and road traffic with broad applications to future smart cities [8–11]. Given the strong locality of the images/videos, instant content fusion around the point of capture in the edge cloud, by exploiting ubiquitous computing, provides an efficient approach for real-time augmentation and redundancy removal to alleviate network congestions. We consider two cases to tackle the features of future IoT mentioned in Section 1.1.1

- **Computing resource allocation in large-scale network**. Given the feature of large volume of data arrivals and massive connections in future IoT networks, large-scale fog computing is studied. By collaborating the MEC servers and data center, and allocating the embedded computing resources, big data analytics can be supported.
- **Storage resource allocation in large-scale network**. A large number of IoT data which is likely to be reused by others, such as temperature and humidity, collected by sensors need to be cached in MEC servers to support mobile applications (e.g., weather forecast). Given the feature of distributed large-scale data storage and delivery across the edge network, a cooperative caching region is designed to improve the storage efficiency.

1.2.2 Technical Challenges in Four Cases

We proceed to articulate the challenges of the four scenarios in details.

Case 1: Multi-user Single-cell Task Scheduling.

In this case, we consider that there are N IoT devices in the coverage of a single base station (BS), and the BS is connected with an edge server. The MECO can reduce the energy consumption of local processing for IoT devices, but increase the energy consumption for transmission. With the consideration of the physical limitations of computing and battery resources of IoT devices, an efficient task scheduling approach need to be proposed to minimize the energy consumption of IoT devices.

Challenges in this case arise from the time-varying nature of practical wireless channels. The BS can only have outdated knowledge of IoT devices' channel conditions at the last slot for predictive computation scheduling decisions at the current slot. However, the IoT devices can observe the devices information (such as data arrivals, computing resources, computing energy) for real-time local processing. In other words, the network states for local processing and computation offloading would be differently aged at the devices (with instantaneous observations of its own states for local processing) and the BS (with outdated channel conditions for computation offloading). Such differently-aged network states would prevent the existing approaches [18, 23–30] from functioning and require the development of hybrid learning approaches to be implemented at the devices for instantaneous local processing decisions and the BS for predictive computation offloading decisions.

Case 2: Multi-user Multi-cell Cooperative Computing.

In this case, we consider that there are N IoT devices covered by M edge servers. The tasks generated by IoT devices can be offloaded multiple hops away and processed anywhere in the network, i.e., locally executed, offloaded to other IoT devices, or remotely processed at the edge servers. The devices in the considered scenario have heterogeneity of computing and wireless capabilities. In particular, inexpensive IoT devices have limited computing and communication capability, and may only interact with one device at a time; while the powerful MEC servers (such as base stations, access points and cloudlets) can maintain simultaneous connections with multiple devices and process complex tasks.

The technical challenges in this case arise from the temporal and spatial couplings of the optimal operations. On the one hand, the causality of operations in multi-hop network would incur strong temporal couplings. Myopic optimization would lead to sub-optimal solutions due to prevalent network randomness (such as channel variations, task arrivals and available computing resources). Offline optimization requires the a-priori knowledge on the network dynamics, which is not practical due to the stochastic nature of the environment. On the other hand, non-uniform cardinalities of devices due to the different transmission capabilities of IoT devices and MEC servers would result in spatial couplings of the optimal operations. The transmission decisions are coupled among nearby devices. Typically centralized approaches make the transmission decisions for the devices in the presence of the global view of the network to avoid transmission collisions due to the non-uniform cardinalities. However, the instantaneous global view is unlikely given the network size and non-negligible multi-hop signaling delays.

Case 3: Large-scale Fog Computing under Limited IoT Buffers.

In this case, we consider that a large amount of data generated by IoT devices are destined for the data center for big data analytics via the edge cloud. Also, the task can be offloaded multiple hops away and processed anywhere in the network. The causality of operations in multi-hop network would incur strong temporal couplings. Lyapunov optimization can be used to decouple temporally coupled variables in this case, and achieve asymptotically optimal solutions with typically an $[\mathcal{O}(V), \mathcal{O}(1/V)]$-tradeoff between the queue lengths and optimality loss [31]. V is a configurable parameter to adjust the weights of stability and optimality (in terms of system utility) in a drift-plus-penalty function.

Challenges in this case arise from the limited buffers of IoT devices, which can not support the operations of Lyapunov optimization. The Lyapunov optimization would require sufficient differences among the queues to steer data from the IoT devices through edge servers to the data center. Unfortunately, inexpensive IoT devices can only have very limited buffers, leading to insufficient queue differences for driving data towards the data center. Moreover, the limited buffers of IoT devices do not align with the typical setting of Lyapunov optimization, where V needs to be large to achieve the asymptotic optimality.

Case 4: Large-scale Cooperative Region for Edge Caching.

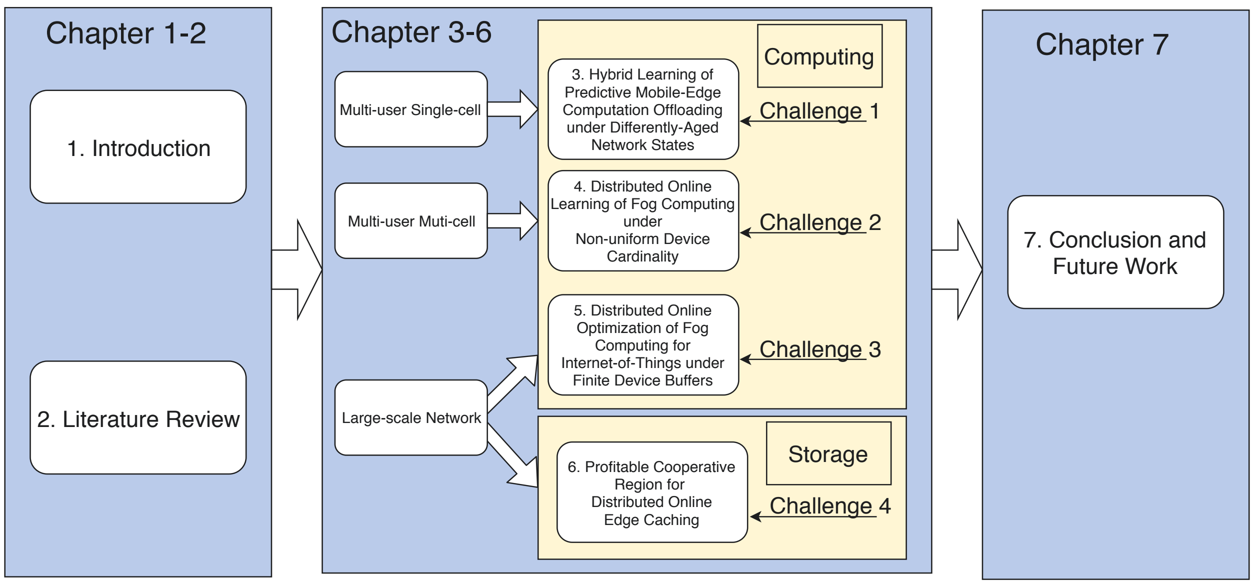
In above three cases, we focus on the computing resource allocation in MEC. However, the storage resource allocation is also an important issue in MEC. In this case, we consider that there are N edge servers, which can cache the IoT data in the local memories to support mobile applications. F IoT files/data are requested from mobile users. The requested data can be retrieved from network backbone and edge cloud.

Challenges in this case arise from the design of cooperative mechanism and cooperative regions. Cooperative edge caching is challenging. This is because edge clouds can be very large and geo-distributed, and undergo spatial and temporal variations resulting from random content request arrivals and background traffic. Centralized coordination [32–49] becomes ineffective with significant signaling delays and outdated network knowledge which would inhibit optimal solutions. However, there has been no proper decentralized solution for cooperative edge caching in the p-resence of random variations. Moreover, in the absence of a global network view, file/data requests could be dispatched unnecessarily large numbers of hops away from the edge server admitting the requests under decentralized designs. File/data could be retrieved from large numbers of hops away as well. The efficiency (or cost-effectiveness) of the file/data delivery would degrade [50].

Setting up appropriate cooperative caching regions, within which requests can be dispatched for file/data retrievals with guaranteed cost-savings compared to retrievals from the core network, is crucial to prevent the above inefficiency. The regions have the potential to guide the placement of file/data, preventing repeatedly and inefficiently caching the same contents at nearby servers. However, the setup of the regions is non-trivial under decentralized network settings of cooperative edge caching.

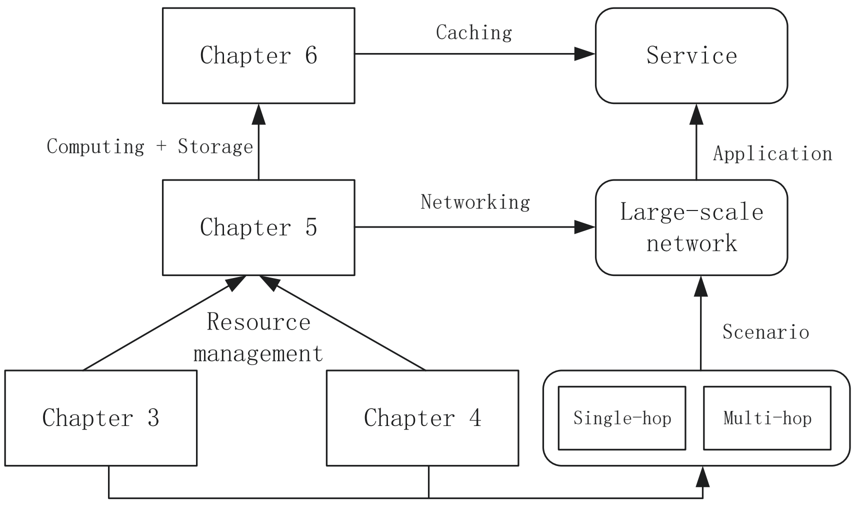
1.3 Thesis Organization

In this thesis, we propose four new approaches to address the challenges listed in Section 1.2.2. "Hybrid Learning of Predictive Mobile Edge Computation Offloading



Chapter 1-2
1. Introduction
2. Literature Review
Chapter 3-6
Multi-user Single-cell
Multi-user Muti-cell
Large-scale Network
3. Hybrid Learning of Predictive Mobile-Edge Computation Offloading under Differently-Aged Network States
4. Distributed Online Learning of Fog Computing under Non-uniform Device Cardinality
5. Distributed Online Optimization of Fog Computing for Internet-of-Things under Finite Device Buffers
6. Profitable Cooperative Region for Distributed Online Edge Caching
Computing
Challenge 1
Challenge 2
Challenge 3
Storage
Challenge 4
Chapter 7
7. Conclusion and Future Work

(a) The organization of thesis.



(b) The relationship between chapters.

Figure 1.3 : The organization and relationship of thesis.

under Differently-Aged Network States" is proposed to address the first challenge in *Case 1*. "Distributed Online Learning of Fog Computing under Non-uniform Device Cardinality" is proposed to address the second challenge in *Case 2*. "Distributed Online Optimization of Fog Computing for Internet-of-Things under Finite Device Buffers" is proposed to address the third challenges in *Case 3*. "Profitable Cooperative Region for Distributed Online Edge Caching" is proposed to address the fourth challenge in *Case 4*.

This organization of this thesis is shown in Fig. 1.3(a), and summarized as follows:

1. As a brief introduction, Chapter 1 introduces the background of future IoT and MEC, and summarizes the technical challenges of MEC for future IoT.
2. Chapter 2 introduces the architectures of IoT and MEC, and summarizes the related works.
3. Chapter 3 proposes a hybrid learning approach to optimize the real-time local processing and predictive computation offloading decisions in a distributed manner given the time-varying nature of practical wireless channels.
4. Chapter 4 develops an energy-efficient resource management approach based on distributed online learning to tackle the heterogeneity of computing and wireless transmission capabilities of edge servers and IoT devices.
5. Chapter 5 proposes an enabling technique for Lyapunov optimization to operate under finite buffers without loss of asymptotic optimality. This is achieved by optimizing the biases (namely, virtual placeholders) of the buffers to create sufficient backlogs.
6. Chapter 6 establishes a new profitable cooperative region for every IoT file/data request admitted at an edge server, which can help automate the placement of contents with reduced density and improved efficiency.
7. Chapter 7 presents the conclusions drawn from the results discussed in earlier chapters of the thesis, and discusses the limitations and future research directions of this study.

As shown in Fig. 1.3(b), this thesis consists of four chapters of works for three types of scenarios, i.e., multi-user single-cell scenario, multi-user multi-cell scenario and large-scale scenario. The relationships between these works are provided in the following.

1. Case 1 is designed for the IoT devices scheduling mechanism in single-cell.

2. Case 2 is designed for the multi-hop transmission mechanism of task processing, dispatching and result delivery between nodes in multiple cells.

3. Case 3 is focused on a large-scale IoT network, where the IoT devices covered by different edge servers generate large amounts of data destined for the data center for big data analytics. Data uploading of IoT devices, and data processing, dispatching, and result delivery between edge servers in network are considered.

4. Case 4 is the service application scenario (i.e., content/file retrieval), where each server optimizes its decisions on the admission, dispatching and grant of file requests, and the file delivery for granted requests.

The four cases in this thesis studies the resource management of fog computing from different aspects, i.e., case 1 and 2 research on wireless resource management of single-hop and multi-hop transmission; case 3 researches on networking in a large-scale wired network; case 4 researches on caching with edge computing. The proposed approaches in cases 1–4 can operate in conjunctions to increase the throughput and scalability of fog computing.

Chapter 2

Literature Review

In this chapter, we will introduce the architectures for IoT and MEC. Then, the related works are summarized with respect to the topics of multi-user single-cell task scheduling, multi-user multi-cell cooperative computing, computing and storage resource allocation in large-scale networks.

2.1 Architecture of IoT and MEC

2.1.1 Architecture of IoT

The architectures of IoT can be classified into [51]: RFID, service oriented architecture, wireless sensor network, supply chain management, industry, healthcare, smart city, logistics, connected living, big data, cloud computing, social computing, and security. Here, we introduce the reference architecture of IoT proposed by Guth et al. [2] in terms of the functional blocks. Fig. 2.1 illustrates the reference architecture of IoT, which depicts the components and interconnections. The components and functions are summarized as following.

- *Sensor and actuator.* Sensors and actuators are the bottom hardware components of IoT architecture. In specific, sensors are used to collect surrounding data, such as temperature or humidity, in the environment, then transform the data to electrical signals. Actuators are used to act upon, control, or manipulate the physical environment, then transform the electrical signals into some kind of physical actions [2]. Both of the sensors and actuators are connected with or integrated into the devices.
- *Device.* By exploiting the software installed in drivers, the devices can process the data collecting from the bottom components, and translate the physical

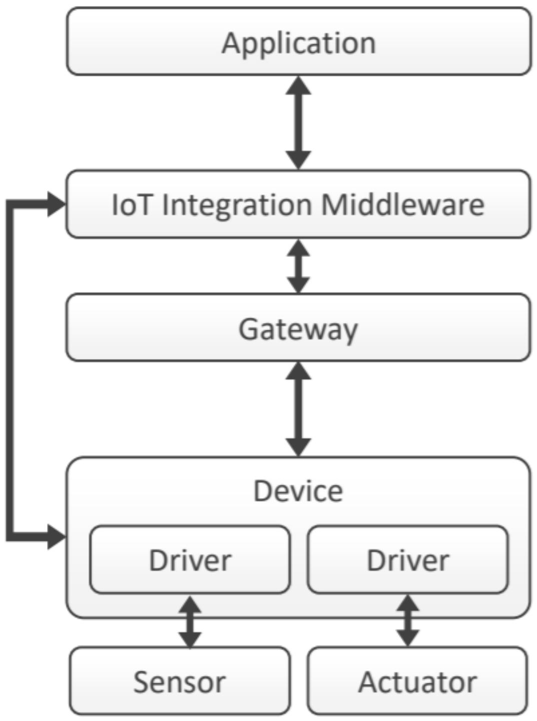


Figure 2.1 : The reference architecture of IoT [2].

parameters into digital information. Devices can be self-contained or connected to another system to facilitate the IoT integration middleware.

- *Gateway.* The gateway is used to supported different protocols of interfaces, such as 3GPP, LTE, and WiFi, and forward the transmission to other entities in the network for devices. Moreover, the gateway may allocate the services and resources for devices.

- *IoT Integration Middleware.* The IoT Integration Middleware is responsible for integrating different kinds of sensors, actuators, devices, and applications. Also, it has most of the functions to support cyber-physical system, such as a rules engine, and graphical dashboards.

- *Application.* The software system operated based on the IoT integration middleware and data collected from the bottom physical components represents the application.

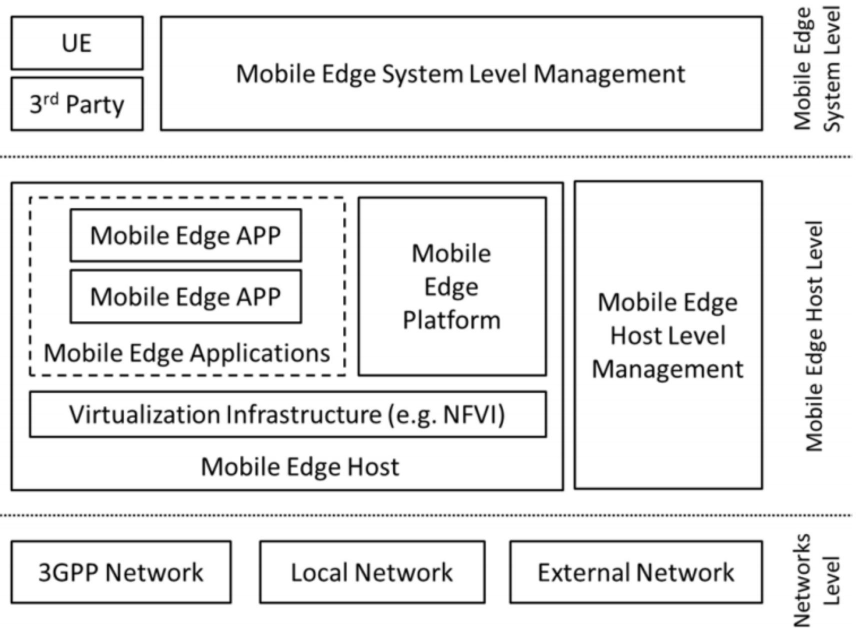


Figure 2.2 : The framework of MEC[3].

2.1.2 Architecture of MEC

1) Framework of MEC:

Figure 2.2 is the basic framework of MEC, where the functional entities of MEC can be classified into networks level, mobile edge hosts level, and mobile edge system level.

- *Mobile edge hosts level.* The mobile edge host level consists of a mobile edge host and a corresponding mobile edge host level management entity, and the mobile edge hosts level can be further categorized into a mobile edge platform, mobile edge applications, and a virtualization infrastructure. The mobile edge host provide computing, caching and networking functions to support the mobile edge applications. The mobile edge platform enables applications to discover, advertise and consume edge services, and provides the virtualization infrastructure with a set of rules for the forwarding plane [3]. The communication between mobile edge platforms is via the interface Mp3 interface, between mobile edge platform and mobile edge application is via the Mp1 interface, between virtualization infrastructures is via Mp7 interface.

- *Networks level.* The networks level consists of external entities, such as 3GPP network, local network and external network representing the association relationship of MEC system and networks.
- *Mobile edge system level.* The top layer is the mobile edge system level management entity, which is responsible for the overall control of the MEC system. The mobile edge system level consists of mobile edge orchestrator and operation support system. In particular, the mobile edge orchestrator is responsible for the application authentication and management. The operation support system is responsible for the service and subsystem management, and request admission from Customer Facing Service portal and users.

2) Architecture of MEC

Three generic MEC architectures are proposed in terms of the different localization of MEC servers.

Edge Network Architecture: Deploy MEC servers to base stations or wireless access points, such as Small Cell Cloud (SCC) [52, 53] and Mobile Micro Clouds (MMC) [54]. The architecture is proposed to enhance the storage and computing capabilities of small cell base stations, thereby improving the performances of cellular networks. The SCC can be controlled by a Small Cell Management entity (SCM) in a centralized manner, or a local SCM (L-SCM) in a distributed manner. With the deployment of small cells, SCC can provide sufficient computing and storage resources. Moreover, the architecture of MMC deploys the computing and storage resources on the base station side, which does not require any management entities, and can be organized in a distributed manner. The edge server of SCC and MMC is in close proximity to the terminal devices. The user experience of service of latency-critical services can be guaranteed.

Core Network MEC Architecture: Deploy the edge servers close to Radio Access Network (RAN) or close to RAN, such as Fast Moving Personal Cloud (MobiScud) [55] and Follow Me Cloud (FMC) [56, 57]. The data center network in MCC can be operated in centralized manner, which has low scalability and cannot meet the increasing number of users. The MEC architecture is operated in a distributed manner, enabling the operator management with the increasing number of devices, and providing low-latency services. Compared to SCC and MMC, this architecture is capable of computing and storage, but remote from the terminal devices, which support the applications with large computation and low delay sensitivity.

"Ubiquitous Computing" Architecture: It is noted that the above architectures have their limitations. A new architecture called "ubiquitous computing" has been proposed and widely adopted in the literature [58, 59]. In particular, ubiquitously deployed edge network nodes, such as switches, routers, and gateways in different subnetworks, can all play a role of edge servers and collaborate with one another, leveraging the embedded computing capabilities of the devices and helping process and route data from the point of capture all the way to their destinations. NFV and SDN are the enabling technologies to orchestrate resources of networks for handling services with efficiency and agility. The ubiquitous computing architecture can realize the rational allocation of computing tasks, data traffic, and storage by coordinating infrastructure resources in the network, thereby integrating the advantages of above architectures to provide different services and applications.

2.1.3 Implementation of MEC for Future IoT

ETSI has identified IoT as one of the most important use cases of MEC [4]. The MEC server can be regarded as the gateway of IoT architecture, which can support different communication technologies and protocols, such as 3GPP, LTE, WiFi, Bluetooth, and ZigBee, for IoT devices. Nevertheless, the MEC server can aggregate the data, and process the data with embedded computing resources. Fig. 2.3 show the illustrative example of IoT gateway service scenario [1, 60].

MEC has been the key enabler for IoT in recent researches [1, 8–13, 61–66]. The works [1, 8–13], discuss the role of MEC for IoT, and introduce the some special use cases. In [61], a UAV-based IoT platform is designed by leveraging MEC. In [62],

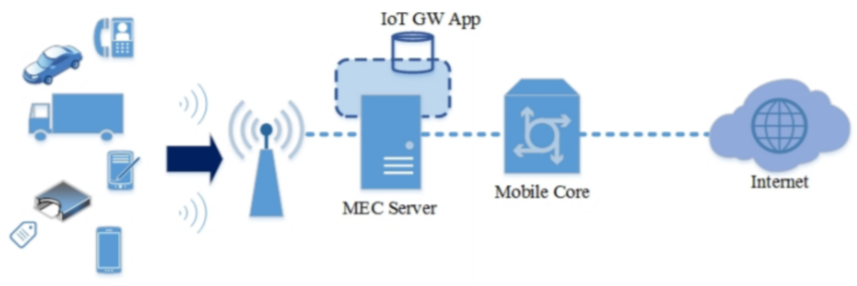


Figure 2.3 : IoT gateway service scenario [4].

a collaborative computation offloading mechanism is proposed under fiber-wireless architecture for IoT. In [63, 64], authors identify MEC services for IoT big-data analytics. SDN scenarios based on IoT and MEC are discussed in [65, 66].

2.2 Summary of Related Work

1) Multi-user Single-cell Task Scheduling:

In the scenario of multi-user single-cell MEC, existing works have proposed static and dynamic approaches to optimize computation offloading and local processing decisions to reduce service delays and energy consumption of mobile devices [16–18, 23–30, 67–70]. In [17, 67–70], the problem of offloading decisions and resource allocations was optimized offline via submodular/convex optimization techniques [17, 67, 68] and non-cooperative games [69, 70]. However, the one-off optimization [16, 17, 67–70] assumed that the wireless/computing environment remains unchanged during task execution and can be acquired/predicted at the time of task arrivals. In [18, 23–30], dynamic approaches were proposed to adjust the offloading decisions and resource allocations according to the current observation of the environment, where Lyapunov optimization [18, 23–25] and Q-learning [27–30] were exploited to enable the online decisions. These dynamic approaches [18, 23–30] also require that the observed network states would be static for the upcoming time slot (e.g., during the task execution).

However, the technical challenge of the differently-aged network states resulting from the time-varying wireless channel conditions in first case would prevent the existing approaches [18, 23–30] from functioning and require the development of hybrid learning approaches to be implemented at the devices for instantaneous local processing decisions and the BS for predictive computation offloading decisions.

2) Multi-user Multi-cell Cooperative Computing:

Existing works of multi-user multi-cell MEC scenario focused on the researches of MEC server selection and computation immigration between servers. The cooperation of MEC server and data center for multi-user task offloading was studied in [71]. A threshold-based scheduling algorithm was proposed by comparing the transmission latency and computation latency, to maximize the total successful offloading probability. In [72], a game-theoretic approach was proposed to minimize the energy consumption of network. A congestion game was formulated to address the couplings between the amounts of the offloading tasks and edge server selection policies. [73] studied a task offloading policy under the scenario where one mobile device can offload tasks to multiple MEC servers. A semi-definite relaxation-based approach was proposed under the fixed CPU frequency to jointly optimize the task offloading decisions and CPU frequency scaling. [74] studied the cooperation among the servers in geo-distribution, and a coalition game was formulated with the utility function representing the cost of virtual machine migration and resource utilization.

However, none of the existing studies [71–74] have taken the heterogeneity of wireless capabilities of future IoT networks into account. Therefore, the technical challenge of couplings between network operations in time and space resulting from the non-uniform cardinality of devices, discussed in Section 1.2.2 can not be addressed by these works.

3) Large-scale Fog Computing:

In the large-scale network scenario, there have been separate studies on MECO that offloads resource-hungry tasks from wireless devices to edge servers [16–18, 67, 69, 70, 75–78], and fog computing that orchestrates network resources for peer-to-peer computing [19, 21, 79–82]. With the assistance of edge servers (e.g., base stations and access points), offloading decisions and resource allocations were typically optimized in a centralized manner [67, 75–78]. For the sake of scalability, distributed scheduling of MECO was studied in [17, 18, 69, 70]. In [69, 70], a non-cooperative game for offloading decisions was formulated to capture the interactions among mobile devices in wireless interference environments. In [17], a heuristic scheme based on submodular optimization was proposed, where the joint optimization of offloading decisions and resource allocations was decomposed for distributed implementations. In [18], Lyapunov optimization was exploited to decouple the offloading and resource allocation between time instants and devices, and achieve the asymptotic optimality in the presence of out-of-date feedback.

In the context of data center or cloud, load balancing for collaborative computing was studied to address the spatial diversities of workload arrivals [79], temperatures [80], and electricity prices [81]. In our recent work [21], the distributed online optimization of fog computing was developed with an emphasis on determining the collaborative regions of edge servers. In [19, 82], the online cooperation of devices was optimized in a centralized manner, where tit-for-tat incentives were used to discourage selfish behaviors of devices and decoupled by Lyapunov optimization. Unfortunately, the offloading was restricted within a single hop in [19, 82], far from unlocking the full potentials of fog computing.

Separate studies on MECO [16–18, 67, 69, 70, 75–78], and fog computing [19, 21, 79–82]. cannot support the large volume of data arrivals and massive connections in future IoT networks. Moreover, none of the existing studies [16–19, 21, 67, 69, 70, 75–83] have taken the limited finite buffers of practical IoT devices into account. However, the direct application of Lyapunov optimization [18, 19, 21, 82] cannot address the problem of interest due to the limited buffers of practical IoT devices.

4) Large-scale Cooperative Caching:

The studies on edge caching are focused on the content placement and delivery across edge cloud. Content placement and delivery have been typically decoupled and separately studied in the literature [84]. Given caching policy and content placement, content delivery has been studied in wireless networks [85–87] and wired networks [33–35]. In [85–87], cache-enabled multicast and cooperative beamforming were studied for the purpose of throughput improvement. In [33–35], multi-hop cache-aware routing was developed to minimize the system cost over backbones. All these schemes were off-line and based on centralized optimization, and are unsuitable for edge clouds where temporal variations and randomness prevail due to background traffic.

Cooperative content placement was studied in [36–43]. By exploiting a typically hierarchical network structure, the placement was decoupled to in-tier and cross-tier subproblems [36–39], and solved by using integer programming [36, 37] and game theory [38], or analytically evaluating the content popularity [39]. In [40], the problem of joint caching, routing, and channel assignment was formulated to maximize system throughput in coordinated small-cell networks, and solved by taking a column generation method. In [41] and [42], the joint optimization of content placement and delivery was formulated as off-line integer programming problems, where scalable video coding (SVC)-based layered video caching [41] and real-time video transcoding [42] were considered. In [42], given caching policy, online content delivery and transcoding were jointly optimized in a centralized manner. In [43], the joint optimization of routing and caching for minimizing a content-access delay was proved to be NP-complete, and approximated algorithms were developed to achieve a $(1 - 1/e)$ optimality. In [44], an online cooperative caching and one-hop wireless transmission scheme was proposed in a cloud-RAN to minimize the normalized delivery time (NDT) metric which captures both the delivery phase of coded caching and the time needed to replenish the cache placement. By modeling the time-varying popularity as a Markov process, both reactive and proactive online caching schemes were developed with a bounded long-term NDT. All these works [36–44] were focused on content deliveries of up to two hops.

The joint optimization of content delivery and placement over unrestricted numbers of hops was studied in [32], where the content delivery over multiple hops had to be accomplished within a single slot by reserving resources along the delivery path in a centralized manner. However, the centralized approach in [32] may be inefficient due to significant signaling delays and outdated knowledge of the large networks.

The existing works of cooperative edge caching have not considered the The cooperative edge caching of interest in future IoT networks has distinctively different and practical settings from the above existing works. It necessitates new formulations, undergoes specific constraints, and leads to distinct solutions from existing researches.

However, there has been no proper decentralized solution for cooperative edge caching, thereby preventing supporting the low-latency demanding mobile applications. Moreover, there has been no adequate design on cooperative caching regions [32–49]. Existing cooperative caching regions are either restrained to within a few hops from the edge server admitting a content request [36–49], or not restrained at all [32–35]. This would lead to either reduced cache hit ratio or significant delays and delivery cost.

Chapter 3

Hybrid Learning of Predictive Mobile-Edge Computation Offloading under Differently-Aged Network States

Given the time-varying nature of practical wireless channels, the BS can only have outdated knowledge of IoT devices' channel conditions at the last slot for predictive computation offloading decisions at the current slot. However, the devices' information, such as data arrivals, and available computing resources, can be observed by IoT devices to make the decisions on real-time local processing. In other words, the network states for local processing and computation offloading would be differently aged at the devices (with instantaneous observations of its own states for local processing) and the BS (with outdated channel conditions for computation offloading). Such differently-aged network states would prevent the existing approaches [18, 23–30] from functioning and require the development of hybrid learning approaches to be implemented at the devices for instantaneous local processing decisions and the BS for predictive computation offloading decisions.

This chapter aims to tackle the challenges arising from the differently-aged network states, and proposes a novel hybrid learning approach that optimizes the instantaneous local processing and predictive computation offloading decisions. The decisions are expect to minimize the overall energy consumption of all the IoT devices, while maintaining the buffer stability of the devices. The proposed hybrid learning approach can be decentralized for individual devices and BS, to learn the network dynamics online and achieve the asymptotically optimal solutions without the a-priori knowledge of network states. The key contributions of the chapter are as follows.

- We establish a new hybrid learning approach for instantaneous local process-

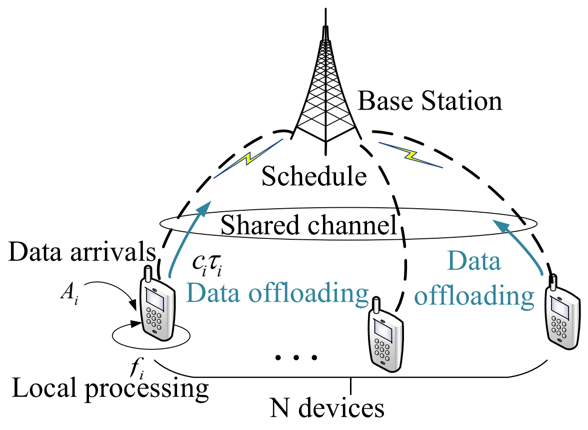


Figure 3.1 : The multi-user single-cell MEC network.

ing and predictive computation offloading decisions by integrating the learning techniques of stochastic gradient descent (SGD) and online convex optimization (OCO) in the primal-dual optimization framework via Lagrange duality. The Lagrange multipliers can be used to exchange the learning results between OCO and SGD and provide a unified hybrid learning framework.

- We decentralize the proposed hybrid learning approach between the BS and IoT devices for scalability. This is achieved by decomposing the primal problems into independent local processing and computation offloading subproblems to be separately pursued at the devices for instantaneous local processing based on local observations and the BS for predictive computation offloading from previous devices' reports.

- We prove the asymptotic optimality of the proposed hybrid learning approach, where the optimality loss resulting from the differently-aged network states can diminish with the decreasing stepsizes of SGD and OCO.

3.1 System Model

Fig. 3.1 shows the multi-user single-cell MEC network of a single BS (which is co-located with an edge server) and N IoT devices. Let $\mathbf{N} = \{1, \cdots, N\}$ collect the indexes for the N IoT devices. The system operates on a slotted basis. At each time slot, the tasks arriving at the IoT devices can be either processed locally or offloaded to the edge server via a shared wireless channel. Each device allocates its computing resources for task processing based on its current observations on local computing capabilities and workloads, while the task offloading via the shared channel needs to be optimized at the BS after the BS collects the reports from the devices.

The key difference of the network of interest (to the state-of-the-art [16–18, 23–30, 67–70]) is that we consider the time-varying nature of practical wireless channels: at the beginning of a time slot, the BS can only have the outdated knowledge of devices' channel conditions at the last time slot for predictive computation offloading decisions at the current slot. As a result, the network states for local processing and computation offloading are differently aged at the devices (with instantaneous observations of its own) and the BS (with outdated channel conditions).

3.1.1 Differently-aged Network States

Let $\mathbf{\Omega}(t) = \{\boldsymbol{\omega}_i(t), \boldsymbol{\omega}(t), \forall i\}$ denote the set of the network states at time slot t, where $\boldsymbol{\omega}_i(t) = \{A_i(t), F_i(t), \varepsilon_i(t)\}$ and $\boldsymbol{\omega}(t) = \{c_i(t), \forall i\}$ are the states at the time slot for local processing of device i and computation offloading at the BS, respectively. $A_i(t) \leq A_i^{\max}$ is the sizes of tasks (in bits) arriving at device i at time slot t. $F_i(t) \leq F_i^{\max}$ and $\varepsilon_i(t) \leq \varepsilon_i^{\max}$ are the available computing resources (in the bits of data that can be processed) and computing cost (in Joules to process a bit of data) of device i at the slot, respectively. $c_i(t)$ denotes the capacity (in bps) of the wireless channel from device i to the BS at time slot t. Due to the finite transmit power of the devices, the channel capacity is upper bounded, i.e., $c_i(t) \leq c_i^{\max}$.

The states for local processing and computation offloading, $\boldsymbol{\omega}_i(t)$ and $\boldsymbol{\omega}(t)$, can be differently-aged at the time of decision at the IoT devices and BS. The devices can locally observe the instantaneous task arrivals, $A_i(t)$, computing resources, $F_i(t)$, and processing cost, $\varepsilon_i(t)$ for local processing decision. In contrast, the reports from the devices received at the beginning of time slot t can be outdated to only capture the channel states $\{c_i(t-1), \forall i\}$ of the last slot due to the continuously changing wireless channels. As a result, the BS can only make the computation offloading schedules predictively with the outdated knowledge of channel conditions.

The differently-aged network states necessitate the design of hybrid learning for predictive computation offloading and instantaneous task processing in the MEC system. In specific, the devices need to apply statistical learning approaches to facilitate online task processing decisions, and the BS needs to run online learning to forecast the channel states from the previous reports for predictive computation offloading. Existing works apply either the statistical learning from instantaneous information [18, 23–30] or online learning from outdated information [88–91], and cannot be applied to the network with differently-aged information. The proposed hybrid learning integrates the statistical and online learning to exploit the differently-aged network information, i.e., predicting the channel states from outdated devices' reports at the BS while utilizing the instantaneous task knowledge at the devices, to increase the effectiveness and performance of learning in MEC. The details will be articulated in Section3.2.

3.1.2 Network Operations

The IoT devices share the wireless channel via time-division multiple-access (T-DMA). Let $\boldsymbol{\tau}(t) = \{\tau_i(t), \forall i\}$ be the schedule decisions of the IoT devices at time slot t, where $\tau_i(t)$ is the transmission duration of device i at the slot. The schedule decisions must satisfy

$$\tau_i(t) \geq 0,\ \forall i, t; \tag{3.1a}$$

$$\sum\nolimits_{i \in \mathbf{N}} \tau_i(t) \leq T,\ \forall t; \tag{3.1b}$$

where (3.1a) is self-explanatory, and (3.1b) states that the total transmission time of all the devices must not exceed the slot duration T.

Let $D_i(t)$ denote the backlog (in bits) of the unprocessed tasks (i.e., workloads) at device i at slot t, as given by

$$D_i(t+1) = [D_i(t) - c_i(t)\tau_i(t) - f_i(t)]^+ + A_i(t),\ \forall i,$$

where $[\cdot]^+ = \max\{\cdot, 0\}$, and $c_i(t)\tau_i(t)$ is the maximum size of data transmitted from device i to the BS within the scheduled time duration of $\tau_i(t)$. Here, $f_i(t)$ is the computing resources allocated for local processing at device i at time slot t and cannot exceed the maximum available resources at the device, i.e.,

$$0 \leq f_i(t) \leq F_i(t), \forall t. \tag{3.2}$$

Let $Q(t)$ denote the backlog (in bits) of the unprocessed tasks at the BS at time slot t, as given by

$$Q(t+1) = [Q(t) - F(t)]^+ + \sum\nolimits_{i \in \mathbf{N}} c_i(t)\tau_i(t),$$

where $F(t)$ is the available computing resources (in bits of raw data to be processed) at the BS, and $\sum_{i=i}^{N} c_i(t)\tau_i(t)$ specifies the sizes of data offloaded from all the selected devices to the BS during slot t.

We need to ensure the stability of all the queues $D_i(t)$ and $Q(t)$ in the system, such that the unbounded growths of the backlogs (and hence, queuing delays) can be prevented. According to queuing theory [92], a queue is stable, if and only if the time-average input to the queue is no more than the time-average output of the queue. The constraints for queue stability can be written as

$$\overline{A_i(t) - c_i(t)\tau_i(t) - f_i(t)} \leq 0, \forall i; \tag{3.3a}$$

$$\overline{\sum\nolimits_{i \in \mathbf{N}} c_i(t)\tau_i(t) - F(t)} \leq 0, \forall i; \tag{3.3b}$$

where $\overline{X(t)} = \lim_{T\to\infty} \frac{1}{T} \sum_{\tau=0}^{T-1} \mathbb{E}[X(\tau)]$ denotes the time-average of any random process $X(t)$.

3.1.3 Problem Formulation

Typical remote wireless devices are battery-powered. We take energy consumption of the devices as the performance metric of the MEC system. The overall energy consumption of the wireless devices at time slot t can be given by

$$\Phi(\mathbf{x}(t)) = \sum\nolimits_{i\in\mathbf{N}} \varepsilon_i(t) f_i(t) + p_i \tau_i(t),$$

where $\mathbf{x}(t) = \{f_i(t), \tau_i(t), \forall i\}$ collects all the variables of local processing $f_i(t)$ and remote offloading $\tau_i(t)$ at slot t. $\varepsilon_i(t)f_i(t)$ and $p_i\tau_i(t)$ are the energy consumptions of device i for local processing and remote offloading, respectively.

We propose to minimize the time-average energy consumption of the devices under the outdated channel states at the BS and the constraints for system stability. The problem can be formulated as

$$\begin{aligned} \Phi^* = &\min_{\mathbb{X}} \overline{\Phi(\mathbf{x}(t))} \\ &\text{s.t. } (3.1),\ (3.2),\ (3.3),\ \forall t, \end{aligned} \tag{3.4}$$

where $\mathbb{X} = \{\mathbf{x}(t), \forall t\}$ collects the variables of local processing and computation offloading across all time slots. Problem (3.4) is challenging with the following features.

1. *Temporal Coupling without a-priori Knowledge:* The time-average constraints on the queue stability, (3.3a) and (3.3b), can result in strong couplings of the variables $\mathbb{X}$ across all time slots. With the temporal coupling, problem (3.4) cannot be solved myopically (which would lead to severe performance degradation). The problem cannot be solved offline either, since the a-priori knowledge on the channel conditions, computing resources, and task arrivals is unavailable and hard to predict due to the stochastic nature of the system.

2. *Differently-aged Network States:* As described in Section 3.1.1, this chapter considers the time-varying nature of practical wireless channels. At the beginning of a time slot, the BS can only have outdated knowledge of devices' channel conditions. In other words, the network states needed for optimizing local processing and offloading would be differently aged at the devices (with instantaneous observations of its own) and the BS (with outdated channel

conditions at the last slot). This would require different approaches to be implemented at the devices for instantaneous local processing decisions and at the BS for predictive offloading decisions.

3.2 Hybrid Learning of Predictive Computation Offloading under Differently-aged Network States

This section presents the new hybrid learning approach that integrates OCO and SGD to achieve instantaneous processing at the devices and predictive computation offloading at the BS in a fully decentralized manner. The proposed approach minimizes the time-average energy consumption of the devices in the presence of differently-aged network states. The approach can be separately implemented at the BS and devices. The devices apply SGD to make online local processing decisions based on the instantaneous observations of its task arrivals and processor states. The BS uses OCO to forecast the channel states from the previous reports of the devices (including SGD-learning results and channel states) for predictive offloading schedules.

An overview of the proposed hybrid learning framework can be provided as follows.

1. *Primal-dual Framework via Lagrange Duality:* The proposed hybrid learning framework first exploits the Lagrange duality to associate the time-average constraints of problem (3.4) with nonnegative Lagrange multipliers to decompose the temporal couplings.

2. *Dual Learning for Optimal Online Decisions on Local Processing per Device via SGD:* SGD can be applied to learn the optimal Lagrange-dual multipliers online from the sequence of observations on the network dynamics.

3. *Predictive Offloading Schedule via OCO:* The decoupled per-slot computation offloading subproblem can be interpreted as an online learning problem, which optimizes the offloading schedule predictively via OCO by forecasting the current network states from the past devices' reports.

Both the SGD-based dual learning and the OCO-based predictive online learning can be integrated to the Lagrange primal-dual framework. The Lagrange multipliers can be used to exchange the learning results between OCO and SGD and provide a unified hybrid learning framework. The details are provided in the following.

By applying Lagrange duality, (3.4) can be reformulated. Let $\boldsymbol{\lambda}(t) = \{\lambda_{i,1}(t), \lambda_2(t), \forall i\}$ collect the set of the Lagrange multipliers $\lambda_{i,1}(t)$ and $\lambda_2(t)$ associated with (3.3a) and (3.3b) at slot t, respectively. The Lagrangian of (3.4) can be given by

$$\mathcal{L}(\mathbb{X}, \boldsymbol{\lambda}(t), \boldsymbol{\Omega}(t)) = \mathbb{E}\big[\mathcal{L}(\mathbf{x}(t), \boldsymbol{\lambda}(t), \boldsymbol{\Omega}(t))\big],$$

where $\mathcal{L}(\mathbf{x}(t), \boldsymbol{\lambda}(t), \boldsymbol{\Omega}(t))$ is the instantaneous Lagrangian, as given by

$$\begin{aligned}\mathcal{L}(\mathbf{x}(t), \boldsymbol{\lambda}(t), \boldsymbol{\Omega}(t)) = \textstyle\sum_{i\in\mathbf{N}} \big[p_i\tau_i(t) + \varepsilon_i(t)f_i(t)\big] + \sum_{i\in\mathbf{N}} \lambda_{i,1}(t)\big[A_i(t) - c_i(t)\tau_i(t) \\ - f_i(t)\big] + \lambda_2(t)\big[\textstyle\sum_{i\in\mathbf{N}} c_i(t)\tau_i(t) - F(t)\big].\end{aligned} \tag{3.5}$$

Given the strong duality of the convex objective and constraints, problem (3.4) can be equivalently reformulated to its dual problem [93], i.e., $\max\limits_{\boldsymbol{\lambda}(t)\succeq 0} \mathcal{L}(\mathbf{x}(t), \boldsymbol{\lambda}(t), \boldsymbol{\Omega}(t))$, where $\succeq$ is taken entry-wise. This is because the solution to the dual Lagrangian with non-negative multipliers, i.e., $D(\boldsymbol{\lambda}(t)) = \min_{\mathbf{x}(t)} \mathcal{L}(\mathbf{x}(t), \boldsymbol{\lambda}(t), \boldsymbol{\Omega}(t))$, provides the lower bound of the primal problem (3.4), i.e., $D(\boldsymbol{\lambda}(t)) \leq \Phi^*$. The duality gap (i.e., the difference between $D(\boldsymbol{\lambda}(t))$ and Φ^*) can diminish by finding the optimal multipliers to maximize the dual Lagrangian, i.e., $\max_{\boldsymbol{\lambda}(t)} D(\boldsymbol{\lambda}(t))$.

The optimization of the primal variables $\mathbf{x}(t)$ and the dual multipliers $\boldsymbol{\lambda}(t)$ can be decoupled and solved iteratively slot by slot [93], as given by

$$\text{Primal-problem: } \mathbf{x}^*(t) = \min_{\mathbf{x}(t)} \mathcal{L}(\mathbf{x}(t), \boldsymbol{\lambda}(t), \boldsymbol{\Omega}(t)); \tag{3.6a}$$

$$\text{Dual-problem: } \boldsymbol{\lambda}^*(t) = \max_{\boldsymbol{\lambda}(t)} D(\boldsymbol{\lambda}(t)); \tag{3.6b}$$

where $\mathbf{x}^*(t) = \{\tau_i^*(t), f_i^*(t), \forall i\}$ and $\boldsymbol{\lambda}^*(t)$ are the optimal variables and dual multipliers at slot t, respectively.

3.2.1 Hybrid Learning under Differently-aged Network States

The dual Lagrange multipliers $\boldsymbol{\lambda}(t)$ are updated slot by slot to maximize the optimal dual solution, i.e., $\max_{\boldsymbol{\lambda}(t)} D(\boldsymbol{\lambda}(t))$. SGD is an effective online learning method with extensive applications to support vector machines, logistic regression and artificial neural networks [94]. It is particularly useful in problem (3.4), where there exists strong temporal coupling and an exact gradient is intractable due to lack of the a-priori knowledge (i.e., the unpredictability) of the network dynamics. According to SGD [94], $\boldsymbol{\lambda}(t)$ can be updated by

$$\lambda_{i,1}(t+1) = \{\lambda_{i,1}(t) + \epsilon[A_i(t) - c_i(t)\tau_i^*(t) - f_i^*(t)]\}^+; \tag{3.7a}$$

$$\lambda_2(t+1) = \{\lambda_2(t) + \epsilon[\sum\nolimits_{i\in\mathbf{N}} c_i(t)\tau_i^*(t) - F(t)]\}^+; \tag{3.7b}$$

where ϵ is the stepsize of SGD and accounts for the optimality of stochastic gradient descent, as will be shown in Theorem 1.

Given $\boldsymbol{\lambda}(t)$, the optimal primal variables at slot t, $\mathbf{x}(t)$, the optimal decisions of computation offloading $\tau_i^*(t)$ and local processing $f_i^*(t)$, can be obtained by solving the Lagrangian in (3.5), as given by

$$\begin{aligned} &\min_{\mathbf{x}(t)} \gamma_t(\mathbf{f}(t)) + \eta_t(\boldsymbol{\tau}(t)) \\ &\text{s.t. } (3.1), (3.2), \end{aligned} \tag{3.8}$$

where the objective is obtained by reorganizing (3.5) with regards to the variables on computation offloading $\boldsymbol{\tau}(t) = \{\tau_i(t), \forall i\}$ and local processing $\mathbf{f}(t) = \{f_i(t), \forall i\}$, i.e., $\mathcal{L}(\mathbf{x}(t), \boldsymbol{\lambda}(t), \boldsymbol{\Omega}(t)) = \gamma_t(\mathbf{f}(t)) + \eta_t(\boldsymbol{\tau}(t))$, and

$$\gamma_t(\mathbf{f}(t)) = \sum\nolimits_{i\in\mathbf{N}} [\varepsilon_i(t) - \lambda_{i,1}(t)] f_i(t); \tag{3.9a}$$

$$\eta_t(\boldsymbol{\tau}(t)) = \sum\nolimits_{i\in\mathbf{N}} [p_i - \lambda_{i,1}(t)c_i(t)]\tau_i(t) + \lambda_2(t) \sum\nolimits_{i\in\mathbf{N}} c_i(t)\tau_i(t). \tag{3.9b}$$

Note that both the objective and constraints of (3.8) can be decoupled in terms of local processing and computation offloading decisions. As a result, primal problem

(3.8) can be decoupled into two subproblems of local processing and computation offloading decisions, as given by

$$\min_{\mathbf{f}(t)} \gamma_t(\mathbf{f}(t)) \text{ s.t. } (3.2); \tag{3.10a}$$

$$\min_{\boldsymbol{\tau}(t)} \eta_t(\boldsymbol{\tau}(t)) \text{ s.t. } (3.1). \tag{3.10b}$$

As described in Section3.1.1, given the stochastic nature of practical wireless channels, the network states for (3.10a) and (3.10b) are differently-aged at the devices and the BS.

A. Optimal Instantaneous Local Processing Decisions

The IoT devices have instantaneous observations of their own parameters and can solve (3.10a) for optimal local processing decisions. The local processing decisions can be decoupled among the devices in both the objective (3.9a) and constraint (3.2). As a result, problem (3.10a) can be decoupled to independently optimize the resource allocation at each device, as given by

$$\min[\varepsilon_i(t) - \lambda_{i,1}(t)]f_i(t), \text{ s.t. } 0 \le f_i(t) \le F_i(t). \tag{3.11}$$

Problem (3.11) is linear programming, and its optimal solution is given by

$$f_i^*(t) = \begin{cases} F_i(t), \text{ if } \lambda_{i,1}(t) \ge \varepsilon_i(t); \\ 0, \text{ otherwise.} \end{cases} \tag{3.12}$$

B. Predictive Computation Offloading via OCO

However, with only the outdated channel states $\boldsymbol{\omega}(t)$ on the last time slot $t-1$, the objective $\eta_t(\boldsymbol{\tau}(t))$ is unknown to the BS, and the optimal computation offloading decisions cannot be obtained from solving (3.10b). We propose to apply OCO to make predictive computation offloading decisions. OCO is an important online learning technique to make a sequence of accurate predictions over time, where the ground truth is only available after making the predictions [88–91, 95]. This is particularly useful for problem (3.10b) to predict the current channel states $\boldsymbol{\omega}(t)$ (and make online computation offloading decisions) from the previous reports of channel states.

The Follow-the-regularized-leader (FTRL) rule is an effective online prediction method in OCO, which uses the information in the previous decision round the in current decision-making with added regularization penalty for stabilization [95]. With a typical $\mathcal{L}_2$ norm regularization penalty, the FTRL rule can be equivalently reformulated as online gradient descent (OGD) of the primal variables [95]. In particular, the current decisions $\boldsymbol{\tau}(t)$ can be made based on the decisions at the last time slot $\boldsymbol{\tau}(t-1)$ and the online gradient $\nabla_{\boldsymbol{\tau}}^{\top}\eta_{t-1}(\boldsymbol{\tau}(t-1), \boldsymbol{\lambda}(t), \boldsymbol{c}(t-1))]$ to minimize the objective η_{t-1} at $\mathbf{x}(t)$. The update can be given by

$$\boldsymbol{\tau}(t) = \Pi_{\mathcal{T}}[\boldsymbol{\tau}(t-1) - \alpha\nabla_{\boldsymbol{\tau}}^{\top}\eta_{t-1}(\boldsymbol{\tau}(t-1), \boldsymbol{\lambda}(t), \boldsymbol{c}(t-1))], \tag{3.13}$$

where $\Pi_{\mathcal{T}}[\cdot]$ denotes the projection of $[\cdot]$ into the feasible domain of the primal variable $\mathcal{T}$ specified by constraint (3.1), and α is the stepsize of OGD.

The projection problem of OGD in (3.13) aims to find the point τ satisfying (3.1) which has the minimum distance to the point after gradient descent, i.e., $[\boldsymbol{\tau}(t-1) - \alpha\nabla_{\boldsymbol{\tau}}^{\top}\eta_{t-1}(\boldsymbol{\tau}(t-1), \boldsymbol{\lambda}(t), \boldsymbol{c}(t-1))]$. The distance minimization problem can be equivalently reformulated to a quadratic programming problem, as given by

$$\begin{aligned}\boldsymbol{\tau}(t) = \arg\min_{\boldsymbol{\tau}} & \nabla_{\boldsymbol{\tau}}^{\top}\eta_{t-1}(\boldsymbol{\tau}(t-1), \boldsymbol{\lambda}(t), \boldsymbol{c}(t-1)) \\ & (\boldsymbol{\tau} - \boldsymbol{\tau}(t-1)) + \frac{\|\boldsymbol{\tau} - \boldsymbol{\tau}(t-1)\|^2}{2\alpha} \\ & \text{s.t. } (3.1),\end{aligned} \tag{3.14}$$

By reorganizing the objective, (3.14) can be rewritten as

$$\begin{aligned}&\min_{\tau} \sum\nolimits_{i\in\mathbf{N}} \frac{\tau_i^2}{2\alpha} + \eta_i(t)\tau_i \\ &\text{s.t. } \tau_i \geq 0, \forall i, \ \sum\nolimits_{i\in\mathbf{N}} \tau_i \leq T.\end{aligned} \tag{3.15}$$

where

$$\eta_i(t) = p_i + [\lambda_2(t) - \lambda_{i,1}(t)]c_i(t-1) - \frac{\tau_i(t-1)}{\alpha}. \tag{3.16}$$

Problem (3.15) is also quadratic programming with strong duality, and its Lagrangian can be given by

$$L(\boldsymbol{\tau}, \mu) = \sum\nolimits_{i\in\mathbf{N}} \big[\frac{\tau_i^2}{2\alpha} + \eta_i(t)\tau_i\big] + \mu(\sum\nolimits_{i\in\mathbf{N}} \tau_i - T). \tag{3.17}$$

The dual problem of (3.15) can be written as $\max\limits_{\mu>0}\min\limits_{\boldsymbol{\tau}\succeq 0} L(\boldsymbol{\tau}, \mu)$. Given the optimal multiplier μ^*, the optimal offloading schedule can be obtained by solving the primal problem $\min\limits_{\boldsymbol{\tau}\succeq 0} L(\boldsymbol{\tau}, \mu)$, as given by

$$\tau_i(t) = \big[-\alpha(\eta_i(t) + \mu)\big]^+. \tag{3.18}$$

The optimal multiplier μ^* can be efficiently solved by bisection search [96] between the interval $[0, \mu^{\max}]$, where $\mu^{\max} = -\min_{i\in\mathbf{N}} \eta_i(t)$ with fast convergence.

Note that the typical $[\mathcal{O}(1/\epsilon), \mathcal{O}(\epsilon)]$-tradeoff between convergence time and optimality loss in terms of the stepsize ϵ indicates that an $\mathcal{O}(1/\epsilon)$ convergence time allows for an $\mathcal{O}(\epsilon)$ close-to-optimal cost. This reveals that the system performance converges to within an optimality bound of $\mathcal{O}(\epsilon)$, which diminishes as the stepsize of stochastic gradient descent $\epsilon \to 0$. Here, the convergence time is represented by the queue backlog of devices. To achieve the asymptotic optimality of system, the stepsize ϵ will take small values, resulting in the long queue backlog of devices. In this case, the backlogged tasks in the queue are long enough to prevent being cleared after locally computation and offloading.

3.2.2 Implementation of Hybrid Learning Framework

The proposed hybrid learning framework under differently-aged network states is summarized in Algorithm 1, where the Lagrange multipliers $\lambda_{i,1}$ and λ_2 are maintained by device i and the BS, respectively. The decisions on local processing and computation offloading are decentralized and made at the devices and the BS, respectively, for scalability and learning accuracy. At the beginning of each time slot

Algorithm 1 The Proposed Hybrid Learning Framework under Differently-aged Network States

```
For each device $i$:
1: Observes its own instantaneous information on computing parameters and task arrivals;
2: Locally optimizes $f_i^*(t)$ by solving (3.10a) based on (3.12);
3: Updates its dual multiplier $\lambda_{i,1}(t)$ by (3.7a);
4: Reports its channel condition $c_i(t-1)$ at the last time slot and the current multiplier $\lambda_{i,1}(t)$ to the BS;
For the BS:
5: Receives the reports from the devices of the outdated channel conditions;
6: Predictive schedules the computation offloading via OCO by solving (3.13) according to (3.18);
7: Updates the multiplier $\lambda_2(t)$ by (3.7b).
```

t, each device i makes its local processing decision based on its own instantaneous observation $\boldsymbol{\omega}_i(t)$ and Lagrange multipliers $\lambda_{i,1}(t)$ in a fully distributed manner. The BS makes the computation offloading decision based on the outdated channel conditions $\boldsymbol{\omega}(t)$ and $\lambda_{i,1}(t)$ collected from each device, and its own Lagrange multiplier $\lambda_2(t)$. This is achieved by separately pursuing the instantaneous local processing subproblem (3.10a) and the predictive computation offloading subproblem (3.13). The BS then notifies the predictively selected devices of the offloading decisions, receives their transmissions, and collects the channel states of all the devices at the end of the time slot.

3.3 Performance and Optimality Analysis

In this section, we prove that the proposed hybrid learning approach under differently-aged network states is asymptotically optimal. The optimality loss resulting from the outdated channel states, can asymptotically diminish by reducing the stepsize of SGD and OCO. The proposed hybrid learning approach is the orchestration of SGD and OCO in a primal-dual optimization framework via Lagrange duality. As a result, we take two steps to evaluate the performance of the hybrid learning approach (as compared to the offline optimum Φ^* for (3.4), which is minimized offline in a posteriori manner and violates causality):

A. Optimality of Primal-dual Framework under Instantaneous States

We first evaluate the gap between the solution for the proposed primal-dual framework under instantaneous network states, denoted by $\mathbf{x}^*(t)$, and the offline optimum Φ^*, where $\mathbf{x}^*(t) = \{\boldsymbol{f}^*(t), \boldsymbol{\tau}^*(t)\}$ is the solution by solving (3.10) under instantaneous channel states and local processing parameters. The gap is proved to diminish asymptotically with the decreasing stepsize ϵ, as will be shown in Theorem 1.

Theorem 1. *The gap between $\Phi(\mathbf{x}^*(t))$ and Φ^* satisfies*

$$\Phi(\mathbf{x}^*(t)) - \Phi^* \leq \epsilon U, \tag{3.19}$$

where $U = \frac{1}{2}\{\sum_{i\in\mathbf{N}}(\max\{A_i^{\max}, [c_i^{\max}T + f_i^{\max}]\})^2 + (\max\{\sum_{i\in\mathbf{N}} c_i^{\max}T, F^{\max}\})^2\}$ is a constant, and ϵ is the stepsize of SGD.

Proof. We start by bounding the variations of the dual multipliers $\lambda(t)$. In light of Lyapunov drift analysis [31], a quadratic drift $\Delta(\boldsymbol{\lambda}(t))$ is introduced to evaluate the variation of the multipliers, which is shown to be upper bounded in the following lemma.

Lemma 1. *At each time slot t, the dual drift of $\Delta(\boldsymbol{\lambda}(t)) = \frac{1}{2}(\|\boldsymbol{\lambda}(t+1)\|^2 - \|\boldsymbol{\lambda}(t)\|^2)$ is upper bounded, as given by*

$$\Delta(\boldsymbol{\lambda}(t)) \leq \frac{\epsilon^2}{2}U + \epsilon\sum_{i\in\mathbf{N}}\lambda_{i,1}(t)[A_i(t) - c_i(t)\tau_i(t) - f_i(t)] + \epsilon\lambda_2(t)[\sum_{i\in\mathbf{N}}c_i(t)\tau_i(t) - F(t)]. \tag{3.20}$$

where $U = \frac{1}{2}\{\sum_{i\in\mathbf{N}}(\max\{A_i^{\max}, [c_i^{\max}T + f_i^{\max}]\})^2 + (\max\{\sum_{i\in\mathbf{N}} c_i^{\max}T, F^{\max}\})^2\}$.

Proof. Squaring on both sides of (3.7a) and (3.7b), we have

$$\begin{aligned}\lambda_{i,1}(t+1)^2 =& \lambda_{i,1}(t)^2 + [\epsilon(A_i(t) - c_i(t)\tau_i(t) - f_i(t))]^2 \\ &+ 2\epsilon\lambda_{i,1}(t)(A_i(t) - c_i(t)\tau_i(t) - f_i(t));\end{aligned} \tag{3.21a}$$

$$\begin{aligned}\lambda_2(t+1)^2 =&\lambda_2(t)^2 + [\epsilon(\sum\nolimits_{i\in\mathbf{N}} c_i(t)\tau_i(t) - F(t))]^2 \\ &+ 2\epsilon\lambda_2(t)(\sum\nolimits_{i\in\mathbf{N}} c_i(t)\tau_i(t) - F(t)).\end{aligned} \tag{3.21b}$$

By substituting (3.21) into the definition of $\Delta(\boldsymbol{\lambda}(t))$, we have

$$\begin{aligned}\Delta(\boldsymbol{\lambda}(t)) = &\frac{\epsilon^2}{2}\sum_{i\in\mathbf{N}}[A_i(t) - c_i(t)\tau_i(t) - f_i(t)]^2 + \frac{\epsilon^2}{2}[\sum_{i\in\mathbf{N}} c_i(t)\tau_i(t) - F(t)]^2 + \epsilon\sum_{i\in\mathbf{N}}\lambda_{i,1}(t)[A_i(t) \\ &- c_i(t)\tau_i(t) - f_i(t)] + \epsilon\lambda_2(t)[\sum_{i\in\mathbf{N}} c_i(t)\tau_i(t) - F(t)] \\ &\overset{(a)}{\leq} \epsilon^2 U + \epsilon\sum_{i\in\mathbf{N}}\lambda_{i,1}(t)[A_i(t) - c_i(t)\tau_i(t) - f_i(t)] + \epsilon\lambda_2(t)[\sum_{i\in\mathbf{N}} c_i(t)\tau_i(t) - F(t)],\end{aligned} \tag{3.22}$$

where inequality (a) is because $[A_i(t) - c_i(t)\tau_i(t) - f_i(t)]^2 \leq (\max\{A_i^{\max}, [c_i^{\max}T + f_i^{\max}]\})^2$, and $[\sum_{i\in\mathbf{N}} c_i(t)\tau_i(t) - F(t)]^2 \leq (\max\{\sum\limits_{i\in\mathbf{N}} c_i^{\max}T, F^{\max}\})^2$. □

Divide both sides of (3.20) by ϵ, then take expectations over $\boldsymbol{\Omega}(t)$, and add $\mathbb{E}[\Phi(\mathbf{x}^*(t))]$ on both sides of (3.20) (where $\mathbf{x}^*(t) = \{\boldsymbol{f}^*(t), \boldsymbol{\tau}^*(t)\}$ is the optimal policy by solving (3.10) with instantaneous network dynamics $\boldsymbol{\Omega}(t)$). We can obtain

$$\begin{aligned}&\frac{\Delta(\boldsymbol{\lambda}(t))}{\epsilon} + \mathbb{E}[\Phi(\mathbf{x}^*(t))] \\ \leq&\epsilon U + \sum_{i\in\mathbf{N}}\lambda_{i,1}(t)[A_i(t) - c_i(t)\tau_i^*(t) - f_i^*(t)] \\ &+ \lambda_2(t)[\sum_{i\in\mathbf{N}} c_i(t)\tau_i^*(t) - F(t)] + \mathbb{E}[\Phi(\mathbf{x}^*(t))] \\ =&\epsilon U + \mathbb{E}\big[\mathcal{L}(\mathbf{x}^*(t), \boldsymbol{\lambda}(t), \boldsymbol{\Omega}(t))\big] \\ =&\epsilon U + \mathcal{D}(\boldsymbol{\lambda}(t)) \\ \leq&\epsilon U + \Phi^*,\end{aligned} \tag{3.23}$$

where $\mathcal{L}(\mathbf{x}^*(t), \boldsymbol{\lambda}(t), \boldsymbol{\Omega}(t))$ is defined in (3.5); $\mathbf{x}^*(t)$ is the optimal primal variable; $\mathbb{E}\big[\mathcal{L}(\mathbf{x}^*(t), \boldsymbol{\lambda}(t), \boldsymbol{\Omega}(t))\big] = \mathcal{D}(\boldsymbol{\lambda}(t))$; and the last inequality in (3.23) is due to the weak duality that $\mathcal{D}(\boldsymbol{\lambda}(t)) \leq \Phi^*$ [93].

By summing up (3.23) from time slots $t = 1$ to $t = \mathbb{T}$, we have

$$\frac{\|\boldsymbol{\lambda}(\mathbb{T}+1)\|^2}{2\epsilon} - \frac{\|\boldsymbol{\lambda}(1)\|^2}{2\epsilon} + \sum_{t=1}^{t=\mathbb{T}}\mathbb{E}[\Phi(\mathbf{x}^*(t))] \leq \epsilon U\mathbb{T} + \Phi^*\mathbb{T}. \tag{3.24}$$

Because $\frac{\|\boldsymbol{\lambda}(\mathbb{T}+1)\|^2}{2\epsilon} \geq 0$ and $\frac{\|\boldsymbol{\lambda}(1)\|^2}{2\epsilon} < \infty$, we have

$$\Phi^*(\mathbf{x}^*(t)) = \frac{1}{\mathbb{T}} \lim_{\mathbb{T}\to\infty} \sum_{t=1}^{t=\mathbb{T}} \mathbb{E}[\Phi(\mathbf{x}^*(t))] \leq \epsilon U + \Phi^*. \tag{3.25}$$

This concludes the proof. □

Theorem 1 reveals that the time-average energy consumption, achieved by the primal-dual framework with SGD dual learning, converges to within an optimality bound of $\mathcal{O}(\epsilon)$, which diminishes as the stepsize of SGD $\epsilon \to 0$. The stepsize ϵ also accounts for the convergence time of the proposed learning process. Given the stepsize ϵ, the convergence time of SGD linearly increases with $\mathcal{O}(1/\epsilon)$ [94]. The typical $[\mathcal{O}(1/\epsilon), \mathcal{O}(\epsilon)]$-tradeoff between the convergence time and the optimality loss in terms of ϵ indicates that an $\mathcal{O}(1/\epsilon)$ convergence time allows for an $\mathcal{O}(\epsilon)$ close-to-optimal solution.

B. Optimality of OCO-based Predictive Offloading under Differently-aged Network States

Next, we assess the optimality loss of the predictive computation offloading decisions, i.e., $\mathbf{x}(t) = \{\mathbf{f}^*(t), \boldsymbol{\tau}(t)\}$ in Section 3.2.2, under the outdated states of the wireless channels, as compared to the solution under instantaneous channel states $\mathbf{x}^*(t)$. The gap between $\Phi(\mathbf{x}(t))$ and $\Phi(\mathbf{x}^*(t))$ is proved to diminish asymptotically with the decreasing OCO stepsize α under sublinear penalty terms, as will be shown in Theorem 2.

Theorem 2. *The optimality loss of the OCO-based predictive computation offloading, resulting from outdated channel states, i.e., the gap between $\Phi(\mathbf{x}(t))$ and $\Phi(\mathbf{x}^*(t))$, satisfies*

$$\Phi(\mathbf{x}(t)) - \Phi(\mathbf{x}^*(t)) \leq \frac{R\mathcal{V}(\{\boldsymbol{\tau}^*(t)\}_{t=1}^{\mathbb{T}})}{\alpha\mathbb{T}} + \frac{\alpha G^2}{2}, \tag{3.26}$$

where R and G are constants, $\mathcal{V}(\{\boldsymbol{\tau}^*(t)\}_{t=1}^{\mathbb{T}}) = \sum_{t=i}^{\mathbb{T}} \|\boldsymbol{\tau}^*(t) - \boldsymbol{\tau}^*(t-1)\|$ is the accumulated variation of the optimal offloading schedules $\boldsymbol{\tau}^*(t)$ under instantaneous channel states, and α is the stepsize of OCO.

Proof. Note that the local data processing decision in the proposed hybrid learning approach is also based on the instantaneous observations (i.e., same as the optimality analysis under instantaneous network states in Theorem 1). As a result, the per-slot optimality loss of the hybrid approach only comes from the predictive offloading schedules based on outdated channel conditions (compared to that under instantaneous channel conditions), i.e.,

$$\Phi(\mathbf{x}(t)) - \Phi(\mathbf{x}^*(t)) = \frac{1}{\mathbb{T}} \lim_{\mathbb{T}\to\infty} \sum_{t=1}^{t=\mathbb{T}} \mathbb{E}\big[\eta_t(\boldsymbol{\tau}(t)) - \eta_t(\boldsymbol{\tau}^*(t))\big]. \tag{3.27}$$

Given Lemma 2, we can establish the relationship between the proposed solution $\boldsymbol{\tau}(t)$ achieved by the proposed hybrid online learning approach and the optimal solution $\boldsymbol{\tau}^*(t)$ under instantaneous channel conditions, as given in the following Lemma.

Lemma 2. *Let $\eta_t(\boldsymbol{\tau}(t))$ denote the utility of predictive computation offloading through the proposed hybrid online learning approach at slot t, and $\eta_t(\boldsymbol{\tau}^*(t))$ denote the objective achieved by the optimal policy in the presence of instantaneous channel states. We have*

$$\eta_t(\boldsymbol{\tau}(t)) \leq \eta_t(\boldsymbol{\tau}^*(t)) + \frac{R\|\boldsymbol{\tau}^*(t) - \boldsymbol{\tau}^*(t-1)\|}{\alpha} + \Psi(t+1) + \frac{\alpha G^2}{2}, \tag{3.28}$$

where $\Psi(t+1) = \frac{\|\boldsymbol{\tau}(t)-\boldsymbol{\tau}^*(t-1)\|^2}{2\alpha} - \frac{\|\boldsymbol{\tau}^*(t)-\boldsymbol{\tau}(t+1)\|^2}{2\alpha}$

Proof. The objective of (3.14) can be equivalently interpreted as $\min_{\boldsymbol{\tau}} \nu(\boldsymbol{\tau}(t)) = \nabla_{\boldsymbol{\tau}}^{\top}\eta(\boldsymbol{\tau}(t-1), \boldsymbol{\lambda}(t), \boldsymbol{\Omega}(t-1))(\boldsymbol{\tau} - \boldsymbol{\tau}(t-1)) - \frac{\|\boldsymbol{\tau}-\boldsymbol{\tau}(t-1)\|^2}{2\alpha}$. Given the strong convexity of ν, there exists $\beta \geq 0$ satisfying $\nu(\boldsymbol{\tau}(t+1)) \leq \nu(\boldsymbol{\tau}^*(t)) - \frac{\beta}{2}\|\boldsymbol{\tau}(t+1) - \boldsymbol{\tau}^*(t)\|^2$ [97, Cor. 1], where $\boldsymbol{\tau}(t+1)$ (i.e., the solution to (3.14) in the proposed approach) is the global optimum of ν at slot $(t+1)$. By using variable substitution, i.e., setting $\beta = \frac{1}{\alpha}$ in the above inequality, and defining $\Psi'(t+1) = \frac{\|\boldsymbol{\tau}^*(t)-\boldsymbol{\tau}(t)\|^2}{2\alpha} - \frac{\|\boldsymbol{\tau}^*(t)-\boldsymbol{\tau}(t+1)\|^2}{2\alpha}$, we have

$$\begin{aligned}&\nabla_{\boldsymbol{\tau}}^{\top}\eta_t(\boldsymbol{\tau}(t))[\boldsymbol{\tau}(t+1)-\boldsymbol{\tau}(t)]+\frac{\|\boldsymbol{\tau}(t+1)-\boldsymbol{\tau}(t)\|^2}{2\alpha}\\ \le&\nabla_{\boldsymbol{\tau}}^{\top}\eta_t(\boldsymbol{\tau}(t))[\boldsymbol{\tau}^*(t)-\boldsymbol{\tau}(t)]+\Psi'(t+1).\end{aligned} \tag{3.29}$$

Adding $\eta_t(\boldsymbol{\tau}(t))$ on both sides of (3.29) leads to

$$\begin{aligned}&\eta_t(\boldsymbol{\tau}(t))+\nabla_{\boldsymbol{\tau}}^{\top}\eta_t[\boldsymbol{\tau}(t+1)-\boldsymbol{\tau}(t)]+\frac{\|\boldsymbol{\tau}(t+1)-\boldsymbol{\tau}(t)\|^2}{2\alpha}\\ \le&\eta_t(\boldsymbol{\tau}(t))+\nabla_{\boldsymbol{\tau}}^{\top}\eta_t[\boldsymbol{\tau}^*(t)-\boldsymbol{\tau}(t)]+\Psi'(t+1)\\ \overset{(a)}{\le}&\eta_t(\boldsymbol{\tau}^*(t))+\Psi'(t+1),\end{aligned} \tag{3.30}$$

where inequality (a) is due to the fact that $\eta_t(\boldsymbol{\tau}(t))+\nabla_{\boldsymbol{\tau}}^{\top}\eta_t[\boldsymbol{\tau}^*(t)-\boldsymbol{\tau}(t)]\le\eta_t(\boldsymbol{\tau}^*(t))$ given the convexity of η_t. By rearranging (3.30), we have

$$\begin{aligned}&\eta_t(\boldsymbol{\tau}(t))\le\eta_t(\boldsymbol{\tau}^*(t))+\Psi'(t+1)-\nabla_{\boldsymbol{\tau}}^{\top}\eta_t[\boldsymbol{\tau}(t+1)-\boldsymbol{\tau}(t)]-\frac{\|\boldsymbol{\tau}(t+1)-\boldsymbol{\tau}(t)\|^2}{2\alpha}\\ \overset{(a)}{\le}&\eta_t(\boldsymbol{\tau}^*(t))+\Psi'(t+1)+\frac{\alpha G^2}{2},\end{aligned} \tag{3.31}$$

where inequality (a) is because $-\nabla_{\boldsymbol{\tau}}^{\top}\eta_t[\boldsymbol{\tau}(t+1)-\boldsymbol{\tau}(t)]\le\|\nabla_{\boldsymbol{\tau}}\eta_t(\boldsymbol{\tau}(t))\|\|[\boldsymbol{\tau}(t+1)-\boldsymbol{\tau}(t)\|\le\frac{\alpha\|\nabla_{\boldsymbol{\tau}}\eta_t(\boldsymbol{\tau}(t))\|^2}{2}+\frac{\|\boldsymbol{\tau}(t+1)-\boldsymbol{\tau}(t)\|^2}{2\alpha}$, and $\|\nabla_{\boldsymbol{\tau}}\eta_t\|^2\le N(\max\{p^{\max},|\lambda_{i,1}|^{\max}c^{\max}\})^2=G^2$.

Note that

$$\begin{aligned}&2\alpha\Psi'(t+1)=\|\boldsymbol{\tau}^*(t)-\boldsymbol{\tau}(t)\|^2-\|\boldsymbol{\tau}^*(t)-\boldsymbol{\tau}(t+1)\|^2\\ =&\|\boldsymbol{\tau}^*(t)-\boldsymbol{\tau}(t)\|^2-\|\boldsymbol{\tau}(t)-\boldsymbol{\tau}^*(t-1)\|^2+\|\boldsymbol{\tau}(t)-\boldsymbol{\tau}^*(t-1)\|^2-\|\boldsymbol{\tau}^*(t)-\boldsymbol{\tau}(t+1)\|^2\\ =&\|\boldsymbol{\tau}^*(t)-\boldsymbol{\tau}*(t-1)\|\|\boldsymbol{\tau}^*(t)+\boldsymbol{\tau}^*(t-1)-2\boldsymbol{\tau}(t)\|\\ &+\|\boldsymbol{\tau}(t)-\boldsymbol{\tau}^*(t-1)\|^2-\|\boldsymbol{\tau}^*(t)-\boldsymbol{\tau}(t+1)\|^2\\ \overset{(a)}{\le}&2R\|\boldsymbol{\tau}^*(t)-\boldsymbol{\tau}^*(t-1)\|+\|\boldsymbol{\tau}(t)-\boldsymbol{\tau}^*(t-1)\|^2-\|\boldsymbol{\tau}^*(t)-\boldsymbol{\tau}(t+1)\|^2\\ =&2R\|\boldsymbol{\tau}^*(t)-\boldsymbol{\tau}^*(t-1)\|+2\alpha\Psi(t+1).\end{aligned} \tag{3.32}$$

where inequality (a) is obtained since $\|\boldsymbol{\tau}^*(t)+\boldsymbol{\tau}^*(t-1)-2\boldsymbol{\tau}(t)\|\le\|\boldsymbol{\tau}^*(t)-\boldsymbol{\tau}(t)\|+\|\boldsymbol{\tau}(t)-\boldsymbol{\tau}^*(t-1)\|\le 2R$. Here, $R^2=NT^2$ and $\|\boldsymbol{\tau}(t)-\boldsymbol{\tau}(t)\|\le R$ holds for any $\boldsymbol{\tau}(t)$

satisfying (3.1). Therefore, (3.32) can be rewritten as

$$\Psi'(t+1) \leq \frac{R\|\boldsymbol{\tau}^*(t) - \boldsymbol{\tau} * (t-1)\|}{\alpha} + \Psi(t+1). \tag{3.33}$$

Finally, (3.28) can be proved by substituting (3.33) into (3.31). □

By summing up (3.28) from time slots $t = 1$ to $t = \mathbb{T}$, and taking expectation over $\Omega(t)$, we can obtain

$$\begin{aligned} &\sum_{t=1}^{t=\mathbb{T}} \mathbb{E}\big[\eta_t(\boldsymbol{\tau}(t))\big] \\ \leq &\sum_{t=1}^{t=\mathbb{T}} \mathbb{E}\big[\eta_t(\boldsymbol{\tau}^*(t)) + \frac{R\|\boldsymbol{\tau}^*(t) - \boldsymbol{\tau}^*(t-1)\|}{\alpha} + \Psi(t+1) + \frac{\alpha G^2}{2}\big] \\ \overset{(a)}{\leq} &\sum_{t=1}^{t=\mathbb{T}} \mathbb{E}\big[\eta_t(\boldsymbol{\tau}^*(t))\big] + \frac{R\mathcal{V}(\{\tau^*(t)\}_{t=1}^{\mathbb{T}})}{\alpha} + \frac{\alpha G^2 \mathbb{T}}{2} + \frac{\|\boldsymbol{\tau}(1) - \boldsymbol{\tau}^*(0)\|^2}{2\alpha}, \end{aligned} \tag{3.34}$$

where the inequality (a) is obtained by plugging $\Psi(t+1)$ in Lemma 2 into the inequality and then removing the negative terms.

By rearranging (3.34), dividing both sides of (3.34) by $\mathbb{T}$, and taking $\mathbb{T} \to \infty$, we have

$$\lim_{\mathbb{T}\to\infty} \frac{1}{\mathbb{T}} \sum_{t=1}^{t=\mathbb{T}} \mathbb{E}\big[\eta_t(\boldsymbol{\tau}(t)) - \eta_t(\boldsymbol{\tau}^*(t))\big] \leq \frac{R\mathcal{V}(\{\tau^*(t)\}_{t=1}^{\mathbb{T}})}{\alpha\mathbb{T}} + \frac{\alpha G^2}{2}, \tag{3.35}$$

where the inequality is due to the fact that $|\boldsymbol{\lambda}(2)\|^2 < \infty$ and $\|\boldsymbol{\tau}(1) - \boldsymbol{\tau}^*(0)\|^2 < \infty$. By plugging (3.35) into (3.27), we have

$$\Phi(\mathbf{x}(t)) - \Phi(\mathbf{x}^*(t)) \leq \frac{R\mathcal{V}(\{\tau^*(t)\}_{t=1}^{\mathbb{T}})}{\alpha\mathbb{T}} + \frac{\alpha G^2}{2}. \tag{3.36}$$

This concludes the proof. □

Theorem 2 reveals that the time-average utility of the proposed hybrid learning approach can converge asymptotically to the online learning optimum, when the $\mathcal{V}(\{\boldsymbol{\tau}^*(t)\}_{t=1}^{\mathbb{T}})$ is sublinear. The optimality loss B_{hybrid} can asymptotically diminish as the stepsize $\alpha \to 0$.

By combining the findings in Theorems 1 and 2, we can establish that the proposed hybrid learning approach can asymptotically converge to the offline optimum, as given by

$$\Phi(\mathbf{x}(t)) - \Phi^* \leq B_{\text{hybrid}}, \tag{3.37}$$

where the optimality loss of the proposed online learning approach, i.e., $B_{\text{hybrid}} = \epsilon U + \frac{R\mathcal{V}(\{\boldsymbol{\tau}^*(t)\}_{t=1}^{\mathbb{T}})}{\alpha \mathbb{T}} + \frac{\alpha G^2}{2}$, can asymptotically diminish, as $\epsilon \to 0$ and $\alpha \to 0$.

3.3.1 Simulation Results and Analysis

In this section, we evaluate the proposed hybrid learning approach with $N = 30$ devices and an edge server, where we run 5000 slots with duration $T = 10$ms. The computing resources of the BS is 50Mbits/sec. The available computing resources of the devices are uniformly and randomly distributed from 0 to 1Mbps, and the energy consumption for locally processing unit size of data (per Mbits) at the devices is randomly and uniformly distributed from 0.08 to 0.12mW. The transmit power of the devices is 200mW (i.e., 23dBm) [98]. The volume of data arriving at devices varies randomly and uniformly from 0 to 1Mps. The stepsize of SGD is $\epsilon = 0.001$ and the stepsize of OCO is $\alpha = 0.1$; unless otherwise specified.

For comparison purpose, we also simulate two other approaches:

1. Online convex optimization approach (denoted by "OCO" in the simulation), where the devices do not observe the devices' information at each time slot. All the decisions of slot t are made based the network states $\boldsymbol{\Omega}(t-1)$ of the last time slot through online convex optimization [88].

2. Round-robin approach (denoted by "Round-robin" in the simulation), where the devices do not observe the devices' information at each time slot, and the computation offloading decisions at slot t are made in a round-robin manner. The devices process the remaining tasks locally after offloading.

3. Lyapunov optimization approach [18] (denoted by "Lya"), where the devices can observe the devices' information at each time slot, but the computation offloading decisions at slot t are made based on the outdated wireless channel conditions at the last time slot $(t-1)$.

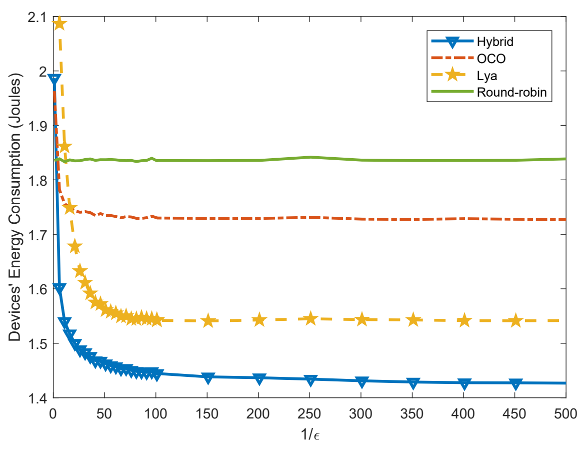


Figure 3.2 : The stabilized system energy consumption versus the reciprocal of the stepsize of SGD, $1/\epsilon$.

We would like to simulate the synchronized information on network status. However, the simulation in the ideal case requires the offline searching of large sizes of actions across a large number of time slots (e.g., 5000 time slots and N=30 IoT devices), which is computationally prohibitive to be simulated in practice. We have clarified that the proposed algorithms are mathematically proved to be asymptotically optimal, as stated in Theorem 2. In other words, the performance of the proposed approaches can asymptotically converge to that of the ideal cases.

Fig. 3.2 plots the devices' energy consumption achieved by the proposed and benchmark approaches, as the reciprocal of the stepsize of stochastic gradient descent, i.e., $1/\epsilon$ increases from 0 to 500. We can see that the device energy consumptions of the proposed hybrid learning, "Lya" approach, and "OCO" approach first decrease and then stabilizes of $1/\epsilon$ increases. This validates the asymptotic optimality of the proposed approach in Theorem 1. The energy consumption of "Lya" and "OCO" approach are larger than the proposed approach, since they lack the capabilities of either predicting channel conditions for computation offloading or exploiting local observations for real-time decisions.

Fig. 3.3 plots the changes of Lagrange multipliers of the proposed approach as t evolves. We can see that the values of Lagrange multipliers under different $1/\epsilon$ all first increase and stabilize at the same value over time. As the stepsize decreases

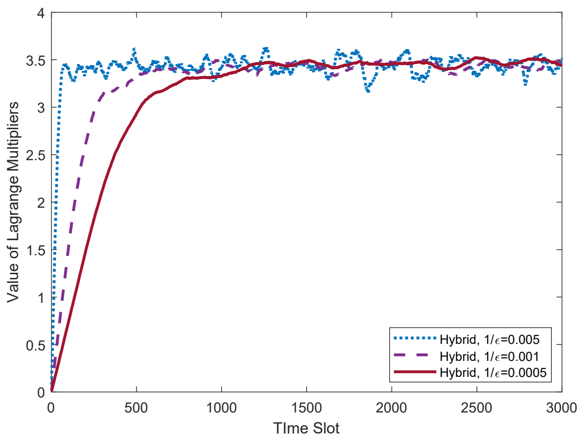


Figure 3.3 : The value of Lagrange multipliers as time evolves.

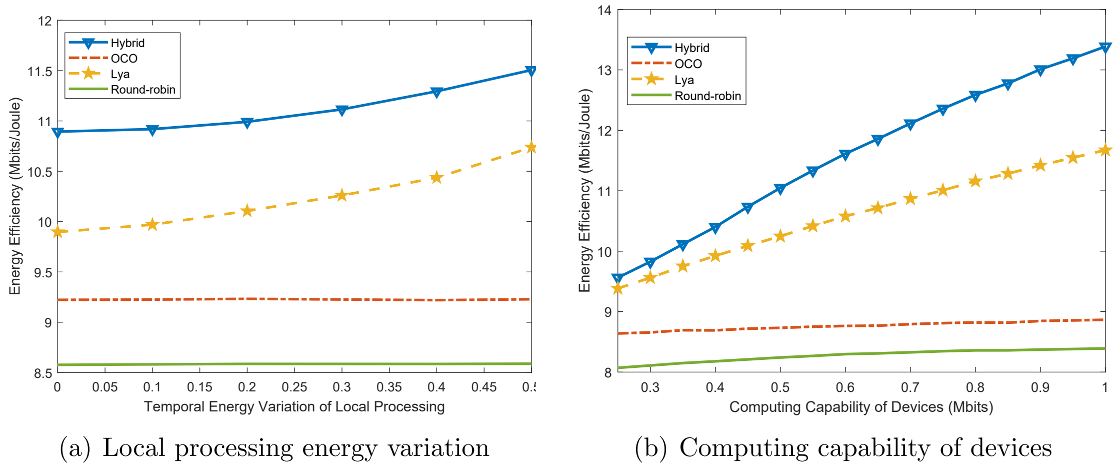


Figure 3.4 : The device energy efficiency as the energy variation of local processing and devices' computing capability increases. The proposed approach can exploit the diversity of devices for processing to improve the energy efficiency.

from $1/\epsilon = 500$ to $1/\epsilon = 2000$, the proposed hybrid learning approach requires increasingly long convergence times to stabilize the system, but there would be increasingly small fluctuation of the multipliers after stabilization and also decrease of the energy consumption, as shown in Fig.3.2. This is the typical between the convergence time and optimality loss as dictated in Theorem 1.

Fig.3.4(a) plots the device energy efficiency achieved by the proposed and benchmark approaches as the temporal energy variation of local processing increases. We can see in Fig.3.4(a) that the energy efficiency of the proposed approach and "Lya" approach increase with the energy variation of local processing. This is because these two approaches with local observation can exploit the diversity gain results from the energy variation of local processing. They have more chances to find a lower value of $\varepsilon_i(t)$ for local processing with the increase of energy variation of local processing. Since the proposed approach can predict the wireless channel conditions and has more diversity for processing, the energy efficiency of the proposed approach is better than the "Lya" approach. On the other hand, the "OCO" approach without local observation capability can only learn the average value of $\varepsilon_i(t)$. In this case, the "OCO" approach can not exploit the diversity when the average value of $\varepsilon_i(t)$ is time-invariant, and the energy efficiency of "OCO" approach does not change with the increase of energy variation of local processing. "Round-robin" approach does not have any knowledge on the devices and wireless information, therefore the energy efficiency does not change.

Fig.3.4(b) plots the device energy efficiency achieved by the proposed and benchmark approaches as the devices' computing capability increases. We can see in Fig.3.4(b) that the energy efficiency of all the approaches increase with the devices' computing capability, but the energy efficiency of the proposed approach and "Lya" approach increase rapidly. This is because the increasing computing capability of devices provides the diversity for processing. The proposed approach and "Lya" approach can observe the real-time energy cost for local processing, and process more tasks in low values of $\varepsilon_i(t)$ with the increase of devices' computing capability. Since the proposed approach can predict the wireless channel conditions and has more diversity for processing, the energy efficiency of the proposed approach is better than the "Lya" approach. On the other hand, since the "OCO" approach and "Round-robin" approach can not observe the devices' information, the increase of devices' computing capability has slight impact on the decisions of local processing.

Fig.3.5 shows the devices' energy efficiency achieved by the proposed and benchmark approaches as the channel variation increases. Recall that the average channel capacity follows a uniform distribution within $[1.4(1-\rho), 1.4(1+\rho)]$Mbps. Here, ρ is the variation percentage of the average channel capacities across the devices and

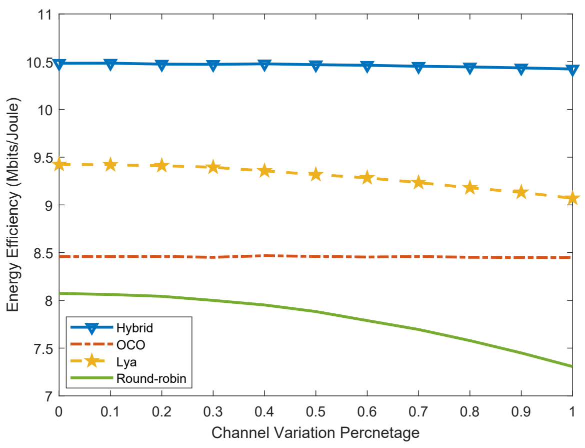


Figure 3.5 : The device energy efficiency as the local processing energy and channel variation increases. The proposed approach can exploit the diversity of devices and channels to improve the energy efficiency.

can be adopted to measure the channel heterogeneity. We can see in Fig.3.5 that the energy efficiency of the proposed approach and "OCO" approach does not change with the variation of ρ. This is because these approached can learn the average channel capacities of the devices from feedback. In this case, the energy efficiency of these approaches do not change given the time-invariant average of channel capacities. On the other hand, "Lya" approach and "Round-robin" approach can not predict the channel capacity, thereby decreasing the energy efficiency.

3.3.2 Discussion on Delay-Sensitive Tasks

The typical $[\mathcal{O}(1/\epsilon); \mathcal{O}(\epsilon)]$-tradeoff indicates that an $\mathcal{O}(1/\epsilon)$ optimality loss can be be achieved with an $\mathcal{O}(\epsilon)$ convergence time. This allows to leverage between convergence time and optimality loss. We can reduce the delay of tasks by increasing the stepsize ϵ and the optimality loss.

For illustration convenience, we assume all tasks are of the same priority, and the queues operate on a simple FIFO basis. This may cause unfairness (i.e., the variations of task delays), as a head-of-line (HOL) task of a queue can be offloaded, become the bottom of another FIFO queue, and hence undergoes a significantly increased queuing delay. The proposed approach can be extended to capture the execution delays and fairness (i.e., the delay variations) of the tasks by setting priority queues with meticulously designed weights. Specifically, we can design the weight of a task (and its corresponding result) to be the age of the task (in time slots). The age of the task grows until the delivery of the result. All the queues operate as priority queues, where tasks can be arranged in the descending order of task age and the oldest tasks are placed at the HOL. The priority queues would not affect the asymptotic optimality of the proposed approach. This is because the proposed approach only depends on the Lagrange multipliers (or in other words, the queue lengths) which are unaffected by the changing order of the tasks within the queues.

Chapter 4

Distributed Online Learning of Fog Computing under Non-uniform Device Cardinality

Distributed online optimization is important given the size of IoT, but challenging due to time variations of random traffic and non-uniform connectivity (or cardinality) of edge servers and IoT devices. This chapter presents a distributed online learning approach to asymptotically minimizing the time-average cost of fog computing in the absence of the a-priori knowledge on traffic randomness, for light-weight, delay-tolerant application scenarios. The proposed approach is able to address non-uniform node cardinalities in the absence of the a-priori network knowledge. The key contributions of this chapter can be summarized as follows.

- By exploiting stochastic gradient descent, we decouple the optimizations between time slots. As a result, task offloading and result delivery can be formulated as Mixed Integer Programming (MIP). By replacing each edge server with cardinality of larger than one with multiple one-cardinality virtual servers, the MIP problem can be further reformulated to a graph matching problem.
- A distributed linear-time algorithm is proposed to solve the graph matching with $\frac{1}{2}$-approximation, which is achieved by having each node spontaneously maximize the total weights of itself and its immediate neighbors.
- We prove that the optimality loss of fog computing can asymptotically diminish by reducing the stepsize of stochastic gradient descent. In other words, the increase of the time-average cost of fog computing, resulting from the proposed distributed $\frac{1}{2}$-approximation algorithm, can be suppressed by extending the learning time.

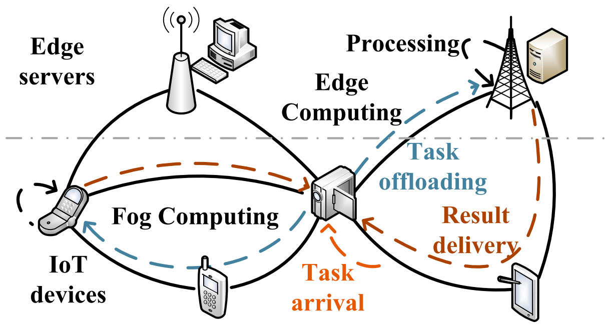


Figure 4.1 : An illustrative example of fog computing in IoT networks.

Corroborated by simulations, the proposed distributed online learning approach is able to increase the throughput by 59% and the energy efficiency by 43%, as compared to the state of the art. With the increase of network scale, the proposed distributed online learning can save 96% the running time of its centralized counterpart which obtains the optimal solution of the graph matching by using the Edmonds' blossom algorithm.

4.1 System Model

Fig. 4.1 illustrates the application of fog computing in IoT networks, where tasks generated at the IoT devices can be offloaded multiple hops away to nearby network nodes (i.e., other devices and edge servers) for processing. As shown in Fig. 4.1, we consider a fog computing network consisting of N IoT devices and M edge servers, such as access points (APs), cloudlets, and base stations (BSs). Running a network timing protocol (NTP), the system operates on a slotted basis with slot length T. The tasks generated by the IoT devices can be offloaded multiple hops away and processed anywhere in the network, i.e., locally executed, offloaded to other IoT devices, or remotely processed at the edge servers. Once processed, the result of a task is returned to the device generating the task. The communication links among the IoT devices (such as WiFi Direct, Bluetooth, and Zigbee), and between devices and edge servers (e.g., WiFi and cellular), can support the task offloading and result delivery within the network. Let $\mathbf{N} = \{1, \cdots, N\}$ and $\mathbf{M} = \{1, \cdots, M\}$ collect the indexes for the devices and edge servers, respectively. The notations used in this

Table 4.1 : Definitions of Notations

Notation	Definition
$\mathbf{N}$	The set of mobile devices
T	Slot length
$A_i(t)$	Tasks generated by device i at slot t
$A_i^{\max}$	Maximum size of tasks generated by device i at a slot
$e_{ij}(t)$	Connectivity between nodes i and j during slot t
ρ_i	CPU cycles for processing a task bit of device i
ξ_i	Ratio of results to unprocessed tasks of device i in size
$l_{ij}(t)$	Link activation decisions for link (i, j) during slot t
K_i	Maximum number of simultaneous transmissions of server i
$C_{ij}(t)$	Capacity of link (i, j) during slot t
$b_{ij}^{(s)}(t)$	Tasks of device s offloaded from devices i to j at slot t
$d_{ij}^{(s)}(t)$	Results for device s returned via devices i to j at slot t
$F_i(t)$	Available computing resources of device i at slot t
$f_i^{(s)}(t)$	Resource allocation of device i for device s's tasks at slot t
$Q_i^{(s)}(t)$	Task queue backlog at device i for device s at slot t
$D_i^{(s)}(t)$	Result queue backlog at device i for device s at slot t
$\varepsilon_{ij}(t)$	Cost for per-bit transmission over link (i, j) during slot t
$\varepsilon_{ij}^{\max}$	Maximum cost for per-bit transmission over link (i, j)
$\varepsilon_i(t)$	Cost to run a CPU cycle of device i during slot t
$\phi(t)$	Total operational cost of the network at slot t
ϵ	Stepsize of the stochastic gradient descent

chapter are summarized in Table 4.1.

4.1.1 Network Model

The topology of the network can be modeled as a stochastic graph $G(t) = \{\mathbf{V}, \mathbf{E}(t)\}$, where $\mathbf{V} = \mathbf{M} \cup \mathbf{N}$ is the set of devices and edge servers of the network and $\mathbf{E}(t) = \{(i, j)|e_{ij}(t) = 1, \forall i, j \in \mathbf{V}\}$ collects the edges (connectivity) among the nodes. $e_{ij}(t) = 1$, if there is a link (i, j) between nodes i and j at time slot t. Let $C_{ij}(t) \in (0, C_{ij}^{\max}]$ denote the capacity of the bi-directional link (i, j) during slot t, and $\varepsilon_{ij}(t) \le \varepsilon_{ij}^{\max}$ denote the cost (e.g., in terms of energy consumption) for per-bit transmission over the link at the slot. The connectivity and channel capacity are assumed to remain unchanged within a slot, and can change between slots.

Each IoT device can only activate a bi-directional link at a time. Edge server i can establish up to K_i active links at time slot t. Let $l_{ij}(t)$ denote the decision of activating the links; where $l_{ij}(t) = 1$ if link (i, j) is activated at slot t, and $l_{ij}(t) = 0$, otherwise. The cardinality constraints of activating communication links can be given by

$$l_{ij}(t) \in \{0, 1\}, \forall i, j \in \mathbf{V}; \tag{4.1a}$$

$$\sum\nolimits_{j \in \mathbf{N}} l_{ij}(t) \leq K_i, \forall i \in \mathbf{M}; \tag{4.1b}$$

$$\sum\nolimits_{j \in \mathbf{V}} l_{ij}(t) \leq 1, \forall i \in \mathbf{N}; \tag{4.1c}$$

where (4.1a) is self-explanatory, and (4.1b) and (4.1c) specify the maximum active links for the edge servers and devices at a slot, respectively.

The tasks arriving at device i during time slot t, can be parameterized as a triplet $\big(A_i(t), \rho_i A_i(t), \xi_i A_i(t)\big)$. $A_i(t)$ is the size of the task generated (in bits) at the slot. It requires $\rho_i A_i(t)$ CPU cycles to be processed, and the size of the results is $\xi_i A_i(t)$ bits. $\rho_i \geq \rho_{\min}$ is the number of CPU cycles for processing a bit of the task, and ξ_i is the ratio of the results to the corresponding unprocessed tasks in bits. Due to the background tasks at the edge servers and IoT devices, $F_i(t) \leq F^{\max}$ denotes the available computing resources of node $i \in \mathbf{V}$ (in CPU cycles per second) during slot t. $\varepsilon_i(t)$ denotes the cost of node i to run a CPU cycle at slot t.

The task arrival $A_i(t)$ is a stochastic process with the maximum $A_i^{\max}$. We assume that tasks can be divided based on the link capacity $C_{ij}(t)$ and the computing resources $F_i(t)$ per slot. The proposed approach can be readily applied for atomic, indivisible tasks by rounding the allocated computing resources and link schedules to the largest integer numbers of atomic tasks which can be supported. This would not compromise the (asymptotic) optimality of the proposed approach, as will be discussed in Section 4.3.

4.1.2 Causality Constraints and Queue Dynamics

We design that each node maintains N queues for the unprocessed tasks generated by all N IoT devices (including the device itself) and another N queues for the corresponding processed results. This is because the result of a task is to be returned to the IoT device generating the task. $\mathcal{Q}(t) = \{Q_i^s(t), D_i^s(t) | \forall i \in \mathbf{V}, s \in \mathbf{N}\}$ collects the total $\big(2N(N+M)\big)$ queues of the $(N+M)$ nodes in the network at time slot t. In specific, as per slot t, $Q_i^s(t)$ is the backlog of the queue, in which node i buffers unprocessed tasks generated by device s; and $D_i^s(t)$ is the backlog of the queue, in which node i buffers processed results destined for device s. Each of the queues is scheduled on a First-In-First-Out (FIFO) basis.

Let $b_{ij}^s(t)$ be the size of unprocessed tasks generated by device s and forwarded to node j through node i at slot t, and $d_{ij}^s(t)$ be the size of processed results returned from node i through node j destined for device s at slot t. We have

$$b_{ij}^s(t) \geq 0,\ d_{ij}^s(t) \geq 0,\ \forall (i,j), s \in \mathbf{N}; \tag{4.2a}$$

$$\sum_{s\in\mathbf{N}} b_{ij}^s(t) + d_{ij}^s(t) + b_{ji}^s(t) + d_{ji}^s(t) \leq C_{ij}(t) l_{ij}(t),\ \forall (i,j); \tag{4.2b}$$

where (4.2a) is self-explanatory, and (4.2b) specifies that the total size of tasks and results transmitted over link (i,j) cannot exceed the link capacity.

Let $f_i^s(t)$ denote the CPU cycles of node i allocated at time slot t to the tasks generated by device s, satisfying the following constraint:

$$f_i^s(t) \geq 0,\ \forall i \in \mathbf{V}, s \in \mathbf{N}; \tag{4.3a}$$

$$\textstyle\sum_{s\in\mathbf{N}} f_i^s(t) \leq F_i(t),\ \forall i \in \mathbf{V}. \tag{4.3b}$$

The backlogs of unprocessed tasks $Q_i^s(t)$ can be updated by

$$Q_i^s(t+1) = \max\big\{Q_i^s(t) - f_i^s(t)/\rho_s - \sum_{j\in\mathbf{V}} b_{ij}^s(t), 0\big\} + \textstyle\sum_{j\in\mathbf{V}} b_{ji}^s(t) + A_i^s(t), \tag{4.4}$$

where $f_i^s(t)/\rho_s$ is the size of tasks that can be processed from the queue at time slot t, and $A_i^s(t)$ is the size of tasks arriving at device i into the queue at the time slot. $A_i^i(t) = A_i(t)$, and $A_i^s(t) = 0$, $\forall s \neq i$. The first term on the right-hand side (RHS) of (4.4) gives the remaining unprocessed tasks in the queue at the end of time slot t,

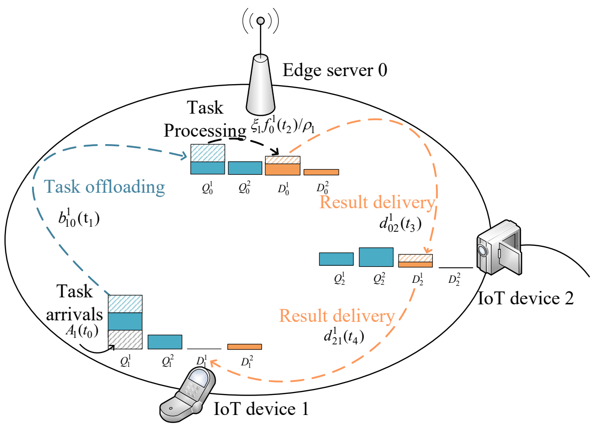


Figure 4.2 : An illustrative example of task offloading and result delivery in the proposed multi-hop fog computing.

after part of the tasks have been processed at node i and offloaded to other nodes. The other two terms give the new task arrivals offloaded from other nodes to node i or generated locally at the node.

The backlogs of processed results $D_i^s(t)$ can be updated by

$$D_i^s(t+1) = \max\left\{D_i^s(t) - \sum\nolimits_{j\in\mathbf{V}} d_{ij}^s(t), 0\right\} + \sum\nolimits_{j\in\mathbf{V}} d_{ji}^s(t) + \xi_s f_i^s(t)/\rho_s,\ \forall s \neq i, \tag{4.5}$$

where $D_i^i(t) = 0$ since node i is the sink for the results.

Fig. 4.2 shows an illustrative example of the procedure of the proposed multi-hop fog computing to process an incoming task in a network of two IoT devices and an edge server. In particular, the incoming task arrives at device 1 and enters the corresponding task queue at time slot t_0, where the size of the task (in bits) is denoted by $A_1(t_0)$; and at time slot t_1, it is offloaded to the task queues of edge server 0 for processing, where the size of offloaded data is $b_{10}^1(t_1)$. At slot t_2, the task is processed at edge server 0 and the computation results enter the corresponding data queue for result delivery to device 1 generating the task. The results are delivered to device 1 hop by hop with the help of device 2, and the sizes of routed data from edge server 0 to device 2 at slot t_3 and from device 2 to device 1 at slot t_4 are denoted by $d_{02}^1(t_3)$ and $d_{21}^1(t_4)$, respectively. Finally, device 1 retrieves the computation results of the task $A_1(t_0)$.

The multi-hop communications can incur additional costs resulting from the queue store and forward when using multi-hop communications. However, in the case of IoT applications, large volumes of data need to be processed, exceeding the computing capability of network with one-hop transmission. Multi-hop transmission can be used to enable the computing capability and throughput of network at the cost of increasing delay of tasks. From Fig. 4.4(a) we can see that compared to the one-hop transmission, the multi-hop transmission can achieve more than 200% gain of throughput with the growth of task arrivals.

4.1.3 Problem Statement

We measure the performance of the fog computing in IoT networks by operational cost. The total operational cost of the network at time slot t can be written as

$$\Phi\big(\mathbf{x}(t)\big) = \sum\nolimits_{i,j\in\mathbf{V}} \varphi_{ij}(t) + \sum\nolimits_{i\in\mathbf{V}} \varphi_i(t), \tag{4.6}$$

where $\mathbf{x}(t) = \{f_i^s(t), b_{ij}^s(t), d_{ij}^s(t), l_{ij}(t), \forall i,j,s\}$ collects all the variables in regards of task processing, task offloading, result delivery and link activation, at time slot t. $\varphi_{ij}(t) = l_{ij}(t)\sum_{s\in\mathbf{N}} \varepsilon_{ij}(t)(b_{ij}^s(t) + d_{ij}^s(t))$ and $\varphi_i(t) = \sum_{s\in\mathbf{N}} \varepsilon_i(t) f_i^s(t)$ are the costs of task offloading and result delivery over link (i,j), and task processing at node i at time slot t, respectively.

The network is stable if and only if all the queues of the network are stable, i.e., the following is met [31]

$$\overline{Q_i^s(t)} < \infty,\ \overline{D_i^s(t)} < \infty,\ \forall i,s \tag{4.7}$$

where $\overline{X(t)} = \lim_{T\to\infty}\frac{1}{T}\sum_{\tau=0}^{T-1}\mathbb{E}[X(\tau)]$ denotes the long-term time-average of any process $X(t)$.

Considering the prevalent randomness of channel conditions, task arrivals and computing resources, we propose to stochastically minimize the overall system cost for delay-tolerant tasks in infinite time horizon, while preserving the stability of all the queues of the network. Quality-of-service (QoS) is not explicitly considered, since as suggested in [16], delay-sensitive applications would be typically processed locally, or offloaded and processed with priority, as will be described in Section 4.5. The problem of interest can be formulated as

$$\begin{aligned} \Phi^* = & \min_{\mathbb{X}} \overline{\Phi\big(\mathbf{x}(t)\big)} \\ & \quad \text{s.t. } (4.1), (4.2), (4.3), (4.4), (4.5), (4.7), \forall t; \end{aligned} \tag{4.8}$$

where $\mathbb{X} = \{\mathbf{x}(t), \forall t\}$ collects the variables $\mathbf{x}(t)$ across all time slots.

Note that problem (4.8) is coupled across infinite time horizon due to the queue dynamics (4.4), (4.5) and (4.7), and also between the large number of IoT devices and edge servers under the non-uniform cardinality constraints (4.1). The optimal solution to (4.8) would require the a-priori knowledge on task arrivals, channel conditions and computing resources across the entire network over infinite time horizon. This would violate causality. Moreover, the instantaneous global view of the entire network may not be available due to the sheer scale and the distributed property of IoT [18]. Non-negligible propagation delays over multiple hops prevent the acquisition of instantaneous global view of the network.

4.2 Fully Distributed Online Learning of Fog Computing

In this section, we propose to fully decentralize the solution for fog computing and asymptotically minimize the time-average operational cost of fog computing by exploiting stochastic online learning. The stochastic gradient descent is exploited to decouple the optimal decisions on task processing, offloading, and result delivery between time slots in the absence of the a-priori knowledge on network randomness. The optimal decisions on task offloading and result delivery can be further decoupled among devices and edge servers at a bounded cost of $\frac{1}{2}$-approximation to the optimum. The cost can asymptotically diminish as the decrease of the stepsize of stochastic online learning, as will be dictated in Section 4.3.

4.2.1 Stochastic Online Learning for Temporal Decoupling

According to queuing theory [92], a queue is stable, if and only if the time-average input rate of the queue is no more than the time-average output rate of the queue. As a result, we reformulate problem (4.8) to suppress the time couplings by transforming (4.4), (4.5) and (4.7) to (4.9), as given by

$$\begin{aligned}&\overline{A_i^s(t) - f_i^s(t)/\rho_s + \sum\nolimits_{j\in\mathbf{N}}(b_{ji}^s(t) - b_{ij}^s(t))} \leq 0,\ \forall i, s;\\ &\overline{\xi_s f_i^s(t)/\rho_s + \sum\nolimits_{j\in\mathbf{N}}(d_{ji}^s(t) - d_{ij}^s(t))} \leq 0,\ \forall i, s.\end{aligned} \tag{4.9}$$

We define the network randomness at slot t by $\boldsymbol{\omega}^t = \{A_i(t), C_{ij}(t), F_i(t), \forall i, j\}$. It is reasonable to assume that $\boldsymbol{\omega}^t$ is independent and identically distributed (i.i.d.) across all time slots, since tasks arrive independently at a large number of devices and the available computing resources are affected by independent background tasks. As a result, the time-average constraint in (4.9) can be further replaced by the expectations per slot over $\boldsymbol{\omega}^t$, as given by

$$\mathbb{E}_{\boldsymbol{\omega}^t}\Big[A_i^s(t) - f_i^s(t)/\rho_s + \sum_{j\in\mathbf{V}}(b_{ji}^s(t) - b_{ij}^s(t))\Big] \leq 0,\ \forall i, s; \tag{4.10a}$$

$$\mathbb{E}_{\boldsymbol{\omega}^t}\Big[\xi_s f_i^s(t)/\rho_s + \sum_{j\in\mathbf{V}}(d_{ji}^s(t) - d_{ij}^s(t))\Big] \leq 0,\ \forall i, s. \tag{4.10b}$$

By replacing (4.4), (4.5) and (4.7) with (4.10), (4.8) can be rewritten as a per-slot expectation minimization problem, as given by

$$\begin{aligned}\widetilde{\Phi}^* = \min_{\mathbb{X}} &\ \mathbb{E}\Big[\Phi\big(\mathbf{x}(t); \boldsymbol{\omega}^t\big)\Big]\\ \text{s.t.} &\ (4.1), (4.2), (4.3), (4.10a), (4.10b), \forall t.\end{aligned} \tag{4.11}$$

Stochastic gradient descent is an effective learning method with extensive applications to support vector machines, logistic regression and artificial neural network-s [94]. It is particularly useful in the cases where an exact gradient is intractable in the absence of the a-priori knowledge on the randomness over infinite time. By taking stochastic gradient descent, (4.11) can be reformulated by interpreting (4.10) as the Lagrange multipliers and iteratively updating the Lagrange multipliers with the stochastic gradient per slot t. As a result, (4.11) can be transformed to (4.12) which can be carried out at each time slot t and is given by

$$\begin{aligned} &\max_{\mathbf{x}(t)} \mu\big(\mathbf{f}(t)\big) + \eta\big(\mathbf{b}(t), \mathbf{d}(t), \mathbf{l}(t)\big) \\ &\text{s.t. } (4.1), (4.2), (4.3). \end{aligned} \tag{4.12}$$

where $\mathbf{f}(t) = \{f_i^s(t), \forall i, s\}$, $\mathbf{b}(t) = \{b_{ij}^s(t), \forall i, j, s\}$, $\mathbf{d}(t) = \{d_{ij}^s(t), \forall i, j, s\}$, and $\mathbf{l}(t) = \{l_{ij}(t), \forall i, j\}$ are the decisions on task processing, task offloading, result delivery, and link activation at time slot t, respectively; and

$$\mu\big(\mathbf{f}(t)\big) = \sum_{i,s\in\mathbf{N}} \left[\frac{\epsilon[Q_i^s(t) - \xi_s D_i^s(t)]}{\rho_s} - \zeta_i(t)\right] f_i^s(t); \tag{4.13a}$$

$$\begin{aligned} \eta\big(\mathbf{b}(t), \mathbf{d}(t), \mathbf{l}(t)\big) = \sum_{i,j,s\in\mathbf{N}} \epsilon\big[Q_i^s(t)(b_{ij}^s(t) - b_{ji}^s(t)) \\ + D_i^s(t)(d_{ij}^s(t) - d_{ji}^s(t))\big] - \zeta_{ij}(t)(b_{ij}^s(t) + d_{ij}^s(t)); \end{aligned} \tag{4.13b}$$

where ϵ is the stepsize of the stochastic gradient descent. The Lagrange multipliers $\lambda_{i,1}^s(t)$ and $\lambda_{i,2}^s(t)$ associated respectively with (4.10a) and (4.10b), can be suppressed in (4.13). This is because, with the consistent stepsize ϵ, the multipliers can be proved to be equal to the product of the stepsize and the queue backlogs, i.e., $\lambda_{i,1}^s(t) = \epsilon Q_i^s(t)$ and $\lambda_{i,2}^s(t) = \epsilon D_i^s(t)$.

Proof. Let $\boldsymbol{\lambda} = \{\lambda_{i,1}^s, \lambda_{i,2}^s, \forall i, s\}$, where $\lambda_{i,1}^s$ and $\lambda_{i,2}^s$ denote the Lagrange multipliers of (4.11) associated with (4.10a) and (4.10b), respectively. The Lagrangian of (4.11) can be given by [93]

$$\mathcal{L}(\mathbb{X}, \boldsymbol{\lambda}) = \mathbb{E}\big[\mathcal{L}(\mathbf{x}(t), \boldsymbol{\lambda})\big],$$

where the instantaneous Lagrangian $\mathcal{L}(\mathbf{x}(t), \boldsymbol{\lambda})$ is given by

$$\begin{aligned} \mathcal{L}(\mathbf{x}(t), \boldsymbol{\lambda}) = \Phi(\mathbf{x}(t)) + \sum_{i,s\in\mathbf{N}} \lambda_{i,1}^s \mathbb{E}\big[A_i^s(t) - f_i^s(t)/\rho_s \sum_{j\in\mathbf{N}}(b_{ji}^s(t) - b_{ij}^s(t))\big] \\ + \sum_{i,s\in\mathbf{N}} \lambda_{i,2}^s \mathbb{E}\big[\xi_s f_i^s(t)/\rho_s + \sum_{j\in\mathbf{N}}(d_{ji}^s(t) - d_{ij}^s(t))\big]. \end{aligned} \tag{4.14}$$

The dual problem is $\max_{\boldsymbol{\lambda}\succeq 0} \mathcal{D}(\boldsymbol{\lambda})$, where $\succeq$ is taken entry-wise and $\mathcal{D}(\boldsymbol{\lambda})$ is given by

$$\begin{aligned} \mathcal{D}(\boldsymbol{\lambda}) = \min_{\mathbb{X}} & \, \mathcal{L}(\mathbb{X}, \boldsymbol{\lambda}) \\ \text{s.t. } & (4.1), (4.2), (4.3), \forall t. \end{aligned} \tag{4.15}$$

At each slot t, $\mathbf{x}(t)$ can be updated by

$$\begin{aligned} \mathbf{x}(t) = \arg\min_{\mathbf{x}(t)} & \, \mathcal{L}(\mathbf{x}(t), \boldsymbol{\lambda}(t)) \\ \text{s.t. } & (4.1), (4.2), (4.3). \end{aligned} \tag{4.16}$$

Without the a-priori knowledge on the statistics of the randomness $\omega(t)$, $\widetilde{\boldsymbol{\lambda}}(t) = \{\widetilde{\lambda}^s_{i,1}(t), \widetilde{\lambda}^s_{i,2}(t), \forall i, s\}$ is an online approximation of the dual multipliers $\boldsymbol{\lambda}$ based on the instantaneous decisions $\mathbf{x}(t)$ per slot t, as given by

$$\widetilde{\lambda}^s_{i,1}(t+1) = \max\left\{\widetilde{\lambda}^s_{i,1}(t) + \epsilon\big[A^s_i(t) - f^s_i(t)/\rho_s + \textstyle\sum_{j\in\mathbf{N}}(b^s_{ji}(t) - b^s_{ij}(t))\big], 0\right\}, \tag{4.17a}$$

$$\widetilde{\lambda}^s_{i,2}(t+1) = \max\left\{\widetilde{\lambda}^s_{i,2}(t) + \epsilon\big[\xi_s f^s_i(t)/\rho_s + \textstyle\sum_{j\in\mathbf{N}}(d^s_{ji}(t) - d^s_{ij}(t))\big], 0\right\}; \tag{4.17b}$$

where ϵ is an appropriate stepsize of the stochastic gradient descent. With $Q^s_i(0) = 0$ and $D^s_i(0) = 0$, by comparing (4.4) and (4.5) with (4.17a) and (4.17b), respectively, the update of dual multipliers per slot can be enclosed in the natural update of queue backlogs, i.e., $\widetilde{\lambda}^s_{i,1}(t) = \epsilon Q^s_i(t)$ and $\widetilde{\lambda}^s_{i,2}(t) = \epsilon D^s_i(t)$.

$\square$

It is worth mentioning that in the absence of the a-priori knowledge on the network dynamics, there is no training dataset available for our proposed approach. The stochastic gradient descent is typically referred to as the online learning of the network dynamics (including network topology, channel capacity, task arrivals and computing resources) from the streaming training data (i.e., the network dynamics $\boldsymbol{\omega}^t$) observed at each time slot t. At every time slot t, stochastic gradient descent is applied to update the Lagrange multipliers $\widetilde{\boldsymbol{\lambda}}(t)$ with the observed network dynamics

$\boldsymbol{\omega}^t$ based on (4.17). The decisions on the task offloading, processing and result delivery of every device are obtained by solving problem (4.12) based on $\widetilde{\boldsymbol{\lambda}}(t)$. This is because the exact gradient is intractable in the absence of the a-priori knowledge on the randomness over the infinite time. The stochastic gradient is a sample (or an approximation) of the exact gradient based on the observed network dynamics $\boldsymbol{\omega}^t$, and would asymptotically converge to within $\mathcal{O}(\epsilon)$ of the optimal multipliers (i.e., the offline optimum which would require the a-priori knowledge on the network dynamics), as stated in Theorem 3.

Note in (4.12) that $\mu\big(\mathbf{f}(t)\big)$ and $\eta\big(\mathbf{b}(t), \mathbf{d}(t), \mathbf{l}(t)\big)$ are decoupled from each other in both the objective and constraints. The optimal solution on task processing for (4.13a), and the optimal solutions for task offloading, result delivery and link activation for (4.13b), can be fully decentralized and implemented in real-time, as described in the next subsection.

4.2.2 Per-slot Optimal Solutions of Online Learning

A. Optimal Decisions on Task Processing

Since $f_i^s(t)$ can be decoupled from $f_j^s(t)$, $\forall i \neq j$, (4.13a) can be further decoupled between the devices and edge servers to independently optimize the resource allocation $\mathbf{f}_i(t) = \{f_i^s(t), \forall s\}$ at each node i, as given by

$$\max_{\mathbf{f}_i(t)} \sum\nolimits_{s\in\mathbf{N}} \alpha_i^s(t) f_i^s(t), \text{ s.t. (4.3);} \tag{4.18}$$

where $\alpha_i^s(t) = \epsilon\big[Q_i^s(t) - \xi_s D_i^s(t)\big]/\rho_s - \zeta_i(t)$. Here, (4.18) is linear programming of weighted-sum maximization [99], and its optimal solution can be given by

$$f_i^s(t) = \begin{cases} F_i(t), \text{ if } s = \arg\max_j \alpha_i^{(j)}(t) \text{ and } \alpha_i^s(t) > 0; \\ 0, \text{ otherwise.} \end{cases} \tag{4.19}$$

B. Optimal Decisions on Task Offloading and Result Delivery

The decisions on task offloading and result delivery cannot be decoupled between links due to the coupling of link activations across the network in constraint (4.1).

Nevertheless, the problem of maximizing (4.13b) subject to (4.1) and (4.2) can be rewritten as

$$\begin{aligned}&\max_{\mathbf{b}(t),\mathbf{d}(t),\mathbf{l}(t)} \sum_{i,j\in\mathbf{V},s\in\mathbf{N}} \{\beta_{ij}^s(t)b_{ij}^s(t)+\gamma_{ij}^s(t)d_{ij}^s(t)\}l_{ij}(t)\\ &\text{s.t. } (4.1), (4.2);\end{aligned} \tag{4.20}$$

where $\beta_{ij}^s(t)=\epsilon\big[Q_i^s(t)-Q_j^s(t)\big]-\zeta_{ij}(t)$ and $\gamma_{ij}^s(t)=\epsilon\big[D_i^s(t)-D_j^s(t)\big]-\zeta_{ij}(t)$. (4.20) is MIP, since the variables $\mathbf{b}(t)$ and $\mathbf{d}(t)$ are continuous and depend on the binary decisions on link activation $\mathbf{l}(t)$ in (4.2).

We propose to decompose (4.20) into two subproblems. By carrying out alternating optimization of the two subproblems, we show that (4.20) can be reformulated to an integer programming problem which can be further transformed to a graph matching problem:

(1) *Given a solution for link activation* $\mathbf{l}(t)$*,* the task offloading and result delivery can be optimized subject to (4.2). The optimization can be decoupled between links. Let $\mathbf{b}_{ij}(t)=\{b_{ij}^s(t),b_{ji}^s(t),\forall s\}$ and $\mathbf{d}_{ij}(t)=\{d_{ij}^s(t),d_{ji}^s(t),\forall s\}$ collect the decisions on task offloading and result delivery over link (i,j) at slot t, respectively. The problem can be given by

$$\max_{\mathbf{b}_{ij}(t),\mathbf{d}_{ij}(t)} \sum_{s\in\mathbf{N}} \eta_{ij}(\mathbf{b}_{ij}(t),\mathbf{d}_{ij}(t)), \text{ s.t. } (4.2); \tag{4.21}$$

where $\eta_{ij}(\mathbf{b}_{ij}(t),\mathbf{d}_{ij}(t))=\beta_{ij}^s(t)b_{ij}^s(t)+\gamma_{ij}^s(t)d_{ij}^s(t)+\beta_{ji}^s(t)b_{ji}^s(t)+\gamma_{ji}^s(t)d_{ji}^s(t)$.

Problem (4.21) is linear programming of weighted-sum maximization [99]. The optimal solution can be obtained by evaluating the weights $\beta_{ij}^s(t)$, $\gamma_{ij}^s(t)$, $\beta_{ji}^s(t)$ and $\gamma_{ji}^s(t)$ at nodes i and j independently. If $\max_s\{\beta_{ij}^s(t),\gamma_{ij}^s(t)\}<0$ or $\max_s\{\beta_{ij}^s(t),\gamma_{ij}^s(t)\}$ $<\max_s\{\beta_{ji}^s(t),\gamma_{ji}^s(t)\}$, node i remains idle at slot t, i.e., neither offloading tasks nor transmitting results to node j. If $\max_s\{\beta_{ij}^s(t)\}>\max_s\{\gamma_{ij}^s(t)\}$, node i does not return results. Instead, it offloads tasks to node j with the task size specified by

$$b_{ij}^s(t)=\begin{cases}C_{ij}(t), \text{ if } s=\arg\max\limits_r \beta_{ij}^{(r)}(t);\\ \quad 0, \text{ otherwise.}\end{cases} \tag{4.22a}$$

If $\max_s\{\beta_{ij}^s(t)\}\le\max_s\{\gamma_{ij}^s(t)\}$, node i does not offload tasks to node j. Instead, it returns results through node j with the result size specified by

$$d_{ij}^s(t) = \begin{cases} C_{ij}(t), \text{ if } s = \arg\max_r \gamma_{ij}^{(r)}(t); \\ \quad 0, \text{ otherwise.} \end{cases} \tag{4.22b}$$

From (4.22), we can see that, for an activated link (i, j), the queue that can make the most use of the link at slot t, i.e., the queue of the maximum weights $\beta_{ij}^s(t)$, $\beta_{ji}^s(t)$, $\gamma_{ij}^s(t)$ and $\gamma_{ji}^s(t)$, is selected to occupy the link capacity C_{ij}(t) in whole.

(2) *Given the solution for (4.21), i.e., (4.22),* problem (4.20) can be reformulated to an integer programming problem of $\mathbf{l}(t)$, as given by

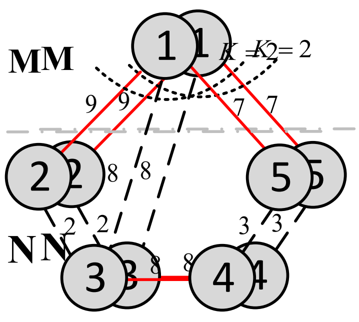
$$\max_{\mathbf{l}(t)} \sum\nolimits_{i,j\in\mathbf{V}} w_{ij}(t) l_{ij}(t), \text{ s.t. } (4.1); \tag{4.23}$$

where $w_{ij}(t) = \max_s\{\beta_{ij}^s(t), \gamma_{ij}^s(t), \beta_{ji}^s(t), \gamma_{ji}^s(t)\}C_{ij}(t)$ can be obtained by substituting (4.22) into (4.20).

Recall the network topology graph $\mathbf{G}(t) = \{\mathbf{V}, \mathbf{E}(t)\}$. $w_{ij}(t)$ is the weight of link (i, j) in graph $\mathbf{G}(t)$. We can translate (4.23) to a standard matching problem in an undirected weighted graph. It is to find the set of disjoint edges which has the maximum total weights [100].

We note that the standard matching problem in an undirected weighted graph requires the cardinality of edge servers to be constrained to no greater than 1 [100]. In contrast, the edge servers can have multiple matched edges in the problem of interest. Therefore, we reconstruct graph $\mathbf{G}(t)$ to $\mathbf{G}'(t) = \{\mathbf{V}', \mathbf{E}'(t)\}$, where edge server $i \in \mathbf{M}$ is replaced with K_i unconnected virtual nodes, denoted by $\{i_1, \cdots, i_K\}$, as illustrated in Fig. 4.3. The K_i virtual nodes have identical weighted edges, i.e., $w_{i_kj}(t) = w_{ij}(t)$. Let $\mathbf{M}'$ be the set of virtual nodes for the edge servers $\mathbf{M}$. In graph $\mathbf{G}'(t)$, $\mathbf{V}' = \mathbf{N} \cup \mathbf{M}'$ collects all the $(N + MK)$ nodes and $\mathbf{E}'(t)$ collects the edges between the (virtual) nodes. The cardinality constraint of the matching problem can be satisfied in $\mathbf{G}'(t)$.

After the construction, the cardinality constraints (4.1b) and (4.1c) can be rewrit-


(a) Graph $\mathbf{G}(t)$

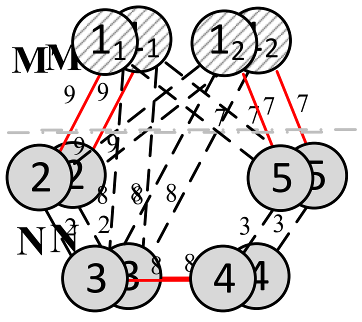

(b) Graph $\mathbf{G}'(t)$

Figure 4.3 : An illustrative example of the graph reconstruction, where there are $N = 4$ devices and $M = 1$ edge server with cardinality $K = 2$.

ten as

$$\sum\nolimits_{j\in\mathbf{V}'} l_{ij}(t) \le 1, \forall i \in \mathbf{V}'. \tag{4.24}$$

By replacing (4.1b) and (4.1c) with (4.24), problem (4.23) becomes

$$\max_{\mathbf{l}(t)} \sum\nolimits_{i,j\in\mathbf{V}'} w_{ij}(t) l_{ij}(t), \text{ s.t. } (4.1a), (4.24), \tag{4.25}$$

which is the standard matching problem in the undirected weighted graph $\mathbf{G}'(t)$, and can be optimally solved by Edmonds' blossom algorithm in a centralized manner with the complexity $\mathcal{O}\big(|\mathbf{E}'||\mathbf{V}'|^2\big)$ [101]. Here, $|\mathbf{E}'(t)|$ and $|\mathbf{V}'|$ are the numbers of edges and vertexes of $\mathbf{G}'(t)$, respectively.

4.2.3 Distributed Online Learning of Fog Computing

As a standard matching problem in an undirected weighted graph, (4.25) can also be sub-optimally solved by sequentially selecting the edges with the maximum weights from the unselected edges, provided the explicit knowledge of the weights of all edges across the network is available to every node [102]. It was proved in [102] that such selections yield $\frac{1}{2}$-approximation to the optimal solution of Edmond's blossom algorithm. This allows us to fully decentralize the solution for (4.25) and, in turn, the solutions for (4.20) and (4.12).

A. Problem Decomposition

We propose to decentralize the solution for (4.20) by decomposing (4.25) into the subproblems of each node $i \in \mathbf{V}'$ maximizing the total weight of itself and its immediate neighbors, as given by

$$\max \sum\nolimits_{i' \in \widetilde{\mathbf{N}}_i, j \in \mathbf{V}'} w_{i'j}(t) l_{i'j}(t), \text{ s.t. (4.1a), (4.24),} \tag{4.26}$$

where $\widetilde{\mathbf{N}}_i$ collects node i itself and its immediate neighbors in graph $\mathbf{G}'(t)$.

Let $\widehat{\mathbf{l}}(t)$ denote the approximation solution of link activation given by (4.26), and $\mathbf{l}^*(t)$ be the optimal solution for (4.25) achieved by Edmonds' blossom algorithm. $W(\mathbf{l}(t)) = \sum_{i,j \in \mathbf{V}'} w_{ij}(t) l_{ij}(t)$ is the sum of the edge weights, given a matching $\mathbf{l}(t)$. The approximation ratio of the decomposed problem (4.26) to the optimum in (4.25) can be established in the following Corollary.

Corollary 1. *The solution to (4.26) achieves $\frac{1}{2}$-approximation to the optimum of (4.25), i.e., $W(\widehat{\mathbf{l}}(t)) \geq \frac{1}{2} W(\mathbf{l}^*(t))$.*

Proof. In (4.26), link (i, j) is selected, i.e., $l_{ij}(t) = 1$, if and only if the link is the most heavily weighted for each of nodes i and j. As a result, (4.26) achieves the same results as the sequential selection heuristic developed in [102]. The $\frac{1}{2}$-approximation ratio of (4.26) to the optimum readily follows the proof in [102]. $\square$

B. Distributed Task Offloading and Result Delivery

Problem (4.26) can be solved in a distributed manner by allowing the nodes to send matching requests and accepting others' requests, as summarized in Algorithm 2. Specifically, each node i defines and updates two sets, namely $\mathbf{N}_i$ and $\mathbf{R}_i$. $\mathbf{N}_i$ is the set of the nodes which can be connected to node i, and can be initialized as its immediate neighbors, i.e., $\mathbf{N}_i = \{j | w_{ij}(t) > 0\}$. $\mathbf{R}_i$ is the set of nodes, from which node i has received matching requests. $\mathbf{R}_i$ can be initialized by $\mathbf{R}_i = \emptyset$.

Node i can send a matching request to the one of its immediate neighbor j which has the largest edge weight in $\mathbf{N}_i$, i.e., $j = \arg\max_{j \in \mathbf{N}_i} w_{ij}(t)$. When receiving the

Algorithm 2 Distributed Task Offloading and Result Delivery

```
Input: Graph $\mathbf{G}'(t)$ with the weights $w_{ij}(t)$
Output: The $\frac{1}{2}$-approximation solution of link activation, task offloading and result delivery
1: Initialize $\mathbf{N}_i = \{j | w_{ij}(t) > 0\}$ and $\mathbf{R}_i = \emptyset$
2: Send a matching request to candidate node $c$ with the largest edge weight, i.e., $c = \arg\max_{j \in \mathbf{N}_i} w_{ij}(t)$
3: while $\mathbf{N}_i \neq \emptyset$ do
4:   if Receiving a matching request from node $j$ then
5:     $\mathbf{R}_i = \mathbf{R}_i \cup \{j\}$
6:     if Node $i$ has sent a matching request to node $j$ then
7:       Establish a matching with node $j$, i.e., $\widehat{l_{ij}}(t) = 1$
8:       Send dropping messages to all the nodes in $\mathbf{N}_i$
9:       Set $\mathbf{N}_i = \emptyset$
10:    end if
11:  end if
12:  if Receiving a dropping message from node $j$ then
13:    $\mathbf{N}_i = \mathbf{N}_i / \{j\}$
14:    if Node $i$ has sent a matching request to node $j$ then
15:      Send a matching request to the next candidate node with the largest edge weight
16:    end if
17:  end if
18: end while
```

matching request, node j adds node i into $\mathbf{R}_j$, i.e., $\mathbf{R}_j = \mathbf{R}_j \cup \{i\}$. If node i happens to be which node j has sent a matching request, node j activates the link with node i, i.e., $l_{ij}(t) = 1$. Then, nodes i and j send dropping messages to their neighbors in $\mathbf{N}_i$ and $\mathbf{N}_j$. When receiving the dropping message from node i, node k removes i from $\mathbf{N}_k$, i.e., $\mathbf{N}_k = \mathbf{N}_k / \{i\}$. If node i is the current candidate to which node k has sent a matching request, node k sends a new matching request to the next candidate with the largest edge weight in $\mathbf{N}_k$. The process repeats until node i is connected or $\mathbf{N}_i = \emptyset$.

Note that each node i only needs to send up to one message to each of its immediate neighbor j, since node i only removes nodes from $\mathbf{N}_i$ by sending matching requests or dropping messages. In particular, a) node i sends a matching request to node j and removes j from $\mathbf{N}_i$, if node j is selected to be the candidate; or b) node i sends dropping messages to the remaining nodes in $\mathbf{N}_i$, if node i is connected (or

Algorithm 3 The Proposed Distributed Online Learning Framework of Fog Computing

```
Input: The network dynamics observed at each node $i$ at time slot $t$, including
    $A_i(t)$, $F_i(t)$, $C_{ij}(t)$, $\varepsilon_i(t)$ and $\varepsilon_{ij}(t)$
Output: The decisions on task offloading, processing and result delivery at the
    time slot
1: Compute the weights $\alpha_i^s(t)$, $\beta_{ij}^s(t)$ and $\gamma_{ij}^s(t)$ based on the queues (i.e., Lagrange
    multipliers) of its own and immediate neighbors
2: Allocate the computing resources by (4.19)
3: Solve the link activation using Algorithm 2
4: Schedule the offloading and routing based on (4.22)
```

matched). As a result, the total number of messages during the proposed distributed algorithm is less than $2|\mathbf{E}'|$, given the total $|\mathbf{E}'|$ edges in $\mathbf{G}'(t)$. The time-complexity of Algorithm 2 is $\mathcal{O}(|\mathbf{E}'|)$.

Algorithm 3 summarizes the proposed distributed online learning framework of fog computing. Note that Algorithm 3 is fully distributed with strong scalability in the era of IoT. In particular, at each time slot t, each node i can optimize the computing resource allocation, task offloading, result delivery and link activation, only based on the knowledge of itself and its immediate neighbors.

The time complexity of Algorithm 3 depends on two parts: i.e., the optimal decisions on task offloading based on (4.19), processing and result delivery based on (4.22) (in lines 1, 2,and 4), and the active link construction (Algorithm 2 in line 3).

1. In the first step, Algorithm 3 computes the weights $\alpha_i^s(t)$, $\beta_{ij}^s(t)$ and $\gamma_{ij}^s(t)$ based on the queues to make decisions. For node i, the time complexity for computing weights $\alpha_i^s(t)$ is $\mathcal{O}(N)$, due to $s \in \boldsymbol{N}$. For node i, weights $\beta_{ij}^s(t)$ and $\gamma_{ij}^s(t)$ can be computed together with the time complexity of $\mathcal{O}(N^2 + NM)$, due to $s \in \boldsymbol{N}$ and $j \in \boldsymbol{N}_i$. $\boldsymbol{N}_i'$ is the set of immediate neighbors for node i, and the number of immediate neighbors for node i is less than the number of IoT devices and edge servers $N + M$. The time complexity of this step is $\mathcal{O}(N^2 + NM)$.

2. In the second step, the time complexity of Algorithm 2 is $\mathcal{O}(|\boldsymbol{E}'|)$, where

$\boldsymbol{E}'$ is the number of edges in the reconstructed network topology graph, and $|\boldsymbol{E}'| \leq N(N+KM-1)/2$. The time complexity for this step is $\mathcal{O}(N^2+KNM)$.

As a result, the time complexity of Algorithm 3 is $\mathcal{O}(N^2 + KNM)$. Compared to the centralized approach solved by Edmonds' blossom algorithm with the complexity $\mathcal{O}\big(|\mathbf{E}'||N + KM|^2\big)$[101], the proposed algorithm with quadratic complexity is scalable for IoT devices, as shown in Fig. 4.8.

4.3 Optimality Analysis

In this section, we prove that the proposed fully distributed online learning of fog computing is asymptotically optimal. We find that the optimality loss, resulting from the sub-optimal matching heuristic, can be compensated and asymptotically diminish by reducing the stepsize of stochastic gradient descent and delaying the convergence of online learning.

4.3.1 Asymptotic Optimality and Convergence Time of Centralized Online Learning

We start by proving that the centralized online learning of fog computing is asymptotically optimal, where the decisions on task offloading and result delivery are optimally solved by Edmonds' blossom algorithm in a centralized manner, as described in Section 4.2.2.

Let $\Phi^*(\mathbf{x}(t))$ denote the long-term time-average cost of the centralized online learning of fog computing, and Φ^* be the offline optimum (minimized in a posteriori manner violating causality). We can establish the asymptotic optimality of the centralized online learning of fog computing, as stated in the following theorem.

Theorem 3. *The gap between $\Phi(\mathbf{x}^*(t))$ and Φ^* satisfies*

$$\Phi^*(\mathbf{x}(t)) - \Phi^* \leq \epsilon U, \tag{4.27}$$

where $U = \frac{1}{2}\{\sum_{i\in\mathbf{N}}[(\xi_i + 1)F^{\max}T/\rho_{\min} + 2\sum_{j\in\mathbf{N}} C_{ij}^{\max}T + A_i^{\max}]\}^2$ is a constant, and ϵ is the stepsize of the stochastic gradient descent.

Proof. Taking squares on both sides of (4.4) and (4.5), and then exploiting the identity inequality for $(\max[a-b,0]+c)^2 \leq a^2+b^2+c^2+2a(c-b)$ for any $a,b,c \geq 0$, we obtain

$$\begin{aligned}[Q_i^s(t+1)]^2 \leq{}& [Q_i^s(t)]^2 + 2Q_i^s(t)[\sum_{j\in\mathbf{N}}(b_{ji}^s(t)-b_{ij}^s(t)) + A_i^s(t) - f_i^s(t)/\rho_s] \\ &+[f_i^s(t)/\rho_s + \sum_{j\in\mathbf{N}} b_{ij}^s(t)]^2 + [\sum_{j\in\mathbf{N}} b_{ji}^s(t) + A_i^s(t)]^2,\end{aligned} \tag{4.28a}$$

$$\begin{aligned}[D_i^s(t+1)]^2 \leq{}& [D_i^s(t)]^2 + 2D_i^s(t)[\sum_{j\in\mathbf{N}}(d_{ji}^s(t)-d_{ij}^s(t)) + \tfrac{\xi_s}{\rho_s}f_i^s(t)] \\ &+[\sum_{j\in\mathbf{N}} d_{ij}^s(t)]^2 + [\sum_{j\in\mathbf{N}} d_{ji}^s(t) + \tfrac{\xi_s}{\rho_s}f_i^s(t)]^2.\end{aligned} \tag{4.28b}$$

Considering a standard quadratic Lyapunov function $\mathfrak{L}(t) = \frac{1}{2}\sum_{i,s\in\mathbf{N}}[Q_i^s(t)^2 + D_i^s(t)^2]$ [31], the drift $\Delta\mathfrak{L}(t)$ readily follows that

$$\begin{aligned}&\Delta\mathfrak{L}(t) = \mathfrak{L}(t+1) - \mathfrak{L}(t) \\ &\leq U + \sum_{i,s\in\mathbf{N}} D_i^s(t)[\sum_{j\in\mathbf{N}}(d_{ji}^s(t)-d_{ij}^s(t)) + \tfrac{\xi_s}{\rho_s}f_i^s(t)] \\ &+ \sum_{i,s\in\mathbf{N}} Q_i^s(t)[\sum_{j\in\mathbf{N}}(b_{ji}^s(t)-b_{ij}^s(t)) + A_i^s(t) - f_i^s(t)/\rho_s],\end{aligned} \tag{4.29}$$

where U is a constant by exploiting the Cauchy–Schwarz inequality $(\sum_i a_i)^2 \geq \sum_i a_i^2$ for $\forall a_i \geq 0$ in (4.28a) and (4.28b).

Taking expectations over $\boldsymbol{\omega}^t$ and adding $\frac{1}{\epsilon}\mathbb{E}[\Phi(\mathbf{x}(t))]$ on both sides (where $\mathbf{x}(t)$ is the optimal policy by solving (4.16)), we arrive at

$$\begin{aligned}&\mathbb{E}[\Delta\mathfrak{L}(t)] + \tfrac{1}{\epsilon}\mathbb{E}[\Phi(\mathbf{x}(t))] \leq U + \tfrac{1}{\epsilon}\mathbb{E}\Big\{\Phi(\mathbf{x}(t)) \\ &+\epsilon\sum_{i,s\in\mathbf{N}} D_i^s(t)\big[\textstyle\sum_{j\in\mathbf{N}}(d_{ji}^s(t)-d_{ij}^s(t)) + \tfrac{\xi_s}{\rho_s}f_i^s(t)\big] \\ &+\epsilon\sum_{i,s\in\mathbf{N}} Q_i^s(t)\big[\sum_{j\in\mathbf{N}}(b_{ji}^s(t)-b_{ij}^s(t)) + A_i^s(t) - \tfrac{f_i^s(t)}{\rho_s}\big]\Big\} \\ &= U + \tfrac{1}{\epsilon}\mathbb{E}\big[\mathcal{L}(\mathbf{x}(t)(\epsilon\mathcal{Q}(t)), \epsilon\mathcal{Q}(t))\big] \\ &= U + \tfrac{1}{\epsilon}D(\epsilon\mathcal{Q}(t)) \leq U + \tfrac{1}{\epsilon}\Phi^*\end{aligned} \tag{4.30}$$

where $\mathcal{L}(\mathbf{x}(t), \boldsymbol{\lambda})$ is defined in (4.14); $\mathbf{x}(t)(\epsilon\mathcal{Q}(t))$ is the optimal primal variable as given by (4.16) (hence, $\mathbb{E}[\mathcal{L}(\mathbf{x}(t)(\epsilon\mathcal{Q}(t)), \epsilon\mathcal{Q}(t))] = D(\epsilon\mathcal{Q}(t))$); and the last inequality in (4.30) is due to the weak duality [93].

Summing up all the telescoping series over time slots $\{0, 1, \cdots, T-1\}$, (4.30) leads to

$$\mathbb{E}[\mathfrak{L}(T)] - \mathfrak{L}(0) + \frac{1}{\epsilon}\sum\nolimits_{t=0}^{T-1}\mathbb{E}[\Phi(\mathbf{x}(t))] \leq UT + \frac{T}{\epsilon}\Phi^*. \tag{4.31}$$

Due to the fact that $\mathfrak{L}(T) \geq 0$ and $\mathfrak{L}(0) < \infty$, we have

$$\Phi^*(\mathbf{x}(t)) = \frac{1}{T}\lim_{T\to\infty}\sum\nolimits_{t=0}^{T-1}\mathbb{E}[\Phi(\mathbf{x}(t))] \leq \Phi^* + \epsilon U. \tag{4.32}$$

This concludes the proof. □

Theorem 3 reveals that the time-average cost achieved by the centralized online learning converges to within an optimality bound of $\mathcal{O}(\epsilon)$, which diminishes as the stepsize of stochastic gradient descent $\epsilon \to 0$. The stepsize ϵ accounts for the convergence time of the proposed online learning. Given the stepsize ϵ, the convergence time of stochastic gradient descent linearly increases with $\mathcal{O}(1/\epsilon)$ [94]. The typical $[\mathcal{O}(1/\epsilon), \mathcal{O}(\epsilon)]$-tradeoff between convergence time and optimality loss in terms of the stepsize ϵ indicates that an $\mathcal{O}(1/\epsilon)$ convergence time allows for an $\mathcal{O}(\epsilon)$ close-to-optimal cost.

4.3.2 Asymptotic Optimality of Distributed Online Learning

We proceed to prove that the $\frac{1}{2}$-approximation resulting from the distributed matching heuristic for task offloading and result delivery can be compensated and asymptotically diminish by decreasing the stepsize of stochastic gradient descent in online learning. Let $\mathbf{x}(t) = \{\mathbf{l}(t), \mathbf{b}(t), \mathbf{d}(t), \mathbf{f}(t)\}$ denote the optimal solutions under the centralized online learning in Section 4.2.2, and $\widehat{\mathbf{x}}(t) = \{\widehat{\mathbf{l}}(t), \widehat{\mathbf{b}}(t), \widehat{\mathbf{d}}(t), \widehat{\mathbf{f}}(t)\}$ denote the solutions achieved by the proposed distributed online learning approach in Algorithm 3. We can establish that the per-slot optimality loss resulting from the $\frac{1}{2}$-approximation, is upper bounded and asymptotically diminishes, as stated in the following theorem.

Theorem 4. *The per-slot optimality loss, resulting from the $\frac{1}{2}$-approximation for distributed task offloading and result delivery, is upper bounded, i.e.,*

$$\Phi(\mathbf{x}(t)) - \Phi(\widehat{\mathbf{x}}(t)) \leq \epsilon B. \tag{4.33}$$

where $B = \frac{(N+MK)}{4} C^{\max} Q^{\max}$ *is a constant. Here,* $C^{\max} = \max_{i,j}\{C_{ij}^{\max}\}$ *is the maximum link capacity across the network, and* $Q^{\max} = \max_{i,s,t}\{Q_i^s(t), D_i^s(t)\}$ *is the maximum backlogs of unprocessed tasks and results at all the nodes, constrained by the physical memory of the devices.*

Proof. The computing resource allocations given by (4.19) are invariant in the $\frac{1}{2}$-approximation solution, i.e., $\mathbf{f}(t) = \widehat{\mathbf{f}}(t)$. As a result, the per-slot optimality loss comes from the $\frac{1}{2}$-approximation decomposition from (4.25) to (4.26), i.e.,

$$\begin{aligned} \Phi(\mathbf{x}(t)) - \Phi(\widehat{\mathbf{x}}(t)) &= \eta\big(\mathbf{b}(t), \mathbf{d}(t), \mathbf{l}(t)\big) - \eta\big(\widehat{\mathbf{b}}(t), \widehat{\mathbf{d}}(t), \widehat{\mathbf{l}}(t)\big) \\ &\leq \frac{1}{2} W(\mathbf{l}^*(t)). \end{aligned} \tag{4.34}$$

Note that the maximum number of active links in graph $\mathbf{G}'(t)$ is $\frac{|\mathbf{V}'|}{2} = \frac{N+MK}{2}$, and $w_{ij}(t) \leq \epsilon C^{\max} Q^{\max}$. Hence, we have $W(\mathbf{l}^*(t)) \leq \epsilon \frac{N+MK}{2} C^{\max} Q^{\max}$, and (4.33) can be proved by substituting this upper bound of $W(\mathbf{l}^*(t))$ into (4.34). □

From Theorems 3 and 4, we can assert that the optimality loss, resulting from the $\frac{1}{2}$-approximation decomposition for distributed online learning, is upper bounded and can asymptotically diminish, as the stepsize of stochastic gradient descent ϵ decreases, i.e.,

$$\Phi^*(\widehat{\mathbf{x}}(t)) - \Phi^* \leq \epsilon(U + B). \tag{4.35}$$

That is to say, online learning can make up for the optimal loss resulting from the distributed graph matching heuristic.

4.4 Simulation Results and Analysis

In this section, we evaluate the proposed distributed online learning of fog computing with $N = 20$ devices and $M = 2$ edge servers, where we run 5000 slots with slot length $T = 50$ms for the simulations of each data point. The CPU capacity of the devices is uniformly distributed from 2 to 4GHz, and the CPU capacity of the edge servers is 20GHz. The CPU powers of the devices and edge servers are 1W and 10W, respectively. The background task varies randomly and uniformly from 0 to

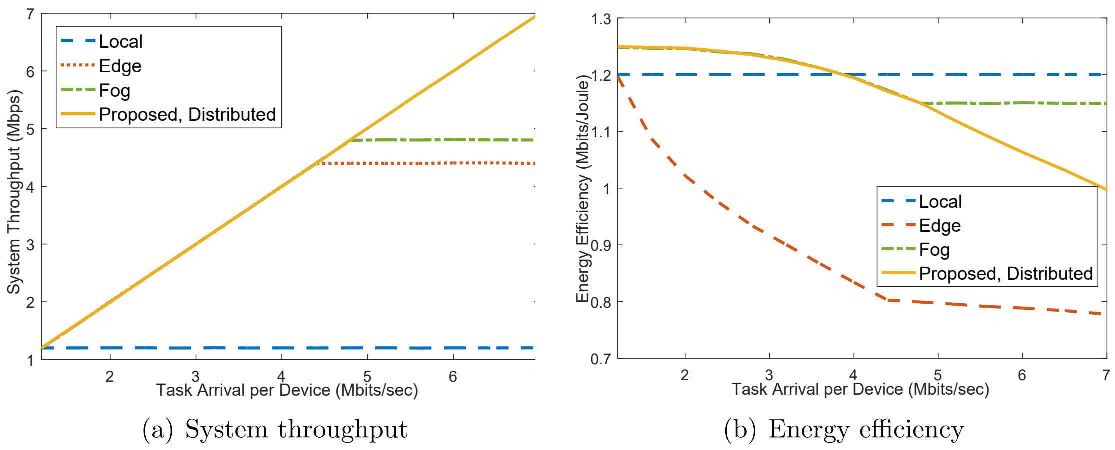
Table 4.2 : Simulation Parameters

Parameters	**Values**
Slot length, T	50ms
Number of device, N	20
Number of edge server, M	2
Maximum transmissions, K	3
Computing capacity of device/edge server	2-4/20GHZ
Computing power of device/edge server	1/10W
Link capacity	5-15Mbps
Transmission powers for D2D/edge server	23/33dBm
Task arrival, $A_i(t)$	4Mbits/sec
CPU cycles required for processing a task bit	2000cycles/bit
Ratio of results to tasks, ξ_i	0.5

40% per time slot. Each device is connected to 6 other devices and is in the coverage of both edge servers. Each edge server can maintain up to $K = 3$ communication links with the devices per slot. The link capacity varies randomly and uniformly from 5 to 15Mbps, and the transmission powers between the devices, and between the devices and edge servers, are 23dBm and 33dBm, respectively.

The tasks only arrive at 5 out of 20 devices to model the spatial variations of task arrivals, where $A_i(t)$ is 4Mbits/sec, $\rho_i = 2000$cycles/bit and $\xi_i = 0.5$. The stepsize of the stochastic gradient descent is set as $\epsilon = 1/80$; unless otherwise specified. Synthetic data are used for the simulations. The parameters of the synthetic dataset are taken from the state of the art [18, 19, 21], for fair comparisons between the proposed approach and the state of the art. Table 4.2 summarizes the parameters used in the simulation.

For comparison purpose, we also simulate four other approaches: (a) local execution approach (denoted by "Local" in the simulation), where the devices buffer and execute all the tasks locally; (b) edge computing approach (denoted by "Edge"), where the devices can only offload tasks to the edge servers for processing, as in MECO [16–18, 67, 69, 70, 75–78], and do not offload tasks to each other; (c) peer-to-peer fog computing approach [19, 21, 79–82] (denoted by "Fog"), where each



(a) System throughput (b) Energy efficiency

Figure 4.4 : The comparing of the system throughput and energy efficiency between the proposed approach and the existing benchmarks, as $A_i(t)$ increases from 1.2 to 7Mbits/sec.

device only offloads its tasks to its peers (i.e., neighboring devices) for processing, and does not offload tasks to the edge servers; and (d) centralized online learning approach (denoted by "Proposed, Centralized"), where the optimal task offloading and result delivery is solved by using the Edmonds' blossom algorithm with the global view at a central controller, hence offering little scalability to large-scale networks. As compared to the proposed approach, the task offloading are confined to either from devices to edge servers in the edge computing approach, or among devices in the peer-to-peer fog computing. The decisions on network operations in the edge computing, peer-to-peer fog computing, and the proposed centralized online learning approaches are all based on the online learning approach derived in Section 4.2, with the proved optimality in Theorem 3.

Fig. 4.4 compares the stabilized system throughput and energy efficiency between the proposed distributed online learning, local execution approach and edge computing approach, where the task arrival $A_i(t)$ increases from 1.2 to 7Mbits/sec. We can see in Fig. 4.4(a) that, by unifying the computing resources at the near-by devices and edge servers, the proposed distributed online learning can process all the arrived tasks and its system throughput linearly increases to 35Mbps with the growth of task arrivals. Edge computing approach can process all the tasks under light traffic conditions, i.e., $A_i(t) \leq 4.4$Mbits/sec, but with the increase of task arrivals, the system throughput of the edge computing approach stabilizes at 22Mbps after exhausting all the computing resources at the edge servers and the five devices with task arrivals. The proposed approach is able to increase up to 59% system throughput, as compared to the edge computing approach. Similar to the edge computing approach, the peer-to-peer fog computing approach can achieve up to 24Mbps system throughput after using up the computing resources at all the 20 mobile devices. The local execution approach undergoes the lowest system throughput by only processing tasks locally, and attains an energy efficiency of 1.2Mbits/Joule under different task arrivals, as shown in Fig. 4.4(b).

We can also see in Fig. 4.4(b) that the energy efficiencies of the proposed distributed online learning and peer-to-peer fog computing approaches only slightly drop, are higher than that of the local execution approach when $A_i(t) \leq$ 4Mbits/sec, and decline almost linearly when $A_i(t) >$ 4Mbits/sec. This is because, under light traffic conditions, the tasks arriving at the inexpensive IoT devices are offloaded via the energy-efficient communication links between proximate devices, and processed at the powerful devices (e.g., smart phones and laptops) with high energy efficiency. The energy efficiency of the edge computing approach declines almost linearly with the increase of task arrivals when $A_i(t) \leq$ 4.4Mbits/sec due to the additional energy consumption for task offloading and processing at edge server. The proposed approach can be up to 43% more energy-efficient than the edge computing approach under the same system throughput when $A_i(t) =$ 4.4Mbits/sec. After reaching their maximum system throughputs, the energy efficiencies of the edge computing and peer-to-peer fog computing only slightly decline with the growth of task arrivals. This is because the system becomes increasingly unstable and additional energy consumption is required for task offloading with the increase of their system throughput; see Fig. 4.4(a).

Fig. 4.5 evaluates the stabilized energy efficiency achieved by the proposed distributed online learning and the other approaches, as N increases from 5 to 30, where $A_i(t) =$ 4Mbits/slot. In the case of $A_i(t) =$ 4Mbits/slot, the proposed distributed online learning and edge computing approaches can process all the tasks, as shown in

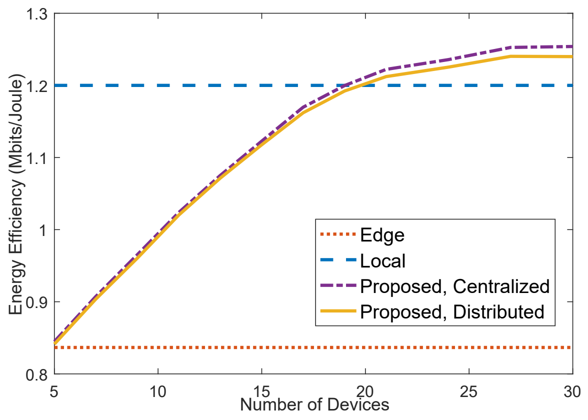


Figure 4.5 : The stabilized energy efficiency of the proposed distributed online learning and the other approaches, as the number of devices N increases from 5 to 30.

Fig. 4.4. We can see in Fig. 4.5 that the energy efficiencies of the local execution and edge computing approaches are 1.2 and 0.83Mbits/Joule, regardless of the number of devices. On the other hand, with the increase of N, the proposed distributed online learning and the centralized online learning can exploit the increasingly available computing resources at nearby devices, and hence their energy efficiencies linearly increase when $N \leq 17$. After that, their energy efficiencies stabilize at the values around 1.08Mbits/Joule, since the computing resources at nearby devices are already sufficient for processing the tasks under $A_i(t) = 4$Mbits/slot. Note in Fig. 4.5 that there exists a small gap of energy efficiency between the proposed distributed online learning and the centralized online learning, i.e., the optimality loss resulting from the $\frac{1}{2}$-approximation for distributed task offloading and result delivery, as described in Section 4.3.

Fig. 4.6 plots the stabilized system energy consumption of the proposed distributed online learning, the edge computing approach, and the centralized online learning, as the reciprocal of the stepsize of stochastic gradient descent, i.e., $1/\epsilon$ increases from 40 to 140. We can see that the the system energy consumptions of the proposed distributed online learning decrease with the increase of $1/\epsilon$, and stabilize when $1/\epsilon \geq 100$. The optimality gap between the proposed distributed online learning and the centralized online learning (as also shown in Fig. 4.5) diminishes

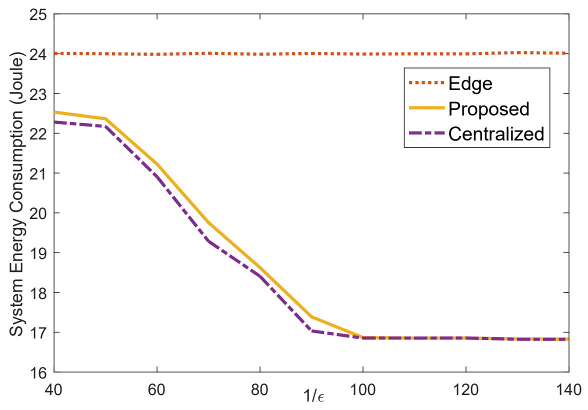


Figure 4.6 : The stabilized system energy consumption versus the reciprocal of the stepsize of stochastic gradient descent, $1/\epsilon$.

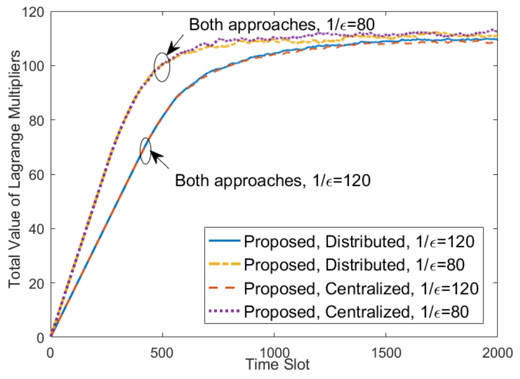
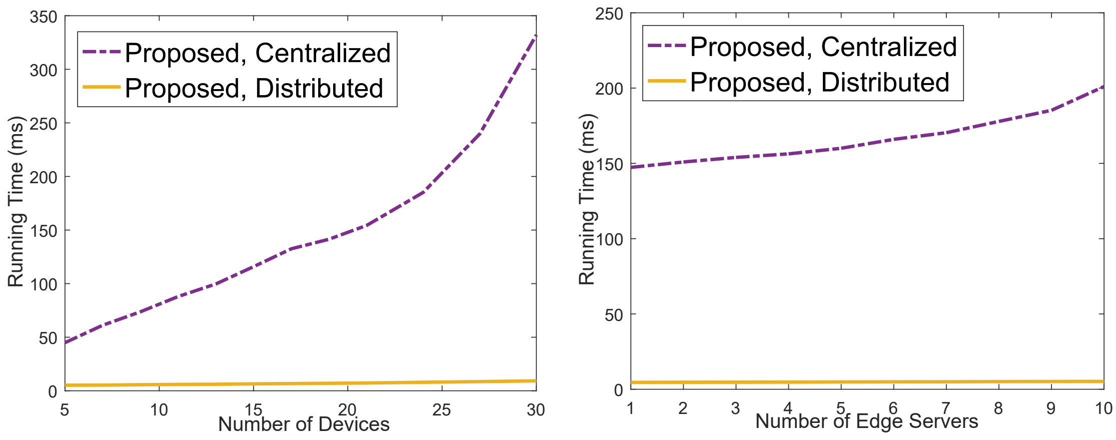


Figure 4.7 : The convergence of the Lagrange multipliers of the proposed distributed online learning and the centralized online learning as t evolves.

with the decrease of ϵ. This validates the asymptotic optimality of the proposed distributed online learning under the $\frac{1}{2}$-approximation for distributed task offloading and result delivery, as dictated in Section 4.3.

Fig. 4.7 shows the change of the Lagrange multipliers of the proposed distributed online learning and the centralized online learning, as t increases from 0 to 2000 time slots. We can see in Fig. 7 that the values of the Lagrange multipliers of both the proposed distributed online learning and the centralized online learning first increase and then stabilize at the same value over time. It is also worth mentioning that in the case of $1/\epsilon = 80$, the Lagrange multipliers of the proposed distributed and centralized approaches are fluctuant after convergence, due to the large ϵ. Given ϵ,



(a) Number of devices (b) Number of servers

Figure 4.8 : The running times of the proposed distributed and centralized online learning approaches, as the numbers of devices and edge servers increase.

the convergence times of the proposed distributed online learning and its centralized counterpart are similar. With the decrease of the stepsize from $1/\epsilon = 80$ to $1/\epsilon = 120$, the online learning approaches require increasingly long convergence times to stabilize the system, but can reduce the energy consumptions, as shown in Fig. 4.6. This is the typical $[\mathcal{O}(1/\epsilon), \mathcal{O}(\epsilon)]$-tradeoff between convergence time and optimality loss in stochastic gradient descent, as discussed in Section 4.3.

Fig. 4.8 shows the running times of the proposed distributed and centralized online learning approaches, as the numbers of devices and edge servers increase. Particularly, the running time of the proposed distributed online learning approach only increases linearly to the numbers of devices and edge servers, while the running time of its centralized counterpart increases quadratically. As a result, when the network scales, the running time of the proposed distributed approach can be as much as 96% lower than that of its centralized counterpart. The 96% saving of running time is due to the significantly lower time-complexity which the proposed distributed algorithm requires to solve a suboptimal, $\frac{1}{2}$-approximation solution for (4.25) than its centralized counterpart requires to use the Edmonds' blossom algorithm to solve an optimal solution for (4.25). The loss of the optimality can diminish by extending the learning time, and the distributed approach can also asymptotically approach the global optimum, as validated in Figs. 4.6 and 4.7.

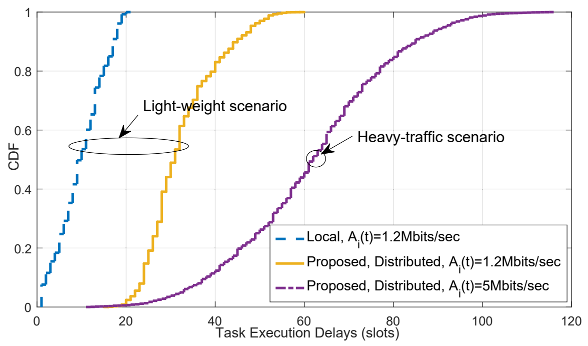


Figure 4.9 : The CDF of the task execution delays of the proposed distributed online learning approach and local execution approach under light-weight and heavy traffic scenarios. Note that the local execution approach cannot process the large size of tasks under heavy-traffic scenario, hence resulting in unstable task queues and unbounded queuing delays (increasing with the simulation durations and hence the CDF of local execution approach under heavy-traffic scenario is not plotted).

Fig. 4.9 plots the cumulative distribution function (CDF) of the task execution delays of the the proposed distributed online learning approach and local execution approach under light-weight and heavy traffic scenarios. In the light-weight scenario with $A_i(t) = 1.2$Mbits/sec, the proposed distributed online learning approach experiences longer task execution delays than the local execution approach, due to the additional time for task offloading and processing remotely, but achieves higher energy efficiency as shown in Fig. 4.4. With the growth of traffic arrivals, the task execution delays of the proposed approach under the heavy-traffic scenario with $A_i(t) = 5$Mbits/slot is longer than that under light-weight scenario, since tasks need to be offloaded increasingly to devices hops away for processing. Note that the local execution approach cannot process the large size of tasks under heavy-traffic scenario, hence resulting in unstable task queues and unbounded queuing delays (increasing with the simulation durations and hence the CDF of local execution approach under heavy-traffic scenario is not plotted.) It is worth mentioning that the maximum difference of the task execution delays of the proposed approach can be more than 100 slots under the heavy traffic scenario. The variations of the task delays result in the unfairness of the proposed approach, as will be discussed in Section 4.5.

Fig. 4.10 plots the stabilized energy efficiency of the proposed distributed online

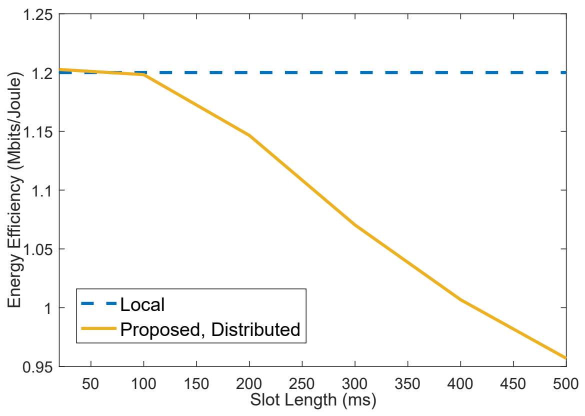


Figure 4.10 : The stabilized energy efficiency of the proposed distributed online learning approach and the local execution approaches, as the slot length T increases from 20 to 500ms.

learning approach and the local execution approach, as the slot length T increases from 20 to 500ms. We can see in Fig. 4.10 that the energy efficiency of the local execution approach is always 1.2Mbits/Joule, as already shown in Figs. 4.4(b) and 4.5. In contrast, the energy efficiency of the proposed distributed online learning approach only slightly declines when $T \leq 100$, and then linearly decreases with the growth of T. This is because a shorter slot length T can provide a finer time resolution for decision-makings on task offloading and processing, and result delivery, thereby achieving more efficient use of system energy.

4.5 Discussion on Delay-Sensitive Tasks

For illustration convenience, we assume all tasks are of the same priority, and the queues operate on a simple FIFO basis. This may cause unfairness (i.e., the variations of task delays), as a head-of-line (HOL) task of a queue can be offloaded, become the bottom of another FIFO queue, and hence undergoes a significantly increased queuing delay, as shown in Fig. 4.9.

The proposed approach can be extended to capture the execution delays and fairness (i.e., the delay variations) of the tasks by setting priority queues with meticulously designed weights. Specifically, we can design the weight of a task (and its corresponding result) to be the age of the task (in time slots). The age of the task grows until the delivery of the result. All the queues operate as priority queues, where tasks can be arranged in the descending order of task age and the oldest tasks are placed at the HOL. The priority queues would not affect the asymptotic optimality of the proposed approach. This is because the proposed approach only depends on the Lagrange multipliers (or in other words, the queue lengths) which are unaffected by the changing order of the tasks within the queues.

The proposed approach can also be extended to a general scenario, where tasks have different priorities and high-priority tasks require short delays. By using priority queues, the weight of a task (and its corresponding result) is the priority of the task. As a result, a high-priority task/result can be placed at the HOL of the queue to expedite the process. The delay of high-priority tasks can be substantially reduced, and their QoS can be improved.

Chapter 5

Distributed Online Optimization of Fog Computing for Internet-of-Things under Finite Device Buffers

Lyapunov optimization has shown to be effective for online optimization of fog computing, asymptotically approaching the optimality only achievable off-line. However, it is not directly applicable to the Internet-of-Things, as inexpensive sensors have small buffers and cannot generate sufficient backlogs to activate the optimization. This chapter proposes an enabling technique for the Lyapunov optimization to operate under limited finite buffers of data sources, i.e., the practical IoT devices, without loss of asymptotic optimality. This is achieved by optimizing the biases (also known as "virtual placeholders") of individual queues to create sufficient queue differences to drive data to flow. The key contributions of the chapter are beyond the direct application of Lyapunov optimization, and can be summarized as follows.

- We cast the Lyapunov optimization problem for distributed online optimization of fog computing subject to the limited buffers of IoT devices. The asymptotic optimality of the solution is proved, provided there are sufficient queues differences to drive the optimization.

- We optimize the virtual placeholders of all queues to create sufficient queue differences and drive Lyapunov optimization to operate under limited finite buffers. The optimization of the virtual placeholders is proved to be a new three-layer shortest path problem, and solved in a distributed manner by extending the celebrated Bellman-Ford algorithm.

- The sizes of the optimal virtual placeholders decline fastest along the shortest paths from the IoT devices through edge servers to the data center, thereby

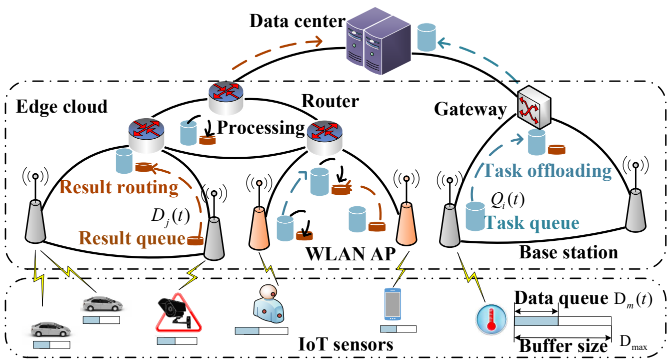


Figure 5.1 : The application of fog computing in IoT networks.

preventing unnecessary detours and reducing end-to-end delays.

- Corroborated by simulations, the Lyapunov optimization would require $V \geq 200$ to achieve the asymptotic optimality. The proposed approach is able to operate under such large V values, while the direct application of the Lyapunov optimization stops working for $V \geq 20$. Moreover, our approach can reduce end-to-end delays by 90%, as compared to the direct application of Lyapunov optimization, even when $V = 5$.

5.1 System Model

Fig. 5.1 illustrates the application of fog computing in IoT networks, where a number of IoT devices, connected to an edge cloud through wireless interfaces, produce big data destined for a data center. The system consists of three types of nodes, namely, *IoT devices*, *edge servers*, and a *data center*. A large number of IoT devices with heterogeneous interfaces to an edge cloud generate large amounts of data destined for the data center (or sink) for big data analytics and services. Edge servers, such as base stations, access points, switches, routers, and gateways in the edge cloud, can collaborate for fog computing, by processing data at the point of capture and delivering results to the data center.

Let $\mathbf{N} = \{0, \dots, N\}$ collect the indexes for the data center and edge servers, where the data center is labeled by index 0 and the N edge servers are labeled from 1 to N. Let $\mathbf{M} = \{1, \dots, M\}$ collect the indexes for the IoT devices, and $\mathbf{M}_i$ denote the set of the IoT devices connected to edge server i. The system operates on a slotted basis with slot length T.

5.1.1 Network Model and Cascaded Queues

We consider a data queue $D_m(t)$ with maximum buffer size $D_{\max}$ at each IoT device m to buffer sensory data, and an unprocessed data queue $Q_i(t)$ and a processed result queue $R_i(t)$ at each edge server i to buffer the data and results at the edge server.

Fig. 5.1 also illustrates the data flows in the edge cloud. From bottom to top, the sensory data can be uploaded from the buffers $D_m(t)$ of the IoT devices to the queues of unprocessed data $Q_i(t)$ at the edge servers. The unprocessed data can be offloaded to the queues of unprocessed data at its neighboring servers, or processed and placed into the result queue $R_i(t)$. Any results are forwarded hop-by-hop to the data center.

A. Data arrival and uploading at the IoT devices

The size of data, arriving at data queue $D_m(t)$ during time slot t, denoted by $A_m(t)$, is assumed to be independent and identically distributed (i.i.d.)* and upper bounded by $A_m^{\max}$. The sensory data in $D_m(t)$ can be uploaded to the edge cloud via wireless links. In the presence of background traffic, the number of subchannels at edge server i available for the IoT devices during slot t, denoted by $K_i(t) \leq L$, can vary between time slots. Multiple IoT devices can be scheduled within a subchannel via time-division multiplexing. The wireless links are assumed to undergo i.i.d. block fading, i.e., the channels remain unchanged during a time slot and can vary between slots. Let $c_m(t)$ denote the channel capacity in the uplink of IoT device m at slot t, and ε_m denote the cost for transmission per unit time, e.g., the transmit

*In many cases, the assumption of i.i.d. data size is reasonable, since the data are sensed from the ambient enviroment of devices. The assumption can be relaxed to be non-i.i.d., e.g., by adopting a Discrete-Time Markov Chain (DTMC) analysis [31]. This can be achieved by extending the one-slot drift $\Delta(t)$ in (5.14) to a T-slot drift, i.e., $\Delta_T(t) = \mathbb{E}[L(t+T) - L(t)]$, as discussed in [31]. By aggregating multiple non-i.i.d. slots (e.g., the recurrence time of a state in the DTMC), the stochastic process of Lyapunov drift between different T slots can be proved to be i.i.d. at the interval of T slots.

power in terms of energy consumption. $c_m(t) \leq c^{\max}$. $c^{\max}$ is the upper bound of the link capacity, given the finite transmit power of the device.

B. Data dispatch, processing and result delivery in the edge cloud

The uploaded data can be partitioned, processed and dispatched among multiple servers. The topology of the edge servers and the data center can be described by a graph $G = \{\mathbf{N}, \mathbf{E}\}$. Let $\mathbf{E}$ collect the wired links between the edge servers. The capacity of each wired link remains unchanged within a slot, and may change between slots due to random background traffic. The capacity of link (i, j) at slot t, denoted by $C_{ij}(t) \in (0, C_{ij}^{\max}]$, accounts for the bi-directional transmission between edge servers i and j. $\epsilon_{ij}(t)$ denotes the cost of transmitting a bit of data from server i to server j with $\epsilon_{ij}^{min} \leq \epsilon_{ij}(t) \leq \epsilon_{ij}^{\max}$.

Let $\widehat{F}_i$ (in CPU cycles per second) denote the computing capability of server i, and $\delta_i(t)$ denote the percentage of background tasks of the server during slot t. The available computing capacity of the server is $F_i(t) = [1 - \delta_i(t)]\widehat{F}_i T$. Let $\epsilon_i(t)$ denote the computing cost of server i per CPU cycle with $\epsilon_i^{min} \leq \epsilon_i(t) \leq \epsilon_i^{\max}$.

5.1.2 Causality Constraints and Queue Dynamics

Over the wireless links, $\tau_m(t)$ denotes the scheduled transmission duration of IoT device m during slot t. Since a typical IoT device is inexpensive, simple and narrow-band [103], it can only access a subchannel at the same time, satisfying

$$0 \leq \tau_m(t) \leq T, \forall m \in \mathbf{M}; \tag{5.1}$$

$$\sum\nolimits_{m \in \mathbf{M}_i(t)} \tau_m(t) \leq K_i(t)T, \ \forall i \in \mathbf{N}, \tag{5.2}$$

where (5.1) indicates that a narrow-band IoT device can only be allocated part of a subchannel; and (5.2) ensures that the total transmission time of the devices must not exceed the number of available subchannels, i.e., $K_i(t)$, for each edge server i within a slot. The amount of data uploaded from IoT device m to it-s connected edge server i, denoted by $d_m(t)$, is proportional to the transmission duration, and upper bounded by the available buffering data at the device, i.e.,

$d_m(t) = \min\{D_m(t), c_m(t)\tau_m(t)\}$. Hence, $D_m(t)$ satisfy the following causality condition:

$$D_m(t+1) = [D_m(t) - d_m(t)] + \min\{A_m(t), D_{\max} - D_m(t) + d_m(t)\}, \tag{5.3}$$

where $\min\{A_m(t), D_{\max} - D_m(t) + d_m(t)\}$ gives the size of the new data that can be admitted at slot t with the maximum buffer size $D_{\max}$. This is different from the setting of most Lyapunov optimization designs [18, 19, 21, 82].

In the edge cloud, $f_i(t)$ is the computing resources of edge server i allocated to process data at slot t, satisfying

$$0 \leq f_i(t) \leq F_i(t), \, \forall i \in \mathbf{N}. \tag{5.4}$$

Per time slot t, let $b_{ij}(t)$ denote the volume of data forwarded from server i to server j, and $r_{ij}(t)$ denote the volume of results dispatched from server i through server j to the data center. Since the total amount of data and results delivered over a link cannot exceed the link capacity, it is easy to establish the following causality constraint:

$$\begin{aligned} & b_{ij}(t) + r_{ij}(t) + b_{ji}(t) + r_{ji}(t) \leq C_{ij}(t), \, \forall (i,j) \in \mathbf{E}; \\ & b_{ij}(t) \geq 0, \, r_{ij}(t) \geq 0, \, \forall (i,j) \in \mathbf{E}. \end{aligned} \tag{5.5}$$

As a result, the backlog of unprocessed data at edge server i, $Q_i(t)$, can be updated per time slot t, according to

$$Q_i(t+1) = \max\{Q_i(t) - f_i(t)/\rho - \sum\nolimits_{j\in\mathbf{N}} b_{ij}(t), 0\} + \sum\nolimits_{j\in\mathbf{N}} b_{ji}(t) + \sum\nolimits_{m\in\mathbf{M}_i} d_m(t), \tag{5.6}$$

where $f_i(t)/\rho$ gives the volume of processed data output from the queue of server i, ρ is the number of CPU cycles for processing a bit of data, and $\sum_{m\in\mathbf{M}_i} d_m(t)$ is the number of data uploaded from the IoT devices at slot t. The first term on the right-hand side (RHS) of (5.6) accounts for the data in the queue at the end of time slot t, after part of the data have been processed at server i or dispatched to other servers. The second and the third terms account for new data dispatched from other servers or uploaded from the IoT devices, respectively.

Likewise, the backlog of results at server i, $R_i(t)$, can be updated as given by

$$R_i(t+1) = \max\{R_i(t) - \sum\nolimits_{j\in\mathbf{N}} r_{ij}(t), 0\} + \sum\nolimits_{j\in\mathbf{N}} r_{ji}(t) + \xi f_i(t)/\rho, \tag{5.7}$$

where $\xi f_i(t)/\rho$ is the size of results injected into the queue via data processing, and ξ is the ratio of a result to its unprocessed data in terms of size. Note that $R_0(t) = 0\ \forall t$, since the data center acts as the sink of data.

The system is stable, if and only if the following is met [31]

$$\overline{D_m(t)} < \infty,\ \overline{Q_i(t)} < \infty,\ \overline{R_i(t)} < \infty,\ \forall i, m, \tag{5.8}$$

where $\overline{X(t)} = \lim_{T\to\infty} \frac{1}{T}\sum_{\tau=0}^{T-1} \mathbb{E}[X(\tau)]$ denotes the time-average of a stochastic process $X(t)$.

5.2 Fog Computing under Large IoT Buffers

5.2.1 Problem Statement and Reformulation

We take cost as a generic measure of the performance of fog computing. The total cost can be written as

$$\varphi(t) = \sum\nolimits_{m\in\mathbf{M}} \varepsilon_m \tau_m(t) + \sum\nolimits_{i,j\in\mathbf{N}} \varphi_{ij}(t) + \sum\nolimits_{i\in\mathbf{N}} \varphi_i(t), \tag{5.9}$$

where $\varphi_{ij}(t) = \epsilon_{ij}(t)(b_{ij}(t) + r_{ij}(t))$ and $\varphi_i(t) = \epsilon_i(t) f_i(t)$ are the costs for data dispatch over link (i,j) and data processing at edge server i at time slot t, respectively.

Considering the stochastic nature of wireless channels, data arrivals and background traffic, we are interested in minimizing the overall time-average system cost while preserving the stability of the network. The problem of interest can be formulated as

$$\begin{aligned} &\min_{\mathbb{X}} \overline{\varphi(\mathbf{x}(t))} \\ &\text{s.t. } (5.1)\text{–}(5.8),\ \forall t, \end{aligned} \tag{5.10}$$

where $\mathbf{x}(t) = \{\tau_m(t), f_i(t), b_{ij}(t), r_{ij}(t), \forall m, i, j\}$ collects all the variables in regards of the data uploading, processing, dispatch, and result delivery at time slot t, and $\mathbb{X}$ collects the variables across all time slots.

Minimizing the time-average cost of fog computing over an infinite time horizon in (5.10) is challenging. In particular, the variables are coupled in time due to the queue dynamics (5.3), (5.6) and (5.7). The queue backlogs in (5.6) and (5.7) can also result in strong couplings of variables in space. For instance, the processing decisions of downstream edge servers depend on the output of their upstream counterparts, i.e., the optimal decisions of different edge servers are coupled.

The Lyapunov optimization is a widely adopted stochastic optimization technique to decouple temporally and spatially coupled variables and derive asymptotically optimal solutions, which has been extensively used for smart grid [104], network function virtualization [20], and fog computing [18, 19, 21, 82]. It is designed to minimize an upper bound of a drift-plus-penalty function at each slot, and proved to achieve an $[\mathcal{O}(V), \mathcal{O}(1/V)]$-tradeoff between the drift (i.e., the queue lengths) and the penalty (i.e., the time-average cost to be minimized).

By applying the Lyapunov optimization, problem (5.10) can be reformulated as

$$\begin{aligned} \max_{\mathbf{f}(t), \widetilde{\mathbf{b}}(t), \widetilde{\mathbf{r}}(t), \boldsymbol{\tau}(t)} \quad & g(\mathbf{f}(t)) + \phi(\widetilde{\mathbf{b}}(t), \widetilde{\mathbf{r}}(t)) + \eta(\boldsymbol{\tau}(t)) \\ \text{s.t.} \quad & (5.1),\ (5.2),\ (5.4),\ (5.5); \end{aligned} \tag{5.11}$$

where $\boldsymbol{\tau}(t) = \{\tau_m, \forall m \in \mathbf{M}\}$, $\mathbf{f}(t) = \{f_i(t), \forall i \in \mathbf{N}\}$, $\widetilde{\mathbf{b}}(t) = \{b_{ij}(t), b_{ji}(t), \forall i, j \in \mathbf{N}\}$, and $\widetilde{\mathbf{r}}(t) = \{r_{ij}(t), r_{ji}(t), \forall i, j \in \mathbf{N}\}$ are the decisions of data uploading, processing, dispatch, and result delivery, respectively; and

$$g(\mathbf{f}(t)) = \sum\nolimits_{i \in \mathbf{N}} [(Q_i(t) - \xi R_i(t))/\rho - V\epsilon_i(t)] f_i(t); \tag{5.12a}$$

$$\begin{aligned} \phi(\widetilde{\mathbf{b}}(t), \widetilde{\mathbf{r}}(t)) = & \sum\nolimits_{i,j \in \mathbf{N}} Q_i(t)[b_{ij}(t) - b_{ji}(t)] \\ & + R_i(t)[r_{ij}(t) - r_{ji}(t)] - V\epsilon_{ij}(t)(b_{ij}(t) + r_{ij}(t)); \end{aligned} \tag{5.12b}$$

$$\eta(\boldsymbol{\tau}(t)) = \sum\nolimits_{i \in \mathbf{N}, m \in \mathbf{M}_i} [D_m(t) - Q_i(t)] d_m(t) - V\varepsilon_m \tau_m(t). \tag{5.12c}$$

Here, V is a predefined non-negative coefficient to adjust the tradeoff between the queue lengths and the optimality loss (in terms of the time-average cost).

Proof. A quadratic Lyapunov function $L(t)$ can be defined as [31]

$$L(t) = \frac{1}{2}\{\textstyle\sum_{i\in\mathbf{N}}[Q_i(t)^2 + R_i(t)^2] + \sum_{m\in\mathbf{M}} D_m(t)^2\}. \tag{5.13}$$

A drift-plus-penalty function can be defined to minimize the time-average system cost while preserving the system stability, as given by

$$\Delta_V(t) = \Delta(t) + V\mathbb{E}\left[\varphi(t)\right], \tag{5.14}$$

where $\mathbb{E}\left[\varphi(t)\right]$ is the expectation of system cost at slot t. The upper bound of $\Delta_V(t)$ is given in the following Lemma.

Lemma 3. *For any queue backlogs and actions, $\Delta_V(t)$ is upper bounded by*

$$\begin{aligned}\Delta_V(t) \le U + V\mathbb{E}\left[\varphi(t)\right] &+ \sum_{m\in\mathbf{M}} D_m(t)\mathbb{E}[A_m(t) - d_m(t)] \\ &+ \sum_{i\in\mathbf{N}} Q_i(t)\mathbb{E}[\sum_{m\in\mathbf{M}_i} d_m(t) - f_i(t)/\rho + \sum_{j\in\mathbf{N}}(b_{ji}(t) - b_{ij}(t))] \\ &+ \sum_{i\in\mathbf{N}} R_i(t)\mathbb{E}[\xi f_i(t)/\rho + \sum_{j\in\mathbf{N}}(r_{ji}(t) - r_{ij}(t))],\end{aligned} \tag{5.15}$$

where

$$U = \frac{1}{2}\{[\sum_{i\in\mathbf{N}}(\xi+1)\widehat{F}_i/\rho + MTc^{\max} + \sum_{i,j\in\mathbf{N}} C_{ij}^{\max}]^2 + [\textstyle\sum_{m\in\mathbf{N}}(A_m^{\max})^2 + (c^{\max}T)^2]\}. \tag{5.16}$$

Proof. Taking squares on both sides of (5.3), (5.6) and (5.7), and then exploiting the inequality for $(\max[a-b,0]+c)^2 \le a^2 + b^2 + c^2 + 2a(c-b)$ for any $a, b, c \ge 0$, and $D_m(t+1) \leq D_m(t) - d_m(t) + A_m(t)$, we can obtain

$$D_m(t+1)^2 - D_m(t)^2 \leq A_m(t)^2 + d_m(t)^2 + 2D_m(t)[A_m(t) - d_m(t)]; \tag{5.17a}$$

$$\begin{aligned} Q_i(t+1)^2 - Q_i(t)^2 \leq{}& [f_i(t)/\rho + \textstyle\sum_{j\in\mathbf{N}} b_{ij}(t)]^2 + [\textstyle\sum_{j\in\mathbf{N}} b_{ji}(t) + \textstyle\sum_{m\in\mathbf{M}_i} d_m(t)]^2 \\ & + 2Q_i(t)[\sum_{j\in\mathbf{N}}(b_{ji}(t) - b_{ij}(t)) + \sum_{m\in\mathbf{M}_i} d_m(t) - f_i(t)/\rho]; \end{aligned} \tag{5.17b}$$

$$\begin{aligned} R_i(t+1)^2 - R_i(t)^2 \leq{}& [\sum_{j\in\mathbf{N}} r_{ji}(t) + \xi f_i(t)/\rho]^2 \\ & + [\sum_{j\in\mathbf{N}} r_{ij}(t)]^2 + 2R_i(t)[\xi f_i(t)/\rho + \sum_{j\in\mathbf{N}}(r_{ji}(t) - r_{ij}(t))]. \end{aligned} \tag{5.17c}$$

Plugging (5.17) into (5.14), we can achieve (5.15) with U satisfying

$$\begin{aligned} 2U \geq{}& \sum_{m\in\mathbf{N}} \mathbb{E}[A_m(t)^2 + d_m(t)^2] + \sum_{i\in N} \mathbb{E}\{[f_i(t)/\rho + \sum_{j\in\mathbf{N}} b_{ij}(t)]^2 \\ & + [\textstyle\sum_{j\in\mathbf{N}} b_{ji}(t) + \textstyle\sum_{m\in\mathbf{M}_i} d_m(t)]^2 + [\textstyle\sum_{j\in\mathbf{N}} r_{ij}(t)]^2 + [\textstyle\sum_{j\in\mathbf{N}} r_{ji}(t) + \xi f_i(t)/\rho]^2\}. \end{aligned} \tag{5.18}$$

Exploiting the inequality $(\sum_i a_i)^2 \geq \sum_i a_i^2$ for $\forall a_i \geq 0$, (5.16) provides an upper bound for (5.18) by replacing the expectations with their maximums. This concludes the proof. □

From Lyapunov optimization, (5.10) can be transformed to minimizing (5.15) at each time slot t, subject to the instantaneous constraints (5.1), (5.2), (5.4) and (5.5). By rearranging (5.15), this problem can be rewritten as (5.11), where both U and $A_m(t)$ are independent of the optimization variables and suppressed. □

It is non-straightforward for the solution of (5.11) to asymptotically achieve the optimality of (5.10) though. This is because the limited finite buffers of practical IoT devices $D_{\max}$ typically cannot satisfy the condition of asymptotic optimality in Lyapunov optimization, i.e., $V \gg 0$. In other words, the bounded queues of the IoT devices are typically not long enough to approach the optimality. As a matter of fact, $D_{\max}$ can be much shorter than the queues at the edge servers, and can even prevent the IoT devices from uploading data and the direct application of Lyapunov optimization from functioning. We propose to set up biases to enlarge the queues of individual devices, thereby not only enabling the Lyapunov optimization but also achieving optimality asymptotically.

5.2.2 Distributed Online Optimization under Large IoT Buffers

The reformulation from (5.10) to (5.11) is a direct application of Lyapunov optimization. Nevertheless, (5.11) provides a new and specific problem and requires specific solutions, as developed in this subsection. We observe in (5.11) that the decisions on data processing, the decisions on data dispatch and result delivery, and the schedules of IoT data uploading are ready to be decoupled. $g(\mathbf{f}(t))$ can be maximized subject to (5.4) for data processing; $\phi(\widetilde{\mathbf{b}}(t), \widetilde{\mathbf{r}}(t))$ can be maximized subject to (5.5) for data dispatch and result delivery; and $\eta(\boldsymbol{\tau}(t))$ can be maximized subject to (5.1) and (5.2) for the schedules of data uploading.

A. Data Processing, Dispatch, and Result Delivery

From (5.12a), the optimization of computing resources can be decoupled between edge servers, as given by

$$\max_{f_i(t)} [(Q_i(t) - \xi R_i(t))/\rho - V\epsilon_i(t)] f_i(t) \text{ s.t. } (5.4), \tag{5.19}$$

which is linear programming of weighted-sum maximization problem [99], and its optimal solution can be given by

$$f_i(t) = \begin{cases} F_i(t), \text{ if } (Q_i(t) - \xi R_i(t))/\rho - V\epsilon_i(t) > 0; \\ 0, \text{otherwise.} \end{cases} \tag{5.20}$$

From (5.12b), the optimization of data dispatch and result delivery can be decoupled between links, and the problem for link (i, j) can be rewritten as

$$\max_{\widetilde{\mathbf{b}}_{ij}(t), \widetilde{\mathbf{r}}_{ij}(t)} \phi_{ij}(\widetilde{\mathbf{b}}_{ij}(t), \widetilde{\mathbf{r}}_{ij}(t)) \text{ s.t. } (5.5), \tag{5.21}$$

where $\phi_{ij}(\widetilde{\mathbf{b}}_{ij}(t), \widetilde{\mathbf{r}}_{ij}(t)) = \beta_{ij}(t)b_{ij}(t) + \beta_{ji}(t)b_{ji}(t) + \gamma_{ij}(t)r_{ij}(t) + \gamma_{ji}(t)r_{ji}(t)$ can be

Algorithm 4 Online Optimization at Edge Servers

```
For each server $i$ at every time slot $t$:
1: Acquire $F_i(t)$, $\epsilon_i(t)$, and $\epsilon_{ij}(t)$
2: Observe the queues of its own and immediate neighbors
3: Allocate computing resources according to (5.20)
4: Schedule data dispatch and result delivery based on (5.22)
```

obtained by decoupling (5.12b) between links; $\beta_{ij}(t) = Q_i(t) - Q_j(t) - V\epsilon_{ij}(t)$; and $\gamma_{ij}(t) = R_i(t) - R_j(t) - V\epsilon_{ij}(t)$.

Here, (5.21) is also a weighted-sum maximization [99], and its optimal solution can be obtained by evaluating the weights $\beta_{ij}(t)$, $\beta_{ji}(t)$, $\gamma_{ij}(t)$ and $\gamma_{ji}(t)$ at edge servers i and j independently. If $\max\{\beta_{ij}(t), \gamma_{ij}(t)\} < 0$ or $\max\{\beta_{ij}(t), \gamma_{ij}(t)\} < \max\{\beta_{ji}(t), \gamma_{ji}(t)\}$, server i remains idle at slot t, i.e., neither dispatching data nor delivering results on link (i, j). Otherwise, server i occupies the link. Particularly, if $\beta_{ij}(t) > \gamma_{ij}(t)$, server i withholds from delivering results via server j. Instead, it dispatches data to server j with the data size specified by

$$b_{ij}(t) = \begin{cases} C_{ij}(t), \text{ if } \beta_{ij}(t) > \gamma_{ij}(t); \\ 0, \text{ otherwise.} \end{cases} \tag{5.22a}$$

If $\beta_{ij}(t) \leq \gamma_{ij}(t)$, server i withholds from dispatching data to server j. Instead, it delivers results through server j with the result size specified by

$$r_{ij}(t) = \begin{cases} C_{ij}(t), \text{ if } \beta_{ij}(t) \leq \gamma_{ij}(t); \\ 0, \text{ otherwise.} \end{cases} \tag{5.22b}$$

From (5.20) and (5.22), we can see that the optimal decisions of data processing, data dispatch, and result delivery can be made locally at every individual server in $\mathcal{O}(1)$ time-complexity, based on the local information of the server itself and its immediate neighbors. Algorithm 4 summarizes the decision-making process at each of the servers.

B. Data Uploading of IoT Devices

From (5.3), it is clear that device m cannot send more data than the data available in its buffer at slot t, i.e., $\tau_m(t) \leq D_m(t)/c_m(t)$. Therefore, (5.1) can be tightened to

$$0 \leq \tau_m(t) \leq T_m(t), \forall m \in \mathbf{M}, \tag{5.23}$$

where $T_m(t) = \min\{D_m(t)/c_m(t), T\}$. By rearranging (5.12c), each server i can optimize its own schedule $\boldsymbol{\tau}_i(t)$ for its associated devices in $\mathbf{M}_i$, as given by

$$\max_{\boldsymbol{\tau}_i(t)} \sum\nolimits_{m\in\mathbf{M}_i} \alpha_m^i(t)\tau_m(t) \text{ s.t. } (5.2) \text{ and } (5.23), \tag{5.24}$$

where

$$\alpha_m^i(t) = [D_m(t) - Q_i(t)]c_m(t) - V\varepsilon_m. \tag{5.25}$$

By interpreting $\alpha_m^i(t)$ as the unit profit of the i-th "item" and $K_i(t)T$ in (5.2) as the capacity of a knapsack, (5.24) becomes the linear relaxation of a knapsack problem. The optimal solution for the knapsack problem can be found by selecting the "item" with higher unit profit to fulfill the knapsack capacity [105].

Without loss of generality, we assume that the devices are sorted in the descending order of unit profit, i.e., $\alpha_m^i(t) \geq \alpha_{m'}^i(t)$ for $m > m'$. The optimal solution is $\tau_m^*(t) = T_m(t)$ if $\sum_{j=1}^m T_j(t) \leq K_i(t)T$. The breaking item is the first device allocated part of its requested transmission duration $T_m(t)$, as specified in (5.23). The index of the breaking item b can be identified, as given by

$$b = \arg\min_m \sum\nolimits_{j=1}^m T_j(t) > K_i(t)T. \tag{5.26}$$

As a result, the optimal uploading schedule of the IoT devices can be given by

$$\tau_m(t) = \begin{cases} T_m(t), \text{ if } m < b; \\ K_i(t)T - \sum_{j=1}^{b-1} T_j(t), \text{ if } m = b; \\ 0, \text{ otherwise,} \end{cases} \tag{5.27}$$

which involves ordering the devices against their unit profits, and can be achieved

Algorithm 5 Online Optimization at IoT Devices

```
For each IoT device $m$ at every slot $t$:
1: Feed back $D_m(t)$, $c_m(t)$ and $\varepsilon_m(t)$ to the edge server
For each edge server $i$ at every time slot $t$:
2: Acquire $K_i(t)$ and $Q_i(t)$
3: Collect feedbacks from IoT devices and evaluate (5.25)
4: Schedule the data uploading of IoT devices by (5.27)
```

by using the quicksort algorithm with a complexity of $\mathcal{O}(M_i \log M_i)$ [106], where M_i is the number of IoT devices associated with edge server i. Algorithm 5 summarizes the proposed algorithm for producing the optimal schedules of IoT data uploading.

Note that (5.27) requires explicit knowledge on $c_m(t)$, and $D_m(t)$ for $m \in \mathbf{M}_i$. In other words, at each time slot t, each IoT device needs to feed back its link capacity $c_m(t)$ and queue backlog $D_m(t)$, so that the edge server can evaluate $\tau_m(t)$ based on (5.27). By referring to our recent work [18], Algorithm 5 can be readily extended to operate based on out-of-date information on $c_m(t)$ and $D_m(t)$ to substantially reduce the feedback.

5.3 Distributed Optimization of Virtual Placeholders for S-mall IoT Buffers

From (5.20), (5.22) and (5.27), we can see that the data uploading, dispatch, processing, and result delivery are all driven by the differences of backlogs between the IoT devices, edge servers, and the data center. It is important to enlarge the differences of queues along the shortest paths from the IoT devices to the data center. However, the queue lengths at the IoT devices are practically constrained by their physical buffers $D_{\max}$, which can be much shorter than the queues at the edge servers. Not only would this prevent achieving the asymptotic optimality of Lyapunov optimization, but would even prevent the direct application of Lyapunov optimization from functioning.

We propose to place virtual placeholders (i.e., biases) into the queues, hence virtually enlarging the queues and increasing their differences. We prove that the optimization of the virtual placeholders can be translated into a new three-layer shortest path problem, and solved in a distributed manner by extending the Bellman-Ford algorithm. As a result, the virtual placeholders enable the distributed online optimization of fog computing under small IoT buffers, inviolate the solutions developed in Section 5.2, and achieve the asymptotic optimality of the solutions.

5.3.1 Optimality and Limitations of Lyapunov Optimization

We show the provisional asymptotic optimality of the solutions developed in Section 5.2.2, and the bounded instantaneous queue backlogs of the resultant fog computing system. Let $\overline{\varphi(t)}$ denote the stochastically minimized time-average cost based on the solutions, and φ^{opt} be the cost of (5.10) minimized off-line (in a posteriori manner) by overlooking causality. The gap between $\overline{\varphi(t)}$ and φ^{opt} can be shown to converge with the growth of V, as revealed in the following theorem.

Theorem 5. *The gap between $\overline{\varphi(t)}$ and φ^{opt} is within $\mathcal{O}(1/V)$, as given by*

$$\overline{\varphi(t)} - \varphi^{\text{opt}} \leq \mathcal{O}(1/V), \tag{5.28}$$

i.e., $\overline{\varphi(t)}$ is asymptotically optimal.

Proof. Let π denote any feasible control policy. We have

$$\begin{aligned}\Delta_V(t) \leq U + V\mathbb{E}\left[\varphi(t)|\pi\right] + \sum_{m\in\mathbf{M}} D_m(t)\mathbb{E}[A_m(t) - d_m(t)|\pi] \\ + \sum_{i\in\mathbf{N}} Q_i(t)\mathbb{E}[\sum_{m\in\mathbf{M}_i} d_m(t) - f_i(t)/\rho + \sum_{j\in\mathbf{N}}(b_{ji}(t) - b_{ij}(t))|\pi] \\ + \sum_{i\in\mathbf{N}} R_i(t)\mathbb{E}[\xi f_i(t)/\rho + \sum_{j\in\mathbf{N}}(r_{ji}(t) - r_{ij}(t))|\pi].\end{aligned} \tag{5.29}$$

According to [31], for any $\sigma > 0$, there exists at least one randomized stationary control policy π^* (i.e., the policy is independent of the backlogs), such that

$$\begin{aligned}
&\mathbb{E}[\varphi(t)|\pi^*] \le \varphi^{\text{opt}} + \sigma;\\
&\mathbb{E}[A_m(t) - d_m(t)|\pi^*] \le \sigma;\\
&\mathbb{E}[\textstyle\sum_{m\in\mathbf{M}_i} d_m(t) - f_i(t)/\rho + \sum_{j\in\mathbf{N}}(b_{ji}(t) - b_{ij}(t))|\pi^*] \le \sigma;\\
&\mathbb{E}[\xi f_i(t)/\rho + \textstyle\sum_{j\in\mathbf{N}}(r_{ji}(t) - r_{ij}(t))|\pi^*] \le \sigma;
\end{aligned} \tag{5.30}$$

Plugging (5.30) into (5.29) and then taking $\sigma \to 0$, we have

$$\Delta_V(t) \le U + V\varphi^{\text{opt}}. \tag{5.31}$$

Taking expectations on both sides of (5.31), we can obtain

$$\mathbb{E}[L(t+1) - L(t)] - V\mathbb{E}[\varphi(t)] \le U + V\varphi^{\text{opt}}. \tag{5.32}$$

Summing up all the telescoping series over $\{0, 1, \cdots, T-1\}$, (5.32) leads to

$$\mathbb{E}[L(T)] - \mathbb{E}[L(0)] + V\sum_{t=0}^{T-1} \mathbb{E}[\varphi(t)] \le UT + VT\varphi^{\text{opt}}. \tag{5.33}$$

Given the fact that $\mathbb{E}[L(T)] \ge 0$ and $L(0) = 0$, we have

$$\overline{\varphi(t)} = \frac{1}{T}\lim_{T\to\infty}\sum_{t=0}^{T-1} \mathbb{E}[\varphi(t)] \le \varphi^{\text{opt}} + \frac{U}{V}. \tag{5.34}$$

This concludes the proof. □

From Theorem 5, the proposed approach is asymptotically optimal, i.e., the optimality gap between the proposed approach and the off-line optimum can asymptotically diminish as V increases. We can further show the instantaneous backlogs of all queues are upper bounded in the following theorem.

Theorem 6. *The backlogs of all the queues at the edge servers are strictly bounded* *at each time slot t, i.e., $Q_i(t) \leq Q_i^{\max}$ and $R_i(t) \leq R_i^{\max}$, $\forall i, t$, as given by*

$$Q_i^{\max} = D_{\max} - \frac{V\varepsilon_m}{c_m} + LTc^{\max} + \sum_{j\in\mathbf{N}_i} C_{ji}^{\max}; \tag{5.35a}$$

$$R_i^{\max} = [Q_i^{\max} - V\rho\epsilon_i]/\xi + \xi\widehat{F}_i/\rho + \sum_{j\in\mathbf{N}_i} C_{ji}^{\max}, \tag{5.35b}$$

where $\mathbf{N}_i$ collects the immediate neighbors of server i.

Proof. We first prove (5.35a) through mathematical induction. The upper bound holds at slot 0 due to the fact that $Q_i(0) = 0$. Suppose that the upper bound holds at slot t. According to (5.25), if $Q_i(t) \geq D_m(t) - V\varepsilon_m/c_m$, $d_m(t) = 0$ and we have $Q_i(t+1) \leq Q_i(t) + \sum_{j\in\mathbf{N}_i} C_{ji}^{\max}$. Otherwise, if $Q_i(t) < D_m(t) - V\varepsilon_m/c_m$, the maximum uploaded data at this slot is $LTc^{\max}$, and hence, $Q_i(t+1) \leq D_m(t) - V\varepsilon_m/c_m + LTc^{\max} + \sum_{j\in\mathbf{N}_i} C_{ji}^{\max}$, i.e., the upper bound also holds at slot $(t+1)$. By replacing $D_m(t)$ with its maximum $D_{\max}$, the proof of (5.35a) is concluded.

Likewise, we can prove (5.35b) through mathematical induction. (5.35b) holds for $t = 0$. We assume that it also holds at slot t. According to (5.20), if $R_i(t) \geq [Q_i^{max} - V\rho\epsilon_i]/\xi$, no data is processed and $R_i(t+1) \leq R_i(t) + \sum_{j\in\mathbf{N}_i} C_{ji}^{\max}$. Otherwise, if $R_i(t) < [Q_i^{max} - V\rho\epsilon_i]/\xi$, the result of processed tasks at slot t cannot exceed $\xi\widehat{F}_i/\rho$, leading to $R_i(t+1) \leq [Q_i^{max} - V\rho\epsilon_i]/\xi + \xi\widehat{F}_i/\rho + \sum_{j\in\mathbf{N}_i} C_{ji}^{\max}$. As a result, (5.35b) also holds at slot $(t+1)$. This concludes the proof. $\square$

From Theorem 6, the upper bounds of the instantaneous queue backlogs at edge server i, i.e., $Q_i^{\max}$ and $D_i^{\max}$ in (5.35), are restrained by the finite IoT buffers $D_{\max}$. Both the upper bounds decrease linearly with V. This is because, from (5.20), (5.22) and (5.27), the queue differences required for data uploading, processing, dispatch, and result delivery increase linearly with V. As a consequence, $Q_i^{\max}$ and $R_i^{\max}$ would become negative with the increase of V in (5.35), preventing the asymptotic optimality in (5.28), or even stopping the proposed approach in Section 5.2 from functioning.

5.3.2 Virtual Placeholder Design

A virtual placeholder can be rigorously optimized for every queue to establish sufficient queue differences along the shortest paths from the IoT devices to the data center. The largest placeholders are placed at the IoT devices, hence making up for the limited finite buffers of the IoT devices. The placeholders need to satisfy the following property [107].

Property 1. There exists a non-negative queue backlog vector $\mathcal{Q}_0 = \{Q_{i,0}, R_{i,0}, D_{m,0}, \forall i, m \in \mathbf{M}_i(t)\}$, as such that: if $\mathcal{Q}(0) \succeq \mathcal{Q}_0$, we have $\mathcal{Q}(t) \succeq \mathcal{Q}_0$ for all $t \geq 0$.

According to Property 1, the placeholders give a threshold (or instantaneous lower bound) $\mathcal{Q}_0$, and once the backlogs of the queues exceed $\mathcal{Q}_0$, the backlogs can by no means fall below $\mathcal{Q}_0$. In other words, the virtual placeholders $\mathcal{Q}_0$ can be neither dispatched nor processed while running Algorithms 4 and 5. They are only used to enlarge the differences between queues to accelerate data processing and result delivery. Decoupled from the actual backlogs, $\widetilde{Q}_i(t)$, $\widetilde{R}_i(t)$ and $\widetilde{D}_m(t)$ are the effective backlogs after placing the placeholders into the queues, as given by

$$\widetilde{Q}_i(t) = Q_i(t) + Q_{i,0}, \ \forall i; \tag{5.36a}$$

$$\widetilde{R}_i(t) = R_i(t) + R_{i,0}, \ \forall i; \tag{5.36b}$$

$$\widetilde{D}_m(t) = D_m(t) + D_{m,0}, \ \forall m. \tag{5.36c}$$

Given the property of the virtual placeholders, Algorithms 4 and 5 can remain unchanged, except that $Q_i(t)$, $R_i(t)$ and $D_m(t)$ are replaced by the effective backlogs $\widetilde{Q}_i(t)$, $\widetilde{R}_i(t)$ and $\widetilde{D}_m(t)$ in (5.36). The asymptotic optimality proved in Theorem 5 is intact, since the placeholder $\mathcal{Q}_0$ at slot 0 does not affect the average utility in the long term.

The virtual placeholders can be formulated in the following Corollary.

Corollary 2. *$\mathcal{Q}_0$ satisfying Property 1 can be given by*

$$R_{i,0} = \begin{cases} 0, \text{if } i = 0; \\ \max\{\min_j\{R_{j,0} + \Delta_{ij}\}, 0\}, \text{ otherwise}; \end{cases} \tag{5.37a}$$

$$Q_{i,0} = \max\{\min_j\{\xi R_{i,0} + \Delta_i, Q_{j,0} + \Delta_{ij}\}, 0\}; \tag{5.37b}$$

$$D_{m,0} = \max\{Q_{i,0} + \Delta^i_m, 0\}, \ m \in \mathbf{M}_i; \tag{5.37c}$$

where $\Delta_{ij} = V\epsilon^{\min}_{ij} - C^{\max}_{ij}$; $\Delta_i = V\rho\epsilon^{\min}_i - \widehat{F}_i/\rho$; $\Delta^i_m = V\varepsilon_m/c^{\max} - c^{\max}T$; and Δ_{ij}, Δ_i and Δ^i_m are restricted by the gap between the minimum cost and maximum capacity of transmission or processing for the servers and IoT devices, and can be adjusted by V.

Proof. Note that $R_{i,0} = 0$ satisfies Property 1, since $R_i(t) \geq 0$ due to the queue dynamics in (5.7). Then, we only need to show that $R_{i,0} = \min_j\{R_{j,0} + \Delta_{ij}\}$ also satisfies Property 1 to prove (5.37a). By exploiting (5.22b), we can prove this through mathematical induction. Suppose that $R_i(t) \geq \min_j\{R_{j,0} + \Delta_{ij}\}$ holds at slot t. According to (5.22b), if $R_i(t) \leq \min_j\{R_{j,0} + V\epsilon^{\min}_{ij}\}$, $\gamma_{ij}(t) \leq 0$ for all neighboring servers, i.e., $R_i(t+1) \geq R_i(t)$, since no result can be dispatched from server i. Otherwise, if $R_i(t) > \min_j\{R_{j,0} + V\epsilon^{\min}_{ij}\}$, the size of results dispatched from server i cannot exceed the maximum link capacity $C^{\max}_{ij}$, and hence, $R_i(t+1) \geq \min_j\{R_{j,0} + \Delta_{ij}\}$. This concludes the proof of (5.37a).

We proceed to prove that $Q_{i,0} = \min_j\{\xi R_{i,0} + \Delta_i, Q_{j,0} + \Delta_{ij}\}$ satisfies Property 1. Similar to the proof of (5.37a), by exploiting (5.22a), the second term $Q_{i,0} \leq \min_j\{Q_{j,0} + \Delta_{ij}\}$ can be proved through mathematical induction. For the first term, suppose that $Q_i(t) \geq \xi R_{i,0}\Delta_i$ holds at slot t. According to (5.20), if $Q_i(t) \leq \xi R_{i,0} + V\rho\epsilon^{\min}_i$, no data will be processed and $Q_i(t+1) \geq Q_{i,0}(t)$. Otherwise, if $Q_i(t) > \xi R_{i,0} + V\rho\epsilon^{\min}_i$, the processed data cannot exceed $\widehat{F}_i/\rho$, and $Q_i(t+1) \geq \xi R_{i,0} + \Delta_i$. This concludes the proof of (5.37b).

Likewise, (5.37c) can be proved by exploiting (5.27) through mathematical induction. Assume that $D_m(t) \geq Q_i(t) + \Delta^i_m$ holds at slot t. According to (5.27), if $D_m(t) \leq Q_i(t) + V\varepsilon_m/c^{\max}$, $\alpha^i_m(t) \leq 0$ and device m will not be scheduled, i.e.,

$D_m(t+1) \geq D_m(t)$. Otherwise, if $D_m(t) > Q_i(t) + \varepsilon_m/c^{\max}$, the uploaded data cannot exceed $c^{\max}T$, and hence, $D_m(t+1) \geq Q_i(t) + \Delta_m^i$. This concludes the proof of (5.37c). $\square$

As shown in Corollary 2, the virtual placeholders increase linearly with V, due to the linearly increasing Δ_{ij}, Δ_i and Δ_m^i in (5.37). This can compensate for the linearly decreasing parts of the instantaneous queue upper bounds in (5.35), since the effective backlogs of the queues in (5.36) are additively enlarged by the virtual placeholders. Hence, the proposed distributed online optimization is enabled to operate under small IoT buffers and achieve asymptotic optimality under large V values.

5.3.3 Three-layer Shortest Path Problem of Placeholders

The placeholder $D_{m,0}$ of device $m \in \mathbf{M}_i$ depends on the data queue placeholder $Q_{i,0}$ of its associated edge server i in (5.37c), while $Q_{i,0}$ of edge server i depends on the result queue placeholder $R_{i,0}$ in (5.37b). By exploiting the optimal substructure of (5.37), the design of the placeholders can be reformulated as a three-layer shortest path problem. The detail of the interpretation is discussed in the following.

On the bottom layer, $R_{i,0}$ in (5.37a) shares the same optimal substructure as shortest path problems [108]. $R_{i,0}$ requires explicit knowledge on $R_{j,0}$ of its neighboring server j, except that $R_{0,0} = 0$ is known in prior for the sink at the data center. As a result, $R_{i,0}$ in (5.37a) can be interpreted as the total distance from server i to the data center, measured by the total of the weights Δ_{ij} on all hops of the shortest path.

We proceed to replace $Q_{i,0}$ and $D_{m,0}$ with $\widehat{Q}_{i,0} = Q_{i,0}/\xi$ and $\widehat{D}_{m,0} = D_{m,0}/\xi$ to transform (5.37b) and (5.37c) as shortest path problems from the nodes on the top and middle layers to the sink $R_{0,0}$ on the bottom layer, respectively, as given by

$$\widehat{Q}_{i,0} = \max\{\min_j\{R_{i,0} + \Delta_i/\xi, \widehat{Q}_{j,0} + \Delta_{ij}/\xi\}, 0\}; \tag{5.38a}$$

$$\widehat{D}_{m,0} = \max\{\widehat{Q}_{i,0} + \Delta_m^i/\xi, 0\},\ m \in \mathbf{M}_i, \tag{5.38b}$$

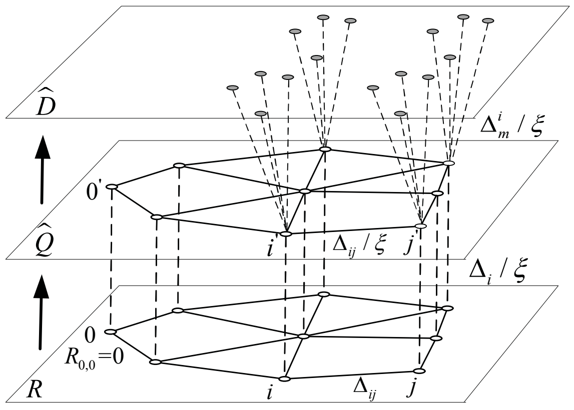


Figure 5.2 : An illustrative example of the three-layer placeholder graph formation.

where, between the layers, $\widehat{Q}_{i,0}$ corresponds to $R_{i,0}$ with weight Δ_i/ξ for data processing; and $\widehat{D}_{m,0}$ is adjacent to $\widehat{Q}_{i,0}$ with weight Δ^i_m/ξ for data uploading.

The new three-layer shortest path problem is illustrated in Fig. 5.2, where the vertexes on the three layers correspond to $R_{i,0}$, $\widehat{Q}_{i,0}$ and $\widehat{D}_{m,0}$, respectively. The solid lines represent data dispatch and result delivery in the edge cloud with weights Δ_{ij} and Δ_{ij}/ξ, and the dotted lines denote data processing at server i with the weight Δ_i/ξ and data uploading from device m with weight Δ^i_m/ξ.

The three-layer shortest path problem can be solved by finding the distance from the data center (i.e., the sink) with $R_{0,0} = 0$ to the node of virtual placeholders on the corresponding layer. The distances from the sink $R_{0,0} = 0$ in the three-layer graph, i.e., $R_{i,0}$, $\widehat{Q}_{i,0}$ and $\widehat{D}_{m,0}$, can be obtained through the distributed single-source shortest path algorithms, such as the celebrated Bellman-Ford algorithm, where an approximation to the correct distance is gradually improved by more accurate values until the eventual convergence to the optimum [109]. After the three-layer shortest path problems are solved, the placeholders $Q_{i,0}$ and $D_{m,0}$ can be finally given by $\xi\widehat{Q}_{i,0}$ and $\xi\widehat{D}_{m,0}$. Algorithm 6 summarizes the optimization of the virtual placeholders.

We note that the IoT buffers on the top layer have the longest distances to the sink on the bottom layer (i.e., the data queue of the data center with $R_{0,0} = 0$) in Fig. 5.2. In the new three-layer shortest path optimization, the sizes of virtual placeholders are based on the distances to the sink, and hence decline fastest along

Algorithm 6 Distributed Optimization of Virtual Placeholders

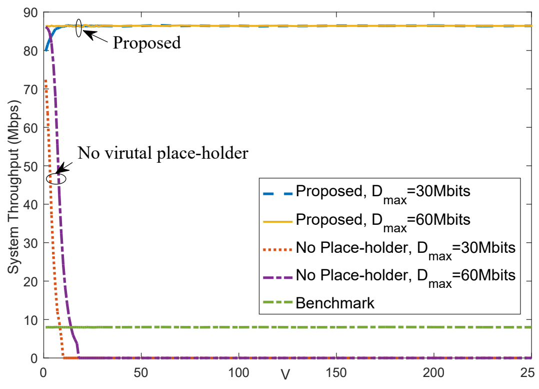
```
1: Construct the three-layer placeholder graph, as illustrated in Fig. 5.2
2: Calculate the distances $R_{i,0}$ and $\widehat{Q}_{i,0}$ and $\widehat{D}_{m,0}$ based on distributed Bellman-
   Ford algorithm [109]
3: Attain $R_{i,0} = R_{i,0}$, $Q_{i,0} = \xi\widehat{Q}_{i,0}$ and $D_{m,0} = \xi\widehat{D}_{m,0}$
4: Obtain the effective backlogs based on (5.36)
```

the shortest paths from the IoT devices through edge servers to the data center. This can drive the data to flow along the shortest paths to the sink, hence preventing unnecessary detours and reducing end-to-end delays.

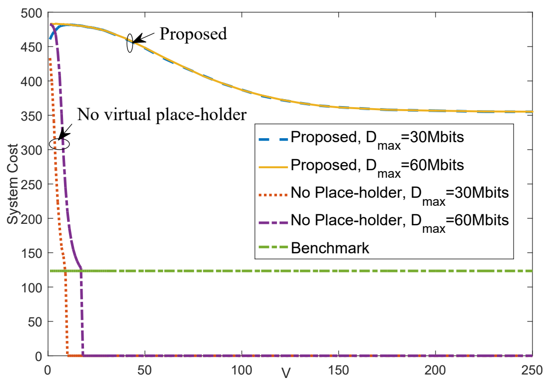
5.4 Simulation Results

Extensive simulations are carried out to evaluate the new distributed online optimization of fog computing in a network of $N = 50$ servers organized in a complete ternary tree. The root is the data center, each of the 33 leaf nodes (i.e., edge servers) is connected to $N = 100$ IoT devices, and the others act as intermediate edge servers (such as gateways, routers and switches) without IoT connections. For each IoT device, sensory data are queued in a buffer with the size of $D_{\max}$, and the data arrival follows a uniform distribution from 0.5 to 1.5Mbits [18]. There are L subchannels per edge server for IoT transmissions. L follows a uniform distribution between 10 and 20; unless otherwise specified. The capacity per subchannel is randomly and uniformly distributed from 125 to 250kbps, accounting for background traffic and time-varying channels [18]. The transmit power of an IoT device is 23dBm. The computing capacity of an edge server ranges from 3 to 15GHz with background computing task uniformly distributed from 0 to 40% [21]. The link capacity between servers randomly and uniformly ranges from 10 to 30Mbps. 10000 slots are simulated for each data point.

For comparison purpose, we also simulate: (a) a benchmark approach (denoted by “Benchmark”) which schedules IoT devices in a round-robin manner and all the data uploaded from the IoT devices are dispatched to the data center for processing, as typically done in traditional cloud computing; and (b) the direct application of



(a) System throughput



(b) System cost

Figure 5.3 : The steady-state system throughput and cost as the parameter V increases from 1 to 250. Note that the proposed direct application of Lyapunov optimization cannot operate for $V \geq 20$ in Fig. 5.3(a), and hence incurs no cost in Fig. 5.3(b).

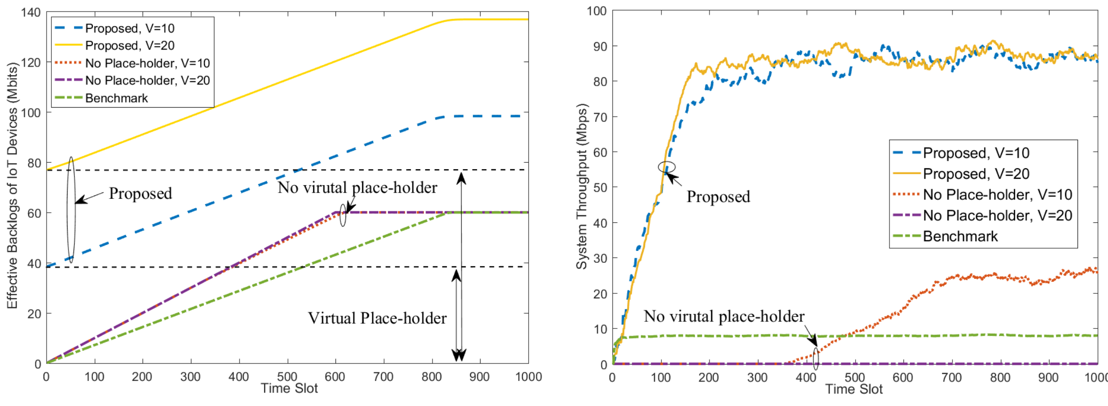
Lyapunov optimization, i.e., the distributed online approach developed in Section 5.2 (denoted by "No Placeholder"), where there is no virtual placeholder.

The benchmark approach provides little scalability with the increase of data, as it separately schedules IoT devices and routes data in the edge cloud with all the data processed at the data center. Compared to the proposed approach with virtual placeholders, the direct use of Lyapunov optimization fails to create sufficient queue differences along the shortest paths, which only operates under large IoT buffers and undergoes much longer delays. It is worth mentioning that the globally optimal centralized offline scheme is computationally prohibitive to simulate, due to the large number of variables over time and space. Nevertheless, Theorem 5 proves that the optimality gap between the proposed Lyapunov optimization approach and the centralized offline optimum would asymptotically diminish with the increase of V.

Fig. 5.3 plots the steady-state system throughput and cost achieved by the proposed approach with virtual placeholders, the proposed approach without virtual placeholders (i.e., the direct application of Lyapunov optimization), and the benchmark approach, as the parameter V increases from 1 to 250, where the IoT buffer sizes $D_{\max} = 30$ or 60Mbits. We see in Fig. 5.3(a) that when $D_{\max} = 60$Mbits, the proposed approach with virtual placeholders can always achieve the maximum system throughput of 86.4Mbps after draining the computing resources at the servers. In the case that $D_{\max} = 30$Mbits, the system throughput first increases with V and then stabilizes at the maximum under $D_{\max} = 60$Mbits when $V \geq 12$. This is because there exist negative parts (i.e., the link rates and computing capacities) in Δ_{ij}, Δ_i and Δ_m^i for the sizes of virtual placeholders, and these static parts would lead to insufficient virtual placeholders under small V values when $D_{\max} = 30$Mbits. In Fig. 5.3(b), the system cost of the proposed approach with virtual placeholders decreases with V and stabilizes when $V \geq 200$. This validates the asymptotic optimality of the approach, as dictated in Theorem 5, i.e., the optimality loss against the global optimum would diminish with the growth of V. Compared with the proposed approach, the benchmark approach always achieves 11% of system throughput but suffers 35% system cost.

It is worth pointing out in Fig. 5.3 that the direct use of Lyapunov optimization without virtual placeholders cannot operate properly due to the limited buffer sizes of IoT devices, and the system throughput quickly diminishes with the growth of V. We can see in both Figs. 5.3(a) and 5.3(b) that the direct use of Lyapunov optimization cannot process any data, when $V \geq 10$ in the case that $D_{\max} = 30$Mbits and when $V \geq 20$ in the case that $D_{\max} = 60$Mbits. This is because, when V is large, the buffer sizes of IoT devices are even smaller than the minimum data sizes required to enable the system, as will be shown in Fig. 5.4.

Fig. 5.4 shows the evolution of the effective backlogs of IoT devices and system throughput, as t increases from 0 to 1500, where $D_{\max} = 60$Mbits and $V = \{10, 20\}$. We can see in Fig. 5.4(a) that the backlogs of the direct use of Lyapunov optimization and the benchmark approach increase and then stabilize at the IoT buffer size over time. The effective backlogs of Lyapunov optimization with virtual placeholders exhibit the similar phenomenon but stabilize at much larger values than $D_{\max}$. This is because virtual placeholders can compensate for the limited buffers of the IoT devices and create the sufficient backlog differences to drive the distributed online Lyapunov optimization; see (5.36). As a result, in Fig. 5.4(b), the system throughput of the proposed approach with virtual placeholders can be significantly improved.



(a) Effective backlogs (b) System throughput

Figure 5.4 : The change of the effective queue backlogs of IoT devices and the system throughput over time. When $V = 20$, the virtual placeholders exceed the buffers $D_{\max} = 60$Mbits, and by no means can the direct use of Lyapunov optimization establish enough queue backlogs to function.

The corresponding time to stabilize the system can be substantially reduced.

We also see in Fig. 5.4(a) that the virtual placeholders (i.e., the sufficient backlogs to drive the distributed online optimization) increase with V, due to the typical $[\mathcal{O}(V), \mathcal{O}(1/V)]$-tradeoff between the queue lengths and optimality loss of Lyapunov optimization. In the case that $V = 20$, the virtual placeholders exceed the maximum IoT buffer $D_{\max} = 60$Mbits, and by no means can the direct use of Lyapunov optimization, as developed in Section 5.3, establish enough queue backlogs to function. This is also validated in Fig. 5.4(b) that the direct use of Lyapunov optimization only operates when the IoT backlogs exceed the corresponding virtual placeholders in the case that $V = 10$, and no data can be processed when $V = 20$.

Fig. 5.5 plots the system throughput of the three approaches, as the buffer size of IoT devices $D_{\max}$ increases from 100 to 1000Mbits, where $V = \{50, 100\}$. We can see that the proposed approach with virtual placeholders can always achieve the maximum system throughput of 86.4Mbps, as also validated in Fig. 5.3(a). With the increase of $D_{\max}$, the system throughput of the direct use of Lyapunov optimization gradually enhances and stabilizes at the maximum system throughput, in both the cases that $V = 50$ and 100. With the increase of V, increasingly large IoT buffer sizes are required to achieve the maximum system throughput in the direct use of

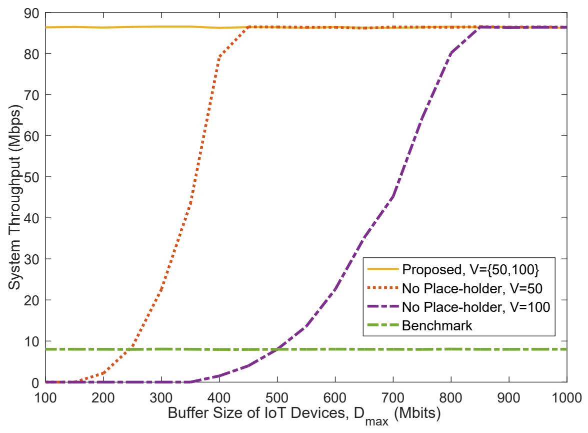
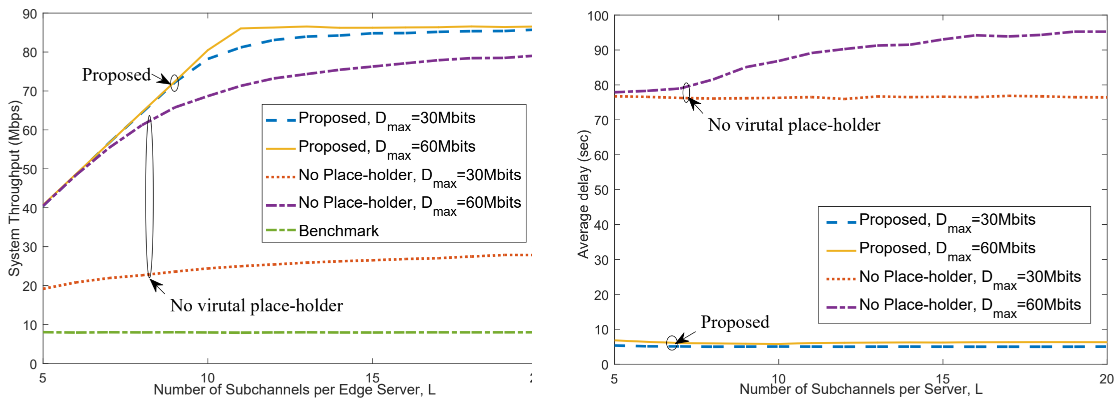


Figure 5.5 : The system throughput as the buffer sizes of IoT devices $D_{\max}$ increase. When $V = 100$, the direct use of Lyapunov optimization requires the buffers to be at least 850Mbits which is not practical in inexpensive IoT devices.



(a) System throughput (b) Average delay

Figure 5.6 : The steady-state system throughput and average delay, as L increases from 5 to 20, where $D_{\max} = \{30, 60\}$Mbits and $V = 5$. It is worth noting that the benchmark approach can lead to congestions around the data center and hence unbounded growth of delays, e.g., over 1000sec, and therefore is not plotted.

Lyapunov optimization. In particular, when $V = 100$, the algorithm requires the IoT buffer sizes to be at least 850Mbits which may not be practical in inexpensive IoT devices.

Fig. 5.6 evaluates the steady-state system throughput and average delay, as the number of subchannels per edge server L increases from 5 to 20, where $D_{\max} = \{30, 60\}$Mbits. It should be noted that, in order to evaluate the average delays of both the proposed approach with virtual placeholders and the direct use of Lyapunov optimization without virtual placeholders, V is set to 5 according to Fig. 5.3. Fig. 5.6(a) shows that the throughput of the the proposed approach with virtual placeholders can be 8 times higher than the 9Mbps throughput of the benchmark, and increases with the number of subchannels and stabilizes at the maximum throughput of 86.4Mbps, since an increasing amount of data can be transmitted from the IoT devices to the servers for remote processing. The placeholder design is also shown to help increase the throughput, especially in the case where the buffer storage $D_{\max} = 30$Mbits is small and the initial queue differences play a key role of driving data to flow and be processed, as shown in Fig. 5.3(a).

We can see in Fig. 5.6(b) that the use of virtual placeholders can substantially reduce the average delay of fog computing by 90%, as compared to the direct use of Lyapunov optimization. This is achieved by driving data along the shortest path, and the delay is relatively stable for different numbers of subchannels L. In contrast, the delay of the direct use of Lyapunov optimization grows when $D_{\max} =$ 60Mbits, due to the increase of system throughput in Fig. 5.6(a), which approaches the processing capacity and leads to unnecessary routing among busy servers and subsequently the growth of the queuing delay. The throughput of the benchmark approach is only 8Mbps independent of the value of L, limited by the capacity of the data center. Under such throughput, the system is not stable and experiences excessive delays over time. This is the consequence of separate operations of wireless transmission and processing in the benchmark approach.

Chapter 6

Profitable Cooperative Region for Distributed Online Edge Caching

Cooperative caching can unify network storage to improve efficiency, but the effective placement and search of files are challenging especially in distributed edge clouds with neither a-priori knowledge on file requests nor instantaneous global view.

This chapter proposes a new profitable cooperative region for every IoT data/file request admitted at an edge server, within which the file can be retrieved profitably as compared to a direct retrieval from the network backbone. A novel approach is proposed for distributed and automated formation of the regions in the absence of the a-priori knowledge on request arrivals and the instantaneous global view of the network. The region narrows down the search for cached files in response to admitted requests. The caching density of the file can also be significantly reduced, e.g., to a cached copy per region. Our approach is based on an asymptotically optimal, distributed framework of cooperative caching, where each server can optimize its decisions on the admission, dispatching and grant of file requests, and the file delivery for granted requests, based on its knowledge on its own and its one-hop neighboring servers. The proposed distributed framework of cooperative caching is based on stochastic gradient descent (SGD) [94], with the following new contributions.

- A new problem is formulated to optimize the admission, dispatching and grant of file requests, and delivery of granted requests, to maximize the time-average profit (or cost-saving) of cooperative caching (as compared to direct retrievals from the network backbone). The problem and its solution provide the stepping stone for the development of the new profitable cooperative caching region in edge clouds.

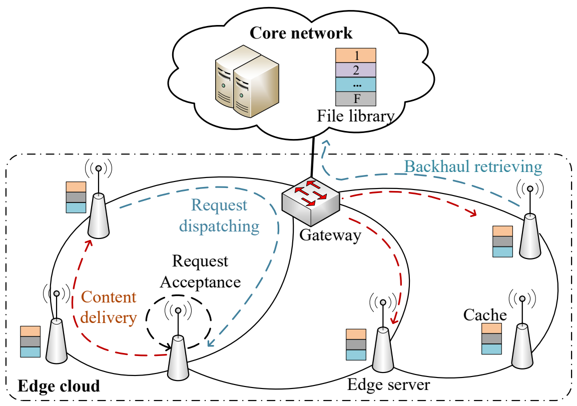


Figure 6.1 : The network of cooperative caching in the edge cloud.

- The new problem is decoupled over time at an asymptotically diminishing loss of optimality by exploiting SGD, and then judiciously optimized at each individual time slot to secure the asymptotic optimality inherited from SGD. The motivation of using SGD for multi-hop cooperative edge caching is novel.

- We design the profitable cooperative caching region for the requests admitted at every server, within which the files can be retrieved profitably. Specifically, the instantaneous upper and lower bounds for the backlog of files that the other servers can deliver profitably to the server of interest, are interpreted as shortest path problems and achieved in a fully distributed manner. The servers with positive gaps between the bounds can form the region.

- Preserving the asymptotic optimality, the proposed profitable cooperative caching region can operate in conjunction with simple caching policies of individual servers, such as least recently/frequently used (LRU/LFU) [110], to automate content placement with reduced density and improved efficiency. Such automated content placement has yet to be addressed in the literature [32–43].

Extensive simulations show substantially improved profit (or cost-effectiveness) of the proposed approach over existing solutions.

6.1 System Model

As illustrated in Fig. 6.1, the network of interest consists of N edge servers equipped with caching memories, where $\mathbf{N} = \{1, \cdots, N\}$ denotes the set of servers. Let $\mathbf{F} = \{1, \cdots, F\}$ denote the set of all files requested across the edge cloud. The network operates on a slotted basis. Each edge server receives requests from mobile users. The requests for the data/files arriving at an edge server can be dispatched to the network backbone or other edge servers that cache the files, for retrieving the files to the server admitting the requests*.

The request arriving at edge server i for file f at slot t can be denoted by its size $A_{i,f}(t)$ in terms of bits of data. Without the a-priori knowledge on traffic arrivals, $A_{i,f}(t)$ is a stochastic process over time with the maximum $A_{i,f}^{\max}$. For generality, we assume that edge server i can admit part of the request, denoted by $a_{i,f}(t)$, to the edge cloud based on the network availability. The rest of the file $\big(A_{i,f}(t) - a_{i,f}(t)\big)$ can be retrieved from the network backbone. $a_{i,f}(t)$ satisfies

$$0 \leq a_{i,f}(t) \leq A_{i,f}(t), \; \forall i, f, t. \tag{6.1}$$

The capacity of link (i, j) between servers i and j can change between time slots due to time-varying background traffic. The capacity of link (i, j) during slot t, denoted by $C_{ij}(t) \in (0, C_{ij}^{\max}]$, accounts for bi-directional transmissions over the link, and is assumed to remain unchanged within the slot. Let $\zeta_{ij}(t)$ denote the cost (e.g., in terms of energy consumption) of delivering a bit over link (i, j) at time slot t, and $\alpha_{i,f}$ denote the cost of server i for retrieving a bit of file f from the network backbone.

As illustrated in Fig. 6.2, each edge server needs to maintain NF request queues to track the requests of F files admitted at all N edge servers (including the edge

*Here, we assume that the user requesting a file remains within the coverage of the same edge server until the file is retrieved. This assumption is reasonable for users with low to moderate mobility, as extensively assumed in the literature [32–43, 45–49]. Nevertheless, the proposed approach can be extended to schedule content delivery to a different server from the server admitting the request, as will be discussed in Section6.3.3.

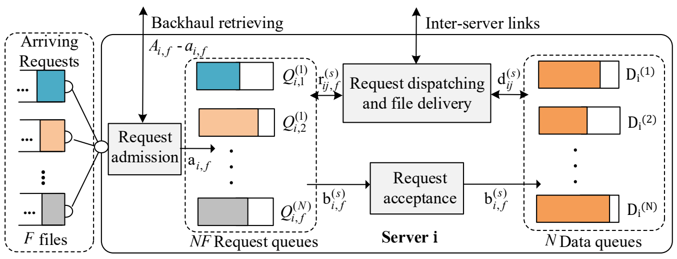


Figure 6.2 : An illustration of the queues and operations at an edge server.

server itself); and N data queues for the files retrieved and destined for different edge servers. As per slot t, $R_{i,f}^s(t)$ denotes the queue backlog of edge server i for the request of file f admitting at edge server s; and $D_i^s(t)$ denotes the queue backlog of edge server i for the file destined for edge server s. For illustration convenience, the files to be delivered to the same edge server are not differentiated in the data queues. The use of $N(F+1)$ queues per server is typically required to retrieve up to F files from N edge servers and deliver the files to where admitted. In this chapter, we relax the requirement of $N(F+1)$ queues per server by establishing the profitable regions for every file and edge server.

Let $r_{ij,f}^s(t)$ denote the size of the request for file f admitted at edge server s and dispatched to edge server j through edge server i, and $d_{ij}^s(t)$ denote the size of the file returned from edge server i to s through edge server j. The dispatch of requests only incurs negligible signaling (typical a few bytes to specify the edge server admitting the request and the index and size of the requested file) over the links, as compared to the delivery of files. As a result, the variables of request dispatch and file delivery satisfy:

$$\sum\nolimits_{s\in\mathbf{N}}[d_{ij}^s(t)+d_{ji}^s(t)]\leq C_{ij}(t),\ \forall(i,j),t; \tag{6.2a}$$

$$\sum\nolimits_{s\in\mathbf{N},f\in\mathbf{F}}[r_{ij,f}^s(t)+r_{ji,f}^s(t)]\leq C_{ij}(t),\ \forall(i,j),t; \tag{6.2b}$$

$$r_{ij,f}^s(t)\geq 0,\ d_{ij}^s(t)\geq 0,\ \forall i,j,f,s,t, \tag{6.2c}$$

where (6.2b) is set to avoid dispatching too many files through link (i,j), and (6.2a) and (6.2c) are self-explanatory. The dispatch of requests between edge servers in

(6.2b) only involves the transmission of a short packet, which is negligible compared to the delivery of contents/files of up to Gigabytes in (6.2a). For example, a 4-byte unsigned integer (e.g., the uint32 data type in C++ that can represent 2^{32} integers within [0,4294967295]) is sufficient to represent the queue backlogs of up to 4.3GB. As a result, constraint (6.2a) captures the capacity (or data rate) of link (i, j), and constraint (6.2b) is set up to avoid dispatching too many requests and stop the requested files from being returned through link (i, j). Given the fact that the a-priori knowledge of the time-varying link capacities is hard to predict, we use the link capacity at the current time slot to approximate the file request constraint.

Let $b^s_{i,f}(t)$ denote the size of file f requested earlier at server s and granted by server i at slot t, and $F_i(t) \leq F^{\max}$ denote the maximum size of requests that server i can grant per slot t in the presence of varying background services, such as firewall and anti-virus software. We have

$$\sum\nolimits_{f\in\mathbf{F},s\in\mathbf{N}} b^s_{i,f}(t) \leq F_i(t),\ \forall i, t; \tag{6.3a}$$

$$0 \leq b^s_{i,f}(t) \leq c_{i,f}F_i(t),\ \forall i, f, s, t, \tag{6.3b}$$

where (6.3b) ensures that an edge server can only support the requests for the files cached locally. Let $c_{i,f} \in \{0, 1\}$ denote the placement of file f at edge server i. If server i caches file f, $c_{i,f} = 1$; otherwise, $c_{i,f} = 0$.

Note that the variables on the admission, dispatching and grant of a request, i.e., $a_{i,f}(t)$, $r^s_{ij,f}(t)$ and $b^s_{i,f}(t)$, and the file delivery of the granted request $d^s_{ij}(t)$, are continuous variables. This is under the assumption that the files are divisible and a file can be delivered in part from different edge servers caching the file, as widely assumed in the state of the art [32–43, 45–49]. However, the proposed approach can be extended to support indivisible files, as will be discussed in Section 6.3.3.

As a result, the backlog of the requests for file f at server i from server s, denoted by $R_{i,f}^s(t)$, can be updated by

$$R_{i,f}^s(t+1) = \max\{R_{i,f}^s(t) - b_{i,f}^s(t) - \sum\nolimits_{j\in\mathbf{N}} r_{ij,f}^s(t), 0\} + \sum\nolimits_{j\in\mathbf{N}} r_{ji,f}^s(t) + a_{i,f}^s(t), \tag{6.4}$$

where $a_{i,f}^s(t)$ is the size of the request for file f admitted to the queue of server i at slot t. $a_{i,f}^i(t) = a_{i,f}(t)$, and $a_{i,f}^s(t) = 0$ $\forall s \neq i$. The first term on the right-hand side (RHS) of (6.4) gives the size of requests at the end of slot t, after part of the requests have been granted or dispatched to other edge servers. The second and the third terms account for new requests dispatched from other edge servers to, or admitted at, server i.

At server i, the backlog of the files which are retrieved to be delivered to server s, i.e., $D_i^s(t)$, can be updated by

$$D_i^s(t+1) = \max\{D_i^s(t) - \sum\nolimits_{j\in\mathbf{N}} d_{ij}^s(t), 0\} + \sum\nolimits_{j\in\mathbf{N}} d_{ji}^s(t) + \sum\nolimits_{f\in\mathbf{F}} b_{i,f}^s(t),\ \forall s \neq i, \tag{6.5}$$

where $D_i^i(t) = 0$ since edge server i sends the files to its associated users and acts as the sink for the files.

6.2 Distributed Optimization of Cooperative Edge Caching

The prominent contribution of this chaper is to establish the profitable cooperative region for each file request admitted at an edge server, thereby narrowing down the search for cached files, reducing the caching density of every file, and accommodating more files. It is prudent to first optimize the distributed operations of the servers on the admission, dispatching and grant of file requests, and the file delivery for granted requests, given file placement. The legitimacy of the proposed profitable cooperative regions relies on the effectiveness and (asymptotic) optimality of the distributed operations of the servers.

6.2.1 Problem Statement

We first propose the fully distributed, asymptotically optimal framework for the operations of the servers. To achieve this, we start with the configuration of $N(F+1)$

queues per edge server, as discussed in Section 6.1. By taking cost as the generic measure of cooperative caching, at any time slot t, the instantaneous cost can be given by

$$\Phi(\mathbf{x}^t) = \sum\nolimits_{i,j\in\mathbf{N}} \phi_{ij}(t) + \sum\nolimits_{i\in\mathbf{N},f\in\mathbf{F}} \alpha_{i,f}(A_{i,f}(t) - a_{i,f}(t)),$$

where $\mathbf{x}^t = \{b^s_{i,f}(t), r^s_{ij,f}(t), d^s_{ij}(t), a_{i,f}(t), \forall i, j, f, s\}$ collects all variables at slot t; $\phi_{ij}(t) = \zeta_{ij}(t)\sum_{s\in\mathbf{N}} d^s_{ij}(t)$ is the cost of routing files over link (i, j) at slot t; and $\alpha_{i,f}(A_{i,f}(t) - a_{i,f}(t))$ is the cost of retrieving file f from backbone to server i at slot t.

We also ensure the stability of the edge cloud, with the following constraint

$$\overline{R^s_{i,f}(t)} < \infty,\ \overline{D^s_i(t)} < \infty,\ \forall i, f, s, \tag{6.6}$$

where $\overline{X(t)} = \lim_{T\to\infty} \frac{1}{T}\sum_{\tau=0}^{T-1} \mathbb{E}[X(\tau)]$ denotes the time-average of a process $X(t)$.

We aim to minimize the time-average cost of cooperative edge caching in the absence of the a-priori knowledge on the network dynamics (e.g., the request arrivals, link capacity, and resource availability), while preserving the system stability. The problem of interest can be formulated as

$$\begin{aligned} \Phi^* = \min_{\mathbb{X}} &\ \overline{\Phi(\mathbf{x}^t)} \\ \text{s.t.}&\ (6.1),\ (6.2),\ (6.3),\ (6.4),\ (6.5),\ (6.6),\ \forall t. \end{aligned} \tag{6.7}$$

where $\mathbb{X} = \{\mathbf{x}^t, \forall t\}$ collects all the variables on the admission and dispatching of requests, and the retrieval of files, across all time slots.

According to queuing theory [92], a queue is stable, if and only if the time-average input rate of the queue is no more than the time-average output rate of the queue. As a result, we reformulate problem (6.7) to suppress the time couplings by transforming (6.4), (6.5) and (6.6) to (6.8), as given by

$$\begin{aligned} &\overline{a^s_{i,f}(t) - b^s_{i,f}(t) + \sum\nolimits_{j\in\mathbf{N}}(r^s_{ji,f}(t) - r^s_{ij,f}(t))} \leq 0,\ \forall i, f, s; \\ &\overline{\sum\nolimits_{f\in\mathbf{F}} b^s_{i,f}(t) + \sum\nolimits_{j\in\mathbf{N}}(d^s_{ji}(t) - d^s_{ij}(t))} \leq 0,\ \forall i, s. \end{aligned} \tag{6.8}$$

As a result, (6.7) can be reformulated as

$$\begin{aligned} \widetilde{\Phi}^* = \min_{\mathbb{X}} \overline{\Phi(\mathbf{x}^t)} \\ \text{s.t. } (6.1), (6.2), (6.3), (6.8), \forall t. \end{aligned} \tag{6.9}$$

With the time-averages in the objective and the constraint (6.8), the optimal solution to (6.9) would require the a-priori knowledge on the random process across the network over the infinite time horizon. This would violate the causality.

6.2.2 Distributed Online Optimization of Edge Caching

The proposed optimization of (6.9) is based on SGD and consists of two stages. By exploiting SGD, we first decouple the problem of interest (6.7) over time into (6.10) to be optimized online slot by slot. Provided that (6.10) is optimally solved per slot, the optimality loss of the decoupling diminishes, as the stepsize ϵ grows. Then, we decouple problem (6.10) into subproblems (6.17a) to (6.17d) which contain the variables on file delivery $\mathbf{d}^t$, request admission $\mathbf{a}^t$ and dispatching $\mathbf{r}^t$, and request grant $\mathbf{b}^t$, respectively. This is because the variables are not coupled in the objective of (6.10), and neither are they in constraints (6.1) – (6.3). Subproblems (6.17a) to (6.17d) can be separately solved to provide the exact optimal solution to (6.10), and secure the asymptotic optimality inherited from SGD. Details are provided in the following.

A. Temporal Decoupling via SGD

By taking SGD, (6.9) can be reformulated by interpreting (6.8) as the Lagrange multipliers and iteratively updating the Lagrange multipliers with the stochastic gradient per slot t. As a result, (6.9) can be transformed to (6.10) which can be carried out at each time slot t in the absence of the a-priori knowledge, as given by

$$\begin{aligned} &\max_{\mathbf{x}^t} g(\mathbf{a}^t) + \mu(\mathbf{b}^t) + \eta(\mathbf{r}^t) + \gamma(\mathbf{d}^t) \\ &\text{s.t. } (6.1), (6.2), (6.3). \end{aligned} \tag{6.10}$$

where

$$g(\mathbf{a}^t) = \sum\nolimits_{i\in\mathbf{N},\mathbf{f}\in\mathbf{F}}[\alpha_{i,f} - \epsilon R^i_{i,f}(t)]a_{i,f}(t); \tag{6.11a}$$

$$\eta(\mathbf{r}^t) = \sum_{i,j,s\in\mathbf{N},\mathbf{f}\in\mathbf{F}} \epsilon R^s_{i,f}(t)(r^s_{ij,f}(t) - r^s_{ji,f}(t)); \tag{6.11b}$$

$$\gamma(\mathbf{d}^t) = \sum_{i,j,s\in\mathbf{N}} \epsilon D^s_i(t)(d^s_{ij}(t) - d^s_{ji}(t)) - \zeta_{ij}(t)d^s_{ij}(t); \tag{6.11c}$$

$$\mu(\mathbf{b}^t) = \sum_{i,s\in\mathbf{N},\mathbf{f}\in\mathbf{F}} \epsilon(R^s_{i,f}(t) - D^s_i(t))b^s_{i,f}(t); \tag{6.11d}$$

In (6.11), the Lagrange multipliers of SGD are suppressed. This is because the evolutions of the multipliers at every time slot are interpreted as queue backlogs: $\lambda^s_{i,f}(t) = \epsilon R^s_{i,f}(t)$ and $\lambda^s_i(t) = \epsilon D^s_i(t)$. Here, ϵ is the stepsize and accounts for the optimality loss of SGD.

Proof. Concatenate the random variables of the system into a vector $\omega^t = \{A_{i,f}(t), C_{ij}(t), F_i(t), \forall i, j, f\}$. ω^t is independent and identically distributed (i.i.d.), since the requests independently arrive at the edge servers and the link capacity is affected by background traffic. (The results of this chapter can be readily extended to non-i.i.d., e.g., Markovian, randomness; see [111].) As a result, we can replace the time-averages in (6.8) by their expectations over i.i.d. random parameters, as given by

$$\mathbb{E}[a^{(s)}_{i,f}(t) - b^{(s)}_{i,f}(t) + \sum\nolimits_{j\in\mathbf{N}}(r^{(s)}_{ji,f}(t) - r^{(s)}_{ij,f}(t))] \leq 0,\ \forall i, f, s; \tag{6.12a}$$

$$\mathbb{E}[\sum\nolimits_{f\in\mathbf{F}} b^{(s)}_{i,f}(t) + \sum\nolimits_{j\in\mathbf{N}}(d^{(s)}_{ji}(t) - d^{(s)}_{ij}(t))] \leq 0,\ \forall i, s. \tag{6.12b}$$

The time-average in the objective of (6.9) can be replaced by its expectation, and (6.9) can be rewritten as

$$\begin{aligned} \widetilde{\Phi}^* = \min_{\mathbb{X}} \mathbb{E}[\Phi(\mathbf{x}^t; \omega^t)] \\ \text{s.t. } (6.1), (6.2), (6.3), (6.12), \forall t. \end{aligned} \tag{6.13}$$

The Lagrangian of (6.13) can be given by [93]

$$\mathcal{L}(\mathbb{X}, \boldsymbol{\lambda}) = \mathbb{E}[\mathcal{L}^t(\mathbf{x}^t, \boldsymbol{\lambda})],$$

where $\mathcal{L}^t(\mathbf{x}^t, \boldsymbol{\lambda})$ is the instantaneous Lagrangian, as given by

$$\begin{aligned}\mathcal{L}^t(\mathbf{x}^t, \boldsymbol{\lambda}) = \Phi(\mathbf{x}^t) + &\sum_{i,s\in\mathbf{N}, f\in\mathbf{F}} \lambda^s_{i,f}\mathbb{E}[a^s_{i,f}(t) - b^s_{i,f}(t) + \sum_{j\in\mathbf{N}}(r^s_{ji,f}(t) - r^s_{ij,f}(t))] \\ + \sum_{i,s\in\mathbf{N}} \lambda^{s\prime}_i \mathbb{E}[\sum_{f\in\mathbf{F}} b^s_{i,f}(t) + \sum_{j\in\mathbf{N}}(d^s_{ji}(t) - d^s_{ij}(t))].\end{aligned} \tag{6.14}$$

Here, $\lambda^s_{i,f}$ and $\lambda^{s\prime}_i$ are the Lagrange multipliers associated with (6.12a) and (6.12b), respectively. $\boldsymbol{\lambda} = \{\lambda^s_{i,f}, \lambda^{s\prime}_i, \forall i, s\} \succeq 0$.

The primal values $\mathbf{x}^t$ can be given by

$$\begin{aligned}&\min_{\mathbb{X}} \mathcal{L}(\mathbb{X}, \boldsymbol{\lambda}) \\ &\text{s.t. } (6.1), (6.2), (6.3), \forall t.\end{aligned} \tag{6.15}$$

An exact gradient is intractable in the absence of the a-priori knowledge on the randomness over infinite time. At every time slot t, SGD is applied to update the Lagrange multipliers with the observed network dynamics ω^t, as given by

$$\lambda^s_{i,f}(t+1) = \max\{\lambda^s_{i,f}(t) + \epsilon[a^s_{i,f}(t) - b^s_{i,f}(t) + \sum\nolimits_{j\in\mathbf{N}}(r^s_{ji,f}(t) - r^s_{ij,f}(t))], 0\}, \tag{6.16a}$$

$$\lambda^s_i(t+1)' = \max\{\lambda^s_i(t)' + \epsilon[\sum\nolimits_{f\in\mathbf{F}} b^s_{i,f}(t) + \sum\nolimits_{j\in\mathbf{N}}(d^s_{ji}(t) - d^s_{ij}(t))], 0\}. \tag{6.16b}$$

With $Q^s_{i,f}(0) = 0$ and $D^s_i(0) = 0$, the updates of the multipliers per slot are linear to the updates of queue backlogs, as observed by comparing (6.4) and (6.5) with (6.16a) and (6.16b), respectively. As a result, we can reformulate (6.9) into (6.10) by replacing the Lagrange multipliers by the queue backlogs. □

B. Per-slot Optimal Operations to Problem (6.10)

Note that $\mathbf{a}^t = \{a_{i,f}(t), \forall i, f\}$, $\mathbf{r}^t = \{r^s_{ij,f}(t), \forall i, j, f, s\}$, $\mathbf{d}^t = \{d^s_{ij}(t), \forall i, j, s\}$, and $\mathbf{b}^t = \{b^s_{i,f}(t), \forall i, f, s\}$ are decoupled from each other; Further, $a_{i,f}(t)$ is decoupled from $a_{j,f}(t)$; $b^s_{i,f}(t)$ is decoupled from $b^s_{j,f}(t)$, $\forall i \neq j$; and $\eta(\mathbf{r}^t)$ and $\gamma(\mathbf{d}^t)$ can be decoupled between links. For each link (i, j), let $\tilde{\mathbf{r}}_{ij}(t) = \{r^s_{ij,f}(t), r^s_{ji,f}(t), \forall f, s\}$ and $\tilde{\mathbf{d}}_{ij}(t) = \{d^s_{ij}(t), d^s_{ji}(t), \forall s\}$. As a result, (6.10) can be solved by separately pursuing (6.17a), (6.17b), (6.17c) and (6.17d):

$$\max_{a_{i,f}(t)} g_{i,f}(t)a_{i,f}(t), \text{ s.t. } (6.1); \tag{6.17a}$$

$$\max_{\tilde{\mathbf{r}}_{ij}(t)} \sum_{s\in\mathbf{N}, f\in\mathbf{F}} \eta^s_{ij,f}(t)r^s_{ij,f}(t) + \eta^s_{ji,f}(t)r^s_{ji,f}(t), \text{ s.t. } (6.2b), (6.2c); \tag{6.17b}$$

$$\max_{\tilde{\mathbf{d}}_{ij}(t)} \sum_{s\in\mathbf{N}} \gamma^s_{ij}(t)d^s_{ij}(t) + \gamma^s_{ji}(t)d^s_{ji}(t), \text{ s.t. } (6.2a), (6.2c); \tag{6.17c}$$

$$\max_{\mathbf{b}_i(t)} \sum_{f\in\mathbf{F}} \mu^s_{i,f}(t)b^s_{i,f}(t), \text{ s.t. } (6.3); \tag{6.17d}$$

where $g_{i,f}(t) = \alpha_{i,f} - \epsilon R^i_{i,f}(t)$, $\eta^s_{ij,f}(t) = \epsilon(R^s_{i,f}(t) - R^s_{j,f}(t)$, $\gamma^s_{ij}(t) = \epsilon[D^s_i(t) - D^s_j(t)] - \zeta_{ij}(t)$, and $\mu^s_{i,f}(t) = c_{i,f}\epsilon(R^s_{i,f}(t) - D^s_i(t))$ can be obtained by decoupling (6.11a), (6.11b), (6.11c) and (6.11d) between links and edge servers. For example, $g_{i,f}(t)$ in (6.17a) can be obtained by decoupling between edge servers, since the objective $g(\mathbf{a}^t)$ in (6.11a) and constraint (6.1) can be respectively rewritten as the sums of the objectives and constraints of individual edge servers.

By closely assessing (6.17a), we see the optimal solution $a_{i,f}(t)$ depends on the sign of the parameter $g_{i,f}(t)$. If $g_{i,f}(t) > 0$ (or in other words, $R^i_{i,f}(t) < \alpha_{i,f}/\epsilon$), then $a_{i,f}(t)$ takes its maximum $A_{i,f}(t)$. Otherwise, $a_{i,f}(t) = 0$. Therefore, the optimal solution can be given by

$$a_{i,f}(t) = \begin{cases} A_{i,f}(t), \text{ if } R^i_{i,f}(t) < \alpha_{i,f}/\epsilon; \\ 0, \text{ otherwise;} \end{cases} \tag{6.18}$$

Problems (6.17b) and (6.17c) are linear programming of weighted-sum maximization [99], and their optimal solutions can be obtained by evaluating the weights

Algorithm 7 Distributed Online Operations of Cooperative Edge Caching

```
For each server $i$ per time slot $t$:
1: Measure $A_{i,f}(t)$, $F_i(t)$, $C_{ij}(t)$ and $\zeta_{ij}(t)$;
2: Observe the queue backlogs of its neighboring server $j$, i.e., $\{R^s_{j,f}(t), D^s_j(t), \forall f, s\}$;
3: Admit requests according to (6.18);
4: Schedule the request dispatching and file routing by (6.19);
5: Satisfy the requests based on (6.20);
6: Update $R^s_{i,f}(t)$ and $D^s_i(t)$ according to (6.4) and (6.5).
```

$\eta^s_{ij,f}(t)$, $\eta^s_{ji,f}(t)$, $\gamma^s_{ij}(t)$ and $\gamma^s_{ji}(t)$, at edge servers i and j. For (6.17b), if $\max_{f,s}\{\eta^s_{ij,f}(t)\} < 0$ or $\max_{f,s}\{\eta^s_{ij,f}(t)\} < \max_{f,s}\{\eta^s_{ji,f}(t)\}$, edge server i does not dispatch requests over link (i, j). Otherwise, edge server i dispatches requests to edge server j, with the request size as specified by

$$r^s_{ij,f}(t) = \begin{cases} C_{ij}(t), \text{ if } (f, s) = \arg\max_{f,s} \eta^s_{ij,f}(t); \\ 0, \text{ otherwise.} \end{cases} \tag{6.19a}$$

For (6.17c), if $\max_s\{\gamma^s_{ij}(t)\} < 0$ or $\max_s\{\gamma^s_{ij}(t)\} < \max_s\{\gamma^s_{ji}(t)\}$, edge server i does not deliver files through link (i, j). Otherwise, edge server i delivers files through edge server j with the following file size

$$d^s_{ij}(t) = \begin{cases} C_{ij}(t), \text{ if } s = \arg\max_s \gamma^s_{ij}(t); \\ 0, \text{ otherwise.} \end{cases} \tag{6.19b}$$

Lastly, (6.17d) is also weighted-sum maximization. If $\max_{f,s}\{\mu^s_{i,f}(t)\} < 0$, edge server i remains idle, i.e., does not satisfy requests. Otherwise, the size of optimally satisfied requests can be given by

$$b^s_{i,f}(t) = \begin{cases} F_i(t), \text{ if } (f, s) = \arg\max\limits_{f,s} \mu^s_{i,f}(t); \\ 0, \text{ otherwise.} \end{cases} \tag{6.20}$$

From (6.18), (6.19) and (6.20), the optimal decisions of the admission, dispatch and grant of requests, and the file delivery for granted requests, can be optimized at individual edge servers, based on the information of the server itself and its one-hop neighbors, as summarized in Algorithm 7. At each time slot t, each server i measures the request arrivals $A_{i,f}(t)$ and server availability $F_i(t)$ of its own, and the conditions of its links, i.e., $C_{ij}(t)$ and $\zeta_{ij}(t)$, in Step 1. The server requests the information on the queue backlogs of its neighboring servers, i.e., $\{R^s_{j,f}(t), D^s_j(t), \forall f, s\}$, in Step 2. Given the information acquired in Steps 1 and 2, edge server i grants the requests in the queue of the most cost-effectiveness according to (6.20) in Step 3. Then, in Step 4, the edge server and the queue with the most cost-effectiveness are activated to dispatch requests and retrieve files based on (6.19). Finally, each edge server updates its queue backlogs $R^s_{i,f}(t)$ and $D^s_i(t)$ according to (6.4) and (6.5) in Step 5.

C. Optimality and Complexity

By referring to [21, Theorem 1], it can be readily proved that Algorithm 1 is asymptotically optimal. This is due to the asymptotic optimality of decoupling (6.7) over time into (6.10) (as inherited from SGD). It is also because the solutions to (6.10) are optimal at every time slot with no further optimality loss. As a result, the time-average cost of edge caching converges to within $\mathcal{O}(\epsilon)$ of the offline optimum, which diminishes as $\epsilon \to 0$. The stepsize ϵ accounts for the convergence time of SGD. Given the stepsize ϵ, the convergence time of SGD linearly increases with $\mathcal{O}(1/\epsilon)$ [94]. The typical $[\mathcal{O}(1/\epsilon), \mathcal{O}(\epsilon)]$-tradeoff between convergence time and optimality loss in terms of the stepsize ϵ indicates that an $\mathcal{O}(1/\epsilon)$ convergence time allows for an $\mathcal{O}(\epsilon)$ close-to-optimal cost. Readers are referred to [21, Theorem 1] for details.

The time-complexity of both (6.19) and (6.20) is dominated by the selection of the most cost-effective queue from the total NF request queues (i.e., the $\arg\max(\cdot)$ operation), which is $\mathcal{O}(NF)$ per link of an edge server. The time-complexity of (6.18) is $\mathcal{O}(1)$ per file. There are a total of up to δ links and F files per server. $\delta < N$ is the maximum degree of the network. As a result, the time-complexity of Algorithm 7 is $\mathcal{O}(\delta NF + F) = \mathcal{O}(\delta NF)$ per edge server.

6.2.3 Extensions to High-Mobility Users and Indivisible Files

A. Users with High Mobility

The proposed approach can be readily applied to schedule file delivery to a different server from the server admitting the request. For a user which is currently in the coverage of edge server i and requests file f, instead of being placed in the request queue $R^i_{i,f}$, the request can be placed into the queue $R^j_{i,f}$. The file is to be delivered to edge server j where the user is expected to download the file from. The edge server j can be predicted by using popular mobility and trajectory prediction methods [112].

B. Indivisible Files

The proposed approach can also be readily extended to the scenario where files are not divisible and must be delivered in whole. In (6.18), the request admission decision is integer, and a file request is either admitted or rejected in whole. The decisions on the dispatching and grant of requests in (6.19a) and (6.20), and the file delivery for granted requests in (6.19b), can be rounded to the largest integer numbers of files which can be accommodated, i.e., $\lfloor \frac{C_{ij}(t)}{S_{\text{file}}} \rfloor$ and $\lfloor \frac{F_i(t)}{S_{\text{file}}} \rfloor$, respectively, where S_{file} is the size of a file, and $\lfloor \cdot \rfloor$ stands for the floor function.

The rounding does not violate the asymptotic optimality of the proposed approach. According to (6.17), the optimality loss of (6.10) resulting from the rounding is upper bounded by $\eta^s_{ij,f}(t)S_{\text{file}}$, $\gamma^s_{ij}(t)S_{\text{file}}$, and $\mu^s_{i,f}(t)S_{\text{file}}$, since the maximum rounding error is S_{file}. Given the upper bounds of the queue backlogs (as stated in Theorem 8), these upper bounds are further bounded by

$$\eta^s_{ij,f}(t)S_{\text{file}} \le \epsilon R^s_{f,\max} S_{\text{file}}; \tag{6.21a}$$

$$\gamma^s_{ij}(t)S_{\text{file}} \le \epsilon D^s_{i,\max} S_{\text{file}}; \tag{6.21b}$$

$$\mu^s_{i,f}(t)S_{\text{file}} \le \epsilon R^s_{f,\max} S_{\text{file}}; \tag{6.21c}$$

which are obtained by suppressing the negative terms in $\eta^s_{ij,f}(t)$, $\gamma^s_{ij}(t)$ and $\mu^s_{i,f}(t)$, and relaxing the rest by their maximums. From (6.21), we see that the optimality loss due to rounding asymptotically diminishes as $\epsilon \to 0$, hence preserving the asymptotic optimality of the proposed approach.

6.3 Distributed Formation of Profitable Cooperative Region

We proceed to establish the profitable cooperative region for every file request admitted at an edge server, given the asymptotically optimal distributed operations of cooperative caching developed in Section 6.2. The regions provide the necessary condition for file placement with guaranteed retrieval and profit. The request only needs to be dispatched within the region to retrieve the file profitably. The number of queues maintained per server and the time-complexity of Algorithm 7 can be dramatically reduced without compromising the asymptotic optimality.

6.3.1 Definition of Profitable Cooperative Caching Regions

The profitable cooperative caching region for each file request admitted at an edge server can be set up by deriving the lower and upper bounds of the queue backlogs at all the other servers. The bounds are based on the topology of the network and the costs of retrieving files among servers and from the network backbone. The profitable cooperative region for a request of file f admitted at edge server s is denoted by $\mathcal{R}_s^f$, beyond which the lower bounds of the backlogs, denoted by $\mathbf{Q}_0$, would exceed the profitable upper bounds, denoted by $\mathbf{Q}_{\max}$, at the servers. As a result, file retrieval beyond the region would be less cost-effective than direct retrieval from the network backbone.

To identify the profitable cooperative region, we start by deriving the lower and upper bounds of the backlogs, $\mathbf{Q}_0$ and $\mathbf{Q}_{\max}$. The lower bounds of the queue backlogs, also known as placeholders [107], can be defined as follows.

Definition 1. $\mathbf{Q}_0 = \{R^s_{i,f,0}, D^s_{i,0}, \forall i, f, s\}$ collects the lower bounds for all the queues of the system, such that, if $\mathbf{Q}(t_0) \succeq \mathbf{Q}_0$ at slot t_0, $\mathbf{Q}(t) \succeq \mathbf{Q}_0$ for $t > t_0$, where $\mathbf{Q}(t) = \{R^s_{i,f}(t), D^s_i(t), \forall i, f, s\}$ collects the backlogs for all the queues at slot t and $\succeq$ is taken entry-wise.

From Definition 1, we can see that, once the queue backlogs exceed $\mathbf{Q}_0$ at slot t_0, by no means can the backlogs be shorter than $\mathbf{Q}_0$ after that. That is to say, $\mathbf{Q}_0$ is neither dispatched nor delivered, without compromising the asymptotic optimality of Algorithm 7. By assessing the operations of Algorithm 7, we can formulate the lower bounds $\mathbf{Q}_0$ satisfying Definition 1 in the following theorem.

Theorem 7. *The lower bounds of the queue backlogs $\mathbf{Q}_0$ can be given by*

$$R^s_{i,f,0} = \max\big\{ \min_j\{D^s_{j,0} - F^{\max}\}, 0\big\}, \tag{6.22a}$$

$$D^s_{i,0} = \begin{cases} 0, \text{ if } i = s; \\ \max\big\{ \min_j\{D^s_{j,0} + w_{ij}\}, 0\big\}, \text{ otherwise;} \end{cases} \tag{6.22b}$$

where $w_{ij} = \zeta^{\min}_{ij}/\epsilon - C^{\max}_{ij}$, and $\zeta^{\min}_{ij}$ is the lower bound of $\zeta_{ij}(t)$.

Proof. For brevity, the proof is suppressed here. Please refer to the proof of Theorem 6 in Chapter 5 for the details. □

Here, (6.22) indicates that the lower bounds of the queues for requests and files to be delivered at a server depend on those of its neighboring servers, as well as the cost of retrieving files from the neighboring servers.

We proceed to derive the upper bounds of the queue backlogs in the following.

Theorem 8. *The instantaneous queue lengths are bounded at each time slot t, and there exist $R^s_{f,\max}$ and $D^s_{i,\max}$ satisfying $R^s_{i,f}(t) \leq R^s_{f,\max}$ and $D^s_i(t) \leq D^s_{i,\max}$, $\forall i, f, s$. The upper bounds $R^s_{f,\max}$ and $D^s_{i,\max}$ are given by*

$$R^s_{f,\max} = \frac{\alpha_{s,f}}{\epsilon} + A^{\max}_{s,f} + \theta_s; \tag{6.23a}$$

$$D^s_{i,\max} = \min\big\{ \max_j\{D^s_{j,\max} - w_{ij}\}, D^s_{\max}\big\}, \tag{6.23b}$$

where

$$D^s_{\max} = \max_f \left\{A^{\max}_{s,f} + \frac{\alpha_{s,f}}{\epsilon}\right\} + 2\theta_s + F^{\max}. \tag{6.24}$$

The parameter $\theta_s = \sum_{j\in\mathbf{N}_s} C^{\max}_{js}$ is the maximum of request and file arrivals at edge server s per slot via request dispatching and file delivery; and $\mathbf{N}_s$ collects the immediate neighbors of server s.

Proof. For brevity, the proof is suppressed here. Please refer to the proof of Corollary 2 in Chapter 5 for the details. □

Note that the upper and lower bounds in Theorems 7 and 8 are server and file specific, and they are derived for developing the new profitable cooperative regions. The profitable cooperative region of the requests for file f admitted at edge server s, i.e., $\mathcal{R}_s^f$, can be decided by evaluating the lower and upper bounds of the queue backlogs. In specific, edge server i can be precluded from $\mathcal{R}_s^f$, if the requests in $Q_{i,f}^s$ or the files in D_i^s can by no means be dispatched or retrieved, respectively. That is to say, the delivery from edge server i is not profitable, i.e., the cost-effectiveness measures, $\eta_{ij,f}^s(t)$, $\gamma_{ij}^s(t)$ and $\mu_{i,f}^s(t)$ in (6.19) and (6.20), are negative across all time slots. $\mathcal{R}_s^f$ can be given by

$$\mathcal{R}_s^f = \big\{ i | D_{i,0}^s < D_{i,\max}^s \text{ and } R_{i,0}^s < R_{f,\max}^s \big\}. \tag{6.25}$$

This is because, for server $i \notin \mathcal{R}_s^f$, $\gamma_{ij}^s(t)$ and $\mu_{i,f}^s(t)$ are always negative according to (6.25). The delivery of file f from server i to server s would be non-profitable, and by no means can the queues be scheduled for the delivery according to (6.19) and (6.20). This narrows down the search of the file. Server s only needs to search within the region $\mathcal{R}_s^f$ for file f.

The lower and upper bounds of the data queues, $D_{i,0}^s$ and $D_{i,\max}^s$ in (6.22b) and (6.23b), have recurrence expressions. $D_{i,0}^s$ and $D_{i,\max}^s$ require the knowledge on $D_{j,0}^s$ and $D_{j,\max}^s$ of its one-hop neighboring server j, except that $D_{s,0}^s = 0$ for (6.22b) and $D_{i,\max}^s \leq D_{\max}^s$ for (6.23b) are known in prior. The calculation of the bounds can be non-trivial due to the dependency between neighboring servers.

6.3.2 Distributed Optimization of Profitable Cooperative Regions

Both (6.22b) and (6.23b) exhibit the optimal substructure of the Bellman equations in dynamic programming [113]. (6.22b) can be regarded as an extended shortest path problem, where $D_{i,0}^s$ can be interpreted as the "distance" from server i to s. The "distance" satisfies the following properties:

- The distance from server s to itself is 0, i.e., $D^s_{s,0} = 0$;
- The distance is sub-additive in the measures of w_{ij}, i.e., $D^s_{i,0} \leq D^s_{j,0} + w_{ij}$, if $w_{ij} \geq 0$;
- The distance is non-negative, i.e., $D^s_{i,0} \geq 0$.

By exploiting the optimal substructure, we can derive the solutions for $D^s_{i,0}$ and $D^s_{i,\max}$ by decomposing them into subproblems.

Exploiting the optimal substructure, the calculation of $D^s_{i,0}$ and $D^s_{i,\max}$ is based on the results of their subproblems. The subproblems, $\widehat{D}^s_{i,0}(h)$ and $\widehat{D}^s_{i,\max}(h)$, denote the lower bounds and upper bounds (i.e., the "distance") up to h hops away from edge server s, respectively. $\widehat{D}^s_{i,0}(h)$ and $\widehat{D}^s_{i,\max}(h)$ can be constructed from their subproblems $\widehat{D}^s_{j,0}(h-1)$ and $\widehat{D}^s_{j,\max}(h-1)$ of its own and its one-hop neighboring servers. In specific, $\widehat{D}^s_{i,0}(h) = \widehat{D}^s_{i,0}(h-1)$ unless it can be further reduced by extending the paths from one of its immediate neighbors, $j \in \mathbf{N}_i$, i.e., $\widehat{D}^s_{j,0}(h-1) + w_{ij}$. As a result, $\widehat{D}^s_{i,0}(h)$ for (6.22b) can be reconstructed as

$$\widehat{D}^s_{i,0}(h) = \max\Big\{ \min_j \big\{ \widehat{D}^s_{i,0}(h-1), \widehat{D}^s_{j,0}(h-1) + w_{ij} \big\}, 0 \Big\}. \tag{6.26a}$$

Likewise, $\widehat{D}^s_{i,\max}(h)$ for (6.23b) can be given by

$$\widehat{D}^s_{i,\max}(h) = \min \left\{ \max_j \left\{ \begin{array}{l} \widehat{D}^s_{i,\max}(h-1), \\ \widehat{D}^s_{j,\max}(h-1) - w_{ij} \end{array} \right\}, D^s_{\max} \right\}; \tag{6.26b}$$

Exhibiting the optimal substructure of the Bellman equations, (6.26) can be solved recursively to compute $D^s_{i,0}$ and $D^s_{i,\max}$ in a distributed manner, as summarized in Algorithm 8. This is due to the fact that (6.26) only requires the knowledge of the server itself and its immediate neighbors. Algorithm 8 initializes $\widehat{D}^s_{i,\max}(0) = D^s_{\max}$ and $\widehat{D}^s_{i,0}(0) = -\infty$, except that $\widehat{D}^s_{s,0}(0) = 0$. $\widehat{D}^s_{i,0}(h)$ and $\widehat{D}^s_{i,\max}(h)$ are updated according to (6.26) based on $\widehat{D}^s_{j,0}(h-1)$ and $\widehat{D}^s_{j,\max}(h-1)$ from its one-hop neighboring servers $j \in \mathbf{N}_i$. $D^s_{i,0} = \widehat{D}^s_{i,0}(h)$ and $D^s_{i,\max} = \widehat{D}^s_{i,\max}(h)$ upon convergence.

Algorithm 8 Distributed Formation of Profitable Cooperative Caching Regions

```
1: Initialize $\widehat{D}^s_{i,\max}(0) = D^s_{\max}$ and $\widehat{D}^s_{i,0}(0) = -\infty$ except for $\widehat{D}^s_{s,0}(0) = 0$, and the
   iteration index $h = 1$;
For each server $i$:
2: repeat
3:     Observe $\widehat{D}^s_{i,0}(h-1)$ and $\widehat{D}^s_{i,\max}(h-1)$ of its own and immediate neighbors;
4:     Update $\widehat{D}^s_{i,0}(h)$ and $\widehat{D}^s_{i,\max}(h)$ by (6.26);
5:     Increase the iteration index $h = h + 1$;
6: until $\widehat{D}^s_{i,0}(h) = \widehat{D}^s_{i,0}(h-1)$ and $\widehat{D}^s_{i,\max}(h) = \widehat{D}^s_{i,\max}(h-1)$
7: $D^s_{i,0} = \widehat{D}^s_{i,0}(h)$ and $D^s_{i,\max} = \widehat{D}^s_{i,\max}(h)$;
8: Establish the profitable region according to (6.25);
```

Identifying the paths with the shortest "distances" through the optimal substructure, Algorithm 8 is an extension of the Bellman-Ford algorithm. The time-complexity of Algorithm 8 depends on the complexity for updating (6.26) per iteration and the number of iterations required for convergence. At each iteration, a server evaluates the results of the last iteration from its one-hop neighboring servers in $\mathcal{O}(N)$ time. There are up to $(N-1)$ iterations for guaranteed convergence, since the maximum length of a path is $(N-1)$ hops in a network of N servers. Algorithm 8 has the time-complexity of $\mathcal{O}(N^2)$.

6.3.3 Implementation of Profitable Cooperative Regions

The profitable cooperative regions can be incorporated in Algorithm 7. Within the region for file f requested at server s, the file only needs to be cached at one of the servers, i.e., server i ($i \in \mathcal{R}^f_s$), to guarantee profitable retrieval of the file to server s. As a result, instead of maintaining all $N(F+1)$ queues, server i only maintains a request queue and a data queue in regards of requests admitted at server s for file f, if server i is in the profitable cooperative region $\mathcal{R}^f_s$. The number of queues per server can be substantially reduced. Recall that the time-complexity of Algorithm 7 depends on the number of queues per server. The complexity of Algorithm 7 can be reduced from $\mathcal{O}(\delta N F)$ to $O(\delta|\mathcal{R}|F)$, where $|\mathcal{R}|$ denotes the average size of the profitable cooperative regions. As will be shown in Fig. 6.5(a), $|\mathcal{R}| \ll N$. The profitable cooperative regions improve the scalability of the proposed approach.

The regions can also operate in conjunction with existing techniques which place contents based on content popularity and its geographical distribution [36–43], to automate the placement of contents and prevent repeatedly caching the same contents within a region. Take the LRU policy for example [110]. Each edge server caches all the requested contents until it runs out of memory. After that, when receiving a new content request, instead of only searching its local cache, an edge server searches the cached contents within its profitable region. If the server fails to locate the requested content in its region, it fetches the content from the network backbone and replaces the least recently used content in its own cache. Nearby servers can benefit from this, if the same content is later requested. They can retrieve the content from the server (rather than directly retrieving from the network backbone and caching).

6.4 Simulation Analysis

In this section, we evaluate the proposed approach for cooperative edge caching in a network of $N = 100$ edge servers and a total of $F = 50$ files. Each server is associated with $M = 3$ servers, and can cache up to 10 files. The files requested across the edge cloud have the size of 15Mbits. The popularity of each file is assumed to follow a Zipf-distribution with the skew parameter $\beta = 0.8$ [114], and the servers cache the files according to their popularity. For per-Mbit size of data, the price of backhaul delivery $\alpha_{i,f}$ is 0.5, and the price of file delivery between edge servers ζ_{ij} is independently and uniformly distributed from 0.05 to 0.15 [34]. The reciprocal of stepsize $1/\epsilon$ is 60; unless otherwise specified.

The proposed approach is compared against the following three benchmark approaches.

- Local (non-cooperative) caching, where a file can be retrieved from either the local cache of the server admitting the request or the network backbone.
- "K-hop" file retrieval is extended from the state-of-the-art offline cooperative caching approach [37], where each edge server dispatches requests to, and

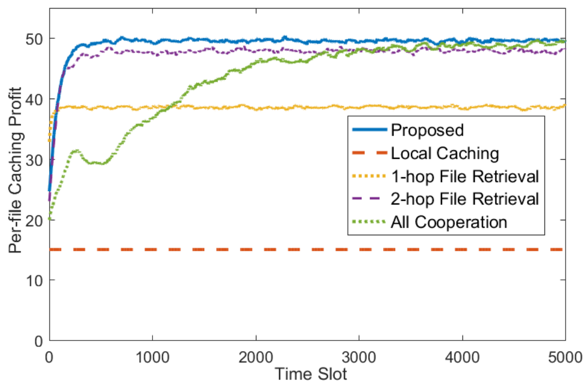
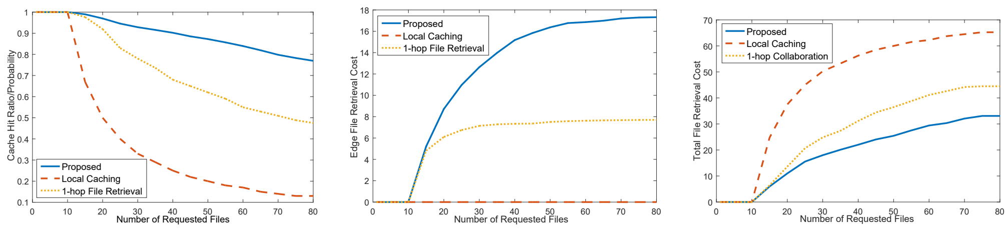


Figure 6.3 : The profit of different caching schemes as t evolves, where $F = 50$. The proposed approach achieves the highest profit and significantly reduces the time for stabilization.

retrieves files from, the servers up to K-hop away. The original offline approach [37] did not respond to random time-varying changes in the network and would degrade dramatically in the presence of random temporal variations. For the purpose of fair comparison, we enhance the offline state of the art with the proposed asymptotically optimal admission, dispatching and grant of requests, and file delivery for granted requests, whereas any file delivery is confined within up to K hops.

- "All-cooperation" file retrieval is extended from a centralized multi-hop cooperative edge caching approach [32], where there was no restriction on the maximum number of hops for file delivery, but the file delivery had to be accomplished within a single slot. We decentralize the centralized approach by allowing buffering at the edge servers, and hence the file delivery is relaxed to multiple slots.

Fig. 6.3 plots the profit of different caching schemes (i.e., the cost-saving compared to retrieving all the files from the network backbone) as t evolves from 0 to 5000, where $F = 50$. The caching profit first increases and then stabilizes with time, and the proposed approach achieves the highest profit of all the schemes. In specific, the profit of local caching is always 15. The stabilized profit of K-hop file retrieval increases with the growth of K. However, it is shown that the K-hop



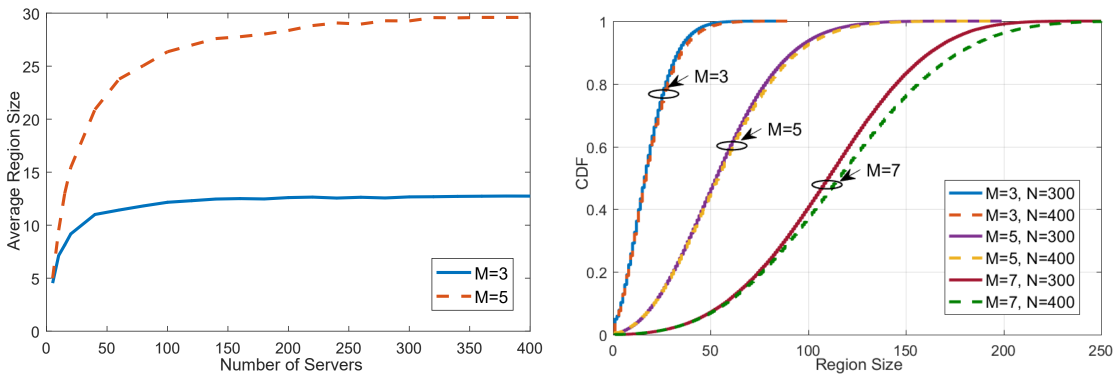
(a) Cache hit ratio/probability (b) Edge file retrieval cost (c) Total file retrieval cost

Figure 6.4 : The cache hit ratio/probability, the corresponding retrieval cost from the edge cloud, and the total file retrieval cost, as the number of requested files increases.

retrieval approach incurs increasing delays to stabilize the network, as K grows. By adaptively establishing the optimal profitable region for every server and file, the proposed approach significantly reduces the time for stabilization.

In Fig. 6.3, the proposed approach stabilizes after $t \geq 500$, and the convergence time of the all-cooperation scheme is about $t \geq 4000$. The proposed approach requires only 12.5% of convergence time of the all-cooperation approach, and achieves at least the same steady-state caching profit as the enhanced all-cooperation approach [32]. This is because, without restraining the maximum offloading hops K, the enhanced all-cooperation approach dispatches the requests unnecessarily further away from the edge server admitting the request, hence slowing down the convergence of the system. The simulation results in the rest of the figures, i.e., Figs. 4 to 8, are the steady-state performance. 5000 slots are run for each data point, and the results are collected and evaluated when $t \geq 2000$ and the system is in the steady state.

Fig. 6.4 evaluates the cache hit ratio (or in other words, the cache hit probability), the corresponding retrieval cost from the edge cloud, and the total file retrieval cost, achieved by the proposed approach, local caching and 1-hop file retrieval, as the number of requested files F increases. We can see in Fig. 6.4(a) that the proposed approach is superior in terms of retrieving the requested files from the edge cloud with up to 6 times the cache hit ratio of the local caching scheme. The cache hit ratio first remains 1 when $F \leq 10$, and then declines with the increasing number of



(a) Average region size

(b) The CDF of region sizes

Figure 6.5 : The average and CDF of the profitable region sizes as the number of servers N increases. The average region size quickly stabilizes with N and substantially increases with the connectivity per server M.

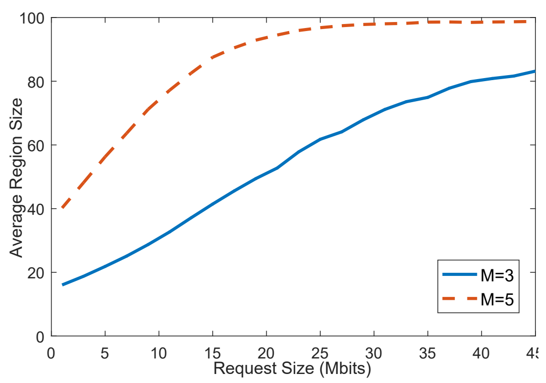
the requested files. This is because each server is set to cache at most 10 files due to the limited memory size of the server. When $F \leq 10$, an edge server can cache all the files locally (i.e., all the requests can be satisfied from the local memory), and the cache hit ratio is always 1.0 in Fig. 6.4(a) with zero file retrieval cost in Fig. 6.4(b). When $F > 10$, the probability that an edge server caches a specific file decreases with the increase of files. The file is increasingly likely to be retrieved from the backbone, resulting in the decline of cache hit ratio in Fig. 6.4(a).

We notice that the local caching scheme can always achieve zero file retrieval cost, as shown in Fig. 6.4(b). This is because the edge file retrieval cost is the cost for cooperatively retrieving files from the edge cloud (which does not include the cost of retrieving files from the network backbone), while the local caching scheme does not support cooperative caching. However, the total cost of the local caching scheme is still the highest, as shown in Fig. 6.4(c), since the scheme does not cache files cooperatively and more files need to be retrieved from the backbone than the other schemes.

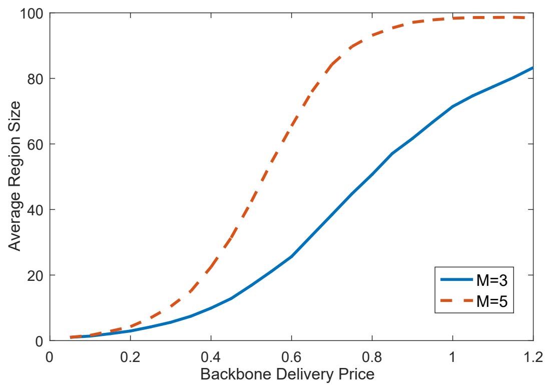
Fig. 6.5 plots the average and the cumulative distribution function (CDF) of the sizes of the profitable cooperative regions, as the number of servers N increases, when the connectivity per server is set to $M = 3$, 5, and 7. The profitable region is plotted from the proposed upper and lower bounds: only the server, which has a positive gap between the upper and lower bounds of its profitable queue backlog for the files another server is interested in, is part of the cooperative caching region for the latter server, as described in Section 6.3. We can see in Fig. 6.5(a) that the average region size stabilizes as N increases. The collaborated region size does not grow with the increasing scale of network. This is because the servers too far away could not deliver files more cost-effectively than the direct retrievals from the network backbone, and would not contribute to enlarging the profitable caching region.

In Fig. 6.5, we see that the number of servers each server is associated with, i.e., M, can have a strong impact on the region sizes. As shown in Fig. 6.5(a), the size of the regions is only 30 out of a total of 400 servers when $M = 5$, and can be further reduced to 12 when $M = 3$. As shown in Fig. 6.5(b), the CDFs of the region sizes under the same connectivity are overlapped, while the growth of connectivity can significantly enlarge the regions. The reason is that the increase of connectivity can help decrease the cost of file delivery between a pair of servers due to the improved delivery path diversity. In this sense, the density of file placement in an edge cloud is expected to grow with the decreasing connectivity of the edge cloud, as the result of the increasing size of the profitable regions.

Fig. 6.6 evaluates the average size of the profitable regions for file retrieval in the edge cloud, as the requested file size and the backbone delivery price (e.g., the unit price for delivering a megabit of data) increase. In Fig. 6(b), the backbone delivery price is up to 1.2, which can be as large as 24 times of the one-hop delivery price ranging from 0.05 to 0.15 [34]. The backbone price is large enough that the profitable regions can cover the entire network of $N = 100$ servers when $M = 5$, as shown in Fig. 6.6(b); in other words, a further increase of the price would not enlarge the profitable cooperative caching regions. We can see that the average region size increases with the request size and file delivery price from the backbone in both Figs. 6.6(a) and 6.6(b). This finding indicates that large files, such as the high-resolution videos, which would incur high file delivery price from the backbone, can be dispatched to large regions of the edge cloud with profitable file delivery. In



(a) Average region size vs. request size, under the backbone delivery price of 0.5.



(b) Average region size vs. backbone delivery price, under the request size of 15Mbits.

Figure 6.6 : The average region size with the increase of request size and backbone delivery price. The size of profitable cooperative regions increases with the request size and backbone delivery price, indicating that SVC-based videos should be placed with different density.

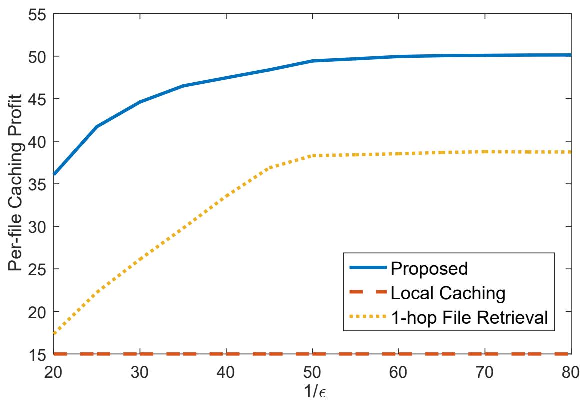


Figure 6.7 : The steady-state caching profit as $1/\epsilon$ increases. The asymptotic optimality of the proposed approach is validated.

contrast, small files, such as low-resolution videos, can only be dispatched to small regions. Consider SVC-based video contents [115]. As indicated by the finding, the base layer of videos containing low-resolution baseline profiles can be placed and cached with a high density across the edge cloud, while the enhancement layer containing additional high-resolution profiles can be placed with a low density. They can be cached at one server optimally identified per region under different request sizes accounting for different video resolutions.

Fig. 6.7 plots the steady-state caching profit of the proposed approach, local caching and 1-hop file retrieval, as $1/\epsilon$ increases from 20 to 80. The asymptotic optimality of the proposed approach is validated in the figure. In particular, the profits of the proposed approach and 1-hop file retrieval first increase with $1/\epsilon$ and stabilize when $1/\epsilon \geq 50$. The proposed approach can achieve higher steady-state profit than 1-hop file retrieval, and the profit of local caching is always 15, as already shown in Fig. 6.3.

Chapter 7

Conclusion and Future Work

This thesis illustrated the implementation of MEC for future IoT in four different cases, respectively, including 1) multi-user single-cell task scheduling, 2) multi-user multi-cell cooperative computing, 3) large-scale fog computing under limited IoT buffers, and 4) large-scale cooperative region for edge caching. Four new approaches are proposed to address the challenges. The novelties and contributions are summarized as follows.

1. In Chapter 3, we established a new hybrid learning approach for instantaneous local processing and predictive computation offloading decisions by integrating the learning techniques of SGD and OCO in the primal-dual optimization framework via Lagrange duality. The proposed hybrid learning approach can be decentralized between the BS and mobile devices for scalability by decomposing the primal problems into independent local processing and computation offloading subproblems separately. Simulation results validated the asymptotic optimality of the proposed hybrid learning approach in the presence of differently-aged network states.

2. In Chapter 4, we presented a fully distributed online learning approach to asymptotically minimizing the time-average cost of fog computing. Stochastic gradient descent was exploited to decouple the optimal operations between time slots, and a distributed heuristic was developed to decouple the spatial couplings in graph matching and achieve $\frac{1}{2}$-approximation to the optimum. The optimality loss resulting from distributed scheduling can be compensated and asymptotically diminish in online learning. Simulations showed that the proposed distributed online learning is significantly superior to the state of the art in terms of throughput and energy efficiency.

3. In Chapter 5, we enabled Lyapunov optimization to operate under finite buffers of IoT devices without loss of asymptotic optimality, by optimizing virtual placeholders of individual queues. Sufficient differences can be therefore established to drive the online optimization. The optimization of the virtual placeholders was proved to be a three-layer shortest path problem, and achieved in a distributed manner by extending the Bellman-Ford algorithm. Corroborated by simulations, the proposed approach can significantly increase the throughput and achieve asymptotically minimized time-average cost, as compared to the direct application of Lyapunov optimization.

4. In Chapter 6, we defined and established the profitable cooperative regions for distributed edge caching, and derived the necessary condition for data/file placement with guaranteed retrieval and profit. The cooperative regions were developed under the asymptotically optimal, distributed framework of cooperative caching, and achieved by formulating and solving extended shortest path problems for the instantaneous bounds of data/file requests yet to be delivered. Extensive simulations showed the average size of the profitable cooperative regions is only 30 out of a total of 400 servers.

This thesis focuses on the separated studies of computing and storage resource allocation in MEC. However, the joint optimization of computing and caching can improve the performances of MEC significantly. Nevertheless, based on the existing works, the approach in chapter 6 can be considered to operate in conjunction with the proposed approaches in Chapter 3–5 to improve the efficiency of MEC. This is because that these approaches are based on queues to operate, and can be integrated in the future work. We consider the service caching problem in the realistic systems. The services need to be pre-installed to enable the corresponding computation tasks to be processed. Caching the popular services in advance can improve the effectiveness of computing resources and reduce the end-to-end delay. In addition, the wireless IoT devices operate in shared wireless media, which have unique characteristics of unpredictable and time-varying channels. Therefore, the joint optimization of caching, computing, and communication is non-trivial due to the couplings of network operations and stochastic nature of user preference, wireless channel conditions, and network states. These will be the focus of our future work.

Bibliography

[1] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb, "Survey on multi-access edge computing for internet of things realization," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2961–2991, 2018.

[2] J. Guth, U. Breitenbücher, M. Falkenthal, F. Leymann, and L. Reinfurt, "Comparison of iot platform architectures: A field study based on a reference architecture," in *2016 Cloudification of the Internet of Things (CIoT)*. IEEE, 2016, pp. 1–6.

[3] ETSI, "Mobile edge computing (mec): Framework and reference architecture," mar, 2016.

[4] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computinga key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[5] S. H. Shah and I. Yaqoob, "A survey: Internet of things (iot) technologies, applications and challenges," in *2016 IEEE Smart Energy Grid Engineering (SEGE)*. IEEE, 2016, pp. 381–385.

[6] D. Lund, C. MacGillivray, V. Turner, and M. Morales, "Worldwide and regional internet of things (iot) 2014–2020 forecast: A virtuous circle of proven value and demand," *International Data Corporation (IDC), Tech. Rep*, vol. 1, 2014.

[7] D. Evans, "The internet of things: How the next evolution of the internet is changing everything," *CISCO white paper*, vol. 1, no. 2011, pp. 1–11, 2011.

[8] X. Lyu, H. Tian, L. Jiang, A. Vinel, S. Maharjan, S. Gjessing, and Y. Zhang, "Selective offloading in mobile edge computing for the green internet of things," *IEEE Netw.*, vol. 32, no. 1, pp. 54–60, Jan 2018.

[9] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb 2018.

[10] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2322–2358, Fourthquarter 2017.

[11] J. He, J. Wei, K. Chen, Z. Tang, Y. Zhou, and Y. Zhang, "Multi-tier fog computing with large-scale iot data analytics for smart cities," *IEEE Internet Things J.*, 2017.

[12] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of mec in the internet of things," *IEEE Consumer Electronics Magazine*, vol. 5, no. 4, pp. 84–91, 2016.

[13] O. Zakaria, J. Britt, and H. Forood, "Internet of things (iot) automotive device, system, and method," Jul. 25 2017, uS Patent 9,717,012.

[14] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, A. Neal *et al.*, "Mobile-edge computing introductory technical white paper," *White paper, mobile-edge computing (MEC) industry initiative*, pp. 1089–7801, 2014.

[15] B. Shi, J. Yang, Z. Huang, and P. Hui, "Offloading guidelines for augmented reality applications on wearable devices," in *Proceedings of the 23rd ACM international conference on Multimedia.* ACM, 2015, pp. 1271–1274.

[16] X. Lyu, H. Tian, W. Ni, Y. Zhang, P. Zhang, and R. P. Liu, "Energy-efficient admission of delay-sensitive tasks for mobile edge computing," *IEEE Trans. Commun.*, 2018.

[17] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3435–3447, 2017.

[18] X. Lyu, W. Ni, H. Tian, R. P. Liu, X. Wang, G. B. Giannakis, and A. Paulraj, "Optimal schedule of mobile edge computing for internet of things using partial information," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2606–2615, Nov 2017.

[19] L. Pu, X. Chen, J. Xu, and X. Fu, "D2D fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3887–3901, 2016.

[20] X. Chen, W. Ni, T. Chen, I. B. Collings, X. Wang, R. P. Liu, and G. B. Giannakis, "Distributed stochastic optimization of network function virtualization," in *IEEE GLOBECOM*, Dec 2017, pp. 1–6.

[21] X. Lyu, C. Ren, W. Ni, H. Tian, and R. P. Liu, "Distributed optimization of collaborative regions in large-scale inhomogeneous fog computing," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 574–586, March 2018.

[22] M. Barcelo, A. Correa, J. Llorca, A. M. Tulino, J. L. Vicario, and A. Morell, "IoT-cloud service optimization in next generation smart environments," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 4077–4090, 2016.

[23] Y. Mao, J. Zhang, S. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 16, no. 9, pp. 5994–6009, 2017.

[24] ——, “Power-delay tradeoff in multi-user mobile-edge computing systems,” in *2016 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2016, pp. 1–6.

[25] C.-F. Liu, M. Bennis, and H. V. Poor, “Latency and reliability-aware task offloading and resource allocation for mobile edge computing,” in *2017 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2017, pp. 1–7.

[26] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, “Delay-optimal computation task scheduling for mobile-edge computing systems,” in *2016 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2016, pp. 1451–1455.

[27] J. Li, H. Gao, T. Lv, and Y. Lu, “Deep reinforcement learning based computation offloading and resource allocation for mec,” in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2018, pp. 1–6.

[28] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, “Learning-based computation offloading for iot devices with energy harvesting,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1930–1941, 2019.

[29] X. Zhou, Z. Chang, C. Sun, and X. Zhang, “Demo abstract: Context-aware video streaming with q-learning for mec-enabled cellular networks,” in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2018, pp. 1–2.

[30] J. Wang, J. Hu, G. Min, W. Zhan, Q. Ni, and N. Georgalas, “Computation offloading in multi-access edge computing using a deep sequential model based on reinforcement learning,” *IEEE Communications Magazine*, vol. 57, no. 5, pp. 64–69, 2019.

[31] M. J. Neely, “Stochastic network optimization with application to communication and queueing systems,” *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[32] L. Pu, L. Jiao, X. Chen, L. Wang, Q. Xie, and J. Xu, “Online resource allocation, content placement and request routing for cost-efficient edge caching in cloud radio access networks,” *IEEE J. Sel. Areas Commun.*, vol. 36, no. 8, pp. 1751–1767, Aug 2018.

[33] W. K. Chai, D. He, I. Psaras, and G. Pavlou, “Cache less for more in information-centric networks,” in *International Conference on Research in Networking*. Springer, 2012, pp. 27–40.

[34] V. Sourlas, P. Flegkas, and L. Tassiulas, “Cache-aware routing in information-centric networks,” in *2013 IEEE International Symposium on Integrated Network Management*, May 2013, pp. 582–588.

[35] P. Sena, A. Ishimori, I. Carvalho, and A. Abelém, “Cache-aware interest routing: Impact analysis on cache decision strategies in content-centric networking,” in *LANC ’16*. New York, NY, USA: ACM, 2016, pp. 39–45.

[36] K. Poularakis and L. Tassiulas, “On the complexity of optimal content placement in hierarchical caching networks,” *IEEE Trans. Commun.*, vol. 64, no. 5, pp. 2092–2103, May 2016.

[37] J. Dai, Z. Hu, B. Li, J. Liu, and B. Li, “Collaborative hierarchical caching with dynamic request routing for massive content distribution,” in *2012 IEEE INFOCOM*, March 2012, pp. 2444–2452.

[38] X. Li, X. Wang, K. Li, Z. Han, and V. C. M. Leung, “Collaborative multi-tier caching in heterogeneous networks: Modeling, analysis, and design,” *IEEE Trans. Wireless Commun.*, vol. 16, no. 10, pp. 6926–6939, Oct 2017.

[39] Q. Li, W. Shi, X. Ge, and Z. Niu, “Cooperative edge caching in software-defined hyper-cellular networks,” *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2596–2605, Nov 2017.

[40] A. Khreishah, J. Chakareski, and A. Gharaibeh, “Joint caching, routing, and channel assignment for collaborative small-cell cellular networks,” *IEEE J. Sel. Areas Commun.*, vol. 34, no. 8, pp. 2275–2284, Aug 2016.

[41] R. Yu, S. Qin, M. Bennis, X. Chen, G. Feng, Z. Han, and G. Xue, “Enhancing software-defined ran with collaborative caching and scalable video coding,” in *2016 IEEE ICC*, May 2016, pp. 1–6.

[42] T. X. Tran, P. Pandey, A. Hajisami, and D. Pompili, “Collaborative multi-bitrate video caching and processing in mobile-edge computing networks,” in *2017 IEEE WONS*, Feb 2017, pp. 165–172.

[43] M. Dehghan, B. Jiang, A. Seetharam, T. He, T. Salonidis, J. Kurose, D. Towsley, and R. Sitaraman, “On the complexity of optimal request routing and content caching in heterogeneous cache networks,” *IEEE/ACM Trans. Network.*, vol. 25, no. 3, pp. 1635–1648, June 2017.

[44] S. M. Azimi, O. Simeone, A. Sengupta, and R. Tandon, “Online edge caching and wireless delivery in fog-aided networks with dynamic content popularity,” *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1189–1202, June 2018.

[45] Y. Guo, Q. Yang, F. R. Yu, and V. C. M. Leung, “Cache-enabled adaptive video streaming over vehicular networks: A dynamic approach,” *IEEE Trans. Veh. Technol.*, vol. 67, no. 6, pp. 5445–5459, June 2018.

[46] G. Gao, W. Zhang, Y. Wen, Z. Wang, and W. Zhu, “Towards cost-efficient video transcoding in media cloud: Insights learned from user viewing patterns,” *IEEE Trans. Multimedia*, vol. 17, no. 8, pp. 1286–1296, 2015.

[47] L. Sun, H. Pang, and L. Gao, “Joint sponsor scheduling in cellular and edge caching networks for mobile video delivery,” *IEEE Trans. Multimedia*, vol. 20, no. 12, pp. 3414–3427, Dec 2018.

[48] J. Kwak, G. Paschos, and G. Iosifidis, "Dynamic cache rental and content caching in elastic wireless CDNs," in *IEEE WiOpt*, 2018, pp. 1–8.

[49] J. Kwak, Y. Kim, L. B. Le, and S. Chong, "Hybrid content caching in 5g wireless networks: Cloud versus edge caching," *IEEE Trans. Wireless Commun.*, vol. 17, no. 5, pp. 3030–3045, 2018.

[50] L. Wang, S. Bayhan, J. Ott, J. Kangasharju, and J. Crowcroft, "Understanding scoped-flooding for content discovery and caching in content networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 8, pp. 1887–1900, Aug 2018.

[51] P. P. Ray, "A survey on internet of things architectures," *Journal of King Saud University-Computer and Information Sciences*, vol. 30, no. 3, pp. 291–319, 2018.

[52] I. Giannoulakis, E. Kafetzakis, I. Trajkovska, P. S. Khodashenas, I. Chochliouros, C. Costa, I. Neokosmidis, and P. Bliznakov, "The emergence of operator-neutral small cells as a strong case for cloud computing at the mobile edge," *Transactions on Emerging Telecommunications Technologies*, vol. 27, no. 9, pp. 1152–1159, 2016.

[53] F. Lobillo, Z. Becvar, M. A. Puente, P. Mach, F. L. Presti, F. Gambetti, M. Goldhamer, J. Vidal, A. K. Widiawan, and E. Calvanesse, "An architecture for mobile computation offloading on cloud-enabled lte small cells," in *2014 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. IEEE, 2014, pp. 1–6.

[54] S. Wang, G.-H. Tu, R. Ganti, T. He, K. Leung, H. Tripp, K. Warr, and M. Zafer, "Mobile micro-cloud: Application classification, mapping, and deployment," in *Proc. Annual Fall Meeting of ITA (AMITA)*, 2013.

[55] K. Wang, M. Shen, J. Cho, A. Banerjee, J. Van der Merwe, and K. Webb, "Mobiscud: A fast moving personal cloud in the mobile network," in *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*. ACM, 2015, pp. 19–24.

[56] T. Taleb and A. Ksentini, "Follow me cloud: interworking federated clouds and distributed mobile networks," *IEEE Network*, vol. 27, no. 5, pp. 12–19, 2013.

[57] T. Taleb, A. Ksentini, and P. Frangoudis, "Follow-me cloud: When cloud services follow mobile users," *IEEE Transactions on Cloud Computing*, 2016.

[58] J. Liu, T. Zhao, S. Zhou, Y. Cheng, and Z. Niu, "Concert: a cloud-based architecture for next-generation cellular systems," *IEEE Wireless Communications*, vol. 21, no. 6, pp. 14–22, 2014.

[59] K. Zhang, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Mobile edge computing and networking for green and low-latency internet of things," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 39–45, 2018.

[60] M. Weyrich and C. Ebert, "Reference architectures for the internet of things," *IEEE Software*, no. 1, pp. 112–116, 2016.

[61] N. H. Motlagh, M. Bagaa, and T. Taleb, "Uav-based iot platform: A crowd surveillance use case," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 128–134, 2017.

[62] H. Guo, J. Liu, and H. Qin, "Collaborative mobile edge computation offloading for iot over fiber-wireless networks," *IEEE Network*, vol. 32, no. 1, pp. 66–71, 2018.

[63] S. Shahzadi, M. Iqbal, T. Dagiuklas, and Z. U. Qayyum, "Multi-access edge computing: open issues, challenges and future perspectives," *Journal of Cloud Computing*, vol. 6, no. 1, p. 30, 2017.

[64] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[65] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How can edge computing benefit from software-defined networking: A survey, use cases, and future directions," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2359–2391, 2017.

[66] B. Blanco, J. O. Fajardo, I. Giannoulakis, E. Kafetzakis, S. Peng, J. Pérez-Romero, I. Trajkovska, P. S. Khodashenas, L. Goratti, M. Paolino *et al.*, "Technology pillars in the architecture of future 5g mobile networks: Nfv, mec and sdn," *Computer Standards & Interfaces*, vol. 54, pp. 216–228, 2017.

[67] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2016.

[68] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Transactions on Communications*, vol. 64, no. 10, pp. 4268–4282, 2016.

[69] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, April 2015.

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

[71] T. Zhao, S. Zhou, X. Guo, Y. Zhao, and Z. Niu, "A cooperative scheduling scheme of local cloud and internet cloud for delay-aware mobile cloud computing," in *2015 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2015, pp. 1–6.

[72] Y. Ge, Y. Zhang, Q. Qiu, and Y.-H. Lu, "A game theoretic resource allocation for overall energy minimization in mobile cloud computing system," in *Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design*. ACM, 2012, pp. 279–284.

[73] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571–3584, 2017.

[74] R. Yu, J. Ding, S. Maharjan, S. Gjessing, Y. Zhang, and D. H. Tsang, "Decentralized and optimal resource cooperation in geo-distributed mobile cloud computing," *IEEE Transactions on Emerging Topics in Computing*, vol. 6, no. 1, pp. 72–84, 2015.

[75] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.

[76] W. Labidi, M. Sarkiss, and M. Kamoun, "Joint multi-user resource scheduling and computation offloading in small cell networks," in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2015 IEEE 11th International Conference on*, Oct 2015, pp. 794–801.

[77] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, June 2015.

[78] K. Wang, K. Yang, and C. Magurawalage, "Joint energy minimization and resource allocation in c-ran with mobile cloud," *IEEE Trans. Cloud Comput.*, 2016.

[79] M. Lin, Z. Liu, A. Wierman, and L. L. H. Andrew, "Online algorithms for geographical load balancing," in *Proc. 2012 Int. Green Comput. Conf. (IGCC)*, June 2012, pp. 1–10.

[80] H. Xu, C. Feng, and B. Li, "Temperature aware workload management in geo-distributed data centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 6, pp. 1743–1753, June 2015.

[81] J. Luo, L. Rao, and X. Liu, "Spatio-temporal load balancing for energy cost optimization in distributed internet data centers," *IEEE Trans. Cloud Comput.*, vol. 3, no. 3, pp. 387–397, July 2015.

[82] Y. Cui, J. Song, K. Ren, M. Li, Z. Li, Q. Ren, and Y. Zhang, "Software defined cooperative offloading for mobile cloudlets," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1746–1760, June 2017.

[83] O. Muz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4738–4755, Oct 2015.

[84] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[85] M. Tao, E. Chen, H. Zhou, and W. Yu, "Content-centric sparse multicast beamforming for cache-enabled cloud ran," *IEEE Trans. Wireless Commun.*, vol. 15, no. 9, pp. 6118–6131, Sept 2016.

[86] B. Zhou, Y. Cui, and M. Tao, "Stochastic content-centric multicast scheduling for cache-enabled heterogeneous cellular networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 9, pp. 6284–6297, Sept 2016.

[87] ——, "Optimal dynamic multicast scheduling for cache-enabled content-centric wireless networks," *IEEE Trans. Commun.*, vol. 65, no. 7, pp. 2956–2970, July 2017.

[88] T. Chen, Q. Ling, and G. B. Giannakis, "An online convex optimization approach to proactive network resource allocation," *IEEE Transactions on Signal Processing*, vol. 65, no. 24, pp. 6350–6364, 2017.

[89] A. Koppel, F. Y. Jakubiec, and A. Ribeiro, "A saddle point algorithm for networked online convex optimization," *IEEE Transactions on Signal Processing*, vol. 63, no. 19, pp. 5149–5164, 2015.

[90] H. Yu and M. J. Neely, "Learning aided optimization for energy harvesting devices with outdated state information," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications.* IEEE, 2018, pp. 1853–1861.

[91] X. Cao, J. Zhang, and H. V. Poor, "A virtual-queue-based algorithm for constrained online convex optimization with applications to data center resource allocation," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 4, pp. 703–716, 2018.

[92] A. O. Allen, *Probability, statistics, and queueing theory.* Academic Press, 2014.

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

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

[95] S. Shalev-Shwartz *et al.*, "Online learning and online convex optimization," *Foundations and Trends® in Machine Learning*, vol. 4, no. 2, pp. 107–194, 2012.

[96] R. L. Burden and J. D. Faires, “2.1 the bisection algorithm,” *Numerical analysis*, 1985.

[97] H. Yu and M. J. Neely, “A simple parallel algorithm with an o(1/t) convergence rate for general convex programs,” *SIAM Journal on Optimization*, vol. 27, no. 2, pp. 759–783, 2017.

[98] E. U. T. R. Access, “Further advancements for E-UTRA physical layer aspects,” 3GPP TR 36.814, Tech. Rep., 2010.

[99] G. Dantzig, *Linear programming and extensions.* Princeton University Press, 2016.

[100] D. B. West *et al.*, *Introduction to graph theory.* Prentice hall Upper Saddle River, 2001, vol. 2.

[101] H. N. Gabow, “A scaling algorithm for weighted matching on general graphs,” *26th Annual Symposium on Foundations of Computer Science*, pp. 90–100, Oct 1985.

[102] R. Preis, “Linear time 1/2-approximation algorithm for maximum weighted matching in general graphs,” in *Annual Symposium on Theoretical Aspects of Computer Science.* Springer, 1999, pp. 259–269.

[103] M. R. Palattella, M. Dohler, A. Grieco, G. Rizzo, J. Torsner, T. Engel, and L. Ladid, “Internet of things in the 5G era: Enablers, architecture, and business models,” *IEEE J. Sel. Areas Commun.*, vol. 34, no. 3, pp. 510–527, March 2016.

[104] X. Chen, W. Ni, T. Chen, I. B. Collings, X. Wang, and G. B. Giannakis, “Real-time energy trading and future planning for fifth generation wireless communications,” *IEEE Wireless Commun.*, vol. 24, no. 4, pp. 24–30, 2017.

[105] D. Pisinger, “Algorithms for knapsack problems,” 1995.

[106] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms.* McGraw-Hill Higher Education, 2001.

[107] L. Huang and M. J. Neely, “Delay reduction via lagrange multipliers in stochastic network optimization,” *IEEE Trans. Autom. Control*, vol. 56, no. 4, pp. 842–857, April 2011.

[108] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas, *Dynamic programming and optimal control.* Athena scientific Belmont, MA, 1995, vol. 1, no. 2.

[109] D. B. West *et al.*, *Introduction to graph theory.* Prentice hall Upper Saddle River, 2001, vol. 2.

[110] D. Lee, J. Choi, J. H. Kim, S. H. Noh, S. L. Min, Y. Cho, and C. S. Kim, "LRFU: A spectrum of policies that subsumes the least recently used and least frequently used policies," *IEEE Trans. Comput.*, vol. 50, no. 12, pp. 1352–1361, Dec. 2001.

[111] T. Chen, A. Mokhtari, X. Wang, A. Ribeiro, and G. B. Giannakis, "Stochastic averaging for constrained optimization with application to online resource allocation," *IEEE Trans. Signal Process.*, vol. 65, no. 12, pp. 3078–3093, June 2017.

[112] Z. Zhao, L. Guardalben, M. Karimzadeh, J. Silva, T. Braun, and S. Sargento, "Mobility prediction-assisted over-the-top edge prefetching for hierarchical vanets," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 8, pp. 1786–1801, Aug 2018.

[113] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas, *Dynamic programming and optimal control.* Athena Scientific Belmont, MA, 1995, vol. 1, no. 2.

[114] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: evidence and implications," in *IEEE INFOCOM*, Mar 1999, pp. 126–134 vol.1.

[115] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the h.264/avc standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, pp. 1103–1120, Sept 2007.