UNIVERSITY OF TECHNOLOGY SYDNEY
Faculty of Engineering and Information Technology

Resource Management of Fog Computing in Future Networks

by

Xinchen Lyu

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

Doctor of Philosophy

Sydney, Australia

2018

Certificate of Authorship/Originality

I, Xinchen Lyu 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.

Production Note:

Signature: Signature removed prior to publication.

Date: 11 April 2019

ABSTRACT

Resource Management of Fog Computing in Future Networks

by

Xinchen Lyu

By enabling task offloading among edge network nodes (such as base stations, switches, and routers), fog computing is able to process the computationally demanding tasks at the point of capture. It has the potential to provide low-latency services, increase network capacity, and relieve network congestions. In this thesis, we focus on three exemplary scenarios of fog computing, including (1) single-cell multiuser fog computing to jointly optimize task offloading and resource allocation for different applications with heterogeneous quality of service requirements; (2) fog computing among selfish devices to design incentive mechanism and efficient management of task offloading, processing, and result retrieving; and (3) fog computing across large-scale edge cloud for scalable and distributed resource management in the presence of a large number of geo-distributed edge servers.

We present five new approaches to address the challenges for efficient and scalable fog computing in the three scenarios. The first three approaches are for the first scenario, i.e., single-cell multiuser fog computing, for three different types of applications, including delay-tolerant, delay-sensitive, and data-partition tasks. The fourth approach is for the second scenario, where distributed tit-for-tat mechanism is proposed to incentivize the cooperation of selfish devices. The fifth approach is for the last scenario, where collaborative regions are created for preventing tasks being offloaded beyond the vicinity of the point of capture in large-scale networks.

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 (BUPT), 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 wife, thank you for encouraging, enduring and inspiring me through the years. Love you always.

Xinchen Lyu

Sydney, Australia, 2018.

List of Publications

Journal Papers

J-1 . **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****,* 2018, DOI: 10.1109/JSAC.2018.2871310.

J-2 . **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-3 . **X. Lyu, W. Ni, H. Tian, R. P. Liu, X. Wang, G. B. Giannakis, A. Paulraj.** *"Optimal Schedule of Mobile Edge Computing for Internet of Things Using Partial Information" in* ***IEEE Journal on Selected Areas in Communications****,* vol. 35, no. 11, pp. 2606-2615, Nov. 2017.

J-4 . **X. Lyu, H. Tian, L. Jiang, A. Vinel, S. Maharjan, S. Gjessing, Y. Zhang.** *"Selective Offloading in Mobile Edge Computing for the Green Internet of Things" in* ***IEEE Network****,* vol. 32, no. 1, pp. 54-60, Jan.-Feb. 2018.

J-5 . **X. Lyu, W. Ni, H. Tian, R. P. Liu, X. Wang, G. B. Giannakis, A. Paulraj.** *"Distributed Online Optimization of Fog Computing for Selfish Devices with Out-of-Date Information" in* ***IEEE Transactions on Wireless Communications****,* 2018, DOI: 10.1109/TWC.2018.2869764.

J-6 . **X. Lyu, H. Tian, W. Ni, Y. Zhang, P. Zhang, R. P. Liu.** *"Energy-Efficient Admission of Delay-Sensitive Tasks for Mobile Edge Computing" in* ***IEEE Transactions on Communications****,* vol. 66, no. 6, pp. 2603-2616,

June 2018.

J-7 . **X. Lyu, H. Tian, C. Sengul, P. Zhang.** *"Multiuser Joint Task Offloading and Resource Optimization in Proximate Clouds" in* ***IEEE Transactions on Vehicular Technology****,* vol. 66, no. 4, pp. 3435-3447, April 2017.

J-8 . **X. Lyu, H. Tian, W. Ni, R. P. Liu, P. Zhang.** *"Adaptive Centralized Clustering Framework for Software-Defined Ultra-Dense Wireless Networks" in* ***IEEE Transactions on Vehicular Technology****,* vol. 66, no. 9, pp. 8553-8557, Sept. 2017.

J-9 . **X. Lyu, H. Tian.** *"Adaptive Receding Horizon Offloading Strategy Under Dynamic Environment" in* ***IEEE Communications Letters****,* vol. 20, no. 5, pp. 878-881, May 2016.

Contents

List of Figures

Abbreviation

Internet of Things - IoT

Quality of service - QoS

Non-deterministic polynomial hard- NP-hard

Mixed integer programming - MIP

Dynamic programming- DP

Time-division multiple access - TDMA

Orthogonal frequency-division multiple access - OFDMA

Alternating direction method of multipliers - ADMM

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 non-identically distributed - i.n.d.

Independent and identical distributed - i.i.d.

Round robin - RR

Proportional fair - PF

Device-to-Device - D2D

Precision time protocol -PTP

Timing-sync protocol for sensor networks - TPSN

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 Introduction

The Internet of Things (IoT) is proposed to equip everyday objects with electronics, software, sensors, and network connectivity, and bring the vision of a connected world into reality [2]. However, computation-intensive services, such as eHealth, automatic driving, and industrial automation, are fast developing and outgrowing the computing and storage capabilities of the IoT devices. Cloud computing offers enormous storage, computing facilities and data sharing opportunities. By offloading the computation and storage from the IoT devices to the cloud through mobile networks, mobile cloud computing can alleviate the computation and storage limitations and prolong the lifetimes of the IoT devices [3]. The ability is crucial for future IoT networks, by bringing together modern computing, Industrial IoT (i.e., IoT 4.0 or smart factory), and artificial intelligence to create intelligent, self-optimizing industrial equipment and facilities.

As the computing units in the core network use shared backhaul resources, mobile cloud computing may not be able to meet the reliability and latency requirements for the IoT services. For instance, the emergency IoT services, such as mobile vehicular connectivity, eHealth and industrial automation, require ultra-low latency and extremely high reliability. In addition, the services from smart sensors generate high volumes of data. Uploading the sensed data to the cloud may waste energy and cause traffic congestion in the core network.

Fog computing is introduced to provide the radio access networks with cloud computing capabilities, and collaborate the edge servers for big data analytics. For instance, macro/pico/femto base stations may be connected to co-located edge servers, so as to reduce latency, ease the traffic on backhaul links, and deliver reliable

Table 1.1 : The comparison of the existing approaches.

Reference	**Scenario**	**Architecture**	**Offloading**	**Scalability**
[5–9]	Single-device	Distributed	Cloud	No
[10–15]	Single-cell	Centralized	Fog server	No
[16–19]	Single-cell	Distributed	Fog server	No
[20–22]	Hierarchical	Centralized	Fog server/cloud	No
[23, 24]	Hierarchical	Distributed	Fog server/cloud	No
[27–29]	Data center	Centralized	One-hop neighbors	No
[25, 26]	Cooperation	Centralized	One-hop neighbors	No

services. Typical characteristics of fog computing include proximity, high energy efficiency, low latency, high throughput, mobility support and location awareness [4]. These features are highly in line with the requirements of the IoT services.

Earlier works on task offloading, such as [5–9], were focused on single-device decision-makings under an implicit assumption of unlimited computational capabilities on cloud platforms. Recent studies have been focused on the offloading decisions and resource allocations in single-cell fog computing where multiple devices are controlled by a single fog server in both centralized [10–15] and distributed manners [16–19]. A hierarchical fog architecture was considered in [20–24], where mobile devices could offload their tasks to either a fog server or a remote cloud. Cooperation of (fog) devices was studied in [25–29]. Table 1.1 summarizes the comparison of the existing approaches. The detailed literature review will be given in Chapter 2.

1.1.1 Challenges of Fog Computing in Future Networks

Fig. 1.1 illustrates an application of fog computing in future networks, where a number of IoT devices, connected to an edge cloud through wireless interfaces, produce big data to be processed at the edge cloud and the data center. We can see in Fig. 1.1 that there are three cases of interest for fog computing in future networks, i.e., **Case 1:** single-cell multiuser fog computing (i.e., the connections between edge

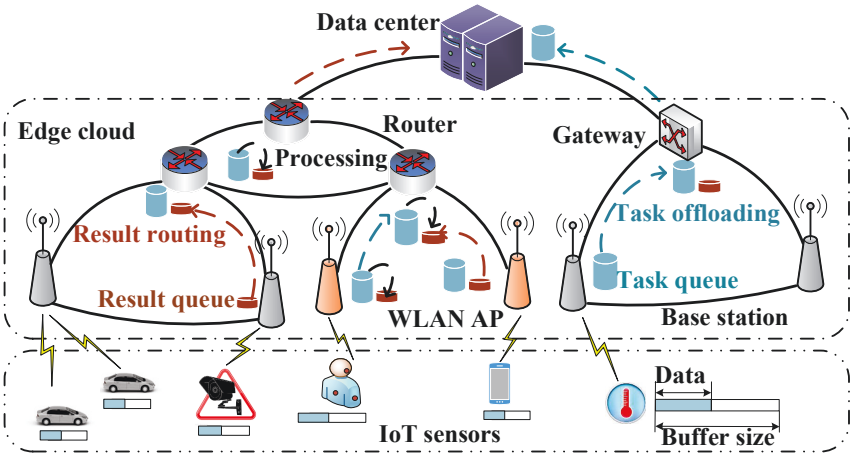


Figure 1.1 : The application of fog computing in IoT networks.

server and devices in the figure), **Case 2:** fog computing among selfish devices (at the bottom of the figure), and **Case 3:** fog computing across large-scale geo-distributed edge cloud (at the middle of the the figure). This following introduces the three cases of interest for fog computing in future networks and discusses the challenges of fog computing in these cases.

Case 1: Single-cell Multiuser Fog Computing.

This is the case where multiple devices are in the coverage of an edge server and offload tasks to the edge server for processing. The single-cell multiuser fog computing establishes the connections between the edge servers (at the middle of Fig. 1.1) and the devices (at the bottom of Fig. 1.1), and is critical for achieving the effectiveness of fog computing.

The main problem to be solved in this case is the joint management (or optimization) of task offloading decisions, wireless resource management, and computing resource allocations at the edge server. Given the limitation of computing and wireless resources at the edge server, the resource management approaches are required to be adequately designed to achieve the effectiveness (in terms of energy saving and latency reduction) and ensure the quality of service (QoS) of fog computing.

The challenges arise from the inherent coupling relations of the wireless and computing allocations, the scalability (in terms of time-complexity and signalling

overhead) in the presence of a large number of devices, and most importantly, distinctively different QoS requirements for different applications. The development of resource management approaches are highly application-specific. Different approaches, or in other words, different approaches need to be developed to satisfy different requirements of various applications. In the following, we mainly focus on three specific types of applications, as listed in the following three sub-cases.

Case 1.1: Delay-tolerant Atomic Tasks.

This type of tasks is general-purpose (e.g., image recognition) where there is no hard deadline for task executions, and the objectives of fog computing can vary significantly based on the preferences of the mobile users (e.g., to save energy or reduce latency). To design a general resource management approach, which can suit the different objectives of the users, is non-trivial. Nevertheless, the atomic feature of these tasks introduces the binary constraints for task offloading (since atomic task has to be offloaded/executed in whole), and the joint optimization of task offloading and resource allocation becomes a non-deterministic polynomial (NP)-hard mixed integer programming (MIP) problem. Effectively solving such problem with low time-complexity is challenging but essential for this type of tasks.

Case 1.2: Delay-sensitive Atomic Tasks.

This type of tasks typically requires high reliability and low latency (e.g., online gaming), where the user experiences would significantly degrade when violating the task deadlines. Task admission and resource allocation are critical to the delay-sensitive tasks due to the finite physical bandwidths of wireless channels and limited computational resources at the edge servers. In coupling with the allocation of both radio/energy resources for task offloading and computational resources for task processing, task admission is typically an NP-hard combinatorial mixed integer programming problem. The computational complexity of task admission would exponentially grow against the number of devices, and become prohibitive in the presence of large numbers of devices (or offloading requests). The scalability and practicality of task admission would degrade. Efficient joint optimization of offloading decisions and resource allocations has yet to be addressed for delay-sensitive

tasks, especially in the case where the number of devices is large, the optimization can be infeasible and admission control would be necessary.

Case 1.3: Data-partition Tasks for IoT Devices.

The data-partition tasks (e.g., data analytics for health-care and smart city) are also known as divisible tasks, such as Gzip compression and feature extraction, where the data are bit-wise independent and the tasks can be partitioned for offloading and execution In the era of IoT, a large number of IoT devices will be under the coverage of a single access point (i.e., edge server), where the devices would offload their sensed data to the edge server for processing. Here, we assume that there is no correlation between the data arriving at different IoT devices (e.g., in different data types of humidity, electricity and images). The challenges arise from the scale of the IoT devices. In particular, the optimal schedules of fog computing requires the explicit knowledge on the IoT devices, but the feedbacks (e.g., channel states and battery information) from thousands of devices would congest the channel, incur prohibitive overhead, and jeopardize practicality. None of the existing works for single-cell or hierarchical fog computing is able to schedule the offloading of a more significant number of IoT devices due to the computationally-prohibitive complexity [10–15, 20–22] and significant singling overhead [16–19, 23, 24].

Case 2: Fog Computing among Selfish Devices.

This is the case of distributed wireless networks (e.g., tactic networks) where the mobile devices can offload their computationally-demanding tasks (e.g., virtual reality and augmented reality) to nearby peers for processing and retrieving the computation results from the peers, as shown at the bottom of Fig. 1.1. Fog computing among mobile devices is able to bridge the gap of limited computing capability of a single device and the demands of computing resources for these complex applications.

The challenges arise from the selfish nature of the devices and the efficiency of fog computing within one-hop neighbors. In [25, 26], selfish devices were incentivized for cooperation via the tit-for-tat mechanisms, but the cooperations (i.e., task offload-

ing) were restrained within the one-hop neighbors, thereby hampering the efficiency of fog computing. However, designing multi-hop fog computing is challenging given the complex routing mechanisms of task offloading and result returning in non-trivial (even time-varying) topology of the mobile devices. Moreover, the tit-for-tat incentive for one device to help another, generated by the latter, can be already outdated when reaching the former, as the result of multi-hop propagation delays.

Case 3: Fog Computing across Large-scale Geo-distributed Edge Cloud.

This is the case of fog computing across the large-scale edge cloud (at the middle of Fig. 1.1), where edge servers across a vast geographical region can cooperate with each other via multi-hop task offloading through the inter-server links between them. It is important to unify the computing/storage capabilities of the edge servers in the geo-distributed edge cloud and support big data analytics and machine learning services. However, the existing works in the data center networks [27–29] were centralized and cannot scale to the distributed edge cloud, since global network information is not practical due to the signaling delays.

The critical challenges arise from the regions for cooperation among the edge servers. In particular, the edge server may offload tasks far beyond the point of capture, resulting in excessive and unnecessary delays of task processing. There exists a tradeoff between the throughput (i.e., effectiveness) and delay (i.e., the size of collaborative regions) for fog computing, i.e., offloading tasks increasingly far away may achieve high throughput at the penalty of excessive delays. This is challenging to optimally determine the collaborative regions so as to reduce the excessive delay and guarantee the effectiveness of collaboration.

1.2 Thesis Organization

In this thesis, we present five new approaches to address these challenges for efficient and scalable fog computing in the above five cases. The first three approaches are designed for *Case 1.1 – Case 1.3*, respectively, i.e., the single-cell multiuser fog computing for different types of applications. The fourth approach is designed for *Case 2*, i.e., the distributed online optimization of fog computing for selfish devices.

The last approach is designed for *Case 3*, i.e., the distributed optimization of fog computing across the large-scale geo-distributed edge cloud.

This thesis is organized as follows:

1. As a brief introduction, Chapter 1 introduces the background of fog computing and discusses the challenges of fog computing in five specific cases.

2. Chapter 2 introduces the reference architecture of fog computing and summarizes the related work and task models for fog computing.

3. Chapter 3 illustrates the joint optimization of task offloading and resource allocation for delay-tolerant atomic tasks, where decomposition techniques are exploited to enable the semi-distributed framework and reduce time-complexity.

4. Chapter 4 presents the energy-efficient admission control of delay-sensitive tasks, where a new quantized dynamic programming technique is leveraged to adjust the tradeoff between optimality (in terms of energy saving) and time-complexity.

5. Chapter 5 elaborates on the optimal scheduling approach for IoT devices powered by renewable energy under partial knowledge on the channel conditions.

6. Chapter 6 shows the distributed online optimization of fog computing for selfish devices under out-of-date signaling on tit-for-tat incentives.

7. Chapter 7 illustrates the distributed optimization of collaborative regions of geo-distributed edge servers, where the regions are created for preventing tasks being offloaded beyond the vicinity in large-scale networks.

8. Chapter 8 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.

Chapter 2

Literature Review

In this chapter, we will elaborate on the use cases of fog computing and its reference architecture, present the current resource management models for task offloading, and then provide the literature review of resource management in fog computing.

2.1 Fog Computing and its Reference Architecture

1) Background and Use Cases of Fog Computing:

Fog computing is a recently emerging distributed computing paradigm, which is closely associated with cloud computing (in data center networks) and IoT services [30, 31]. In particular, part of the data arriving at the IoT devices can be processed (analyzed) locally at the edge devices (i.e., fog), while the others are pushed to the cloud in the core network. The development of fog computing frameworks provides the operators with the choices for processing data wherever it is most appropriate and efficient. For example, some applications may require the data to be analyzed/processed in ultra-low latency and high reliability, e.g., in a manufacturing (or connected vehicle) use case where machines/vehicles are required to be able to respond to an incident in less than 2 ms delay [32].

Fog computing can satisfy such low-latency and high-reliability service requirements and increase the network processing capacity, by exploiting the low-latency network connections between the edge devices and analytics endpoints [33]. Fog computing can also reduce the amount of required bandwidth (of the backhaul links), as compared to traditional cloud computing where the data need to be sent all the way back to a data center in the core network for processing. This is highly aligned with future IoT era/networks where a large volume of data can be streamed into the devices for processing or analysis. It can also be in scenarios (e.g., tactic

networks), where there is no network link to send data, so it must be processed close to where it is created to fulfill the real-time requirement [34].

Fog computing is still its emerging stages (as compared to cloud computing), but, given the aforementioned benefits, there are a variety of use cases that have been identified as potential ideal scenarios for fog computing in both industrial and academic groups, as listed in the following [1].

- *Connected Cars*: With the advent of semi-autonomous and self-driving cars, the vehicles need to have the ability to locally analyze the sensing data (e.g., from cameras) in real-time, and decide the operations (e.g., driving speed and directions). The incident with high priority is required to be analyzed locally at the vehicles, while the others can be sent back to the core network for the maintenance purpose. The fog computing framework can empower the communications among the vehicles to cooperatively process the data (given the temporal variations of task arrivals) in real-time.

- *IoT Applications*: Like connected cars, IoT applications (e.g., smart city/grid) are increasingly using real-time data to manage the system more efficiently. Moreover, the large volume of data generated by the IoT devices (sensors) cannot be processed in a centralized manner in a data center. Fog computing framework, which can process the data aggregated from the sensors close to the point of capture, is able to address both the issues in these applications.

- *Real-time Data Analytics*: Real-time data analytics are critical in many practical systems, for example, manufacturing systems that are required to react to incidents in no time, and financial systems that exploit real-time data to inform trading decisions. These applications may require not only the real-time requirements but also the reliability/security of data processing. Fog computing deployments can help facilitate the storage of data in a variety of places with backups to fulfill both the requirements.

2) OpenFog Reference Architecture for Fog Computing:

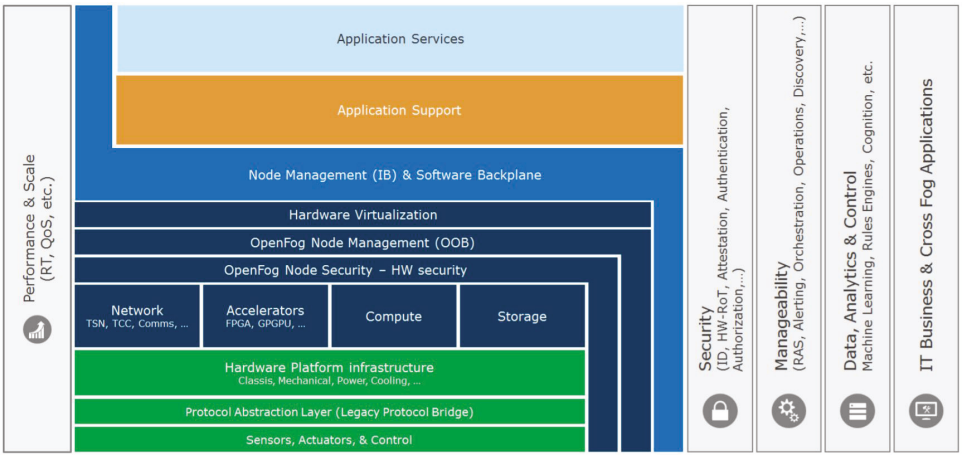


Figure 2.1 : The OpenFog reference architecture for fog computing [1].

As mentioned above in the use cases, a fog computing framework can have a variety of components and gateways, including gateways, routers, switches, and other edge devices; which may require a dedicated design of the fog computing architecture to exploit the potential of fog computing. The OpenFog Consortium, from a global membership of 55 organizations led by Cisco, has outlined three goals for developing a fog framework [1]. In particular, fog computing should be horizontally scalable, meaning it will support multiple industry vertical use cases; be able to work across the cloud to things continuum; and be a system-level technology, that extends from things, over network edges, through to the cloud and across various network protocols.

Fig. 2.1 shows the OpenFog reference architecture for fog computing [1]. The OpenFog research architecture [1] is a mid-level guidepost document, full of invaluable architecture recommendations for anyone wishing to implement fog components, fog nodes, entire fog networks, or fog-based applications. It describes in detail several illustrative fog use cases in transportation, smart cities, and visual security. The research architecture also discusses eight core fog capabilities that we call the Pillars of OpenFog, including Security, Scalability, Openness, Autonomy, Manageability, Agility, Hierarchical organization and Programmability. The detailed architecture stack shows the interrelationships between various hardware, software infrastructure, and application software layers, as well as various cross-cutting concerns, such

as security, performance, manageability, analytics, and control.

2.2 Resource Management Models in Fog Computing

OpenFog reference architecture [1] (as shown in Fig. 2.1) provides a programmable and scalable architecture for fog computing. The operators can design and apply their specific designed resource management scheme in the OpenFog reference architecture to optimize the resource allocation (such as task offloading, load balancing, and radio resource management) of fog computing network.

Given the various use cases of fog computing, the resource management in fog computing needs to take the unique features of the specific applications (e.g., data analytics and automatic driving) in the use cases into account. We proceed to introduce the computation task models for different applications. The models provide mechanisms for abstracting various/different functions and applications into optimization problems and facilitating theoretical analysis as discussed in the following sections.

In general, a computation task can be modeled as a triplet (D, ρ, ξ), which indicates that the task has an input data size of D bits, processing a bit of input data requires ρ CPU cycles, and the computation data result is ξD bits. In in many applications of Gzip data compression, feature extraction, and data analytics, the required CPU cycles are linear to the number of bits of the input data, and the ratio ρ is also known as the computation density of the application [35]. The commonly-used three parameters can capture the essential properties of applications (e.g., computation and communication demands) [35]. Based on whether and how the input data can be partitioned, the computation tasks can be categorized into three types, as illustrated in the following.

1. *Binary Task Model*: A highly integrated tasks cannot be partitioned and has to be executed in whole locally or offloaded to the edge server for remote processing. In this case, the input data D of the task cannot be partitioned, and the offloading decisions are modeled as a binary variable.

2. *Data-partition Task Model*: In practice, many applications are composed of multiple components, which could be partitioned to enable the fine-grained task offloading. The data-partition task model is the case where the input data are bit-wise independent, and can be divided for execution. In this case, the offloading decisions can be modeled as a continuous variable.

3. *Dependent Task Model*: In other cases, there may exist dependency among different components of a task/application due to the inherent sequential/parallel execution of the codes. In particular, such dependency can be modeled as a task-call graph (e.g., directed acyclic graph), where a node in the graph is a component in the application and a directed edge between two nodes indicates that the node (task) in the latter cannot be started to process until the node (task) in the former has been accomplished. Based on the task features, there can be three types of dependency models of the task-call graph, including sequential, parallel, and general dependency [35].

2.3 Summary of Related Work

1) Single-device Offloading Decisions:

Earlier works on task offloading, such as [5–9], were focused on single-device decision-makings under an implicit assumption of unlimited computational capabilities on cloud platforms. In [5], an application was decided to be offloaded entirely to a cloud or executed at a mobile device. In [6–9], applications were partitioned into tasks or code blocks to improve efficiency. In [6], a mobile application was partitioned into a sequence of tasks which were processed sequentially. In [7], partitioned tasks were processed in parallel, and dynamic programming (DP) was employed to minimize the processing delay under energy constraints. In [8], an online heuristic approach was developed to partition tasks. In [9], a directed acyclic graph was used to represent code blocks, and a genetic algorithm was developed to partition the code blocks.

2) Multi-user Offloading in Single-cell Fog Computing:

Recent studies have been focused on the offloading decisions and resource allocations in single-cell fog computing where multiple devices are controlled by a single fog server in both centralized [10–15] and distributed manners [16–19]. In [10], the average offloading priority function was defined to decide the close-to-optimal offloading decisions of the devices in both cases of time-division multiple access (TDMA) and orthogonal frequency-division multiple access (OFDMA). In [11], a heuristic approach was proposed to minimize the energy consumption under latency constraints of the tasks. In [12], both offline and online approaches were proposed for the joint optimization, where a single task was offloaded to the MEC server while the others were executed locally. In [13] and [14], transmission and computational resources were jointly optimized to save energy, but there was no attempt to optimize offloading decisions. In [13], a distributed optimization framework was proposed using successive convex optimization, and a closed-form expression for the maximum energy saving was derived in a single-user case. In [14], both independent and joint optimizations of computational and transmission resources were formulated to be non-convex optimization problems, which were reformulated and iteratively solved using minimum mean square error criteria. In [15], joint optimization of offloading and resource allocation was considered, but under a relaxed assumption that the problem was feasible. A suboptimal solution was developed by decoupling offloading decisions and resource allocations. Tasks were incrementally offloaded, and resources were correspondingly allocated by exploiting second-order cone programming iteratively, until either the task deadlines were violated or energy saving diminished.

For the sake of scalability, distributed scheduling of fog computing was later proposed by applying game theory [16, 17], decomposition techniques [18] and dynamic programming [19]. In [16, 17], a non-cooperative game for offloading decisions was formulated to capture the interactions among mobile devices in wireless interference environments. The utilities of individual game players were designed with an emphasis on the stability of the games. Only Pareto-optimal solutions could be achieved, if stabilized, which unnecessarily can be translated to the global optimal-

ity in terms of the utility of the entire cell. In [18], 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 [19], energy-efficient admission of delay-sensitive services was proposed, where, by exploiting dynamic programming and decomposition techniques, the devices can spontaneously decide to withhold their offloading requests without compromising the asymptotic optimality of the approach.

3) Multi-user Offloading in Hierarchical Fog Computing:

A hierarchical fog architecture was considered in [20–24], where mobile devices could offload their tasks to either a fog server or a remote cloud. These approaches [20–24] are extensions of those in the single-cell fog computing scenario [10–15], where the binary offloading decisions (for local execution or offloading to the fog server) were relaxed to be integer variables to indicate the destination of task offloading (i.e., fog servers or the cloud).

In [21], a computation offloading game was formulated to model the competition between devices for the limited computing resources. In [20, 22], the synchronization of the cloud and edge devices was studied, and the end-to-end delays were analyzed with a focus on engineering practice. The game was proved to be a potential game, thereby confirming the existence of the Nash equilibrium. In [23, 24], different queuing models were applied to capture the delay performance of task processing at a mobile device, a fog node, and the cloud. In [23], a multi-objective optimization problem was formulated to obtain a Pareto-optimal solution among energy consumption, delay, and the payment to the fog service provider. In [24], each device aimed to minimize the execution cost of itself and its socially connected neighbors.

4) Cooperation among Multiple Fog Devices:

In a different yet relevant context of collaborative computing between data centers in a cloud, centralized off-line load balancing was studied to deal with the spatial diversities of workload arrivals [27], temperatures [28], and electricity prices [29]. In [27], by exploiting the receding horizon control mechanism, the geographical load

balancing executed and adjusted the provisioning decision for the current time given the predicted future workloads in a time window. In [28], the temperature diversity and relations to the energy efficiency of the cooling system were exploited to develop a temperature-aware load balancing approach based on the alternating direction method of multipliers (ADMM) algorithm. In [29], the online load balancing approach was proposed to minimize the system cost under time-varying electricity price from the smart grid, where queuing theory and price estimation method were developed from the empirical datasets.

The cooperation of fog devices was recently studied in [25, 26], where tit-for-tat incentives were used to discourage selfish behaviors of devices under the assumption that the incentives generated at individual devices can be instantly obtained at a central controller. The decisions of cooperations among devices were optimized given the up-to-date knowledge on the incentives by using Lyapunov optimization [25, 26]. Moreover, offloading was confined within a single hop with limited numbers of devices to cooperate [25, 26], hence preventing fog computing for large tasks.

Chapter 3

Multiuser Joint Task Offloading and Resource Optimization

In this chapter, we jointly optimize the offloading decision (i.e., which users should be offloaded, and which users should use local execution), and resource utilization in a single-cell network for the applications of binary task model (i.e., the tasks are highly integrated and need to be offloaded in whole). Our solution takes the inherent heterogeneity across mobile devices and computation tasks. We design quality of experience (QoE)-based utility function, which measures the utility of task completion time and energy consumption of offloading compared to local execution. Then, we formulate the problem as a system utility maximization problem, which jointly optimizes the offloading decision, and communication and computation resources. The novelties and contributions of this chapter are as follows.

- We reduce the problem to a submodular maximization problem and prove its NP-hardness by decomposing it into two subproblems (1) optimization of communication and computation resources, and (2) offloading decision. Communication and computation resources are optimized through convex and quasi-convex optimization.

- Based on the results of resource optimization, we prove the system utility to be a submodular set function in terms of offloading decision, and compute the local optimum through submodular optimization.

- The proposed heuristic offloading decision algorithm (HODA) reduces the complexity of finding the local optimum to $O(K^3)$, compared to the $O(2^K)$ complexity of the optimal solution, where K is the number of mobile users. As the number of mobile users increases, which results in the shortage of resources,

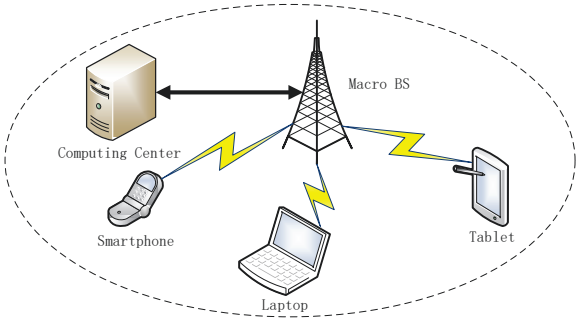


Figure 3.1 : The single-cell multi-user fog computing scenario.

HODA is able to manage the bottleneck by selecting the users that will benefit most from offloading, and its performance is within 5% of the optimal on average.

3.1 System Model

In this chapter, we consider a single-cell multi-user fog computing scenario shown in Fig. 3.1. In this scenario, the single LTE macro base station (BS) has a computing center with limited computation resources and serves K mobile devices in its range. These mobile devices may have different computation and energy resources, such as in the case of smartphones, tablets, and laptops.

Each mobile user i has a computation task CT_i, which can be described in terms of

1. The input, D_i(bit), including system settings, program codes, and input parameters;
2. The number of CPU cycles required to accomplish the computation task, C_i;
3. The output (e.g., the computation result).

We assume tasks are atomic, and cannot be divided into subtasks. The information about D_i and C_i can be obtained by applying program profilers.

Each computation task can be either executed locally or remotely in a macro computing center.

1) Local Computation:

F_i^l denotes the local computation capability of mobile user i in terms of instructions per second. T_i^l is the task completion time and is

$$T_i^l = \frac{C_i}{F_i^l}. \tag{3.1}$$

According to [36, 37], CPU power consumption is a super-linear function of execution frequency, and is given as

$$P_i^l = \alpha \cdot (F_i^l)^\gamma, \tag{3.2}$$

where P_i^l denotes the local power consumption of mobile user i, and typically, $\alpha = 10^{-11}$ and $\gamma = 2$ [16, 17].

The energy consumption of mobile user i, E_i^l (Joule), becomes

$$E_i^l = P_i^l \cdot T_i^l = \alpha (F_i^l)^{\gamma-1} C_i. \tag{3.3}$$

It should be noted that both T_i^l and E_i^l are determined only by F_i^l and C_i, which are intrinsic features of a mobile device and its computation task, and therefore, are known.

2) Remote Computation:

A typical remote computing approach consists of three stages

1. The mobile user uploads CT_i to macro BS through the uplink channel;
2. The macro computing center allocates f_i computation resources to the task, and then executes it on behalf of the mobile user;
3. Macro BS transmits output data back to the mobile user.

As mentioned above, we ignored the overhead of the output data in the last stage [16, 17].

Compared with the local computation, remote computation saves mobile users' energy and computation resources for task execution, but spends additional time and energy in uplink transmission. For the uplink transmission, the intra-cell interference is well-mitigated in the LTE network. Therefore, the data rate of a mobile user can be given as [38]

$$R_i(p_i) = W\log_2(1 + \frac{p_i \cdot h_i}{N_0}), \tag{3.4}$$

where W(Hz) is the user bandwidth and p_i denotes the transmission power, which can be configured by the mobile user subject to a maximum transmission power constraint. B(Hz) is the system bandwidth, and therefore, at most $N = B/W$ mobile users are allowed to transmit at the same time. h_i denotes the channel gain from the mobile user i to macro BS including path-loss and fading, and N_0 is the noise power. According to (4.4), the mobile user i can adjust its data rate from 0 to $W\log_2(1 + p_0h_i/N_0)$ by controlling its transmission power, where p_0 denotes maximum transmission power allowed.

The total remote computation completion time for mobile user i, $T_i^r(f_i, p_i)$, is composed of two parts

$$T_i^r(f_i, p_i) = T_i^t(p_i) + T_i^e(f_i), \tag{3.5}$$

where $T_i^t(p_i)$ and $T_i^e(f_i)$ are uplink transmission time and remote execution time, respectively. $T_i^t(p_i)$ depends on the size of the input, D_i, and the rate achieved by the user, R_i(bps)

$$T_i^t(p_i) = \frac{D_i}{W\log_2(1 + a_ip_i)}, \tag{3.6}$$

where $a_i = h_i/N_0$. Similar to (4.1), $T_i^e(f_i)$ is

$$T_i^e(f_i) = \frac{C_i}{f_i}, \tag{3.7}$$

where $p_i \neq 0$ and $f_i \neq 0$, so infinite transmission and execution times are avoided.

The energy consumption of the mobile user i for remote computation is E_i^r (J). We only consider the energy consumption of the upload, since the energy consump-

tion for task execution at the mobile is saved through offloading [16, 17].

$$E_i^r(p_i) = \frac{p_i}{\zeta} \cdot T_i^t(p_i) = \frac{p_i}{\zeta} \frac{D_i}{W \log_2(1 + a_i p_i)}, \tag{3.8}$$

where ζ is the power amplifier efficiency.

3.2 Problem Formulation

In a mobile cloud computing system, the QoE of a user i is determined mainly by task completion time, T_i, and energy consumption, E_i. Specifically, T_i and E_i can be obtained as

$$T_i = s_i \cdot T_i^r + (1 - s_i) \cdot T_i^l; \tag{3.9}$$

$$E_i = s_i \cdot E_i^r + (1 - s_i) \cdot E_i^l; \tag{3.10}$$

where $s_i \in \{0, 1\}$ denotes the offloading decision for CT_i (i.e., the task is offloaded when $s_i = 1$).

The performance improvement in time and energy consumption compared with local execution can be given by $\frac{T_i^l - T_i}{T_i^l}$ and $\frac{E_i^l - E_i}{E_i^l}$, respectively. The user preferences on task completion time and energy consumption, $\beta_i^T, \beta_i^E \in [0, 1]$, can be determined by the remaining battery life and the requirement of task completion time. For instance, a mobile user i of short battery life can increase β_i^E and decrease β_i^T, so as to save more energy at the expense of longer task completion time. Taking user preferences into account, we define the utility function of mobile user i as

$$\begin{aligned} v_i(s_i, f_i, p_i) &= \beta_i^T \frac{T_i^l - T_i}{T_i^l} + \beta_i^E \frac{E_i^l - E_i}{E_i^l} \\ &= s_i(\beta_i^T \frac{T_i^l - T_i^r}{T_i^l} + \beta_i^E \frac{E_i^l - E_i^r}{E_i^l}). \end{aligned} \tag{3.11}$$

Here, the terms of T_i and E_i are functions of the offloading decisions s_i, as shown in (3.9) and (3.10). Note that the user utility, $v_i(s_i, f_i, p_i)$, measures the improvement in QoE by offloading compared with local execution. So, if the decision is local execution, and hence, $s_i = 0$, utility function is equal to 0. Also, due to limited

bandwidth and remote computation resources, offloading too many tasks for remote computation will lead to longer remote execution times. In this case, utility function can take negative values, if remote task completion time is much longer than local execution.

Similarly, the resource providers have preferences on different mobile users, $\rho_i \in [0, 1]$ (e.g., based on the payments offered by the mobile users). For instance, the resource providers can prioritize the users with higher revenues for offloading by increasing the corresponding preferences. The system utility is defined as $\sum_{i=1}^{|\mathbf{K}|} \rho_i v_i(s_i, f_i, p_i)$, which not only measures the overall utility of mobile users but also considers the interest of the resource provider.

Hence, sharing of remote resources for task offloading becomes a system utility maximization problem. We formulate the problem as follows

$$\mathbf{P} \quad \begin{aligned} &\max_{\mathbf{s},\mathbf{f},\mathbf{p}} \sum_{i=1}^{|\mathbf{K}|} \rho_i v_i(s_i, f_i, p_i) \\ &\text{s.t. } C1 s_i = \{0, 1\} \quad , \forall i \in \mathbf{K} \\ &\quad C2 0 < p_i \leq p_0 \ , \forall i \in \mathbf{S} \\ &\quad C3 f_i > 0 \ , \forall i \in \mathbf{S} \\ &\quad C4 \textstyle\sum_{i \in \mathbf{S}} f_i \leq f_0 \\ &\quad C5 \textstyle\sum_{i \in \mathbf{K}} s_i \leq N; \end{aligned} \tag{3.12}$$

where $\mathbf{s}$, $\mathbf{f}$ and $\mathbf{p}$ are the vectors of offloading decisions, s_i; allocation of computation resources, f_i; and the uplink transmission powers, p_i, respectively. $\mathbf{K}$ denotes the set of all mobile users, and the set $\mathbf{S}$ represents the offloading set, i.e., $\mathbf{S} = \{i \,|s_i = 1\}$. Constraint C1 states that a task can be either executed locally or offloaded. According to Constraint C2, uplink transmission power must be positive, and must not exceed the maximum transmission power p_0. Constraint C3 ensures that all mobile users in $\mathbf{S}$ are assigned computation resources. Constraint C4 guarantees that the total resources assigned are less than the maximum instructions per second allowed at the macro computing center, denoted as f_0. Constraint C5 states that at most $N = B/W$ mobile users are allowed to transmit simultaneously.

3.3 Decomposition-Based Problem Reduction

Offloading decisions are coupled with the optimization of communication and computation resources in the problem $\mathbf{P}$. Moreover, since offloading decision $\mathbf{s}$ is an integer vector and the resource allocations $\mathbf{f}, \mathbf{p}$ are continuous vectors, the problem $\mathbf{P}$ is formulated as mixed integer nonlinear programming (MINLP) [39]. To be able to solve $\mathbf{P}$, we decompose it into two dependent subproblems

1. Joint optimization of communication and computation resources, for a particular offloading decision;

2. Optimization of the offloading decision based on the results of the resource optimization.

Therefore, the problem $\mathbf{P}$ can be rewritten as

$$\begin{aligned} &\max_{\mathbf{S}} v(\mathbf{S}) \\ &\text{s.t. } C5\, |\mathbf{S}| \leq N, \end{aligned} \tag{3.13}$$

where $v(\mathbf{S})$ is the maximum system utility for a particular offloading decision $\mathbf{S}$ when communication and computation resources are jointly optimized, and can be denoted as

$$\begin{aligned} v(\mathbf{S}) = \max_{\mathbf{f},\mathbf{p}} &\sum_{i \in \mathbf{S}} \rho_i v_i(s_i, f_i, p_i) \\ &\text{s.t. } C2, C3 \text{ and } C4. \end{aligned} \tag{3.14}$$

In the rest of the section, we discuss how to solve the first subproblem of resource optimization, and use the results to reduce $\mathbf{P}$ to a submodular maximization problem.

3.3.1 Joint Optimization of Communication and Computation Resources

According to (3.14), the problem of jointly optimizing communication and computation resources for a particular offloading decision can be written as

$$\mathbf{P1} \quad \begin{aligned} &\max_{\mathbf{f},\mathbf{p}} \sum_{i\in\mathbf{S}} \rho_i v_i(s_i, f_i, p_i) \\ &= \sum_{i\in\mathbf{S}} \rho_i \beta_i^T \frac{T_i^l - T_i^r}{T_i^l} + \rho_i \beta_i^E \frac{E_i^l - E_i^r}{E_i^l} \\ &= \sum_{i\in\mathbf{S}} \rho_i (\beta_i^T + \beta_i^E) - \sum_{i\in\mathbf{S}} \rho_i (\beta_i^T T_i^r / T_i^l + \beta_i^E E_i^r / E_i^l) \\ &\text{s.t. } C2, C3 \text{ and } C4. \end{aligned} \tag{3.15}$$

Note that **P1** can be transformed to minimize total overhead in offloaded mobile users. From (4.1)-(3.11), we can obtain the equivalent optimization problem as

$$\mathbf{P2} \quad \begin{aligned} &\min_{\mathbf{f},\mathbf{p}} \sum_{i\in\mathbf{S}} \rho_i (\beta_i^T T_i^r / T_i^l + \beta_i^E E_i^r / E_i^l) \\ &= \sum_{i\in S} (\eta_i + \gamma_i p_i) / \log_2(1 + a_i p_i) + \tau_i F_i^l / f_i \\ &\text{s.t. } C2, C3 \text{ and } C4, \end{aligned} \tag{3.16}$$

where $\eta_i = \frac{\rho_i \beta_i^T D_i}{W \cdot T_i^l}$, $\gamma_i = \frac{\rho_i \beta_i^E D_i}{W E_i^l \zeta}$ and $\tau_i = \rho_i \beta_i^T$ for simplicity.

Note that, in **P2**, the allocation of the uplink transmission power $\mathbf{p}$ and computation resources $\mathbf{f}$ are decoupled from each other in both the objective and the constraints. Thus, **P2** can be solved by optimizing communication and computation resources independently.

1) Optimization of Uplink Transmission Power:

Each mobile user assigns its transmission power solving the following

$$\mathbf{P3} \quad \begin{aligned} &\min_{p_i} f(p_i) \\ &\text{s.t. } 0 < p_i \leq p_0, \end{aligned} \tag{3.17}$$

where

$$f(p_i) = \frac{\eta_i + \gamma_i p_i}{\log_2(1 + a_i p_i)}. \tag{3.18}$$

Note that the second-order derivative of (3.18),

$$f''(p_i) = \frac{a_i/\ln 2}{(1+a_i p_i)^2 \log_2^3(1+a_i p_i)}\{[a_i(\eta_i+\gamma_i p_i) - 2\gamma_i(1+a_i p_i)]\log_2(1+a_i p) + 2a_i(\eta_i+\gamma_i p_i)/\ln 2\}. \tag{3.19}$$

$f''(p_i)$, is not always positive in the domain of $f(p_i)$, and hence, $f(p_i)$ is not convex in the domain.

Lemma 1. *$f(p_i)$ is quasiconvex in the domain.*

Proof. $f(p_i)$ is twice differentiable on $\mathbf{R}$. We next check the quasiconvex second-order condition that the point x_0 satisfying $f'(x_0) = 0$ also satisfies $f''(x_0) \geq 0$ [40].

The first-order derivative of (3.18) is

$$f'(p_i) = \frac{\gamma_i \log_2(1+a_i p_i) - a_i/\ln 2 \frac{\eta_i+\gamma_i p_i}{1+a_i p_i}}{\log_2^2(1+a_i p_i)}. \tag{3.20}$$

$f'(x_0) = 0$, when

$$\phi(x_0) = \gamma_i \log_2(1+a_i x_0) - a_i/\ln 2 \cdot \frac{\eta_i+\gamma_i x_0}{1+a_i x_0} = 0. \tag{3.21}$$

Substituting x_0 into (3.19), we obtain

$$f''(x_0) = \frac{a_i^3}{\gamma_i \ln^2 2} \frac{(\eta_i+\gamma_i x_0)^2}{(1+a_i x_0)^3 \log_2^3(1+a_i x_0)} \geq 0.$$

This concludes the proof of Lemma 1. □

A general approach to quasiconvex optimization problems is the bisection method, solving a convex feasibility problem each time [40]. However, solving convex feasibility problems by an interior cutting plane method requires $O(\frac{m^2}{\varepsilon^2})$ iterations, where m is the dimension of the problem [41]. Note that the first-order derivative of (3.21) is

$$\phi'(p_i) = \frac{{a_i}^2}{\ln 2} \frac{\eta_i+\gamma_i p_i}{(1+a_i p_i)^2} > 0,$$

Algorithm 1 Uplink Transmission Power Bisection

Calculate $\phi(p_0) = \gamma_i \log(1 + a_i p_0) - {a_i}/{\ln 2} \cdot \frac{\eta_i + \gamma_i p_0}{1 + a_i p_0}$
if $\phi(p_0) \leq 0$ **then**
  $p_i^* = p_0$
else
  Initialize $p_s = 0$ and $p_t = p_0$
  repeat
    $p_l = (p_t + p_s)/2$
    if $\phi(p_l) \leq 0$ **then**
      $p_s = p_l$
    else
      $p_t = p_l$
    end if
  until $(p_t - p_s) \leq \varepsilon$
  $p_i^* = (p_t + p_s)/2$
end if

where

$$\phi(0) = -\frac{a_i \eta_i}{\ln 2} < 0.$$

This implies that $\phi(p_i)$ is a monotonically increasing transcendental function, and negative at the starting point $p_i = 0$. Therefore, we develop a low-complexity bisection method, calculating $\phi(p_i)$ instead of solving a convex feasibility problem each time, to obtain the optimal power allocation p_i^* as shown in Algorithm 6. If $\phi(p_0) > 0$, $\lceil \log_2(p_0/\varepsilon) \rceil$ iterations are required before Algorithm 6 terminates [40]. Note that in **P3** each mobile user can pre-calculate the optimal transmission power p_i^* before knowing the offloading decision.

2) Optimization of Computation Resources:

We formulate the problem of optimizing computation resources as

$$\mathbf{P4} \quad \begin{aligned} &\min_{\mathbf{f}} g(\mathbf{f}) \\ &\text{s.t. } C3 \text{ and } C4, \end{aligned} \tag{3.22}$$

where

$$g(\mathbf{f}) = \sum_{i \in \mathbf{S}} {\tau_i F_i^l}/{f_i}. \tag{3.23}$$

Note that the domain of $g(\mathbf{f})$ is convex. The Hessian matrix of (3.23) is composed of elements either $\frac{\partial^2 g}{\partial f_i^2} = \frac{2\tau_i F_i^l}{{f_i}^3} > 0$ or $\frac{\partial g^2}{\partial f_i \partial f_j} = 0 \ \ (i \neq j)$. Hence, the Hessian matrix is positive-definite, and therefore, $g(\mathbf{f})$ is convex [40].

The Lagrangian of **P4** is

$$L(\mathbf{f}, \lambda) = \sum_{i \in \mathbf{S}} \tau_i F_i^l / f_i + \lambda (\sum_{i \in \mathbf{S}} f_i - f_0). \tag{3.24}$$

Note that constraint C3 is slack based on KKT conditions, which has already been eliminated in (3.24) [40]. Thus, we can obtain the dual problem of **P4** as

$$\max_{\lambda > 0} \min_{\mathbf{f} \succ 0} L(\mathbf{f}, \lambda) = \sum_{i \in \mathbf{S}} \tau_i F_i^l / f_i + \lambda (\sum_{i \in \mathbf{S}} f_i - f_0). \tag{3.25}$$

Note that f_i^* satisfies $\frac{\partial L}{\partial f_i}|_{f_i^*} = 0$, and can be obtained as

$$f_i^* = \frac{\sqrt{\tau_i F_i^l}}{\sqrt{\lambda}}. \tag{3.26}$$

Substituting (3.26) into (3.25), we can obtain the Lagrangian dual function of **P4** as

$$\varphi(\lambda) = \min_{\mathbf{f} \succ 0} L(\mathbf{f}, \lambda) = 2\sqrt{\lambda} \sum\nolimits_{i \in \mathbf{S}} \sqrt{\tau_i F_i^l} - \lambda f_0. \tag{3.27}$$

By solving $\varphi'(\lambda^*) = 0$, we obtain the optimal Lagrangian multiplier λ^* as

$$\lambda^* = (\sum\nolimits_{i \in \mathbf{S}} \sqrt{\tau_i F_i^l} / f_0)^2. \tag{3.28}$$

Substituting (3.28) into (3.26), we can obtain the optimal allocation of computation resources as

$${f_i}^* = \frac{\sqrt{\tau_i F_i^l}}{\sum\limits_{i \in \mathbf{S}} \sqrt{\tau_i F_i^l}} f_0. \tag{3.29}$$

Substituting (3.29) into (3.23), the minimum value of $g(\mathbf{f})$ is

$$g(\mathbf{f})^{\min} = \left(\sum_{i \in \mathbf{S}} \sqrt{\tau_i F_i^l} \right)^2 / f_0 \tag{3.30}$$

3.3.2 Submodular Maximization Problem

For a particular offloading decision, the computation and communication resources are optimized as described in Section 3.3.1. Specifically, according to (3.15)-(3.30), the system utility, $v(\mathbf{S})$, can be obtained as

$$v(\mathbf{S}) = \sum_{i \in \mathbf{S}} \rho_i (\beta_i^T + \beta_i^E) - \sum_{i \in \mathbf{S}} f(p_i^*) - g(\mathbf{f})^{\min}, \tag{3.31}$$

where p_i^* can be pre-calculated through Algorithm 6, and $g(\mathbf{f})^{\min}$ can be obtained by the closed-form expression (3.30). Substituting (3.31) into (3.13), $\mathbf{P}$ is reduced to an integer nonlinear programming problem

$$\begin{aligned} \mathbf{P5:} \quad & \max_{\mathbf{S}} \sum_{i \in \mathbf{S}} \rho_i (\beta_i^T + \beta_i^E) - \sum_{i \in \mathbf{S}} f(p_i^*) - g(\mathbf{f})^{\min} \\ & \text{s.t.} \quad |\mathbf{S}| \leq N. \end{aligned} \tag{3.32}$$

Next, we focus on solving the reduction problem **P5** instead of the original MINLP problem $\mathbf{P}$.

Lemma 2. *The system utility $v(\mathbf{S})$ is submodular.*

Proof. The marginal value $\Delta_i v(\mathbf{A})$ denotes the increasing system utility by adding element i into $\mathbf{A}$

$$\begin{aligned} \Delta_i v(\mathbf{A}) &= v(\mathbf{A} \cup \{i\}) - v(\mathbf{A}) \\ &= \rho_i (\beta_i^T + \beta_i^E) - (\Delta(i) + \Delta(i \,|\mathbf{A})), \end{aligned} \tag{3.33}$$

where $\Delta(i)$ and $\Delta(i \,|\mathbf{A})$ represent the constant and variable parts of the marginal value in (3.33). These are

$$\Delta(i) = \frac{\eta_i + \gamma_i p_i^*}{\log_2(1 + a_i p_i^*)} + \frac{\tau_i F_i^l}{f_0}, \tag{3.34}$$

and

$$\Delta(i \,|\mathbf{A}) = \frac{2\sqrt{\tau_i F_i^l} \sum\limits_{j \in \mathbf{A}} \sqrt{\tau_j F_j^l}}{f_0}. \tag{3.35}$$

While $\Delta(i)$ does not depend on the offloading decision set, $\Delta(i\,|\mathbf{A})$ monotonically increases with the offloading decision set $\mathbf{A}$. Hence, according to (3.33), $\Delta_i v(\mathbf{A})$ is a monotonically decreasing function of the current offloading decision $\mathbf{A}$, i.e., for all $\mathbf{A} \subset \mathbf{B}$ and $i \notin \mathbf{B}$, $\Delta_i v(\mathbf{A}) \geq \Delta_i v(\mathbf{B})$ is satisfied. Thus, $v(\mathbf{S})$ is submodular [42]. □

Note that the marginal value $\Delta_i v(\mathbf{A})$ in (3.33) can be negative, and therefore, $v(\mathbf{S})$ is non-monotone, i.e., $\mathbf{A} \supset \mathbf{B}$ cannot imply $v(\mathbf{A}) \geq v(\mathbf{B})$. Besides, $v(\mathbf{S})$ is not symmetric, i.e. $v(\mathbf{A}) \neq v(\mathbf{A}^c)$. Thus, $v(\mathbf{S})$ is an asymmetric non-monotone submodular function. The problem **P5** is an asymmetric non-monotone submodular function maximization problem with cardinality constraint, and is NP-hard according to [42].

Theorem 1. *The problem **P** is NP-hard.*

Proof. The proof is omitted here for simplicity. Please refer to [18] for details. □

3.4 Heuristic Offloading Decision Algorithm

The optimal offloading decision of **P5** can be computed by enumerating and comparing all the possible offloading decisions. However, the complexity of such enumeration is $O(2^K)$. Note that the problem **P** is NP-hard. Therefore, in the rest of this section, we propose the heuristic offloading decision algorithm (HODA) to obtain the local optimum in polynomial complexity.

3.4.1 Locally Optimal Offloading Decision

Definition 1. Offloading decision $\mathbf{S}$ is locally optimal, if and only if neither supersets nor subsets of $\mathbf{S}$ result in higher system utility than $\mathbf{S}$, i.e., for all $\mathbf{A} \subsetneq \mathbf{S}$ or $\mathbf{A} \supsetneq \mathbf{S}$ and $|\mathbf{A}| \leq N$, $v(\mathbf{S}) \geq v(\mathbf{A})$.

Note that determining whether an offloading decision is locally optimal needs comparing all the supersets or subsets with the original decision, which again has a high complexity of $O(2^K)$. Next, we discuss how the offloading decision expands or contracts to increase system utility, and exploit the submodularity of system utility to reduce the complexity of determining locally optimal offloading decision to $O(K)$.

Definition 2. Offloading decision $\mathbf{A}$ is better than offloading decision $\mathbf{B}$, denoted as $\mathbf{A} \triangleright \mathbf{B}$, if and only if the utility of $\mathbf{A}$ is higher than $\mathbf{B}$, i.e., $v(\mathbf{A}) \geq v(\mathbf{B})$. The relation $\triangleright$ is transitive, i.e., $\mathbf{A} \triangleright \mathbf{B}$ and $\mathbf{B} \triangleright \mathbf{C}$ implies $\mathbf{A} \triangleright \mathbf{C}$.

Based on the Definition 2, the offloading decision set can expand or contract while increasing the system utility. Specifically, if $\mathbf{B} \supsetneq \mathbf{A}$, $|\mathbf{B}| \leq N$ and $\mathbf{B} \triangleright \mathbf{A}$, the offloading decision $\mathbf{A}$ can expand to $\mathbf{B}$. Similarly, if $\mathbf{C} \subsetneq \mathbf{A}$ and $\mathbf{C} \triangleright \mathbf{A}$, the offloading decision $\mathbf{A}$ can contract to $\mathbf{C}$.

Definition 3. An offloading decision $\mathbf{A}$ is inextensible, if and only if $|\mathbf{A}| = N$ or there exists no set $\mathbf{B} \supsetneq \mathbf{A}$ satisfying $\mathbf{B} \triangleright \mathbf{A}$. An offloading decision $\mathbf{A}$ is indecomposable, if and only if there exists no subset $\mathbf{C} \subsetneq \mathbf{A}$ satisfying $\mathbf{C} \triangleright \mathbf{A}$.

If an offloading decision is both inextensible and indecomposable, it is also locally optimal. In order to find a locally optimal offloading decision, we can repeat extending the original offloading decision until it cannot be extended, while ensuring the decision is indecomposable at the same time. However, again, determining whether an offloading decision is inextensible or indecomposable also has a complexity of $O(2^K)$. In the following, we prove two lemmas showing that we can reduce this complexity to $O(K)$, when we take into account the submodularity of the utility function.

Lemma 3. *Offloading decision* $\mathbf{A}$*,* $|\mathbf{A}| < N$ *is inextensible, if and only if there exists no element* $i \in \mathbf{A}^c$ *satisfying* $(\mathbf{A} \cup \{i\}) \triangleright \mathbf{A}$*.*

Proof. First, let's assume that $\mathbf{A}$ is inextensible. If there exists an element $i \in \mathbf{A}^c$ satisfying $(\mathbf{A} \cup \{i\}) \triangleright \mathbf{A}$, $\mathbf{A}$ can expand to set $\mathbf{A} \cup \{i\}$ at least, which results in contradiction with $\mathbf{A}$ is inextensible.

Second, let's assume that there does not exist any element $i \in \mathbf{A}^c$ satisfying $(\mathbf{A} \cup \{i\}) \triangleright \mathbf{A}$, i.e., $\Delta_i v(\mathbf{A}) \leq 0$. For any set $\mathbf{I} \supsetneq \mathbf{A}$, let $\mathbf{A} = \mathbf{T}_1 \subset \mathbf{T}_2 \subset \cdots \subset \mathbf{T}_k = \mathbf{I}$ be a chain of sets, where $\mathbf{T}_i - \mathbf{T}_{i-1} = \{e_i\}$. For $2 \leq i \leq k$, due to submodularity, $v(\mathbf{T}_i) - v(\mathbf{T}_{i-1}) = \Delta_{e_i} v(\mathbf{T}_{i-1}) \leq \Delta_{e_i} v(\mathbf{A}) \leq 0$. Summing up all these inequalities, we obtain that $v(\mathbf{I}) - v(\mathbf{A}) \leq 0$, i.e., for all $\mathbf{I} \supsetneq \mathbf{A}$, $\mathbf{A} \triangleright \mathbf{I}$, and therefore, $\mathbf{A}$ is inextensible. □

Lemma 4. *Offloading decision* $\mathbf{A}$ *is indecomposable, if and only if there exists no element* $i \in \mathbf{A}$ *satisfying* $(\mathbf{A} - \{i\}) \triangleright \mathbf{A}$.

Proof. This can be proven similar to Lemma 3, and therefore, omitted for the sake of brevity. □

Lemmas 3 and 4 show that instead of evaluating all the supersets or subsets of the original decision, we can evaluate only adding elements from $\mathbf{A}^c$ or removing elements from $\mathbf{A}$. This has a complexity of $O(K)$. Specifically, if all $i \in \mathbf{A}^c$ satisfy $\mathbf{A} \triangleright (\mathbf{A} \cup \{i\})$, and all $i \in \mathbf{A}$ satisfy $\mathbf{A} \triangleright (\mathbf{A} - \{i\})$, offloading decision $\mathbf{A}$ is locally optimal.

3.4.2 User Classification

In this section, we discuss how to build the initial offloading set. Mobile users vary in terms of the capabilities of their mobile devices, and the current network conditions they experience. These differences may make offloading more beneficial for some users, and local execution for others. For example, a mobile user with better uplink channel condition and lower computation capability will benefit more from offloading.

Next, we define two conditions to determine the users for local execution and offloading, respectively.

Condition 1. If $\Delta_i v(\mathbf{A})^{\max} = \rho_i(\beta_i^T + \beta_i^E) - \Delta(i) \leq 0$, mobile user i executes task locally.

As mentioned in the proof of Lemma 3, $\Delta_i v(\mathbf{A}) > 0$ indicates that $\mathbf{A}$ can expand to $\mathbf{A} \cup \{i\}$. Note that both (3.34) and (3.35) are positive, and $\Delta(i\,|\mathbf{A}) = 0$ if and only if $\mathbf{A} = \emptyset$. Therefore, $\Delta_i v(\mathbf{A})$ in (3.33), i.e., the marginal value when i is added, cannot be positive when Condition 1 is satisfied. Therefore, users that satisfy this condition choose to execute tasks locally. $\mathbf{S}_{local} = \{i\,|\Delta_i v(\mathbf{A})^{\max} \leq 0\}$ denotes the set of mobile users which satisfy Condition 1.

Next, the macro cell checks the following condition to decide whether mobile user i should be offloaded. After determining $\mathbf{S}_{local}$, $\Delta_i v(\mathbf{A})$ gets the minimum value, when $\mathbf{A}$ is the maximum set which does not include i, i.e., $\mathbf{A} = \mathbf{K} - \mathbf{S}_{local} - \{i\}$.

Condition 2. If $\Delta_i v(\mathbf{A})^{\min} = \rho_i(\beta_i^T + \beta_i^E) - (\Delta(i) + \Delta(i\,|\mathbf{K} - \mathbf{S}_{local} - \{i\})) \geq 0$, the macro cell selects mobile user i for offloading.

Note that, if this condition is satisfied, adding mobile user i increases utility, even if $\mathbf{A}$ reached its largest possible size. $\mathbf{S}_{remote} = \{i \,|\Delta_i v(\mathbf{A})^{\min} \geq 0\}$ is the set of mobile users that satisfy Condition 2, and is indecomposable according to Definition 3.

Based on this classification, $\mathbf{S}_{remote}$ should be included in the locally optimal offloading decision set, and $\mathbf{S}_{local}$ must be excluded from this set. Finally, $\mathbf{S}_{search} = \mathbf{K} - \mathbf{S}_{local} - \mathbf{S}_{remote}$ constitutes the remaining mobile users, which cannot be predetermined to run local or remote executions.

3.4.3 Heuristic Offloading Decision Algorithm

Based on the discussions and results in Sections 3.4.1 and 3.4.2, we adopt a semi-distributed offloading approach composed of two stages, shown in Algorithm 7.

1) Algorithmic Process:

In Stage I, mobile users run Algorithm 6 to compute optimal uplink transmission powers p_i^* based on their current conditions and device characteristics (i.e., device computation capacity and power consumption, wireless channel conditions, user preferences, and computation tasks). Also, the users check whether they satisfy Condition 1, deciding the set $\mathbf{S}_{local}$. The mobile users that are not in $\mathbf{S}_{local}$ send their offloading requests to the macro base station, including information about their current conditions and characteristics. The rest send NULL messages to indicate that they will not be offloading.

In Stage II, macro base station waits until it collects all the requests. Based on Condition 2, mobile users are classified into $\mathbf{S}_{remote}$ and $\mathbf{S}_{search}$. The offloading decision set is initialized to $\mathbf{S} = \mathbf{S}_{remote}$. If the initialized offloading users require

Algorithm 2 Heuristic Offloading Decision Algorithm (HODA)

Stage I: at each mobile user i
1: i computes p_i^* (Algorithm6)
2: **if** $i \notin \mathbf{S}_{local}$ $(\Delta_i v(\mathbf{A})^{\max} > 0)$ **then**
3: send an offloading request
4: **else**
5: send NULL
6: **end if**
Stage II: at the macro base station
7: Wait until all the requests are received
8: Determine $\mathbf{S}_{remote}$ and $\mathbf{S}_{search}$ (Condition 2)
9: $\mathbf{S} \leftarrow \mathbf{S}_{remote}$
10: **if** $|\mathbf{S}| \geq N$ **then**
11: **while** $|\mathbf{S}| > N$ **do**
12: $i \leftarrow argmin_{i \in \mathbf{S}}\{v_i(s_i, f_i, p_i)\}$
13: $\mathbf{S} \leftarrow \mathbf{S} - \{i\}$
14: **end while**
15: **else**
16: **repeat**
17: $i \leftarrow argmax_{i \in \mathbf{S}_{search}}\{v_i(s_i, f_i, p_i)\}$ s.t. $\Delta_i v(\mathbf{S}) > 0$ **and** $(\mathbf{S} \cup \{i\})$ is indecomposable
18: $\mathbf{S}_{search} \leftarrow \mathbf{S}_{search} - \{i\}$
19: $\mathbf{S} \leftarrow \mathbf{S} \cup \{i\}$
20: **until** $\mathbf{S}$ is inextensible
21: **end if**
22: Send offloading decision $\mathbf{S}$ to mobile users

more than the available bandwidth, i.e. $|\mathbf{S}| > N$, the macro base station repeats deleting the user of minimum utility in $\mathbf{S}$ until $|\mathbf{S}| = N$. Otherwise, the macro base station heuristically selects the best mobile user in $\mathbf{S}_{search}$, which has the maximum utility, $v_i(s_i, f_i, p_i)$, while ensuring $\Delta_i v(\mathbf{S}) > 0$ and $\mathbf{S} \cup \{i\}$ is indecomposable. The macro BS repeats adding the best user in $\mathbf{S}_{search}$ into $\mathbf{S}$ until $\mathbf{S}$ is inextensible. Then, the decision $\mathbf{S}$ is sent back to the mobile users, and on receiving this information, the mobile users can offload their tasks accordingly.

Theorem 2. *HODA computes the local optimum of problem* $\boldsymbol{P}$.

Proof. As in Sections 3.3 and 3.4.1, the offloading solution is locally optimal in $\mathbf{P}$, if and only if it is both inextensible and indecomposable. HODA initializes the offloading set $\mathbf{S} = \mathbf{S}_{remote}$, which is indecomposable. Next, the offloading set contracts

to the maximum size allowed by constraint C5, or expands to the inextensible set while ensuring it is indecomposable. The result of HODA is both inextensible and indecomposable, and therefore, is locally optimal in $\mathbf{P}$. □

2) Complexity Analysis:

In order to reduce complexity, the proposed approach migrates the computation of p_i^* and $\mathbf{S}_{local}$ to the mobile users. At the macro station, the size of searching users, $\mathbf{S}_{search}$, is further reduced by building the initial offloading set, $\mathbf{S}_{remote}$, through Condition 2. Moreover, Lemmas 3 and 4 reduce the complexity of determining inextensible and indecomposable sets to $O(K)$.

Specifically, at the mobile side, the complexity of Stage I is determined by Algorithm 6. Here, the complexity of the bisection method is $O\left(\log(p_0/\varepsilon)\right)$. At the macro station, the complexity of Stage II is determined by the complexity of building set $\mathbf{S}$, which takes $O(K)$ iterations. If $|\mathbf{S}| \geq N$, selecting the user with maximum utility in $\mathbf{S}$ requires the complexity of $O(K)$. Otherwise, within each iteration, by storing the result of $\sum_{i\in\mathbf{S}} \sqrt{\beta_i^T F_i^l}$, finding the maximum utility user with $\Delta_i v(\mathbf{S}) > 0$ while checking that $\mathbf{S} \cup \{i\}$ remains indecomposable has a complexity $O(K^2)$. Hence, the complexity of Stage II is $O(K^3)$.

3) Further Discussion:

As described in Section 3.4.3, HODA adopts a semi-distributed request-and-decision framework (1) the mobile users sends offloading requests independently in Stage I; (2) the macro computing center decides the offloaded users and pre-allocates the computation and communication resources for offloading in Stage II. Such framework can readily adapt to other scenarios:

1. **Resource multiplexing**: In the general case when computation tasks arrive dynamically, we can execute HODA as an online algorithm to achieve the resource multiplexing in the time dimension. The mobile users can apply personal offloading policies to determine whether to send offloading requests, while the macro computing center always collects the requests and then allows the resource sharing based on the currently available resources. Therefore, the

released communication and computation resources could be reused by the upcoming computation tasks.

2. **Multiple computing centers**: The availability of the computing center, f_0, varies over time due to the stochastic nature of users' offloading requests. Thus, the single macro computing center can become a bottleneck. To address this problem, the resource providers can either increase the capability of the macro computing center or deploy multiple computing centers. In the case of multiple computing centers, the mobile users can apply the offloading policies that make offloading decisions among multiple devices independently (e.g., [9]), and send offloading requests to the selected computing center.

3. **Fairness**: The macro computing center cannot accommodate all the offloading requests from all users, and the users with bad wireless channels may seldom be allowed for offloading. To achieve fairness among users, the resource provider can apply fair scheduling (e.g., proportional fairness) by adjusting its preferences. For instance, the resource provider can increase ρ_i to prioritize user i for offloading.

3.5 Evaluation

We evaluate the efficiency of our heuristic offloading decision algorithm (denoted by HODA) based on Monte Carlo simulations. Since no work jointly optimizes offloading decision together with communication and computation resources, we compare HODA against:

1. **Local execution** There is no offloading. All tasks are always locally executed.

2. **Enumeration algorithm** All 2^K offloading decisions are enumerated and compared to find the global optimum.

3. **All remote joint optimization algorithm (ARJOA)** All tasks are offloaded for remote computing. Then, we add joint optimization of communication and computation resources proposed in Section 3.3.1.

Table 3.1 : Simulation Parameters

Parameters	**Assumptions**
Macrocell Radius	500m
System Bandwidth B	20MHz
UE Bandwidth W	1MHz
Pathloss from Mobile User to Macro BS	$128.1 + 37.5\log_{10}(r)$
Thermal Noise Density	-174dBm/Hz
Max UE Tx Power p_0	23dBm
Lognormal Shadowing Standard Deviation	10dB
Input Data Size D	420kB
Total Number of CPU Cycles C	1000MCycles
Local Computation Capability f^l	0.5GHz−1.5GHz
Mobile Users' Preferences β_i^T, β_i^E	$0.25 - 0.75$
Resource Providers' Preferences ρ_i	1
Maximum Remote Computation Resources f_0	20GHz

4. **Independent offloading and joint optimization algorithm (IOJOA)** The mobile users make offloading decisions independently. Also, the communication and computation resources are jointly optimized as in Section 3.3.1.

Note that, when all the system bandwidth is already occupied, mobile users in ARJOA and IOJOA choose to execute their computation tasks locally, so as to avoid the unexpected latency from both the communication and computation sides.

We simulate 5000 runs in a network composed of a single macrocell with a radius of 500m, where the mobile users are randomly distributed. We set the system bandwidth $B = 20$MHz and UE bandwidth $W = 1$MHz, and therefore $N = 20$. The radio communication parameters follow the 3GPP specification [43]. As an example of a complex application, we adopt the face recognition application in [16, 17, 44], where the input data size, $D = 420$kB, and the required number of CPU cycles, $C = 1000$MCycles. The communication and computation parameters used in our simulations are summarized in Table I.

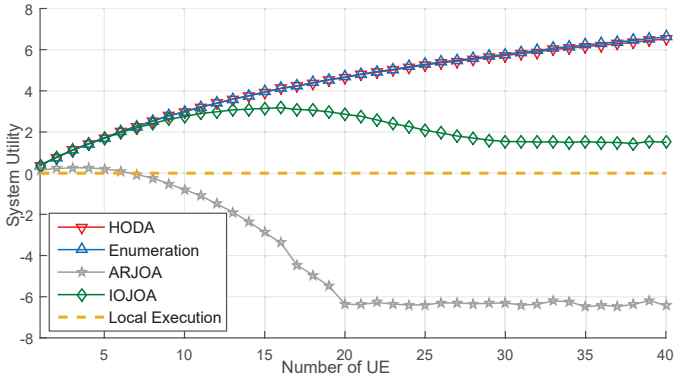


Figure 3.2 : The comparison of system utility as K varies from 1 to 40. HODA performs very close to the optimal solution.

3.5.1 Analysis of System Utility

1) System Utility with Varying Number of Users:

We first evaluate the system utility as the number of users, K, increases from 1 to 40. Fig. 3.2 shows a comparison of the system utility achieved by HODA, local execution, enumeration, IOJOA and ARJOA. In the case of local execution, the system utility is always 0, and therefore, its performance does not change with increasing K. HODA performs very close to the optimal solution computed by the enumeration algorithm. Its performance slightly degrades when K gets larger, and remains within 5% of the optimal. Furthermore, as the number of mobile users increases, both the enumeration algorithm and HODA have a wider choice of mobile users with varying diversity, so that they are able to improve the system utility continuously. However, the performance of IOJOA and ARJOA starts degrading as K gets larger since all users compete for the limited communication and computation resources. With approximately 7 users, ARJOA's performance drops below local execution. IOJOA performs similarly to HODA when $K < 10$, but loses more than 65% system utility when $K > 30$. Since the limited communication resources prevent offloading more tasks, both IOJOA and ARJOA reach the stable system utility as K gets larger. While IOJOA performs better than ARJOA due to selecting better users to offload, and hence, alleviating the competition on resources, it never reaches the performance of HODA or enumeration. This shows that optimizing offloading

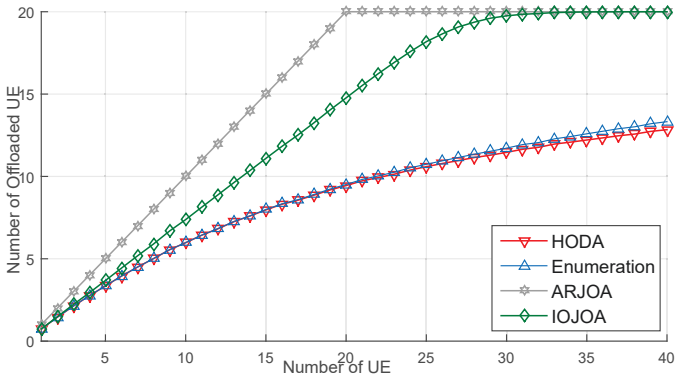


Figure 3.3 : The average number of offloaded users as K increases from 1 to 40. Both HODA and the enumeration algorithm offload fewer users as K increases, improving system utility.

decisions is essential to improve system utility.

2) The Number of Offloaded Users with Varying Number of Users:

Fig. 3.3 shows how many users HODA, IOJOA, ARJOA, and the enumeration algorithm decide to offload as K varies from 1 to 40. As K increases, since ARJOA offloads all the users, the number of offloaded users increases linearly at first. However, due to the constraint of communication resources, ARJOA cannot offload more users when $K > 20$. IOJOA selects fewer better computation tasks to offload, but still faces the bottleneck of communication resources when $K > 30$. In contrast, both the enumeration algorithm and HODA maintain a similar but smaller number of offloaded users, due to the limited computation resources. For HODA as K increases, the size of $\mathbf{S}_{local}$ increases linearly. This is expected as device characteristics are assigned uniformly random in simulation and the mobile users that satisfy Condition 1 execute their tasks locally. As K increases, computation resources become scarcer, and users cannot satisfy Condition 2. As a result, the size of $\mathbf{S}_{remote}$ gets smaller. Hence, $\mathbf{S}_{search}$ becomes larger, providing a wide selection of users to choose from. Compared to the enumeration algorithm, HODA selects slightly fewer users for offloading, as it prioritizes users with higher utility rather than total system utility.

3) The Variation of System Utility:

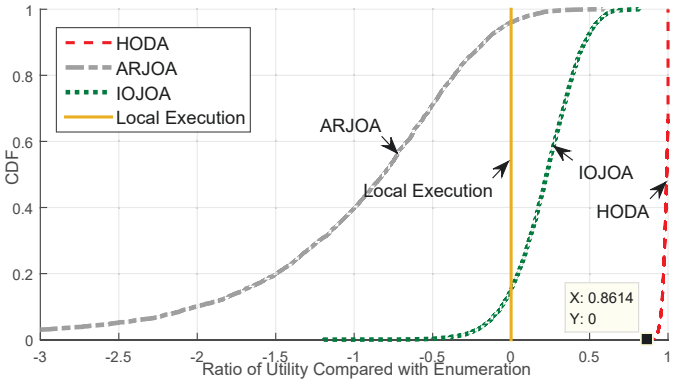


Figure 3.4 : The comparison of CDF of the ratio of the system utility of a solution compared to the enumeration algorithm when $K = 40$. HODA does not exhibit high variations.

Next, we mainly evaluate the scenario of 40 mobile users, as the significant resource bottlenecks show the benefit of joint decision and offloading optimization. To understand how the performance of the different algorithms varies across different simulation runs, we plot the CDF of the ratio of the system utility of a given solution compared to the enumeration algorithm when $K = 40$ in Fig. 3.4. The ratio of system utility is the ratio of the utility of a solution and the optimal utility. Since ARJOA and IOJOA offload more UEs, they have negative system utility, even worse than local execution. HODA performs significantly better in achieving comparable performance to the optimal. Fig. 3.4 shows that HODA does not exhibit high variations, where the worst case performs 86% of the optimal.

3.5.2 Analysis of per-user Metrics

1) The Distribution of User Utility:

Fig. 3.5 shows the CDF of the user utility achieved by HODA, local execution, enumeration, IOJOA and ARJOA when $K = 40$. In local execution, all the users have the same utility of 0 according to (3.11). As in Fig. 3.3, both IOJOA and ARJOA offload 20 computation tasks when $K = 40$, and as expected, have 50% of the users with 0 utility. On the other hand, HODA and enumeration have about 68% of its users with 0 utility, and hence, offload about 32% users. Both algorithms perform similarly, although HODA may achieve better system utility for remote

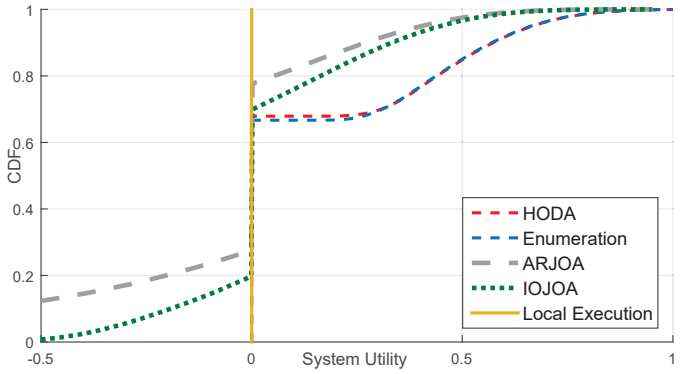


Figure 3.5 : The comparison of CDF of user utility when $K = 40$. All users have non-negative utility with HODA.

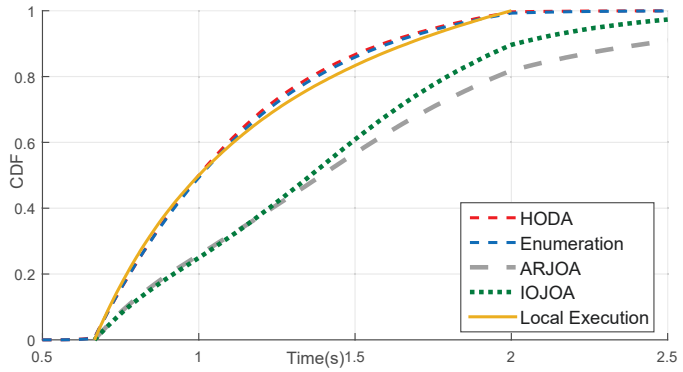


Figure 3.6 : The comparison of CDF of task completion time when $K = 40$. Offloading too many users leads to an increase in task completion time.

users as it prioritizes maximum utility users when building the offloading set. Note that with 40 mobile users, the resources are extremely limited in this scenario, and therefore, in ARJOA, randomly selecting 20 users for offloading leads to 28% of the mobile users achieve a utility lower than local execution. IOJOA also suffers but less drastically, with 20% users achieving negative utility.

2) The Distribution of Task Completion Time:

Fig. 3.6 shows the CDF of task completion time of any user achieved by HODA, local execution, enumeration, IOJOA and ARJOA when $K = 40$. HODA, local execution and enumeration algorithm perform close to each other. For the majority of mobile users, the task completion time of HODA and the enumeration algorithm is only slightly shorter than local execution. On the other hand, as IOJOA and

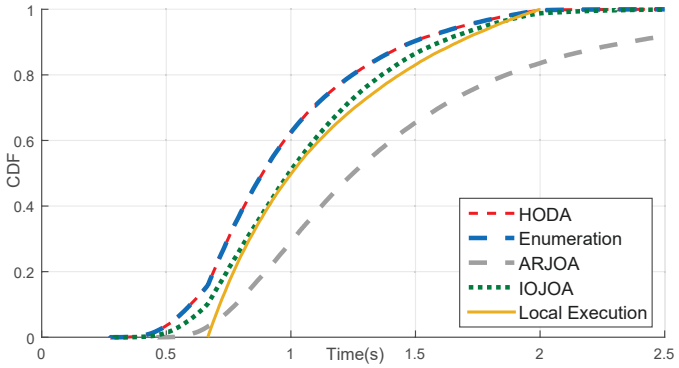


Figure 3.7 : The comparison of CDF of task completion time when $K = 10$. HODA improves task completion time.

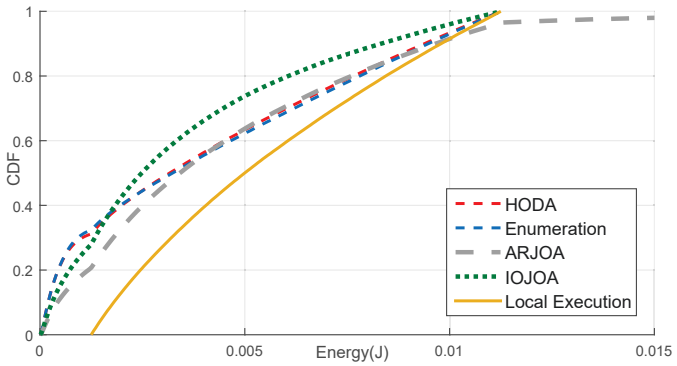


Figure 3.8 : The comparison of CDF of energy consumption when $K = 40$. Selectively offloading tasks improves energy efficiency.

ARJOA offload more users, the task computation times are much longer when the resources are limited. In this case, ARJOA again suffers the worst task completion time. Specifically, the task completion time of the 50% mobile users of HODA is less than 1s, which is 30% improvement over ARJOA.

When we decrease the number of users to $K = 10$ as shown in Fig. 3.7, for a small number of users, both HODA and the enumeration algorithm are able to improve task completion times compared to local execution and IOJOA, which perform similarly. The worst performance is experienced by ARJOA, which offloads all tasks creating a resource bottleneck.

3) The Distribution of Energy Consumption:

Fig. 3.8 shows the CDF of energy consumption achieved by HODA, local execu-

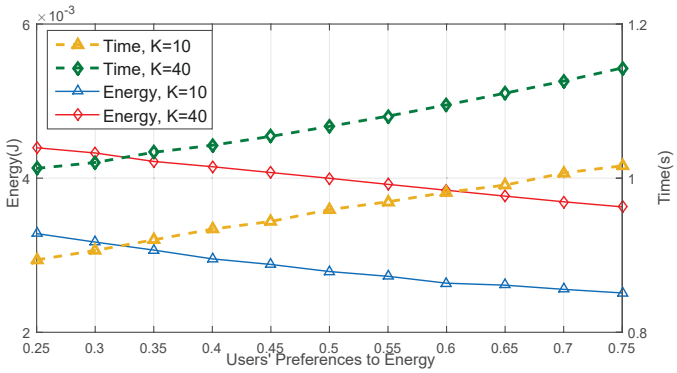


Figure 3.9 : The average latency and energy consumption of HODA as β_i^E =0.25-0.75 with $\beta_i^T = 0.5$ when $K = 10, 40$. Increasing β_i^E reduces the energy consumption at the expense of longer latency.

tion, enumeration, IOJOA and ARJOA when $K = 40$. Since offloading saves energy in general, local execution has the worst performance with the highest energy consumption. The most energy-efficient algorithm is IOJOA as it offloads each user independently, and for these selected tasks remote computation has better energy performance than local computation. The energy consumption of HODA is close to the enumeration algorithm. For the 50% mobile users, the energy consumption of HODA is less than 3.5mJ, saving 30% energy compared with local execution. While IOJOA offloads 50% of its users, compared to the 32% in HODA and enumeration (see Fig. 3.5), IOJOA fails to select users with a better utility to offload, which results in a crossing point among IOJOA, HODA and enumeration in Fig. 3.8.

4) The Analysis of Users' Preferences:

Fig. 3.9 shows the users' average latency and energy consumption of HODA as the users' preferences to energy consumption, β_i^E, vary from 0.25 to 0.75 with $\beta_i^T = 0.5$ when $K = 10, 40$. With fixed $\beta_i^T = 0.5$ and increasing β_i^E, mobile users prefer lower energy consumption. Hence, the users' average energy consumption decreases approximately linearly with β_i^E at the cost of suffering longer latency. In the case of $K = 40$, mobile users compete for limited resources, and have less probability to offload their tasks. Therefore, the users experience more average latency and energy consumption than the case of $K = 10$.

In summary, simulation results show that the limited resources in fog computing necessitate joint optimization of offloading decision, and communication and computation resources. HODA is able to approximate well the performance of the optimal enumeration algorithm in terms of the system utility, task completion time and energy consumption. While the enumeration algorithm always aims to maximize the system utility, HODA serves user preferences better, by prioritizing users with maximum utility in its offloading decisions, and hence, achieves better per-user metrics at the expense of fewer offloaded users.

Chapter 4

Energy-efficient Admission Control of Delay-Sensitive Tasks

This chapter proposes an asymptotically optimal online programming of task admission which can guarantee task delays and achieve $(1-\epsilon)$-approximation of the maximum energy saving at a time-complexity of $\mathcal{O}(NK^2/\epsilon)$ linear to the device number N. (K is the number of subchannels. ϵ is linear to the quantization interval of energy.) The contributions of this chapter are as follows.

- We discover that, after the tasks that cannot catch deadlines by local execution are pre-admitted for offloading, the admission of the remaining tasks is integer programming (IP) with the optimal substructure. By relaxing the energy saving as a continuous variable, the subproblems under the substructure can recursively produce the optimal admission schedule at a polynomial complexity depending on the number of subproblems.
- We optimally discretize the energy saving to holistically control the number of the subproblems, leveraging the optimality loss and complexity at an $[\mathcal{O}(\epsilon), \mathcal{O}(1/\epsilon)]$-tradeoff.
- The proposed approach is efficient to implement. Only part of the devices need to report their information while the asymptotic optimality is unaffected. Other devices can evaluate their energy savings of offloading against local execution, and spontaneously withhold offloading requests if there is no energy saving or the deadlines would be violated.

Evident from extensive simulations, the proposed protocol exhibits an attractive property that the signalling overhead can decrease with the increasing number of devices at no cost of optimality, especially when N is large. This is because the

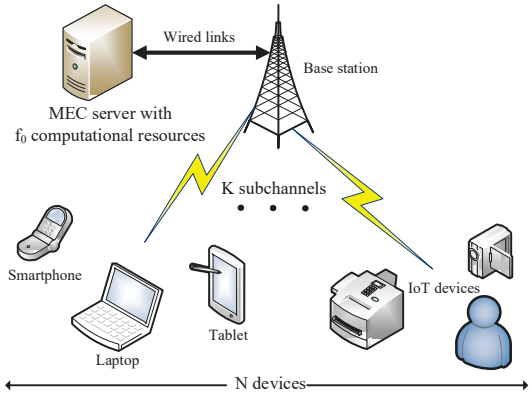


Figure 4.1 : The single-cell multi-user fog computing system with delay-sensitive tasks.

number of devices pre-admitted grows, draining the resources and increasingly preventing other devices from offloading.

4.1 System Model

Fig. 4.1 shows a single-cell multi-user fog computing system, where an LTE Base Station (BS) is physically co-located with an edge server. The connection between the edge server and the BS can be via comparatively delay- and fault-free wired links, such as fiber. With finite computational resources, the edge server serves the tasks offloaded from a number of devices within the coverage of the BS. The tasks can be processed locally at the device, or offloaded to the edge server.

The task of device i to be processed can be defined by a triplet $(D_i, C_i, T_i^{\text{req}})$, where D_i specifies the size of the task in bits, C_i is the number of CPU cycles required to accomplish the task, and T_i^{req} is the task deadline, i.e., the maximum delay that the task can tolerate. (Recall the triplet computation task model in Chapter 2. Here, the required CPU cycles can be given by $C_i = D_i\rho_i$). Assume that every task is atomic, i.e., cannot be further split given strong dependence over different parts of the task [16, 17, 25]. D_i and C_i, acquired by program profilers, depend on the type of task and are consistent for a specific type. The output of a task is generally much smaller than the task size, and can be returned to the device with negligible transmission delay (e.g., face recognition and language processing [16, 17]).

Let F_i^l denote the local computational capability of device i in cycles per second. T_i^l is the time required to perform the task locally at the device, as given by

$$T_i^l = C_i / F_i^l. \tag{4.1}$$

The CPU power consumption is widely modeled to be a super-linear function of F_i^l, as given by [16, 17, 36]

$$P_i^l = \alpha (F_i^l)^\gamma, \tag{4.2}$$

where P_i^l denotes the local power consumption of device i, and α and γ are pre-configured model parameters depending on the chip architecture. Typically, $\alpha = 10^{-11}$ Watt/cycle$^\gamma$ and $2 \leq \gamma \leq 3$ [16, 17, 36].

The energy consumption of device i for *local* computation, denoted by E_i^l, is therefore given by

$$E_i^l = P_i^l \cdot T_i^l = \alpha (F_i^l)^{\gamma - 1} C_i. \tag{4.3}$$

Let f_0 denote the quantity of the available computational resources at the edge server in cycles per second. In the case that device i is admitted for offloading, the edge server allocates f_i computational resources, and performs the task on behalf of the device.

With consideration of typical static IoT networks, we assume that the channels are frequency-flat and each device i feeds back its own channel, denoted by h_i, to the BS infrequently. The BS schedules the devices accordingly. The uplink data rate of device i can be given by

$$R_i = W \log_2 (1 + p_i h_i / N_0), \tag{4.4}$$

where W is the channel bandwidth and N_0 is the noise power. With a limited number of subchannels, the BS can support at most K concurrent data transmissions at the same time. p_i is the uplink transmission power, and can be pre-configured based on the capability and location of the device. Nevertheless, the algorithm developed in this chapter can be extended to more complex scenarios, where the channels are

frequency-selective and can be selected for different devices to save energy further. However, signalling overhead would grow under this channel-aware scheduling, as all the devices need to feed back instantaneous channels to the BS, as will be discussed in Section 4.4.3.

For device i, the total delay of remote task processing, denoted by T_i^r, can be given by

$$T_i^r = T_i^t + T_i^e = D_i/R_i + C_i/f_i, \tag{4.5}$$

where $T_i^t = D_i/R_i$ and $T_i^e = C_i/f_i$ are the uplink transmission time and remote execution time, respectively. Typically, the computational resources allocated to each task at the edge server, f_i, are up to 10^{10} cycles per second [17]. Taking an example of face recognition, the required processing density is 31680 cycles/bit [45]. On the other hand, the uplink data rate R_i for IoT devices is typically less than 250kbps. The processing speed of about $\frac{10^{10}}{31680} \approx 320$ kbps is similar to the uplink rate. The processing delay at the edge server is non-negligible, as compared with the transmission delay.

If admitted for remote processing, the energy consumption of device i, denoted by E_i^r, is given by

$$E_i^r = (p_i/\zeta_i)T_i^t = p_i D_i/\zeta_i R_i, \tag{4.6}$$

where ζ_i is the power amplifier efficiency of device i.

It is the edge server that makes and advises admission decisions based on the availability of its own resources and the requests of all devices. The request that device i sends consists of information on task and device parameters, such as C_i, D_i, T_i^{req}, p_i, T_i^l, and E_i^l.

4.2 Energy-efficient Offloading and Resource Optimization

In this section, we propose to minimize the energy consumption of devices under the latency constraints for delay-sensitive tasks. This can be formulated as an MIP problem. Propositions are developed to reformulate the MIP as IP, which can be solved by a new quantized DP algorithm, named Energy-efficient offloading and

Resource Optimization Scheme (EROS).

4.2.1 Problem Formulation

EROS is designed to minimize the total energy consumption of devices under latency constraints. It may increase energy consumptions at some individual devices which need to give way to others with tighter energy budget or more stringent deadlines, but improve the sustainability of the entire network in the long term. The problem of interest can be formulated as

$$\begin{aligned} \mathbf{P}: \min_{\mathbf{s},\mathbf{f}} & \sum_{i\in\mathbf{N}} s_i E_i^r + (1-s_i)E_i^l, \\ \text{s.t.} \quad & \text{C1: } s_i \in \{0,1\}, \forall i \in \mathbf{N}, \\ & \text{C2: } \textstyle\sum_{i\in\mathbf{N}} s_i \le K, \\ & \text{C3: } \textstyle\sum_{i\in\mathbf{N}} f_i \le f_0, \\ & \text{C4: } T_i \le T_i^{\text{req}}, \ \ \forall i \in \mathbf{N}, \end{aligned} \tag{4.7}$$

where $\mathbf{s}$ and $\mathbf{f}$ are the vectors of admission decisions $s_i \in \{0,1\}$, and allocated computational resources f_i, respectively. Particularly, the task of device i is admitted for offloading if $s_i = 1$, or rejected otherwise; and the edge server allocates f_i to the task. $\mathbf{N}$ denotes the set of active devices; T_i and E_i denote the time and the energy consumption for executing the task of device i, respectively, and can be written as

$$T_i = s_i T_i^r + (1-s_i)T_i^l; \tag{4.8}$$

$$E_i = s_i E_i^r + (1-s_i)E_i^l. \tag{4.9}$$

Here, C1 states that a task can be either executed locally or offloaded for remote processing. C2 specifies the constraint of concurrent offloaded tasks due to limited frequency subchannels. C3 ensures that the total computational resources assigned are no more than the available computational capability. C4 is the latency constraints for task deadlines.

Note that $\mathbf{P}$ is NP-hard MIP [39]. This is because the admission decision $\mathbf{s}$ is binary while the resource allocation decision $\mathbf{f}$ is continuous. The MIP problem is

typically solvable, e.g., by using branch and bound method or exhaustive search, but at prohibitive NP time-complexity. To circumvent this impasse, the following two new propositions are first put forth to decouple $\mathbf{f}$ from C3 and C4 and transform $\mathbf{P}$ to an IP problem which exhibits the optimal substructure of DP.

Property 1. Resource-restrained devices with $C_i/f_i^l > T_i^{\text{req}}$ are pre-admitted for offloading to satisfy task deadlines.

Define $\mathbf{N}_r = \{i \mid T_i^l > T_i^{\text{req}}\}$. Devices in $\mathbf{N}_r$ are too resource-restrained to accomplish tasks on their own before deadline, and hence pre-admitted for remote processing according to Proposition 1. The edge server allocates the minimum computational resources to meet the deadline, as specified in the following.

Remark 1. For a resource-restrained device i, the remote computational resources are pre-allocated, as given by

$$f_i = s_i f_i^{\min} = s_i C_i/(T_i^{\text{req}} - D_i/R_i). \tag{4.10}$$

Proof. For the offloaded task, C4 indicates that $s_i T_i^r \leq T_i^{\text{req}}$, i.e., $s_i(D_i/R_i + C_i/f_i) \leq T_i^{\text{req}}$. Rearranging the inequality, we have

$$f_i \geq s_i f_i^{\min}, \tag{4.11}$$

where $f_i^{\min} = C_i/(T_i^{\text{req}} - D_i/R_i)$ for notational simplicity. Note that the edge server is designed to allocate the minimum computational resources to the offloaded traffic here, since the energy consumption of device i is independent of f_i. □

In the case that $\mathbf{P}$ is feasible, both the computational and transmission resources are sufficient to satisfy the deadlines of all devices. Under this assumption, the remaining computational resources and frequency subchannels available at the edge server are given by

$$\tilde{f}_0 = f_0 - \sum\nolimits_{i \in \mathbf{N}_r} f_i^{\min} \geq 0;\ \tilde{K} = K - N_r \geq 0. \tag{4.12}$$

Property 2. If all the devices are capable of accomplishing tasks on their own, i.e., $T_i^l \leq T_i^{\text{req}}$, $\forall i \in \mathbf{N}$, constraints C3 and C4 are equivalent to C5: $\sum_{i\in\mathbf{N}} s_i f_i^{\min} \leq f_0$.

Proof. Since $T_i^l \leq T_i^{\text{req}}$ is satisfied for any device $i \in \mathbf{N}$ by substituting (4.11) into C3, we can obtain the necessary condition of C3 and C4, as given by

$$f_0 \geq \sum\nolimits_{i\in\mathbf{N}} f_i \geq \sum\nolimits_{i\in\mathbf{N}} s_i f_i^{\min}. \tag{4.13}$$

On the other hand, C5 can also be proved to be the sufficient condition of C3 and C4. If C5 is satisfied, the minimum resources $f_i = s_i f_i^{\min}$ are assigned to the task of device i to fulfill C3 and C4. Therefore, we conclude that if $T_i^l \leq T_i^{\text{req}}$ for $\forall i \in \mathbf{N}$, C3 and C4 are equivalent to C5. □

Proposition 2 reveals that the MIP problem (4.7) can be reformulated to an IP problem by replacing C3 and C4 with C5, on the condition that all devices are capable of local task execution. From Proposition 1, resource-restrained devices in $\mathbf{N}_r$ can be pre-admitted for offloading. Proposition 2 holds for the remaining devices, and therefore the remaining problem can be recast as IP.

Let $\mathbf{N}_u = \mathbf{N} \setminus \mathbf{N}_r = \{1, 2, \ldots, N_u\}$ collect the devices that are capable of accomplishing tasks on their own before deadline, and $\mathbf{s}_u = \{s_i | i \in \mathbf{N}_u\}$ collect the offloading decisions for the devices in $\mathbf{N}_u$. Devices in $\mathbf{N}_u$ satisfy the condition required in Proposition 2. Applying (4.13), the energy minimization problem under the latency constraints can be reformulated as an IP problem, as given by

$$\begin{aligned} \textbf{P1:} \max_{\mathbf{s}_u} & \sum_{i\in\mathbf{N}_u} s_i E_i^s, \\ \text{s.t.} \quad & \text{C1: } s_i \in \{0,1\}, \forall i \in \mathbf{N}_u, \\ & \text{C2: } \textstyle\sum_{i\in\mathbf{N}_u} s_i \leq \tilde{K}, \\ & \text{C5: } \textstyle\sum_{i\in\mathbf{N}_u} s_i f_i^{\min} \leq \tilde{f}_0, \end{aligned} \tag{4.14}$$

where the objective of minimizing the total energy is equivalent to maximizing the energy saving $E_i^s = E_i^l - E_i^r$ through remote processing. Analogous to (4.10), the remote computational resource schedule is $f_i = s_i f_i^{\min}$.

In the case that **P** is infeasible, the available transmission or computational resources at the edge server and devices can by no means satisfy the deadlines of all devices. Some of the resource-restrained devices $i \in \mathbf{N}_r$ that need to offload to meet their deadlines, as specified in Proposition 1, have to be denied for offloading. Then, **P** becomes to select a subset of the resource-restrained devices and maximize energy saving, as given by

$$\begin{aligned} \textbf{P1':} \max_{\mathbf{s}_r} & \sum_{i\in\mathbf{N}_r} s_i E_i^s, \\ \text{s.t. } & \text{C1: } s_i \in \{0,1\}, \forall i \in \mathbf{N}_r, \\ & \text{C2: } \textstyle\sum_{i\in\mathbf{N}_r} s_i \le K, \\ & \text{C5: } \textstyle\sum_{i\in\mathbf{N}_r} s_i f_i^{\min} \le f_0, \end{aligned} \tag{4.15}$$

where $\mathbf{s}_r = \{s_i | i \in \mathbf{N}_r\}$ denotes the admission decisions of devices in $\mathbf{N}_r$. The computational resources can also be allocated based on Remark 1, i.e., $f_i = s_i f_i^{\min}$.

Both inherited from **P**, **P1** and **P1'** account for the feasible and infeasible scenarios, respectively. Given the same structure and the same goal of selecting the most energy-efficient subset of devices to offload, **P1** and **P1'** can be solved by a unified solver (with the only difference of input parameters) which is to be developed in the rest of this section. For illustration convenience, the solver is described against **P1** in the chapter, but can be readily used for **P1'**.

4.2.2 Quantized DP Algorithm

We note that **P1** yields the optimal substructure of DP, and can be partitioned into overlapping subproblems. The solution for **P1** can be efficiently constructed from the solutions for its subproblems by using DP techniques. Specifically, the solution for minimizing the energy of the first i devices can be that either device i offloads its task for remote processing or processes the task locally, given the solutions to the subproblems for the first $(i-1)$ devices.

Let $\phi_i(e,l)$ denote the minimum value of the subproblem defined in (16), i.e., the minimum computational resources for the first i devices while saving e units of energy and offloading $l \in \{0,\ldots,\tilde{K}\}$ tasks:

$$\phi_i(e,l) = \min_{\mathbf{s}_u} \left\{ \textstyle\sum_{j=1}^{i} s_j f_j^{\min} \big| \sum_{j=1}^{i} s_j E_j^s = e, \sum_{j=1}^{i} s_j = l \right\}. \tag{4.16}$$

According to Bellman equation [46], $\phi_i(e,l)$ can be solved recursively based on the results of the preceding subproblems $\phi_{i-1}(e,l)$, as given by

$$\phi_i(e,l) = \min\{\phi_{i-1}(e,l), \phi_{i-1}(e-E_i^s, l-1) + f_i^{\min}\}. \tag{4.17}$$

The solution for $\phi_i(e,l)$ is chosen between local task execution, i.e., $\phi_{i-1}(e,l)$, or task offloading for remote processing, i.e., $\phi_{i-1}(e-E_i^s, l-1) + f_i^{\min}$. The Bellman equation exploits the optimal-substructure property, and reduces the time-complexity by finding the solution from the subproblem of the shortest size [46].

The admission decision for device i in the solution to subproblem $\phi_i(e,l)$, denoted by $s_i(e,l) \in \{0,1\}$, is given by

$$s_i(e,l) = \begin{cases} 1, \text{ if } \phi_i(e,l) = \phi_{i-1}(e-E_i^s, l-1) + f_i^{\min}; \\ 0, \text{ otherwise.} \end{cases} \tag{4.18}$$

We note that e can take 2^i possible discrete values for the i-th ($i = 1, \cdots, N_u$) subproblem in (4.16). It can be computationally prohibitive to enumerate the possible discrete values of e in (4.18), especially in the case N is large. To eliminate the computationally prohibitive enumeration, we propose to first relax e to be a continuous variable ($0 \leq e \leq \overline{E}$) which can be properly initialized so that the final optimal value of e takes one of the 2^{N_u} possible discrete values. $\overline{E} = \sum_{i \in \mathbf{N}_u} \max(E_i^s, 0)$ is the upper bound of energy saving for **P1**. Specifically, we initialize $\phi_i(e,l) = \infty$ except that $\phi_i(0,0) = 0$. If $e = e'$ does not belong to the 2^i possible values, $\sum_{j=1}^{i} E_j^s \neq e'$, subproblem (4.16) is inactive, and $\Phi_i(e',l)$ is not updated and remains ∞. The superordinate subproblems of $\phi_i(e',l)$ are ∞, larger than their counterparts of $\phi_i(e",l)$ with $e"$ taking one of the 2^i possible discrete values. e' cannot be part of the final optimal solution.

Although the computationally prohibitive enumeration is avoided, the continuous relaxation of $e \in [0, \overline{E}]$ would lead to an infinite number of subproblems. Nevertheless, the number of subproblems is much flexible and controllable, as compared to the rigid enumeration. To control the number of subproblems and improve the tractabil-

ity of (4.18), we propose to further discretize the energy saving e, and restrain the number of subproblems to be finite. (We also optimize the quantization interval to holistically leverage the optimality loss and complexity at an $[\mathcal{O}(\epsilon), \mathcal{O}(1/\epsilon)]$-tradeoff, as to be articulated in Sections 4.3 and 4.4.1.) The uniform quantizer of e is given by

$$q_\delta(e) = k, \text{ if } (k-1)\delta < e \leq k\delta, \tag{4.19}$$

where δ is the quantization interval. The energy saving of each device can also be discretized, and let $e_i^s = q_\delta(E_i^s)$ denote the quantized energy saving of device i. As a result, the upper bound of quantized total energy saving for **P1**, denoted by $\widehat{e}$, can be given by

$$\widehat{e} = \lceil \overline{E}/\delta \rceil + \tilde{K} \tag{4.20}$$

where $\lceil \overline{E}/\delta \rceil$ corresponds to the quantized upper bound of total energy saving. Note that, according to (4.19), the proposed quantizer can overestimate the energy saving by no more than δ, i.e., $0 \leq \delta e_i^s - E_i^s < \delta$. Given the constraint of $\tilde{K}$ available subchannels in **P1**, the edge server cannot admit more than $\tilde{K}$ devices for offloading. Thus, the quantization error due to the discretization of e_i^s cannot exceed $\delta\tilde{K}$.

The number of subproblems $\phi_i(e, l)$ is the product of the numbers of devices, N_u, the number of available subchannels, $\tilde{K}$, and the upper bound of quantized energy saving, $\widehat{e}$. There are $N_u\tilde{K}\widehat{e}$ subproblems in total. After solving these subproblems for the total of N_u devices, the optimal solution can be given by

$$e^* = \delta \max_{l=0,\ldots,\tilde{K}} \{e \mid \phi_{N_u}(e, l) \leq \tilde{f}_0\}, \tag{4.21}$$

where e^* is the maximum energy saving by the proposed quantized DP algorithm. Let l^* denote the number of offloaded tasks corresponding to e^*.

Backward induction has been widely used to solve DP problems and can determine a sequence of optimal actions by reasoning backwards [47]. It starts by first assessing the last decision in the optimal solution, i.e., $s_{N_u}(e^*/\delta, l^*)$, and then uses the outcome to determine the second-to-last decision. This continues until the optimal decisions are decided for all the devices. The proposed energy-efficient offloading

Algorithm 3 Energy-efficient offloading and Resource Optimization Scheme (EROS)

Pre-admit Resource-restrained Devices
if $T_i^l > T_i^{\text{req}}$ **then**
  Offload and schedule resources based on *Proposition 1*
end if
Quantized Dynamic Programming
Discretize: $e_i^s = q_\delta(E_i^s)$, $\forall i \in \mathbf{N}_u$
Initialize: $\phi_i(e, l) = \infty$ except that $\phi_i(0, 0) = 0$
for Each device $i = 1$ to N_u **do**
  for $l = 0$ to $\tilde{K}$, $e = 0$ to $\widehat{e}$ **do**
    $\phi_i(e, l) = \min\{\phi_{i-1}(e, l), \phi_{i-1}(e - e_i^s, l - 1) + f_i^{\min}\}$
    Record $s_i(e, l)$ by (4.18)
  end for
end for
Find the optimal solution e^* by (4.21)
Backward Induction
Initialize: $e = e^*/\delta$ and $l = l^*$
for $i = N_u$ down to 1 **do**
  Task admission: $s_i = s_i(e, l)$
  Trace backward: $e = e - s_i E_i^s$, $l = l - s_i$
end for
Task Admission
Devices offload tasks according to the result of s_i
Schedule computational resources: $f_i = s_i f_i^{\min}$

and resource optimization scheme is summarized in Algorithm 3.

4.3 Asymptotically Optimal Quantization Interval

There is a tradeoff between the time-complexity and the optimality loss of Algorithm 3 due to the quantization of energy; see (4.19). Particularly, narrowing the quantization interval δ reduces the optimality loss, but increases the time-complexity. In this section, we quantify the tradeoff by inferring the upper and lower bounds of the solution for **P1**, where the linear programming (LP) relaxation

of **P1** is carried out, as given by

$$\begin{aligned} \textbf{P2:} \max_{\mathbf{s}_u} & \sum_{i\in\mathbf{N}_u} s_i E_i^s, \\ \text{s.t. C2 and C5,} & \\ \text{C6: } & s_i \in [0,1],\ \forall i \in \mathbf{N}_u. \end{aligned} \tag{4.22}$$

Here, the binary constraint C1 is relaxed to be C6. Clearly, **P2** gives the upper bound for **P1** in terms of energy saving.

The Lagrangian problem of **P2** can be written as

$$L(\lambda,\mu) = \max_{\mathbf{s}_u\in\text{C6}} \sum_{i\in\mathbf{N}_u} s_i E_i^s + \lambda(\tilde{K} - \sum_{i\in\mathbf{N}_u} s_i) + \mu(\tilde{f}_0 - \sum_{i\in\mathbf{N}_u} s_i f_i^{\min}). \tag{4.23}$$

The Lagrangian problem $L(\lambda,\mu)$ is separable, and can be restructured as

$$L(\lambda,\mu) = \lambda\tilde{K} + \mu\tilde{f}_0 + \sum\nolimits_{i\in\mathbf{N}_u} L_i(\lambda,\mu), \tag{4.24}$$

where

$$L_i(\lambda,\mu) = \max_{s_i\in[0,1]} s_i(E_i^s - \lambda - \mu f_i^{\min}). \tag{4.25}$$

As a result, the Lagrangian function can be maximized if and only if $L_i(\lambda,\mu)$ is maximized for all $i = 1, 2, \cdots$. The maximization of $L_i(\lambda,\mu)$ can be efficiently solved as

$$s_i^*(\lambda,\mu) = \begin{cases} 1, \text{ if } \theta(i,\lambda,\mu) > 0; \\ 0, \text{ if } \theta(i,\lambda,\mu) < 0, \end{cases} \tag{4.26}$$

where $\theta(i,\lambda,\mu) = E_i^s - \lambda - \mu f_i^{\min}$ for notational simplicity.

Strong duality holds in LP problems [40]. By substituting (4.26) into (4.23), the dual problem of (4.22) can be formulated, as given by

$$\min_{\lambda>0,\mu>0} \lambda\tilde{K} + \mu\tilde{f}_0 + \sum\nolimits_{i\in\mathbf{N}_u} s_i^*(\lambda,\mu)(E_i^s - \lambda - \mu f_i^{\min}), \tag{4.27}$$

where the optimal Lagrangian multipliers λ^* and μ^*, subject to a hyperplane search problem, can be obtained through multi-dimensional search at a linear complexity of $\mathcal{O}(N_u)$ [48].

According to (4.26) and the optimal Lagrangian multipliers, $\mathbf{N}_u$ can be divided into three subsets: $\mathbf{N}_u^+ = \{i \mid \theta(i, \lambda^*, \mu^*) > 0\}$, $\mathbf{N}_u^0 = \{i \mid \theta(i, \lambda^*, \mu^*) = 0\}$, and $\mathbf{N}_u^- = \{i \mid \theta(i, \lambda^*, \mu^*) < 0\}$. From (26), clearly, $s_i = 1$ for $i \in \mathbf{N}_u^+$; and $s_i = 0$ for $i \in \mathbf{N}_u^-$.

For the evaluation of the upper bound of **P1**, among all the devices $i \in \mathbf{N}_u^0 \neq \emptyset$ and $\theta(i, \lambda^*, \mu^*) = 0$, one of the devices $r^0 = \arg\max_{r_0 \in \mathbf{N}_u^0} \{s_{r^0} E_{r^0}^s\}$ with $s_{r^0} = (\tilde{f}_0 - \sum_{j \in \mathbf{N}_u^+} f_j^{\min})/f_{r^0}^{\min} \in (0, 1)$ can be selected to optimize **P2**. This is the case where some resources remain available at the edge server after all devices in $\mathbf{N}_u^+$ are accepted for offloading, but the remaining resources cannot satisfy in whole any unsatisfied offloading request (from the devices in $\mathbf{N}_u^0$). Given the resource, the device which can save the most energy among the unselected devices, i.e., device r^0, is selected to offload part of its task, thereby maximizing the total energy saving in the absence of the binary constraint on $s_j \in \{0, 1\}$. This provides the upper bound for **P1**, as given by

$$e^{\mathrm{LP}} = \sum\nolimits_{i \in \mathbf{N}_u^+} E_i^s + s_{r^0} E_{r^0}^s. \tag{4.28}$$

For the evaluation of the lower bound of **P1**, $s_i = 0$ can be taken for any device $i \in \mathbf{N}_u^0 \neq \emptyset$, and hence the lower bound is given by $e_f = \sum_{i \in \mathbf{N}_u^+} E_i^s$. This is the case where, after the devices in $\mathbf{N}_u^+$ are admitted, the remaining available resources are just wasted. Since this solution is integer but not optimized, it can save no more energy than the optimal solution to **P1**, and hence provides the lower bound.

The relationship among the lower bound, e_f, the optimal solution to **P1**, e^{opt}, and the LP upper bound, e^{LP}, is revealed in the following Lemma.

Lemma 5. $e_f \leq e^{\mathrm{opt}} \leq e^{\mathrm{LP}} \leq 2e_f$.

Proof. Note that e_f is a lower bound for **P1**, while the LP relaxation provides the upper bound e^{LP}. We can obtain that $e_f \leq e^{\mathrm{opt}} \leq e^{\mathrm{LP}}$. Besides, $e^{\mathrm{LP}} = \sum_{i \in \mathbf{N}_u^+} E_i^s + s_{r^0} E_{r^0}^s \leq 2 \max\{\sum_{i \in \mathbf{N}_u^+} E_i^s, E_{r^0}^s\} = 2e_f$. Then, we prove $e_f \leq e^{\mathrm{opt}} \leq e^{\mathrm{LP}} \leq 2e_f$. □

Following Lemma 5, the quantization interval δ can be adjusted to achieve $(1-\varepsilon)$-approximation of the optimum e^{opt} for any $\varepsilon > 0$, as dictated in the following Lemma.

Lemma 6. *The proposed quantized DP algorithm can achieve* $(1-\varepsilon)$*-approximation of the optimum for any* $\varepsilon > 0$*, by setting the quantization interval* $\delta = e_f\varepsilon/\tilde{K}$*.*

Proof. Let $\mathbf{s}_u^{\text{opt}}$ and $\mathbf{s}_u^*$ denote the optimal admission decisions for **P1** and the decision obtained by the proposed quantized DP algorithm, respectively. Let $p(\mathbf{s}) = \sum_i s_i E_i^s$ and $q(\mathbf{s}) = \sum_i s_i e_i^s$ denote the original and quantized energy saving of the admission decision $\mathbf{s}$, respectively. Then, the optimal energy saving can be given by $e^{\text{opt}} = p(\mathbf{s}_u^{\text{opt}})$ and the solution of EROS is $\overline{e^*} = p(\mathbf{s}_u^*)$.

According to (4.19), we have

$$\delta(e_i^s - 1) \leq E_i^s < \delta e_i^s. \tag{4.29}$$

Hence, we can obtain that $p(\mathbf{s}_u^{\text{opt}}) < \delta q(\mathbf{s}_u^{\text{opt}})$ and $p(\mathbf{s}_u^*) \geq \delta[q(\mathbf{s}_u^*) - |\mathbf{s}_u^*|]$, and therefore, we have

$$e^{\text{opt}} - \overline{e^*} = p(\mathbf{s}_u^{\text{opt}}) - p(\mathbf{s}_u^*) < \delta[q(\mathbf{s}_u^{\text{opt}}) + |\mathbf{s}_u^*| - q(\mathbf{s}_u^*)]. \tag{4.30}$$

Note that $q(\mathbf{s}_u^*) \geq q(\mathbf{s}_u^{\text{opt}})$ holds, since the proposed EROS produces the optimal solution for the problem after energy quantization. Thus, we also obtain

$$\delta[q(\mathbf{s}_u^{\text{opt}}) + |\mathbf{s}_u^*| - q(\mathbf{s}_u^*)] \leq (e_f\varepsilon/\tilde{K})\,|\mathbf{s}_u^*| \leq e_f\varepsilon \leq e^{\text{opt}}\varepsilon. \tag{4.31}$$

From (4.30) and (4.31), we have $e^{\text{opt}} - \overline{e^*} < e^{\text{opt}}\varepsilon$. Hence, for any $\varepsilon > 0$, the proposed quantized DP algorithm can achieve $(1-\varepsilon)$-approximation of the optimum, i.e., $\overline{e^*} > (1-\varepsilon)e^{\text{opt}}$. □

4.4 Discussions and Extensions

In the proposed EROS, devices send offloading requests to the edge server, and offload their tasks according to the result of task admission. Signalling is streamlined, and reduced. In this section, we analyze the complexity, discuss fairness and task partitioning of EROS, extend the proposed EROS to frequency-selective subchannels, and illustrate the impact of inter-cell interference.

4.4.1 Complexity Analysis

The following Lemma exhibits the tradeoff between the performance and time-complexity of the proposed EROS.

Lemma 7. *EROS is able to achieve* $(1-\varepsilon)$*-approximation of the optimum at the complexity of* $\mathcal{O}(NK^2/\varepsilon)$.

Proof. Recall that N_u and $\tilde{K}$ denote the numbers of remaining devices and sub-channels after pre-admission by Proposition 1, respectively. The time-complexity of EROS depends on the number of subproblems to be solved. As mentioned in Section 4.2.2, the number of subproblems is $N_u\tilde{K}\widehat{e}$, where the time-complexity for each subproblem using (4.17) is $\mathcal{O}(1)$. The time-complexity of backward induction is $\mathcal{O}(N_u)$ [47]. Thus, the overall time-complexity of EROS is $\mathcal{O}(N_u\tilde{K}\widehat{e})$.

We can further tighten the upper bound of quantized energy saving in (4.20) by replacing $\overline{E}$ with the LP upper bound, e^{LP}, as given by $\widehat{e} = \lceil e^{\mathrm{LP}}/\delta \rceil + \tilde{K}$. Therefore, we have

$$\mathcal{O}(N_u\tilde{K}\widehat{e}) = \mathcal{O}(N_u\tilde{K}^2 + N_u\tilde{K}e^{\mathrm{LP}}/\delta). \tag{4.32}$$

From Lemma 6, we show that $(1-\varepsilon)$-approximation of the optimum can be achieved by using $\delta = e_f\varepsilon/\tilde{K}$. By substituting this into (4.32), the time-complexity of EROS is $\mathcal{O}(N_u\tilde{K}^2 + (e^{\mathrm{LP}}/e_f)N_u\tilde{K}^2/\varepsilon) = \mathcal{O}(N_u\tilde{K}^2 + N_u\tilde{K}^2/\varepsilon) = \mathcal{O}(N_u\tilde{K}^2/\varepsilon)$. Since the pre-admission using Proposition 1 reduces the number of devices to be assessed, the complexity of EROS is $\mathcal{O}(NK^2/\varepsilon)$. □

Additional measures can be taken to further reduce the complexity and overhead. Upon the receipt of $\tilde{f}_0$, each device in $\mathbf{N}_u$ can check the following condition to determine whether to send offloading requests.

Property 3. If $(T_i^r)_{\min} = T_i^t + C_i/\tilde{f}_0 > T_i^{\mathrm{req}}$ or $E_i^r \geq E_i^l$, device i executes its task locally.

Proposition 3 describes two cases. In the first case, even allocating all the remaining resources $\tilde{f}_0$ at the edge server to device i cannot satisfy the task deadline,

i.e., $(T_i^r)_{\min} = T_i^t + C_i/\tilde{f}_0 > T_i^{\text{req}}$. In the second case, offloading would not save energy for the device, i.e., $E_i^r \geq E_i^l$. In both cases, the device is pre-denied and chooses to execute its task locally.

Based on Propositions 1 and 3, signalling can be reduced and streamlined while the asymptotic optimality of the system is preserved. Particularly, devices with limited resources and tight deadlines are given priority to send offloading requests to, and get satisfied by, the edge server; see Proposition 1. The edge server then broadcasts the remaining resources, based on which devices can spontaneously decide to process tasks locally and withhold requests, if the remaining resources is insufficient to meet their deadlines; see Proposition 3. Only the rest of the devices send offloading requests to the edge server which (i.e., the server) conducts the proposed quantized DP algorithm to admit devices and allocate resources accordingly.

Lemma 7 dictates an $[\mathcal{O}(\varepsilon), \mathcal{O}(1/\varepsilon)]$-tradeoff between the optimality loss and time-complexity of the proposed quantized DP algorithm running at the edge server. This gives the edge server an opportunity to reduce the energy consumption of the network by leveraging its hardware capability. In practice, an edge server can choose the smallest ϵ value based on its capability, thereby attaining the minimum achievable energy consumption of the system.

4.4.2 Fairness and Task Partitioning

The proposed approach can be readily extended to provide fairness between devices in terms of energy saving in the long term. By exploiting the idea of proportional fairness [49], a coefficient $\frac{1}{\kappa_i}$ can be multiplied to the energy consumption of each device in the objective of $\mathbf{P}$, i.e., $\min_{\mathbf{s},\mathbf{f}} \sum_{i \in \mathbf{N}} \frac{1}{\kappa_i}[s_i E_i^r + (1 - s_i) E_i^l]$. κ_i is the time-average of the past energy saving of device i. Device i with small κ_i is given priority to offload tasks, achieving fairness in the long term. At every instant, the optimal substructure of Bellman equation can be preserved, and the reformulated problem can be readily solved by using the proposed quantized DP algorithm.

In a different context of task partitioning, a task can be partitioned into atomic subtasks (e.g., in a tree structure). Some of the atomic subtasks can be offloaded,

and offloading needs to be holistically designed to ensure the consistency (i.e., the correct order) of task processing. However, existing studies, such as [5–9], implicitly assumed unlimited computational and transmission capabilities, and did not schedule between multiple devices. To this end, there is no comparable algorithm to the algorithm proposed in this chapter.

In our earlier work [50], we partitioned a delay-sensitive task of a single device to be processed against limited resources in the most energy-efficient manner. This has the potential to be implemented in conjunction with the proposed algorithm to support task partitioning. In specific, each device can partition its own task into atomic subtasks, and specify the deadlines of the subtasks. By using the partitioning technique in [50], the deadlines can be designed to guarantee the integrity of task processing, given available resources. The proposed quantized DP algorithm can be used to asymptotically optimally schedule the atomic subtasks and assign resources. Given the inherent independence between task partitioning and scheduling in this design, we can still separately focus on the scheduling approach developed in this chapter. More closely coupled designs of task partitioning and scheduling are non-trivial, have yet to be developed, and will be our future work.

4.4.3 Channel-aware Scheduling

The proposed algorithm has the potential to be extended to more complex scenarios, where the channels are frequency-selective, instantaneously measured at the devices, fed back to the BS, and selected for different devices to further reduce energy consumption.

Define $s_i \in \{0, \cdots, K\}$ as such that device i locally executes its task if $s_i = 0$, or offload the task to the edge server via subchannel s_i. The constraints C1 and C2 of problem $\mathbf{P}$ can be accordingly reformulated as

$$\begin{aligned} &\text{C1':}\ s_i \in \{0, \cdots, K\},\ \forall i, \\ &\text{C2':}\ s_i \neq s_j,\ \forall s_i \neq 0,\ i \neq j; \end{aligned} \tag{4.33}$$

where C2' indicates that a subchannel must be exclusively occupied by a single

device at a time.

The changes of the constraints do not affect Proposition 1, where the resource-restrained devices $\mathbf{N}_r$ are pre-admitted for offloading. We note that the minimum computational resources (i.e., $f_i^{\min}$) in Proposition 2 now depend on the subchannel that device i is allocated, and needs to be updated to $f_{i,l}^{\min}$, i.e., the minimum computational resources of device i using the specific subchannel l:

$$f_{i,l}^{\min} = C_i/(T_i^{\text{req}} - D_i/R_i^l), \tag{4.34}$$

where R_i^l is the transmit rate of device i at subchannel l. C5 in problem **P1** can be updated by

$$\text{C5':}\ \sum\nolimits_{i\in\mathbf{N}} s_i f_{i,s_i}^{\min} \le f_0. \tag{4.35}$$

Unlike **P1**, the channel allocation for the pre-admitted devices $\mathbf{N}_r$ are now coupled with the admission and channel allocation for the remaining undetermined devices $\mathbf{N}_u$. Given frequency-selective channels, the problem of interest **P3** becomes

$$\begin{aligned} \textbf{P3:}\ & \max_{\mathbf{s}} \textstyle\sum_{i\in\mathbf{N}} s_i E_{i,s_i}^s, \\ & \text{s.t. } s_i \neq 0,\ i \in \mathbf{N}_r, \\ & \quad \text{C1', C2' and C5'}, \end{aligned} \tag{4.36}$$

where E_{i,s_i}^s denotes the energy saving of device i by offloading its task through subchannel s_i. The channels of the pre-admitted devices in $\mathbf{N}_r$ are selected together with the other devices, and the admission of $\mathbf{N}_r$ is guaranteed by enforcing the constraint $s_i \neq 0$, $i \in \mathbf{N}_r$.

Let $\mathbf{O} = \{o_1, \cdots, o_K\}$ stand for the channel occupation status of the K subchannels and $o_i \in \{0, 1\}$, i.e., subchannel i is unoccupied if $o_i = 0$, or occupied otherwise. Define $\phi_i(e, \mathbf{O})$ as the minimum computational resources for the first i devices, while e units of energy are saved and the subchannel status is $\mathbf{O}$. We notice that the optimal substructure still holds by replacing the original subproblem $\phi_i(e, l)$ with $\phi_i(e, \mathbf{O})$ in problem **P3**.

According to Bellman equation [46], $\phi_i(e, \mathbf{O})$ can be presented in a recurrence expression based on the results of the preceding subproblems $\phi_{i-1}(e, \mathbf{O})$, as given by

$$\phi_i(e, \mathbf{O}) = \min \left\{ \begin{array}{l} \phi_{i-1}(e, \mathbf{O}), \text{ if } i \notin \mathbf{N}_r \\ \phi_{i-1}(e - E^s_{i,l}, \mathbf{O}(l) = 0) + f^{\min}_{i,l}; \forall o_l = 1 \end{array} \right\}, \tag{4.37}$$

where $\mathbf{O}(l) = 0$ stands for the vector $\mathbf{O}$ with $o_l = 0$. Also, the channel allocation can be recorded by

$$s_i(e, \mathbf{O}) = \left\{ \begin{array}{l} l, \text{ if } \phi_{i-1}(e, \mathbf{O}) = \phi_{i-1}(e - E^s_{i,l}, \mathbf{O}(l) = 0) + f^{\min}_{i,l}; \\ 0, \text{ otherwise.} \end{array} \right. \tag{4.38}$$

The maximum energy saving is $e^* = \delta \max_l \{e \mid \phi_N(e, \mathbf{O}) \leq f_0\}$, and the channel-aware scheduling can also be obtained through backward induction, as done in Algorithm 1.

However, signalling overhead would grow under the channel-aware scheduling, as all the devices need to feed back their instantaneous channels so that the BS can schedule the devices in the channel-aware manner. Moreover, the computational complexity would also grow, since the channel selection is integer programming coupled with the device selection in a multiplicative manner in the proposed algorithm. The number of subproblems grows from $NK\widehat{e}$ subproblems $\phi_i(e, l)$ in frequency-flat channels to $N2^K\widehat{e}$ subproblems $\phi_i(e, \mathbf{O})$ in frequency-selective channels. Consequently, the time-complexity grows from $\mathcal{O}(NK^2/\epsilon)$ to $\mathcal{O}(NK2^K/\epsilon)$.

4.4.4 The Impact of Inter-cell Interference

The proposed asymptotically optimal approach can be applied in the presence of inter-cell interference. Interference coordination and fractional frequency reuse are typically adopted for inter-cell interference mitigation [51]. Based on the Central Limit Theorem, the residual inter-cell interference can be modeled to be Gaussian, given a typically large number of transmitters beyond the cell of interest [52]. (4.4) can be rewritten as $R^l_i = W \log_2(1 + \frac{p_i h^l_i}{N_0 + N^l_I})$ to account for the Gaussian interference per subchannel, where R^l_i and h^l_i are the achievable data rate and the channel gain of device i in subchannel l, and N^l_I is the interference power measured at the BS in the subchannel. In the case of a flat-fading channel with i.i.d. Gaussian

interference per subchannel, the resultant problem involving R_i^l (not R_i) fully complies with the original interference-free setting, and can be solved by running the proposed algorithm, i.e., Algorithm 1. In the case of a frequency-selective channel with independent and non-identically distributed (i.n.d.) Gaussian interference per subchannel, the resultant problem fully complies with the channel-aware variation of the original interference-free setting, and can also be readily solved by using the proposed approach in the same way as discussed in Section 4.4.3.

4.5 Simulation Results

In this section, Monte Carlo simulations are carried out to evaluate the proposed algorithm. The number of subchannels is $K = 20$, and the bandwidth per subchannel is $W = 180$kHz. The total number of devices N is up to 25; unless otherwise specified. These devices are uniformly distributed in a cell with a radius of 250m. Consider complex applications like face recognition [45], of which the input data size is $D = 85$kB and the required number of CPU cycles is $C = 1000$ million cycles. The local computational capability F_i^l is uniformly distributed within $[0.5, 1.5]$ GHz. Other parameters used in the simulations are summarized in Table 4.1 [43]. 5000 independent runs are conducted for each data point.

Apart from the proposed task admission algorithm (i.e., EROS), we also simulate the following algorithms for comparison purpose.

1. **Branch and Bound (B&B):** After reformulating the MIP problem P to the IP problem **P1**, we take the B&B algorithm [53] to achieve the optimal solutions for **P**. B&B is a classical and popular solver for discrete and combinatorial optimization problems, conducting a structured enumeration of candidate solutions following a tree structure [53]. In the case of **P1**, the B&B method can be set up to assess s_j $(j = 1, \cdots, N_u)$, following a binary tree. When assessing a particular s_j, the upper and lower bounds of **P1** are achieved by taking the binary values of s_i, $i = 1, \cdots, j$, from each of the remaining branches, relaxing s_k to be continuous within $[0, 1]$ for $k = j + 1, \cdots, N_u$, and solving the relaxed problems with linear programming. The branches with upper bounds lower

Table 4.1 : Simulation Parameters

Parameters	**Assumptions**
Macrocell Radius	250m
Number of Subchannels	20
Bandwidth per Subchannel	180kHz
Pathloss from Device to Macro BS	$128.1 + 37.5\log_{10}(r)$
Thermal Noise Density	-174dBm/Hz
Transmit Power	23dBm
Lognormal Shadowing Standard Deviation	10dB
Input Data Size D	85kB
Required Number of CPU Cycles C	1000Million cycles
Local Computational Capability F_i^l	0.5GHz−1.5GHz
Latency Request T^{req}	$\{1, 1.5\}$s
Remote Computational Capability f_0	15GHz

than the lower bound of some other branches are removed – bounding, since those branches offer no prospect of the optimal solution. By this means, the optimal solution can be achieved by enumerating a potentially reduced number of candidate solutions. Unfortunately, in the worst-case scenario where no branches can be removed until the final stage of assessing the last layer, all candidate solutions are enumerated from the root along every branch of the binary tree. As a consequence, the worst-case complexity of the B&B method can be as high as exhaustive search.

2. **All Request Admission Algorithm (ARAA):** The edge server accepts all the offloading requests, and computational resources are equally allocated. In the case that the wireless bandwidth is insufficient to admit all the devices, K out of the devices are randomly admitted for remote processing.

3. **Local execution (Local):** There is no offloading. All tasks are executed locally.

We have also simulated the relaxed version of EROS which allows partial offloading

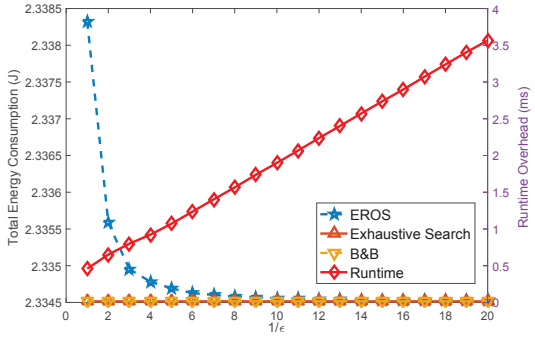


Figure 4.2 : The $[\mathcal{O}(\varepsilon), \mathcal{O}(1/\varepsilon)]$-tradeoff between the optimality loss and time-complexity.

by dropping the binary constraint C1, where a task can be arbitrarily partitioned; i.e., data-partition task model. This is due to the fact that there is no other comparable partial offloading approach, as mentioned in Section 4.4.2. Unfortunately, the relaxed algorithm supporting partial offloading, referred to as "Partial", can violate the integrity of tasks that cannot be partitioned in many cases. Moreover, the relaxed algorithm is less challenging than EROS. Without the binary constraint C1, partial offloading can be straightforwardly implemented by using the standard Matlab linear programming toolbox. The energy saving of partial offloading can also be substantially overestimated, especially in the case of large tasks, tight deadlines, or limited resources, as will be shown in Figs. 4.3 and 4.4.

Fig. 4.2 shows the tradeoff between the optimality loss and time-complexity of the proposed EROS, as $1/\varepsilon$ varies from 1 to 20. Here, $N = 20$ and $T^{\text{req}} = 1$s. We see that the time-complexity of EROS grows linearly with $1/\varepsilon$, while the total energy consumption of devices decreases with $1/\varepsilon$ and approaches to the optimum achieved by exhaustive search. The $[\mathcal{O}(\varepsilon), \mathcal{O}(1/\varepsilon)]$-tradeoff is confirmed, as stated in Lemma 3. B&B achieves the minimum energy consumption, as can be validated by comparing to the results of exhaustive search. EROS achieves nearly the minimum energy consumption achieved by B&B when $1/\varepsilon \geq 10$. For this reason, in the following simulations, we set $(1 - \varepsilon) = 0.9$, i.e., $1/\varepsilon = 10$. It is worth mentioning that both 2.3385J and 2.3345J are the energy consumptions under different values of $\epsilon = 1$ and $\epsilon = 0.05$. Their difference is not energy saving.

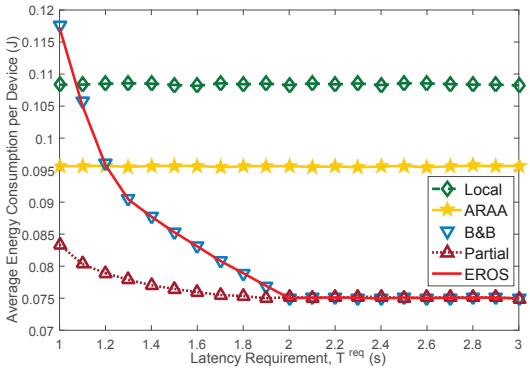


Figure 4.3 : The comparison of devices' average energy consumption as T^{req} increases from 1s to 3s, where $N = 20$.

Fig. 4.3 shows the average energy consumptions of EROS, Partial, B&B, ARAA and Local, where $N = 20$ and T^{req} ranges from 1s to 3s. The energy savings of the proposed EROS compared with the existing solutions are evaluated. With the growth of T^{req}, the energy consumptions of EROS, Partial, and B&B first decrease and then stabilize at 0.075 Joule per device when $T^{\text{req}} \geq 2$s. This is because increasingly relaxed deadlines allow a growing number of devices to be selected for offloading, thereby increasingly exploiting diversity and reducing energy consumption. EROS and B&B can save up to 31% of energy as compared with Local. Compared with EROS and B&B, Partial could dramatically save 28% more energy, since it allows part of the most energy-efficient tasks to be offloaded, but violates the integrity of the tasks. When the deadlines are so loose that almost all devices can be selected for offloading, the energy consumption stops decreasing and stabilizes. The loose deadlines can also help eliminate the difference between atomic tasks and partial offloading, since tasks can be offloaded in whole and integrity does not degrade.

Fig. 4.4 shows the number of devices with satisfied deadlines achieved by EROS, Partial, B&B, ARAA and Local, as f_0 increases from 10GHz to 30GHz, where $N = 20$ and $T^{\text{req}} = 1$s. EROS and B&B can satisfy the deadlines of all devices when $f_0 \geq 17$ GHz. However, when $f_0 < 17$GHz, the numbers of devices satisfied by EROS and B&B slightly drop to 17 as f_0 decreases to 10GHz. This is the infeasible scenario of (4.15) due to the insufficient computational and transmission resources.

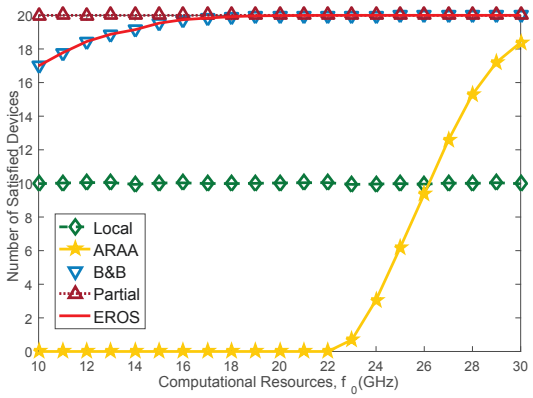
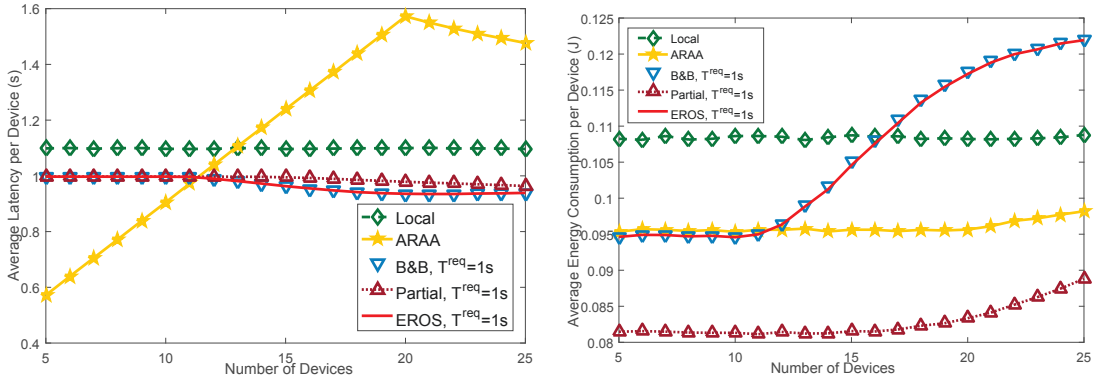


Figure 4.4 : The comparison of the number of satisfied devices as f_0 increases from 10GHz to 30GHz, where $N = 20$ and $T^{\text{req}} = 1$s.



(a) Average latency (b) Average energy consumption

Figure 4.5 : The comparison of devices' average latency and energy consumption where $T^{\text{req}} = 1$s. EROS and B&B increase energy consumptions to satisfy the stringent deadline.

On the other hand, Partial can satisfy all deadlines irrespective of f_0, by overlooking task integrity and continuously leveraging both local and remote computational resources. In contrast, ARAA cannot satisfy any deadline when $f_0 \leq 22$GHz, and satisfies only up to 18 devices when $f_0 = 30$GHz. Local can always satisfy only half of the deadlines due to the uniform distribution of local computational capacity.

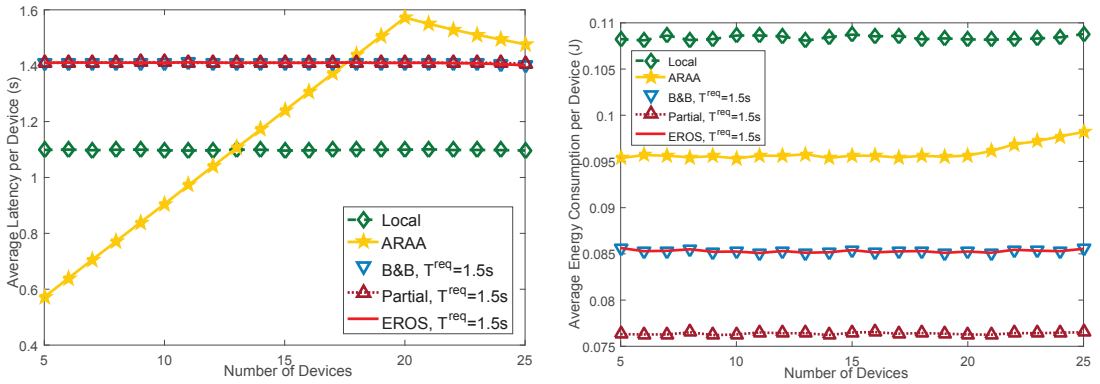
Fig. 4.5 compares the average latency and energy consumption between EROS, Partial, B&B, ARAA and Local, where $T^{\text{req}} = 1$s. This is the case where the deadlines are stringent, and effective task admission is critical to meet the deadlines by leveraging local and remote computational capabilities. We see in Fig. 4.5

that EROS, Partial, and B&B can either save energy or reduce latency, compared with ARAA and Local. In Fig. 4.5(a), EROS and B&B provide nearly identical latency, both satisfying the deadline. Partial can also meet the deadline. The other algorithms all violate the deadline $T^{\text{req}} = 1$. The average latency of ARAA grows linearly with N when N is small to medium (i.e., $N \leq 20$); and declines when N is large (i.e., $N > 20$), as the result of the increasing number of devices executing tasks locally with the average latency of 1.1s (c.p., 1.6s for remote processing). ARAA also violates $T^{\text{req}} = 1$ for $N > 13$.

In Fig. 4.5(b), we see that the proposed EROS and B&B provide indistinguishably close energy consumptions, validating the asymptotic optimality of EROS. We also see the energy consumptions of EROS and B&B are low and stable when N is small, i.e., $N < 12$, and grow as N increases from 12 to 20. The growths of energy consumptions slow down, when N is large, i.e., $N > 20$. This is because EROS and B&B exploit the increasing diversity pertaining to the growing number of devices, thereby saving more energy. As shown in Fig. 4.3, Partial could substantially save energy by breaching task integrity, but would undergo the increase of energy consumption when $N \geq 17$ as in EROS and B&B. In contrast, the energy consumptions of ARAA and Local are relatively stable, and can be lower than that of EROS and B&B when N is large, at the cost of unsatisfied deadlines, as discussed in Fig. 4.5(a).

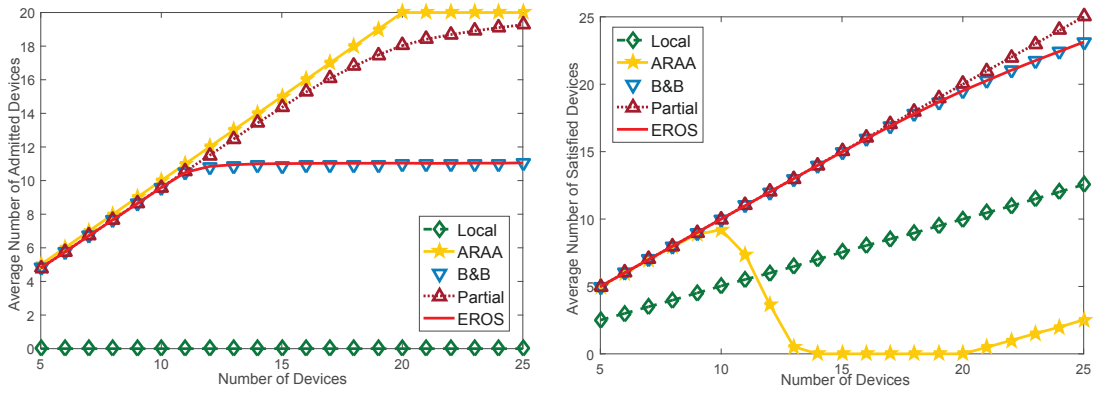
Fig. 4.6 plots the average latency and energy consumption of the algorithms where $T^{\text{req}} = 1.5$s. This is the case where the deadlines are relatively loose. We can see that EROS, Partial, and B&B can substantially save energy consumptions given the loose deadline. Their energy consumptions are much lower than ARAA and Local, since ARAA and Local admit devices independently of deadlines.

Fig. 4.7 shows the numbers of admitted and satisfied devices under EROS, Partial, B&B, ARAA and Local, where $T^{\text{req}} = 1$s. We see that B&B and EROS admit at most 11 devices for offloading, while satisfying the deadlines of almost all tasks. This is because the limited resources at the edge server cannot accommodate more tasks concurrently under the stringent deadlines. Partial can admit more



(a) Average latency (b) Average energy consumption

Figure 4.6 : The comparison of devices' average latency and energy consumption where $T^{\text{req}} = 1.5$s. EROS and B&B can substantially save energy.



(a) Admitted devices (b) Satisfied devices

Figure 4.7 : The comparison of the numbers of admitted and satisfied devices where $T^{\text{req}} = 1$s. Without task admission, ARAA satisfies no task deadlines.

devices for offloading than EROS and B&B, and satisfy all deadlines, but the number of offloaded devices is still less than $K = 20$. This is because partial offloading must not violate the physical constraint of K subchannels. In contrast, Local does not admit any devices, as shown in Fig. 4.7(a). The number of satisfied devices is proportional to the probability that a device can satisfy its deadline locally, and therefore grows linearly with the number of devices, as shown in Fig. 4.7(b). ARAA always admits as many devices as possible without consideration on deadlines, as shown in Fig. 4.7(a). As a consequence, the satisfaction of the devices degrades rapidly, as shown in Fig. 4.7(b). Only in the case that $N > 20$, a small number of

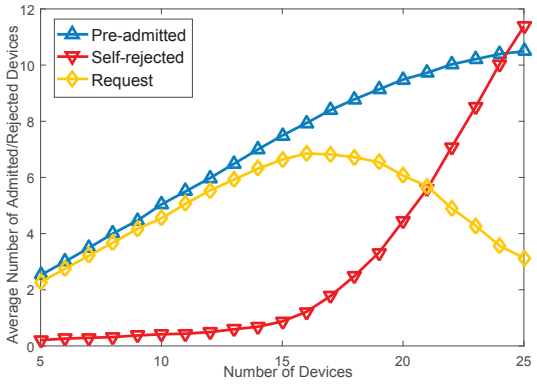


Figure 4.8 : The comparison of the numbers of pre-admitted and pre-denied devices where $T^{\text{req}} = 1$s. The number of devices to be assessed is consistent with the result in Fig. 4.9.

devices that are not admitted may satisfy their deadlines through local processing. The increase of such devices is proportional to the probability that a device can satisfy its deadline locally, and therefore yields the same slope as Local in Fig. 4.7(a).

Fig. 4.8 plots the numbers of the devices that can be pre-admitted, the devices that spontaneously decide to process tasks locally and not to send offloading requests, and the devices that send requests to be selected for offloading, as N increases. The total number of the three types of device is N. It is worth pointing out that the number of subchannels K limits the number of devices concurrently offloading, not the number of devices in total. As shown in the figure, both the numbers of the devices pre-admitted and the devices withholding requests increase with N, while the number of the devices that request to be scheduled first increases and then declines. This is because, by pre-admitting any resource-restrained devices and pre-rejecting those which can neither save energy nor meet deadlines by offloading, the number of devices that need to feed back channels and task information can be substantially reduced. The feedbacks can even reduce with the growth of N, since the number of resource-restrained devices that need to be pre-admitted for offloading grows, hence increasingly draining the available computational resources and stopping devices from offloading and feeding back.

Fig. 4.9 compares runtime between B&B and EROS, where $T^{\text{req}} = 1$s and the

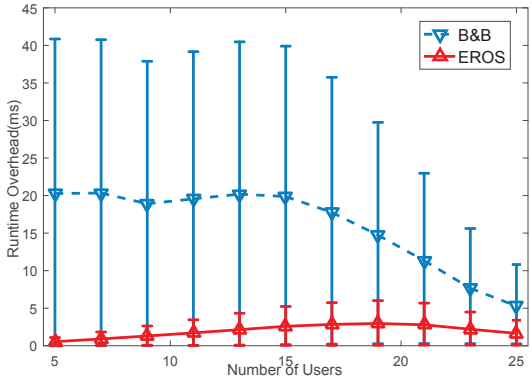


Figure 4.9 : The comparison of runtime between EROS and B&B where $T^{\text{req}} = 1$s. EROS is superior in terms of efficiency and stability.

confidence interval is 95%. We can see that the proposed EROS is superior in terms of efficiency and stability. Both the average and variance of the runtime of EROS are substantially lower than those of B&B, respectively. In contrast, B&B is neither computationally scalable nor reliable, and provides limited value in practice. We also see the concavity of the average runtime, i.e., the runtimes of EROS and B&B first increase and then decline as N grows, as can be evidenced by Fig. 4.8.

Chapter 5

Optimal Schedule for Internet-of-Things under Partial Information

This chapter presents a new asymptotically optimal offloading schedules, which are tolerant of partial out-of-date network knowledge and stochastically maximize a time-average network utility balancing system throughput and fairness. The contributions are summarized as follows.

- A perturbed Lyapunov technique is proposed to decouple time couplings of offloading schedule, energy intake, and data admission, and convert the stochastic optimization of offloading schedules to deterministic optimizations over time slots without loss of optimality. An $[\mathcal{O}(V), \mathcal{O}(1/V)]$-tradeoff can be achieved between stability and utility, so that the optimality loss of the proposed approach (as compared to the offline optimum) asymptotically diminishes at the cost of increasing queue length;

- Particularly, the deterministic optimization per slot is meticulously formulated to a linear relaxation of a knapsack problem. The breaking item of the problem, achieving the asymptotic optimality, is identified. Its lower bound in the presence of partial out-of-date information is derived as the threshold. As a result, only the devices meeting the threshold are designed to feed back per slot, hence preserving the asymptotic optimality with reduced feedbacks;

- Extensive simulations show that our approach is able to substantially improve system utility by 26% and dramatically reduce feedbacks per slot by over 98%. Only less than 60 out of 5000 devices need to feed back per slot. The feedbacks can decline with an increasingly large number of devices, due to the increasingly prominent urgency of some devices.

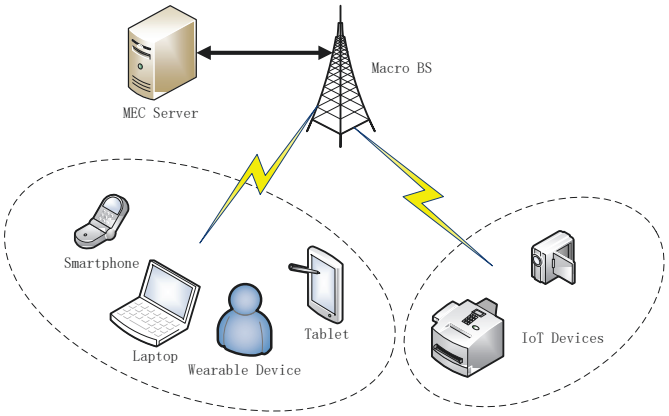


Figure 5.1 : An illustrative example of single-cell fog computing for IoT.

5.1 System Model

Fig. 5.1 shows the fog computing system which consists of a base station (BS) and N IoT devices, e.g., smart meters for supply monitoring, actuators for industrial automation, and sensors for smart home and E-health. Let $\mathbf{N} = \{1, 2, \ldots, N\}$ collect the indexes for N IoT devices. The system operates in a slotted structure, $t \in \mathbb{T} = \{0, 1, \ldots\}$ with slot length T. Equipped with energy harvesting capabilities, the IoT devices can be powered by renewable energy from the ambient environment. An edge server is deployed at the BS, and sensory data from the IoT devices can be offloaded through wireless links and processed at the edge server. In the presence of other uplink traffic to the BS, the number of available subchannels for IoT data offloading, denoted by $K(t)$, can vary between time slots. The notations used in this paper are summarized in Table 5.1.

Independent and identical distributed (i.i.d.) flat block fading channels are assumed, i.e., the channels remain unchanged during a time slot and vary between slots [25]. Let $c_i(t)$ denote the channel capacity in the uplink of device i at slot t. $c_i^{\min} \leq c_i(t) \leq c_i^{\max}$, where $c_i^{\min}$ and $c_i^{\max}$ are the minimum and maximum capacities of the link, respectively, given finite transmit powers of IoT devices.

Within a subchannel, the transmissions of multiple IoT devices can be accommodated via TDMA. Consider that an IoT device is inexpensive, simple and narrowband [54]. It can only access a subchannel at the same time. Let $\tau_i(t)$ denote the

Table 5.1 : Definitions of Notations

Notation	**Definition**
$\mathbf{N}$	The set of IoT devices
T	The slot length
$K(t)$	The number of available subchannels during slot t
$c_i(t)$	The channel capacity of device i during slot t
$\tau_i(t)$	The transmission duration of device i within slot t
$d_i(t)$	The size of data offloaded from device i during slot t
$A_i(t)$	The size of data generated by device i at slot t
$a_i(t)$	The size of data admitted at device i within slot t
$e_i(t)$	The energy consumption of data offloading during slot t
$E_{\mathrm{H}}^i(t)$	The amount of energy harvested by device i at slot t
$e_{\mathrm{H}}^i(t)$	The amount of energy stored at device i during slot t
ξ_i	The CPU cycles for processing a task bit of device i
$F(t)$	The total available CPU cycles at slot t
$\overline{X}$	The time-average of any stochastic process $X(t)$
$\phi(\overline{\boldsymbol{a}})$	The system utility balancing throughput and fairness
$Q_i(t)$	The data queue backlog of device i at slot t
$E_i(t)$	The battery level of device i at slot t
$C(t)$	The task backlog in terms of CPU cycles at slot t
$G_i(t)$	The virtual queue backlog of device i at slot t
$\alpha_i(t)$	The unit profit of device i at slot t
$\widehat{X}_i(t)$	The value of X under partial out-of-date knowledge
$\underline{\alpha}_b(t)$	The feedback threshold at slot t

transmission duration of device i within slot t, i.e.,

$$0 \leq \tau_i(t) \leq T, \forall i \in \mathbf{N}, \tag{5.1a}$$

and the maximum amount of data offloaded from device i to the edge server can be given by $\widehat{d}_i(t) = c_i(t)\tau_i(t)$. In some cases, a device can be assigned to different subchannels at different times within a slot, since the narrow-band device can switch between subchannels to offload tasks [54].

We define an offloading schedule $\boldsymbol{\tau}(t) = \{\tau_1(t), \ldots, \tau_N(t)\}$ by collecting the schedules of all IoT devices during slot t. The total transmission time of the devices must not exceed the number of available subchannel times the slot length, as given by

$$\sum\nolimits_{i \in \mathbf{N}} \tau_i(t) \leq K(t)T. \tag{5.1b}$$

Let $A_i(t)$ denote the size of sensory data arriving at device i at slot t. $A_i(t)$ is an i.i.d. random process with the maximum $A_i^{\max}$. Admission control is critical for the IoT devices with limited buffer size and the edge server with restrained computing capacity. As a result, device i may only be able to admit part of the arrived data, denoted by $a_i(t)$, satisfying

$$0 \leq a_i(t) \leq A_i(t), \, \forall i, t. \tag{5.2}$$

Let $Q_i(t)$ be the backlog of the data queue at IoT device i, as updated by

$$Q_i(t+1) = [Q_i(t) - d_i(t)] + a_i(t), \tag{5.3}$$

where

$$d_i(t) = \min\{Q_i(t), c_i(t)\tau_i(t)\}, \tag{5.4}$$

since the device cannot offload more than what it has. The corresponding energy consumption of the device for offloading the data, $e_i(t)$, can be written as$e_i(t) = p_i\tau_i(t)$, where p_i is the transmit power of device i.

The energy harvesting at an IoT device can also be modeled as a stochastic arrival process [55]. Let $E_{\mathrm{H}}^i(t)$ denote the amount of renewable energy harvested by device i during time slot t. Assume that $E_{\mathrm{H}}^i(t)$ is i.i.d. with the maximum $E_{\mathrm{H}}^{i,\max}$. Consider a finite battery at the device. Device i can only store part of the newly harvested energy at each time slot t, denoted by $e_{\mathrm{H}}^i(t)$, satisfying

$$0 \leq e_{\mathrm{H}}^i(t) \leq E_{\mathrm{H}}^i(t), \, \forall i, t. \tag{5.5}$$

Let $E_i(t)$ denote the battery backlog of device i at slot t. It can be updated by

$$E_i(t+1) = [E_i(t) - e_i(t)] + e_{\mathrm{H}}^i(t), \tag{5.6}$$

where

$$e_i(t) \leq E_i(t), \forall i, t, \tag{5.7}$$

since the energy used for offloading must not exceed the energy in its battery.

Upon the receipt of the task from device i at time slot t, i.e., $d_i(t)$ in (5.4), the edge server can process the task with $f_i(t) = \xi_i d_i(t)$ CPU cycles or put the task into the queue for later processing, where ξ_i is the number of CPU cycles required per task bit of device i for the task.

Let $C(t)$ be the required CPU cycles to process the task queued at the edge server, as updated by

$$C(t+1) = \max\left[C(t) - F(t), 0\right] + \sum\nolimits_{i \in \mathbf{N}} f_i(t), \tag{5.8}$$

where $F(t)$ specifies the total available CPU cycles at slot t. In the presence of other concurrent services, $F(t)$ is stochastic with the maximum $F^{\max}$. $\sum_{i \in \mathbf{N}} f_i(t)$ gives the total of the newly offloaded tasks during time slot t.

5.2 Online Offloading in Single-cell Fog Computing

By leveraging between throughput and fairness in fog computing, the system utility can be defined as

$$\phi(\overline{\boldsymbol{a}}) = \sum\nolimits_{i \in \mathbf{N}} \log(1 + \overline{a_i}), \tag{5.9}$$

where $\overline{X} = \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[X(t)]$ defines the time-average of any stochastic process $X(t)$, and $\overline{\boldsymbol{a}} = \{\overline{a_1}, \overline{a_2}, \cdots, \overline{a_N}\}$ collects the time-average total of admitted data at all N devices. Particularly, $\phi(\overline{\boldsymbol{a}})$ is a concave logarithm function that is able to balance the network throughput and fairness by discouraging individual devices to consume excessive energy for little utility gains [56].

We aim at maximizing the system utility under all the causality constraints while preserving the stability of all the queues in the system. All the queues are time-average stable if and only if [57]

$$\overline{\mathrm{C}} < \infty, \overline{E_i} < \infty, \overline{Q_i} < \infty, \forall i. \tag{5.10}$$

If the queues are stable, the total data admission $\sum_{i\in\mathbf{N}} \overline{a_i}$ is equal to the time-average system throughput in the long term.

The problem of interest can be formulated as given by

$$\begin{aligned} &\max_{\boldsymbol{\tau}(t),\boldsymbol{e}_{\mathrm{H}}(t),\boldsymbol{a}(t)} \phi(\overline{\boldsymbol{a}}) \\ &\text{s.t. (5.1), (5.2), (5.3), (5.7), (5.5), (5.6), (5.8), (5.10).} \end{aligned} \tag{5.11}$$

where $\boldsymbol{\tau}(t)$, $\boldsymbol{e}_{\mathrm{H}}(t)$ and $\boldsymbol{a}(t)$ are the offloading schedules, energy intake, and data admission to be optimized, respectively.

Note that $c_i(t)$, $A_i(t)$, $E_{\mathrm{H}}^i(t)$, $K(t)$ and $F(t)$ are all time-varying with little predictability. An offline optimization of (5.11) would violate causality; whereas a myopic optimization per time slot would compromise the optimality in the long term, due to time coupling of variables in (5.3), (5.6), (5.8), and (5.10).

5.2.1 Perturbed Lyapunov Optimization

Lemma 8. *By defining auxiliary variable $\gamma_i(t)$ for each IoT device i, satisfying*

$$\overline{\gamma_i} \le \overline{a_i}, \forall i; \tag{5.12a}$$

$$0 \le \gamma_i(t) \le A_i^{\max}, \forall i, t. \tag{5.12b}$$

Problem (5.11) can be equivalently reformulated as

$$\begin{aligned} &\max_{\boldsymbol{\tau}(t),\boldsymbol{e}_{\mathrm{H}}(t),\boldsymbol{a}(t),\boldsymbol{\gamma}(t)} \overline{\phi(\boldsymbol{\gamma})} \\ &\qquad \textit{s.t. (5.1), (5.2), (5.3), (5.7), (5.5), (5.6), (5.8), (5.10), (5.12)} \end{aligned} \tag{5.13}$$

Proof. Given that (5.1), (5.2), (5.3), (5.7), (5.5), (5.6), (5.8), and (5.10) are part of both (5.11) and (5.13). We only need to prove that the maximum for (5.13) is no less than that for (5.11) in the presence of (5.12). Let $\overline{\phi(\boldsymbol{\gamma})}$ denote the maximum utility of (5.13), where $\overline{\boldsymbol{a}^*}$ is the corresponding time-average data admission. Since $\phi(\boldsymbol{x})$ is a monotonically increasing function, (5.12a) guarantees that $\phi(\overline{\boldsymbol{a}^*}) \geq \phi(\overline{\boldsymbol{\gamma}})$. Using Jensen's inequality [40], we have $\phi(\overline{\boldsymbol{\gamma}}) \geq \overline{\phi(\boldsymbol{\gamma})}$ and, in turn, $\phi(\overline{\boldsymbol{a}^*}) \geq \overline{\phi(\boldsymbol{\gamma})}$.

Let ϕ^{opt} denote the solution for (5.11), where $\boldsymbol{a}^{\text{opt}}(t)$ is the corresponding data admission at slot t. Clearly, under (5.12) in (5.13), performing the strategy $\boldsymbol{\gamma}(t) = \overline{\boldsymbol{a}^{\text{opt}}(t)}$ for all time slots t, achieves $\overline{\phi(\boldsymbol{\gamma})} = \phi^{\text{opt}}$. As a result, $\phi(\overline{\boldsymbol{a}^*}) \geq \phi^{\text{opt}}$, i.e., (5.13) is equivalent to (5.11). □

By defining a device-specific virtual queue, (5.12a) can be reformed with improved tractability, where the backlog of the virtue queue, denoted by $G_i(t)$, is updated by

$$G_i(t+1) = \max\{G_i(t) + \gamma_i(t) - a_i(t), 0\}. \tag{5.14}$$

According to queuing theory, $G_i(t)$ is stable if and only if (5.12a) is satisfied [58]. We can replace (5.12a) with the stability of $G_i(t)$, i.e.,

$$\overline{G_i} < \infty, \forall i. \tag{5.15}$$

Stochastic optimization, more specifically, Lyapunov optimization, is able to eliminate time coupling of variables, and enable online decision-making, while preserving asymptotic optimality [57]. A perturbed Lyapunov function of the system can be defined as

$$L(t) = \frac{1}{2}\left\{C(t)^2 + \sum_{i\in\mathbf{N}}[Q_i(t)^2 + (E_i(t) - \theta_i)^2 + G_i(t)^2]\right\},$$

where $\boldsymbol{\theta} = \{\theta_i, \forall i \in \mathbf{N}\}$ collects the perturbing weights (biases) to ensure sufficient energy available in the batteries for optimal schedules durations at any slot, such that the temporal correlation of the schedules due to (5.6) can be decoupled. Let $\Theta(t) = \{C(t), Q_i(t), E_i(t), G_i(t), \forall i\}$ collect all the backlogs at slot t. A drift-plus-

penalty function can be defined by

$$\Delta_V(t) = \Delta(t) - V\mathbb{E}\left[\sum\nolimits_{i\in\mathbf{N}} \log(1+\gamma_i(t))|\Theta(t)\right], \tag{5.16}$$

where $\Delta(t) \triangleq \mathbb{E}\left[L(t+1) - L(t)|\Theta(t)\right]$ is the conditional Lyapunov drift, i.e., the conditional expectation of the difference of the perturbed Lyapunov function between two consecutive time slots, and $V \geq 0$ is a predefined coefficient to tune the tradeoff between the system utility and queue stability.

Minimizing an upper bound of $\Delta_V(t)$ per slot t can prevent unbounded growths of backlogs, while maximizing the time-average conditional expectation of the system utility [57]. The upper bound can be given in the following lemma.

Lemma 9. *For any queue backlogs and actions, the drift-plus-penalty $\Delta_V(t)$ is upper bounded by*

$$\begin{aligned}\Delta_V(t) \leq D - V\mathbb{E}\left[\textstyle\sum_{i\in N} \log(1+\gamma_i(t))|\Theta(t)\right] C(t)\mathbb{E}[\textstyle\sum_{i\in N} f_i(t) - F(t)|\Theta(t)] \\ + \textstyle\sum_{i\in N} Q_i(t)\mathbb{E}[a_i(t) - d_i(t)|\Theta(t)] \textstyle\sum_{i\in N} [E_i(t) - \theta_i]\mathbb{E}[e^i_{\mathrm{H}}(t) - e_i(t)|\Theta(t)] \\ + \textstyle\sum_{i\in N} G_i(t)\mathbb{E}[\gamma_i(t) - a_i(t)|\Theta(t)]\end{aligned} \tag{5.17}$$

where

$$\begin{aligned}D = \tfrac{1}{2}\textstyle\sum_{i\in N}\{(p_iT)^2 + \left(E^{i,\max}_{\mathrm{H}}\right)^2 + (c_i^{\max}T)^2 + 3(A_i^{\max})^2\} \\ + \tfrac{1}{2}\{\textstyle\sum_{i\in N}(\xi_i c_i^{\max}T)^2 + (F^{\max})^2\}\end{aligned} \tag{5.18}$$

Proof. Taking squares on both sides of (5.3), (5.8), (5.6) and (5.14), and then exploiting $(\max[a-b,0]+c)^2 \leq a^2+b^2+c^2+2a(c-b)$ for any $a,b,c \geq 0$, we have:

$$Q_i(t+1)^2 - Q_i(t)^2 \leq a_i(t)^2 + d_i(t)^2 + 2Q_i(t)[a_i(t) - d_i(t)], \tag{5.19a}$$

$$C(t+1)^2 - C(t)^2 \leq (\textstyle\sum_{k\in\mathbf{N}} f_k(t))^2 + F(t)^2 + 2C(t)[\textstyle\sum_{k\in\mathbf{N}} f_k(t) - F(t)], \tag{5.19b}$$

$$E_i(t+1)^2 - E_i(t)^2 \leq e^i_{\mathrm{H}}(t)^2 + e_i(t)^2 + 2[E_i(t) - \theta_i][e^i_{\mathrm{H}}(t) - e_i(t)], \tag{5.19c}$$

$$G_i(t+1)^2 - G_i(t)^2 \leq a_i(t)^2 + \gamma_i(t)^2 + 2G_i(t)[\gamma_i(t) - a_i(t)]. \tag{5.19d}$$

Plugging (5.19) into (5.16) yields the upper bound in Lemma 10 with D satisfying

$$\begin{aligned} D \geq \tfrac{1}{2} \sum_{i \in N} \{\mathbb{E}[e_i(t)^2] + \mathbb{E}[e_{\mathrm{H}}^i(t)^2] + \mathbb{E}[\gamma_i(t)^2] + 2\mathbb{E}[a_i(t)^2] + \mathbb{E}[d_i(t)^2]\} \\ + \tfrac{1}{2}\{\mathbb{E}[(\textstyle\sum_{i \in N} f_i(t))^2] + \mathbb{E}[F(t)^2]\}. \end{aligned} \tag{5.20}$$

(5.18) provides an upper bound for (6.15) by replacing all the expectations with the maximums in (6.15). □

Exploiting Lemma 10, we decouple the optimal decisions of (5.11) between time slots to minimize (6.10) subject to the instantaneous constraints, as given by

$$\begin{aligned} \max_{\boldsymbol{\tau}(t), \boldsymbol{e}_{\mathrm{H}}(t), \boldsymbol{a}(t), \boldsymbol{\gamma}(t)} & f(\boldsymbol{\gamma}(t)) + g(\boldsymbol{\tau}(t)) - \eta(\boldsymbol{e}_{\mathrm{H}}(t)) - \mu(\boldsymbol{a}(t)) \\ \text{s.t. } & (5.1), (5.2), (5.7), (5.5), (5.12\text{b}); \end{aligned} \tag{5.21}$$

where $F(t)$ is suppressed, since it is independent of the variables; and

$$f(\boldsymbol{\gamma}(t)) = \textstyle\sum_{i \in \mathbf{N}} [V \log(1 + \gamma_i(t)) - G_i(t)\gamma_i(t)]; \tag{5.22a}$$

$$g(\boldsymbol{\tau}(t)) = \sum_{i \in \mathbf{N}} \{[Q_i(t) - C(t)\xi_i]d_i(t) + [E_i(t) - \theta_i]e_i(t)\}; \tag{5.22b}$$

$$\eta(\boldsymbol{e}_{\mathrm{H}}(t)) = \textstyle\sum_{i \in \mathbf{N}} [E_i(t) - \theta_i]e_{\mathrm{H}}^i(t); \tag{5.22c}$$

$$\mu(\boldsymbol{a}(t)) = \textstyle\sum_{i \in \mathbf{N}} [Q_i(t) - G_i(t)]a_i(t). \tag{5.22d}$$

Let $\overline{\boldsymbol{a}^*}$ denote the optimal solution for **P1**, and ϕ^{opt} be the maximum system utility which can only be achieved offline under the assumption of full knowledge across all time slots, i.e., the optimum for (5.11). As shown in the following theorem, the optimum for **P1**, $\phi(\overline{\boldsymbol{a}^*})$, converges to the optimum for (5.11), ϕ^{opt}, as V increases.

Theorem 3. *The gap between the optimum for (5.21), $\phi(\overline{\boldsymbol{a}^*})$, and the optimum for (5.11), ϕ^{opt}, is less than D/V, i.e.,*

$$\phi^{\mathrm{opt}} - \phi(\overline{\boldsymbol{a}^*}) \leq D/V. \tag{5.23}$$

Proof. Let π denote any feasible control strategy for (5.11). The minimum upper bound of $\Delta_V(t)$ in (6.10) can be given by

$$\begin{aligned}&\Delta_V(t) \leq D - V\mathbb{E}\left[\sum_{i\in N}\log(1+\gamma_i(t))|\Theta(t),\pi\right] + C(t)\mathbb{E}[\sum_{i\in N} f_i(t) - F(t)|\Theta(t),\pi]\\&+\sum_{i\in N} Q_i(t)\mathbb{E}[a_i(t)-d_i(t)|\Theta(t),\pi] + \sum_{i\in N}[E_i(t)-\theta_i]\mathbb{E}[e^i_{\mathrm{H}}(t)-e_i(t)|\Theta(t),\pi]\\&+\sum_{i\in N} G_i(t)\mathbb{E}[\gamma_i(t)-a_i(t)|\Theta(t),\pi].\end{aligned} \tag{5.24}$$

According to [57], 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:

$$\begin{aligned}&-\phi(\boldsymbol{\gamma}(t)|\pi^*) \leq -\phi^{\mathrm{opt}} + \sigma,\ \mathbb{E}[\sum_{i\in N} f_i(t) - F(t)|\pi^*] \leq \sigma,\ \mathbb{E}[a_i(t)-d_i(t)|\pi^*] \leq \sigma,\\&\mathbb{E}[e^i_{\mathrm{H}}(t)-e_i(t)|\pi^*] \leq \sigma,\ \mathbb{E}[\gamma_i(t)-a_i(t)|\pi^*] \leq \sigma.\end{aligned} \tag{5.25}$$

Plugging (6.18) into (5.24) and then taking $\sigma \to 0$ yields

$$\Delta_V(t) \leq D - V\phi^{\mathrm{opt}}. \tag{5.26}$$

Taking expectation on both sides of (5.26), we obtain

$$\mathbb{E}[L(t+1) - L(t)] - V\mathbb{E}[\phi(\gamma(t))] \leq D - V\phi^{\mathrm{opt}}. \tag{5.27}$$

Summing up all the telescoping series of (5.27) over $\{0, 1, \cdots, T-1\}$, we can obtain

$$\mathbb{E}[L(T)] - \mathbb{E}[L(0)] - V\sum\nolimits_{t=0}^{T-1}\mathbb{E}[\phi(\gamma(t))] \leq DT - VT\phi^{\mathrm{opt}}. \tag{5.28}$$

Consider that all queues are empty initially. Given the fact that $\mathbb{E}[L(T)] \geq 0$ and $L(0) = \sum_{i\in N}\theta_i^2$, we have

$$\frac{1}{T}\lim_{T\to\infty}\sum\nolimits_{t=0}^{T-1}\mathbb{E}[\phi(\gamma(t))] \geq \phi^{\mathrm{opt}} - \frac{D}{V}, \tag{5.29}$$

i.e., $\overline{\phi(\gamma(t))} \geq \phi^{\mathrm{opt}} - \frac{D}{V}$. In Lemma 8, we have proved that $\phi(\overline{\boldsymbol{a}^*}) \geq \overline{\phi(\gamma(t))}$. Therefore, we obtain $\phi^{\mathrm{opt}} - \phi(\overline{\boldsymbol{a}^*}) \leq \frac{D}{V}$. □

Theorem 4. *With $\overline{\boldsymbol{a}^*}$, the backlogs of all the queues in the system is strictly bounded*

at each time slot t, as given by

$$Q_i(t) \leq V/\ln 2 + 2A_i^{\max}; \tag{5.30}$$

$$E_i(t) \leq \theta_i + E_{\mathrm{H}}^{i,\max}; \tag{5.31}$$

$$C(t) \leq \beta + \sum\nolimits_{i \in N} \xi_i c_i^{\max} T; \tag{5.32}$$

where

$$\beta = \max_{i \in \mathbf{N}} \{ \frac{V/\ln 2 + 2A_i^{\max}}{\xi_i} + \frac{E_{\mathrm{H}}^{i,\max} p_i}{c_i^{\min} \xi_i} \}. \tag{5.33}$$

Proof. We first prove the upper bound of the auxiliary variable backlogs, i.e., $G_i(t) \leq \frac{V}{\ln 2} + A_i^{\max}$ through mathematical induction. Specifically, the upper bound holds at slot $t = 0$ given the fact that $G_i(t) = 0$, $\forall i \in \mathbf{N}$. Suppose that the upper bound holds at time slot t. Then, if $G_i(t) \geq \frac{V}{\ln 2}$, $\gamma_i(t) = 0$ according to (5.37), and $G_i(t)$ cannot increase at slot t, i.e., $G_i(t+1) \leq G_i(t)$. Otherwise, if $G_i(t) < \frac{V}{\ln 2}$, the difference between $G_i(t)$ and $G_i(t+1)$ is less than $A_i^{\max}$ according to (5.14) and (5.12b), i.e., $G_i(t+1) \leq \frac{V}{\ln 2} + A_i^{\max}$. As a result, the upper bound also holds at slot $t+1$. This concludes the proof of the upper bound of $G_i(t)$.

Given the upper bound of $G_i(t)$, we prove (5.30). Since (5.30) holds for $t = 0$, we suppose that it holds at slot t. From (5.41), if $Q_i(t) \geq G_i(t)$, device i does not admit new data, and therefore, $Q_i(t+1) \leq Q_i(t) \leq G_i(t) + A_i^{\max}$. On the other hand, if $Q_i(t) < G_i(t)$, the increased backlog must not exceed $A_i^{\max}$, i.e., $Q_i(t+1) \leq G_i(t) + A_i^{\max}$. Substituting the upper bound of $G_i(t)$, we have $Q_i(t+1) \leq \frac{V}{\ln 2} + 2A_i^{\max}$. In other words, (5.30) holds at slot $t+1$. By mathematical induction, (5.30) is proved.

Next, we proceed to prove (5.31). Since (5.31) holds at slot $t = 0$, we suppose that it holds at slot t. If $E_i(t) \geq \theta_i$, device i must not harvest energy according to (5.39), i.e., $E_i(t+1) \leq E_i(t) \leq \theta_i + E_{\mathrm{H}}^{i,\max}$. Otherwise, if $E_i(t) < \theta_i$, we have $E_i(t+1) \leq \theta_i + E_{\mathrm{H}}^{i,\max}$ due to $e_{\mathrm{H}}^i \leq E_{\mathrm{H}}^{i,\max}$. In other words, (5.31) holds at slot $t+1$. This concludes the proof of (5.31).

Finally, we prove (5.32). Since it holds at slot $t = 0$, we suppose that it holds at slot t. If $C(t) \geq \beta$, we have $\alpha_i(t) \leq 0$ for any device $i \in \mathbf{N}$, i.e., the MEC server

does not schedule any device according to (5.46), resulting in $C(t+1) \leq C(t)$. If $C(t) < \beta$, we have $C(t+1) \leq \beta + \sum_{i \in N} \xi_i c_i^{\max} T$ by substituting the maximum link capacity. As a result, (5.32) holds at $t+1$. This concludes the proof of (5.32). □

Theorem 5. *The weight perturbation parameter θ_i can be adjusted by*

$$\theta_i = [V/\ln 2 + 2A_i^{\max}]c_i^{\max}/p_i + p_i T, \tag{5.34}$$

such that, when a device is scheduled to offload at time slot t, there is enough energy in its battery backlog for optimal schedules, i.e., $E_i(t) \geq p_i T$.

Proof. When θ_i is adjusted according to (5.34) and $E_i(t) < p_i T$ at slot t, we have $E_i(t) - \theta_i < -\frac{[V/\ln 2 + 2A_i^{\max}]c_i^{\max}}{p_i}$. Plugging this into (6.20), we obtain $\alpha_i(t) < [Q_i(t) - C(t)\xi_i]c_i(t) - [V/\ln 2 + 2A_i^{\max}]c_i^{\max} < Q_i(t)c_i(t) - [V/\ln 2 + 2A_i^{\max}]c_i^{\max}$, i.e., $\alpha_i(t) < 0$ from (5.30). This proves that a device is not scheduled if $E_i(t) < p_i T$, given θ_i in (5.34). □

Theorems 3, 4 and 5 show an $[\mathcal{O}(V), \mathcal{O}(1/V)]$-tradeoff between the queue lengths (or in other words, the queuing delay according to Little's theorem [58]) and the optimality loss. Here, the optimality loss is the gap of system utility between the proposed approach and the causality-violating offline optimization of the offloading schedule. That is to say, by fine-tuning V, our approach allows the queuing delay to be adjustable at the cost of the optimal system utility.

5.2.2 Optimal Offloading Schedules

Note that $\boldsymbol{\tau}(t)$, $\boldsymbol{e}_{\mathrm{H}}(t)$, $\boldsymbol{a}(t)$ and $\boldsymbol{\gamma}(t)$ are decoupled in both the objective and constraints in (5.21), and therefore can be optimized separately as follows to achieve $\overline{\boldsymbol{a}^*}$.

1) Optimal Auxiliary Parameter: The subproblem of optimizing auxiliary variables can be written as

$$\begin{aligned} &\max_{\boldsymbol{\gamma}(t)} f(\boldsymbol{\gamma}(t)) \\ &\text{s.t. } 0 \leq \gamma_i(t) \leq A_i^{\max}, \forall i \in \mathbf{N}, \end{aligned} \tag{5.35}$$

where $\gamma_i(t)$ is decoupled from $\gamma_j(t)$ for any $j \neq i$. Therefore, $\gamma_i(t)$ can be optimized independently by solving

$$\begin{aligned} &\max_{\gamma_i(t)} V \log(1+\gamma_i(t)) - G_i(t)\gamma_i(t) \\ &\text{s.t. } 0 \leq \gamma_i(t) \leq A_i^{\max} \end{aligned} \tag{5.36}$$

The first-order partial derivation of $f(\boldsymbol{\gamma}(t))$ can be given by $\frac{\partial f}{\partial \gamma_i} = \frac{V}{(1+\gamma_i(t))\ln 2} - G_i(t)$. If $\frac{\partial f}{\partial \gamma_i}|_{\gamma_i(t)=0} \leq 0$, $f(\boldsymbol{\gamma}(t))$ decreases monotonically in the region $\gamma_i(t) \geq 0$ and the optimum $\gamma_i(t)^* = 0$. Otherwise, the optimum is either at the stationary point $\frac{\partial f}{\partial \gamma_i} = 0$, i.e., $\gamma_i(t)^* = \frac{V}{G_i(t)\ln 2} - 1$, or on the boundary, i.e., $\gamma_i(t)^* = A_i^{\max}$. Therefore,

$$\gamma_i(t)^* = \begin{cases} 0, \text{ if } \frac{V}{\ln 2} - G_i(t) \leq 0; \\ \min\{\frac{V}{G_i(t)\ln 2} - 1, A_i^{\max}\}, \text{ otherwise.} \end{cases} \tag{5.37}$$

2) Optimal Energy Intake: Decoupled from (5.21), the subproblem of optimizing energy intake can be written as

$$\begin{aligned} &\min_{\boldsymbol{e}_{\mathrm{H}}(t)} \textstyle\sum_{i\in\mathbf{N}} [E_i(t) - \theta_i] e_{\mathrm{H}}^i(t) \\ &\text{s.t. } 0 \leq e_{\mathrm{H}}^i(t) \leq E_{\mathrm{H}}^i(t), \forall i \in \mathbf{N}, \end{aligned} \tag{5.38}$$

where the optimal solution can be readily given by

$$e_{\mathrm{H}}^i(t)^* = \begin{cases} 0, \text{ if } E_i(t) \geq \theta_i; \\ E_{\mathrm{H}}^i(t), \text{ otherwise.} \end{cases} \tag{5.39}$$

The devices consistently push their battery backlogs towards $\boldsymbol{\theta}$. As shown in Theorem 5, sufficient energy is available in the batteries for optimal schedules by judiciously designing $\boldsymbol{\theta}$.

3) Optimal Data Admission: The decoupled subproblem of optimizing data admission can be written as

$$\begin{aligned} &\min_{\boldsymbol{a}(t)} \textstyle\sum_{i\in\mathbf{N}} [Q_i(t) - G_i(t)] a_i(t) \\ &\text{s.t. } 0 \leq a_i(t) \leq A_i(t), \forall i \in \mathbf{N}, \end{aligned} \tag{5.40}$$

where the optimal admission decision can be given by

$$a_i(t)^* = \begin{cases} 0, \text{ if } Q_i(t) \geq G_i(t); \\ A_i(t), \text{ otherwise.} \end{cases} \tag{5.41}$$

4) Optimal Offloading Schedule: $\boldsymbol{\tau}(t)$ is optimized to maximize $g(\boldsymbol{\tau}(t))$, while satisfying (5.1a) and (5.1b) at each time slot t. From (5.3) and (5.4), it is clear that device i should not transmit more data than $Q_i(t)$ at slot t, i.e., $\tau_i(t) \leq Q_i(t)/c_i(t)$. Besides, (5.7) requires that the transmission energy cannot exceed the available energy in the battery backlog, i.e., $\tau_i(t) \leq E_i(t)/p_i$. Therefore, (5.1a) can be tightened to

$$0 \leq \tau_i(t) \leq T_i, \, \forall i \in \mathbf{N}, \tag{5.42}$$

where $T_i = \min\{Q_i(t)/c_i(t), E_i(t)/p_i, T\}$. By rearranging (5.22b), $\boldsymbol{\tau}(t)$ can be given by

$$\begin{aligned} &\max_{\boldsymbol{\tau}(t)} \textstyle\sum_{i\in\mathbf{N}} \alpha_i(t)\tau_i(t) \\ &\text{s.t. (5.1b) and (5.42),} \end{aligned} \tag{5.43}$$

where

$$\alpha_i(t) = [Q_i(t) - C(t)\xi_i]c_i(t) + [E_i(t) - \theta_i]p_i. \tag{5.44}$$

By interpreting $\alpha_i(t)$ as the unit profit of an "item" i and $K(t)T$ as the capacity of a knapsack, (5.43) becomes the linear relaxation of a knapsack problem. The optimal solution for linear relaxation knapsack can be found by selecting the "item" with higher unit profit to fulfill the knapsack capacity [59].

Without loss of generality, we assume the devices are sorted in the descending order of unit profit, i.e., $\alpha_i(t) \geq \alpha_j(t)$ for $i > j$. The optimal solution is $\tau_i = T_i$ if $\sum_{j=1}^{i} T_j \leq K(t)T$. Define the breaking item to be the first device that is not allocated its full transmission duration specified in (5.42) [59]. The index of the breaking item b can be identified, as given by

$$b = \arg\min_i \sum\nolimits_{j=1}^{i} T_j > K(t)T. \tag{5.45}$$

Algorithm 4 Online Optimization of Offloading in Fog Computing for IoT

At each device i:
1: Acquire $Q_i(t)$, $E_i(t)$, $c_i(t)$, $A_i(t)$ and $E_{\mathrm{H}}^i(t)$
2: Perform the optimal solutions given by (5.37), (5.39) and (5.41)
3: Feed back $Q_i(t)$, $E_i(t)$ and $c_i(t)$ to the edge server
4: Update $G_i(t+1)$ based on (5.14)
At the edge server:
5: Acquire $K(t)$, $F(t)$ and $C(t)$
6: Collect feedbacks from all the devices and evaluate $\alpha_i(t)$
7: Schedule the offloading by (5.46)

As a result, the optimal offloading schedule can be given by

$$\tau_i(t)^* = \begin{cases} T_i, \text{ if } i < b; \\ K(t)T - \sum_{i=1}^{b-1} T_i, \text{ if } i = b; \\ 0, \text{ otherwise,} \end{cases} \tag{5.46}$$

which, involving ordering the devices against their unit profits, can be achieved by using the quicksort algorithm with a complexity of $\mathcal{O}(N \log N)$ [47].

Note that the optimal solutions in (5.37), (5.39) and (5.41) can be locally computed at individual IoT devices, since the queue backlogs, wireless channels, data arrivals and energy harvesting can be measured at the devices without the coordination of the edge server. However, (5.46) requires the knowledge on all the IoT devices, i.e., $c_k(t)$, $Q_k(t)$, $E_k(t)$, and $G_k(t)$ for $k \in \mathbf{N}$. At each time slot t, the edge server needs each IoT device to feed back its link capacity $c_k(t)$ and backlogs $Q_k(t)$, $E_k(t)$ and $G_k(t)$ to evaluate $\tau_i(t)^*$ in (5.46). We propose to distribute part of these optimizations (5.37), (5.39) and (5.41) at the IoT devices, and the rest (5.46) at the edge server, as summarized in Algorithm 4.

5.3 Offloading Schedules under Partial Feedbacks

As discussed, the link capacity and backlogs of each IoT device are needed in the edge server to evaluate (5.46) for optimal offloading schedules. Consider practical IoT, however, the number of IoT devices can be hundreds to thousands, while the

number of available subchannels can be far less, i.e., $K(t) \ll N$. Getting each device to feed back as per slot could be infeasible due to explosive overhead, and also inefficient since the majority of the feedbacks would not contribute to the optimization of the schedule.

From (5.46), it is clear that the IoT devices with high unit profits are given priority to offload. Particularly, the optimality of (5.46) would not be compromised, if the devices with unit profits less than the breaking item, i.e., $\alpha_i(t) < \alpha_b(t)$, do not feed back. To this end, there is an opportunity arising to substantially reduce feedbacks at every slot.

In this section, we generalize the proposed online optimization with up-to-date knowledge on devices' backlogs at the edge server, to the scenario where the knowledge is out-of-date at the edge server from slot to slot. Lyapunov optimization under partial information and out-of-date knowledge is established, which captures the case of practical interest where only part of IoT devices feed back their backlogs. Using this, we derive a threshold for $\alpha_i(t)$, denoted by $\underline{\alpha}_b(t)$, which enables a small number of devices to self-nominate to feed back, while preserving the asymptotic optimality stated in Theorem 3. Following are the details.

From (6.20), the edge server evaluates $\alpha_i(t)$ based on $C(t)$, $c_i(t)$, $Q_i(t)$ and $E_i(t)$, where $C(t)$ is known to the server whereas the other three parameters need device i to feed back. In the absence of up-to-date feedbacks from most devices, we define $\widehat{Q}_i(t)$ and $\widehat{E}_i(t)$ for the data and energy backlogs of device i, respectively, to capture the out-of-date knowledge at the edge server.

The edge server can update $\widehat{Q}_i(t)$ and $\widehat{E}_i(t)$, as follows. In the case that device i does not feed back at slot t, the edge server keeps the out-of-date backlogs unchanged, i.e., $\widehat{Q}_i(t+1) = \widehat{Q}_i(t)$ and $\widehat{E}_i(t+1) = \widehat{E}_i(t)$. In the case that device i feeds back at slot t, the edge server refreshes the backlogs up-to-date, as given by

$$\begin{aligned} \widehat{Q}_i(t+1) &= Q_i(t) - d_i(t) + a_i(t), \\ \widehat{E}_i(t+1) &= E_i(t) - e_i(t) + e_{\mathrm{H}}^i(t). \end{aligned} \tag{5.47}$$

From (5.3) and (5.6), we can see that these backlogs do not exceed the actual backlogs, i.e., $\widehat{Q}_i(t) \leq Q_i(t)$ and $\widehat{E}_i(t) \leq E_i(t)$ for $i \in \mathbf{N}$ and $t \in \mathbb{T}$. In the absence of feedback from device i, the channel capacity of the device can be replaced by the minimum capacity of the link $c_i^{\min}$.

By referring to (6.20), the edge server can evaluate the unit profit of device i based on the out-of-date backlogs, as given by

$$\widehat{\alpha}_i(t) = [\widehat{Q}_i(t) - C(t)\xi_i]c_i^{\min} + [\widehat{E}_i(t) - \theta_i]p_i. \tag{5.48}$$

Clearly, $\widehat{\alpha}_i(t) \leq \alpha_i(t)$, $\forall i \in \mathbf{N}$, $t \in \mathbb{T}$. The unit profit of the breaking item $\underline{\alpha}_b(t)$ can be achieved by sorting devices in the descending order of $\widehat{\alpha}_i(t)$ and identifying the breaking item based on (5.45). Algorithm 4 can be slighted modified to accommodate the reduced feedbacks. Specifically, the edge server broadcasts $\underline{\alpha}_b(t)$ as the threshold before step 1 based on its out-of-date information. Device i compares the threshold with its unit profit $\alpha_i(t)$, and self-nominates to feed back in step 3, if $\alpha_i(t) \geq \underline{\alpha}_b(t)$. Algorithm 4 resumes in the rest steps.

Corollary 1. *The out-of-date knowledge on the backlogs of devices does not compromise the optimality of* ***P3****.*

Proof. The optimal $\boldsymbol{e}_{\mathrm{H}}(t)$, $\boldsymbol{a}(t)$ and $\boldsymbol{\gamma}(t)$ for (5.21) are obtained based on the real-time measurement and calculation at each device. We only need to prove the optimality of $\boldsymbol{\tau}(t)$. Since $\underline{\alpha}_b(t) \leq \alpha_b(t)$ always holds, the optimal set of devices to be selected in (5.46) to offload, i.e., $\{i|\alpha_i(t) \geq \alpha_b(t)\}$, is a subset of the devices feeding back their link capacity and up-to-date backlogs, i.e., $\{i|\alpha_i(t) \geq \underline{\alpha}_b(t)\}$. This preserves the optimality of $\boldsymbol{\tau}(t)$. □

5.4 Simulation Result and Analysis

In this section, we evaluate the proposed optimal offloading schedules of MEC system in Matlab with $T = 1$ sec. Focused on scheduling a large number of devices using partial out-of-date knowledge, we assume that the uploaded tasks can be computed correctly, and the randomness of externalities, such as sensory data, energy

arrival, and channel conditions, can be parameterized. Particularly, the computational resource $F(t) \sim U[0,3]$ GHz, the arrival of sensory data $A_i(t) \sim U[0,10]$ kbps, the number of available subchannels $K(t) \sim U[0,30]$, where $U[a,b]$ denotes a random uniform distribution within $[a,b]$. We also assume that the arrival of renewable energy $E_{\mathrm{H}}^i(t) \sim U[0,20]$ mJ/s, the transmit power $p_i \sim U[10,23]$ dBm, and the average link capacity $c_i \sim U[20,80]$ kbps, where $c_i(t) \sim U[0.5c_i, 2c_i]$. The number of devices N is set to be 500, unless otherwise specified. Every result of the simulations is the average of 5000 independent runs.

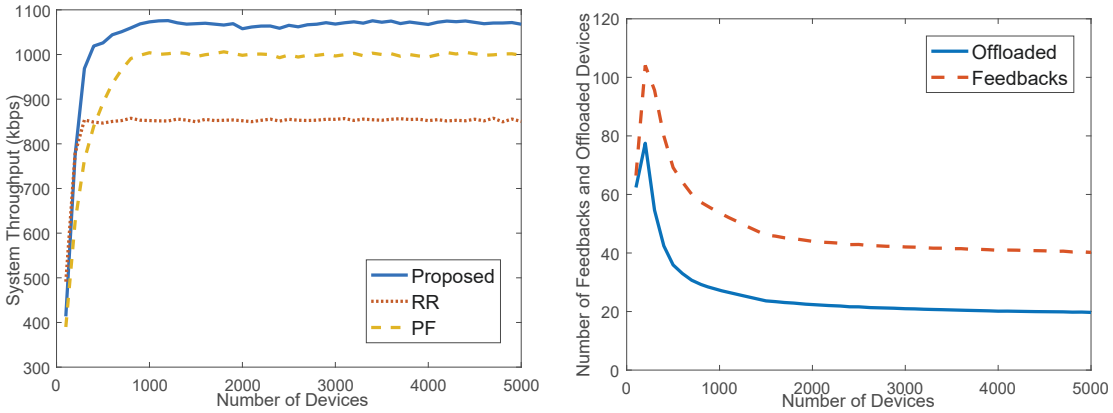
For comparison purpose, we also simulate

- Round Robin (RR) method which gives no priority to any devices;
- Proportional Fair (PF) based on the popular weighted fair queueing model, where devices are prioritized and scheduled based on $c_i(t)/\sum_{\tau=0}^{t-1} d_i(t)$.

We note that, in the case of RR, devices are scheduled in a predetermined order and only a subset of devices to be scheduled at the upcoming slot need to feed back their states on channels, battery levels, and queue backlogs. In the case of PF, however, each device needs to feed back its priority and states, based on which the edge server selects the devices to offload. In other words, PF still requires explicit up-to-date knowledge on all the devices.

Fig. 5.2 plots the throughput of the proposed approach and the corresponding average number of devices that feed back per slot, where N increases from 100 to 5000. We can see in Fig. 5.2(a) that the throughput of the proposed approach converges as the number of devices gets large, e.g., $N > 1000$. This is due to the limited wireless link capacity and computational resources of the network. Nevertheless, our approach provides higher throughput than RR and PF, by prudently leveraging the data queues and energy battery.

More importantly, we can see in Fig. 5.2(b) that the number of devices self-nominated to feed back their backlogs and channels is substantially reduced in the proposed approach based on partial and out-of-date information. Particularly, the

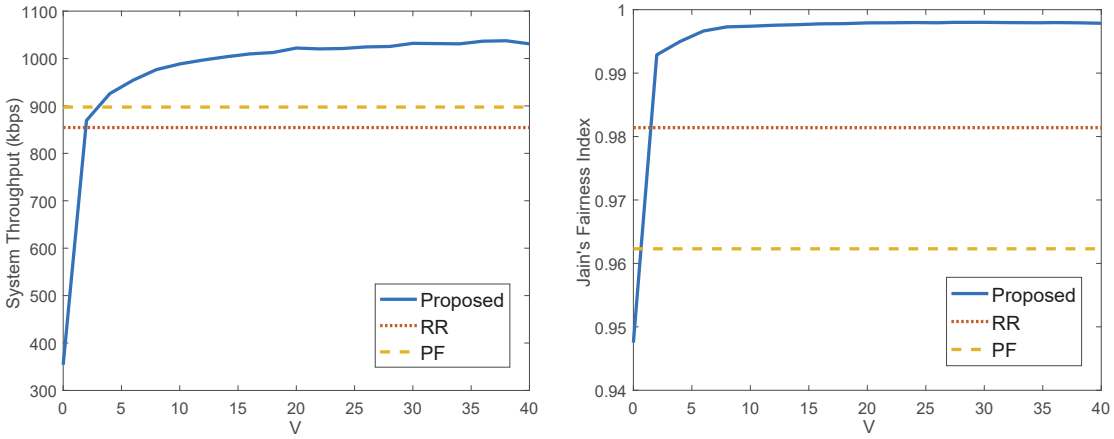


(a) System throughput (b) Feedbacks and offloaded devices

Figure 5.2 : The changes of throughput and the corresponding average number of feedbacks as N increases from 100 to 5000 where $V = 40$.

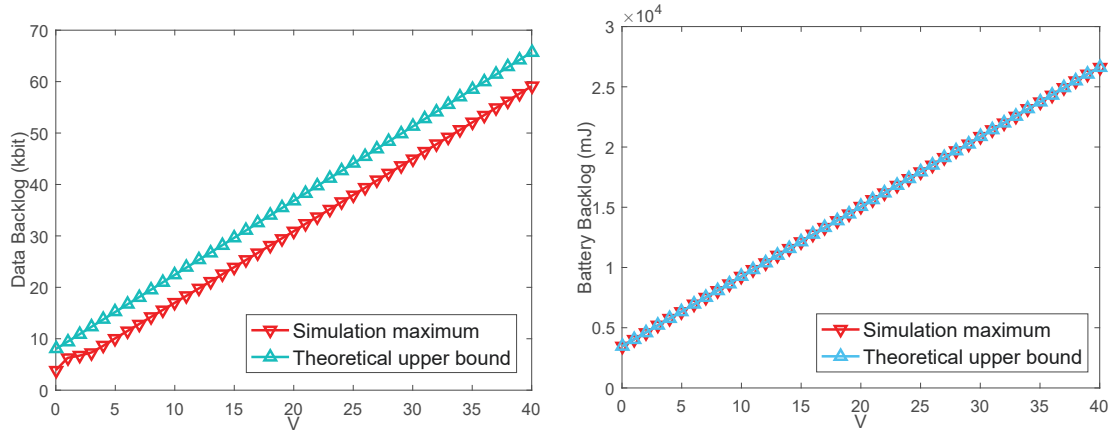
number of self-nominating devices is about twice the number of devices to be optimally scheduled. It is interesting to see in Fig. 5.2(b) that there is a surge of throughput around $N = 200$, after which the number of self-nominating devices declines with the increasing number of devices. The reason is that the total data arrival of all devices is low and likely to be instantly offloaded in the case of $N < 200$. In contrast, the total data arrival is so high in the case of $N > 200$, and the data queues build up at the devices. At every slot, the urgency of a small number of devices for offloading tasks becomes increasingly prominent, leading to a growth of $\underline{\alpha}_b(t)$ and subsequent decrease of self-nominating devices.

Fig. 5.3 plots the system throughput and fairness of the proposed approach, as V increases. Here, $V \in [0, 40]$ is displayed to provide a reasonable dynamic range to show the impact of V on both the queue length and throughput. We can see that both the throughput and Jain's index increase rapidly with V when $V \leq 5$, and then slow down increasing and start to stabilize when $V \geq 30$. As revealed in Theorem 3, the reason for this is that increasing V leads the utility to grow, which leverages both the throughput and fairness. For comparison, we also plot the throughput and Jain's index of RR and PF, both of which are independent of V and therefore remain unchanged. We can see that the proposed scheme approach is able to outperform RR and PF by 18% in terms of throughput and up to 4.5% in



(a) System throughput (b) Fairness among devices

Figure 5.3 : The system throughput and Jain's fairness index of the proposed approach, RR and PF, where V ranges from 0 to 40 and $N = 500$.



(a) Data backlog (b) Battery backlog

Figure 5.4 : The comparison of the simulation results on the maximum backlogs of the data and energy queues where V increases from 0 to 40, where $N = 500$.

terms of Jain's index; see $V = 40$.

Fig. 5.4 compares the simulation results on the maximum backlogs of the data and energy queues where V increases from 0 to 40. We can see that the maximum backlog of battery tightly fits its upper bound given in (6.10), while the maximum backlog of data is consistently lower than its upper bound by about 5 Kbits. This corroborates the upper bounds and the theorem – Theorem 7. The consistent differences between the backlogs and upper bounds also provide opportunities to predict

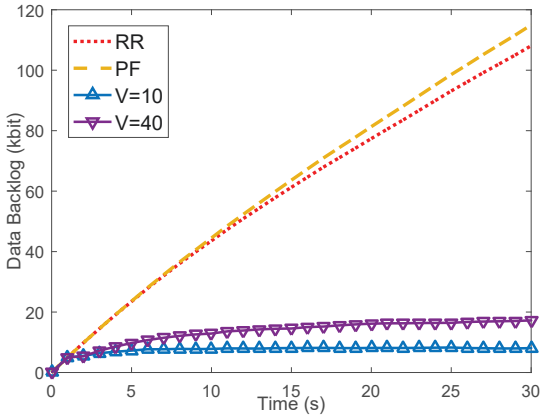
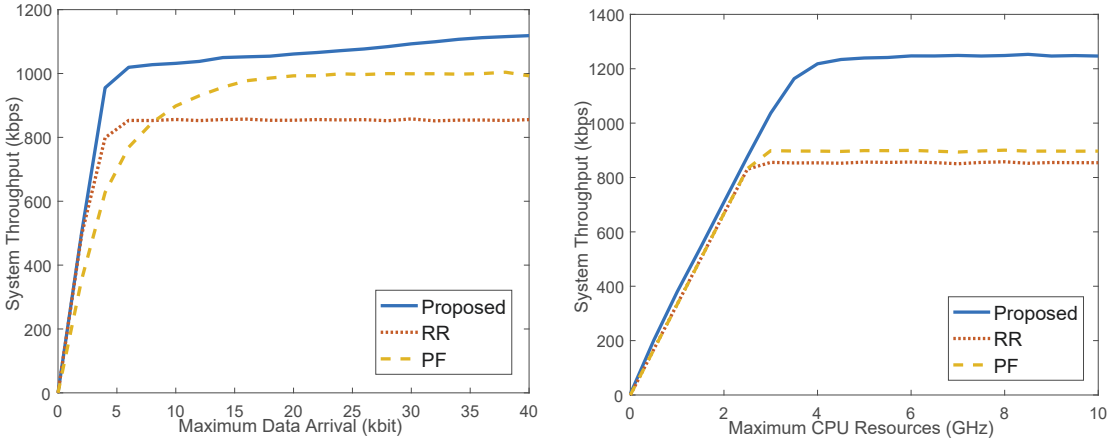


Figure 5.5 : The time variation of the average data backlog of the proposed approach, RR and PF along the time, where $N = 500$.

the value for V and the expected maximum utility by using the upper bounds (6.10), given the sizes of data buffer and battery at the devices.

Fig. 5.5 plots the time variation of the data backlog of the proposed approach in comparison with those of RR and RF along the time. We can see that the average data backlogs are able to quickly stabilize, as the time elapses. Moreover, a small value of V is able to further speed up the stabilization and also reduce the backlog, as compared to a large V value. This finding is consistent with Fig. 5.4. In Fig. 5.5, we also see that the average backlogs of both RR and PF continue increasing almost linearly to the time, and have yet to stable by the end of the simulation time. The superiority of the proposed approach in terms of system stability is confirmed.

Fig. 5.6 compares the throughput between the proposed approach, RR and PF, where the data arrival at the devices and the available computing resources at the MEC server are varied separately. We show in Fig. 5.6(a) that the throughput of the proposed approach increases with the growth of data arrival. Particularly, the throughput grows quickly and then starts to stabilize when $A_i^{\max} \geq 5$ kbps. This is because the data arrival of $A_i^{\max} = 5$ kbps starts to congest the network, and to saturate the offloading throughput. RR and PF also experience the congestion of the network and the saturation of the offloading throughput. However, the saturated throughput of RR and PF is much lower than that of the proposed approach. We can also see the network congestion and throughput saturation in Fig. 5.6(b), where the



(a) Throughput versus data arrival, where $F_{\max} = 3$ GHz

(b) Throughput versus CPU resources, where $A_i^{\max} = 10$ kbps

Figure 5.6 : The comparison of throughput between the proposed approach, RR and PF, versus data arrival and CPU resources, where $N = 500$ and $V = 40$.

computing resources are increasingly available at the MEC server and the offloading throughput is restrained by the wireless link capacity of the network.

Chapter 6

Distributed Fog Computing for Selfish Devices

This chapter presents a new distributed online optimization of fog computing, where selfish devices are encouraged to cooperate over unrestricted numbers of hops to minimize the time-average energy consumption on delay-tolerant, computationally demanding tasks. The contributions can be summarized as follows.

- The proposed approach optimizes cooperations of N selfish devices over unrestricted numbers of hops. The approach is fully distributed, where every device optimizes its own schedule of task processing and offloading, and result delivery with a time complexity of less than $\mathcal{O}(N^2)$. The schedules can collectively provide asymptotic optimality across the entire system.

- A new decentralized tit-for-tat mechanism is designed to incentivize selfish devices to collaborate. The tit-for-tat incentive of a device is the gap between the helps (measured by energy consumption) the device has received and offered; indicates how much help the device can offer at the next time slot; and can be evaluated at all devices based on simple signalling in which every device reports how much help it offered in the past time slot. With the tit-for-tat incentives of all other devices, each device can optimize its task processing and offloading based on the expected help from others.

- In the presence of multi-hop signalling delays, the aforementioned simple signalling can be delayed and the tit-for-tat incentives can be outdated. By taking the worst-case multi-hop signalling delays into account, we prove the optimality loss of the proposed approach resulting from outdated tit-for-tat incentives is upper bounded and can asymptotically diminish. The proposed approach preserves the asymptotic optimality under multi-hop signalling delays.

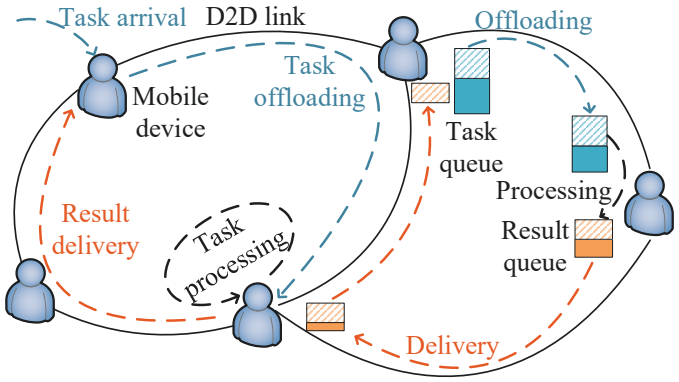


Figure 6.1 : An illustrative example of a distributed wireless network supporting fog computing.

Corroborated by extensive simulations, the proposed approach is able to substantially reduce the time-average energy consumption of the state of the art by 50%, and accommodate significantly more tasks by engaging selfish devices hops away.

6.1 System Model

Fig. 6.1 shows an illustrative example of distributed wireless networks supporting fog computing. The network under consideration consists of N mobile devices. $\mathbf{N} = \{1, \cdots, N\}$ collects the indexes for the devices. The computational tasks generated by each of the mobile devices can be either locally executed, or offloaded to other devices via Device-to-Device (D2D) links for fog computing. Different from most existing assumptions, we allow a task to be offloaded multiple hops away. Once executed, the result of the task is returned to the device generating the task. The notations used in this paper are summarized in Table 6.1.

The system run on a slotted basis with the slot length T. The devices can be synchronized by using existing time synchronization protocols, such as IEEE 1588 Precision Time Protocol (PTP) [60] and Timing-Sync Protocol for Sensor Networks (TPSN) [61]. Every pair of neighboring devices transmit to each other in the sequel, exchange their timestamps of the transmissions and receptions, measure their round-trip delay, and calibrate their time offset. The devices can be synchronized with a typical offset of microseconds. By running TPSN, the time synchronization precision

Table 6.1 : Definitions of Notations

Notation	Definition
$\mathbf{N}$	The set of mobile devices
T	The slot length
$A_i(t)$	The size of tasks generated by device i at slot t
ρ_i	The CPU cycles for processing a task bit of device i
ξ_i	The ratio of results to unprocessed tasks of device i in size
$C_{ij}(t)$	The capacity of link (i,j) during slot t
$\epsilon_{ij}(t)$	The energy consumption per bit of link (i,j) during slot t
$F_i(t)$	The available computing resources of device i at slot t
ϵ_i	The energy consumption per CPU cycle of device i
$Q_i^{(s)}(t)$	The task queue backlog at device i for device s at slot t
$D_i^{(s)}(t)$	The result queue backlog at device i for device s at slot t
$E_{ij}(t)$	The energy consumption on link (i,j) during slot t
$E_i(t)$	The energy for processing tasks of device i during slot t
$E(t)$	The overall energy consumption of all devices during slot t
$b_{ij}^{(s)}(t)$	The tasks of device s offloaded from device i to j at slot t
$d_{ij}^{(s)}(t)$	The results for device s returned via device i to j at slot t
$f_i^{(s)}(t)$	The resources of device i for device s's tasks at slot t
$E_i^j(t)$	The energy of device i to help device j's tasks at time slot t
$C_i(t)$	The energy of device i to help others at slot t
$S_i(t)$	The energy of other devices to help device i at slot t
$Z_i(t)$	The tit-for-tat virtual backlogs of device i at slot t
$\widehat{Z}_s^{(i)}(t)$	The out-of-date backlogs at device i for device s at slot t
$\delta_{\max}$	The diameter of stochastic connectivity graph over all slots

with errors of less than 2 μs can be stably achieved between two devices [62].

For a typical time slot of tens of milliseconds, such synchronization error of microseconds is negligible, and the time slots can be accurately configured among the devices. The schedule of time slots can be initialized by a nominated root device based on the clock time of the device. The devices within the one-hop range of the root device can synchronize with the root device by running TPSN. So on and so forth, all the devices can synchronize with the root device and configure the time slots accordingly.

We assume that the locations and wireless channels of the devices remain unchanged within a slot, and can change between slots. A stochastic graph $G(t) = \{\mathbf{N}, \mathcal{E}(t)\}$ is adopted to describe the topology of the network, where $\mathbf{N}$ is represented by vertexes and $\mathcal{E}(t) = \{(i,j)|e_{ij}(t) = 1, \forall i,j \in \mathbf{N}\}$ collects edges (i,j) between any pair of devices i and j with an active D2D link. $e_{ij}(t) = 1$ if devices i and j have a direct D2D link. Let $\epsilon_{ij}(t)$ denote the energy consumption per bit for the transmission from device i to j at time slot t. It is upper bounded by $\epsilon_{ij}^{\max}$, due to the finite transmit power of a device.

The capacity of the link during slot t, denoted by $C_{ij}(t)$, is assumed to be an independent and identically distributed (i.i.d.) stochastic process with an upper bound $C_{ij}^{\max}$. $C_{ij}(t)$ can be observed by monitoring the transmit (Tx) buffer for link (i,j) [63]. The re-transmission of data (i.e., tasks) is captured in $C_{ij}(t)$. This is because, by monitoring the Tx buffer for link (i,j), $C_{ij}(t)$ can adapt to the time-varying wireless channel conditions and account for re-transmissions following failed transmissions. The D2D link can be asymmetric, i.e., $C_{ij}(t) \neq C_{ji}(t)$, due to a potential hidden node problem.

Let $\widehat{F}_i$ denote the computational capability of device i (in CPU cycles per second), and $\delta_i(t)$ denote the percentage of background tasks at the device during slot t. Therefore, the available computational capacity of the device is $F_i(t) = [1-\delta_i(t)]\widehat{F}_i T$ for fog computing. Also let ϵ_i denote the energy consumption rate at device i (in Joules per CPU cycle).

A computational task (such as face recognition, augmented reality, and e-health services), generated by device i, can be characterized by a triplet $(A_i(t), \rho_i A_i(t), \xi_i A_i(t))$ [25]. $A_i(t)$ is the size of the task in bits generated at slot t, it requires $\rho_i A_i(t)$ CPU cycles to process, and the result is $\xi_i A_i(t)$ bits. ρ_i is the number of CPU cycles for processing a bit of the task, and $\rho_{\min}$ and $\rho_{\max}$ denote the minimum and maximum of the number, respectively. ξ_i is the ratio of results to the corresponding unprocessed tasks in terms of size with the maximum $\xi_{\max}$.

Note that $A_i(t)$ is an i.i.d. stochastic process with the maximum $A_i^{\max}$. The assumption of i.i.d. task arrivals has been widely adopted in the literature [23–

26, 64, 65] and experimentally validated [66]. For example, the task arrivals at a mobile device, such as mobile applications and background services, are typically modeled as Poisson process [23, 24]. Within non-overlapping equal-length time windows (such as time slots of a slotted system), the numbers of occurrences of a Poisson process are known to be i.i.d. [58]. Hence, the number of Poisson task arrivals per time slot is i.i.d. [23, 24].

For illustration convenience, we consider divisible tasks, also known as the data-partition model [35, Section II-A-2], [10, 23, 24, 64, 65], such as Gzip compression and feature extraction, for establishing the proposed approach and its asymptotic optimality. The tasks can be partitioned according to $C_{ij}(t)$ and $F_i(t)$ in (6.21) and (6.25). The proposed approach can also be applied to some statically partitioned tasks following the parallel dependency model [35, Fig. 4(b)], and preserve the asymptotic optimality, as will be discussed in Section 6.2.

We propose that each device maintains N queues for unprocessed tasks of all N devices (including the device itself) and another N queues for the corresponding results. The separate $2N$ queues enable all N devices to collaborate and ensure the results to be returned to the device generating the task. Each of the queues is scheduled on a First-In-First-Out (FIFO) basis. As per slot t, $Q_i^{(s)}(t)$ denotes the backlog of the queue, in which device i buffers unprocessed tasks originating from device s; and $D_i^{(s)}(t)$ denotes the backlog of the queue, in which device i buffers results destined for device s.

Fig. 6.2 illustrates the operations of an individual device in multi-hop fog computing, where there are two key decision gates. One of the decision gates is to decide the task to be executed based on the cost-effectiveness measure $\alpha_i^{(s)}(t)$, as to be given in (6.21). The other decision gate is to decide the transmission of unprocessed tasks or processed results over link (i, j) based on the cost-effectiveness measures $\beta_{ij}^{(s)}(t)$ and $\gamma_{ij}^{(s)}(t)$, as to be given in (6.25) and (6.26), respectively.

We also design a tit-for-tat mechanism based on the time-average energy consumption (or in other words, the accumulative energy consumption). Let $E_i^j(t) = \epsilon_i f_i^{(j)}(t) + \sum_{r \neq i} \epsilon_{ir}(t)[b_{ir}^{(j)}(t) + d_{ir}^{(j)}(t)]$ be the energy that device i consumes to help

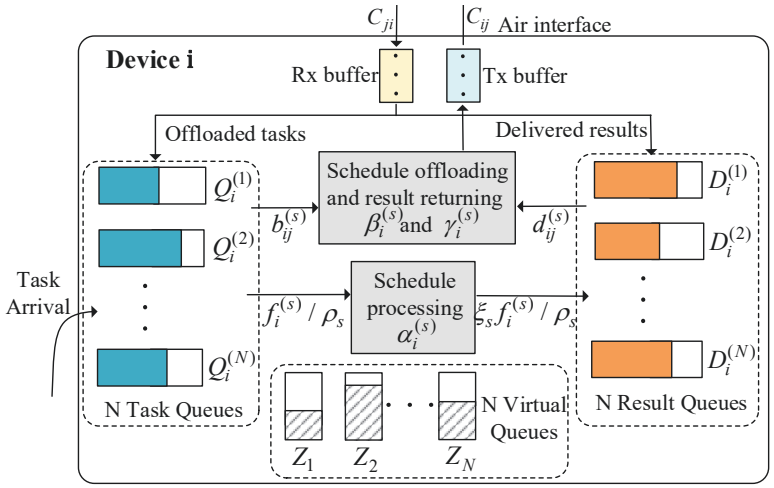


Figure 6.2 : An illustration on task processing and offloading, and the return of results in the proposed approach. The virtual queues are designed to capture the tit-for-tat incentives which motivate selfish devices to cooperate, as will be introduced in Section 6.2.

device j's tasks, including those on task execution and transmission, at time slot t, $i \neq j$. Let $C_i(t)$ specify the energy consumption of device i to help other devices at time slot t:

$$C_i(t) = \sum\nolimits_{r \neq i} E_i^r(t). \tag{6.1}$$

Let $S_i(t)$ specify the energy consumption of other devices to help device i at time slot t:

$$S_i(t) = \sum\nolimits_{r \neq i} E_r^i(t). \tag{6.2}$$

The tit-for-tat mechanism requires $\overline{S_i(t)} \leq \overline{C_i(t)}$, $\forall i \in \mathbf{N}$, 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 $X(t)$. In this sense, how much help a device can get from others (measured by a unified metric of energy) depends on how much the device has offered to help others with their tasks.

Motivated by future helps from others when needed, a selfish device offers to help when it is lightly loaded and has spare resources. By this means, participations in fog computing are incentivized, and selfish behaviors are discouraged. From (6.1) and (6.2), device i needs to have the knowledge on the energy consumption of other devices to update the tit-for-tat incentives of itself and the others for the decision

gates of task processing, offloading and result delivery in Fig. 6.2. The information needs to propagate hop by hop from the other devices until reaching device i. With the increase of network size, the information can be increasingly out-of-date due to multi-hop propagation delays, and so do the tit-for-tat incentives.

6.2 Distributed Online Optimization of Fog Computing for Selfish Devices

In this section, we present the new asymptotically optimal, distributed online optimization for the multi-hop offloading and processing of tasks with guaranteed result retrieval, provided the up-to-date information is always available at each device. As discussed in Section 6.1, the tit-for-tat incentives can be out-of-date subject to multi-hop propagation delays. In Section 6.3, we will prove that the optimality loss due to the out-of-date tit-for-tat incentives is bounded, and can diminish to preserve the asymptotic optimality.

6.2.1 Problem Statement

In fog computing, energy consumption provides a unified measure for the cost of both computations and transmissions. Also, mobile devices are energy restrained. For these reasons, we aim to reduce the overall time-average energy consumption of fog computing. The overall energy consumption of all the devices at slot t, denoted by $E(t)$, can be written as

$$E(t) = \sum\nolimits_{i\in\mathbf{N}} \sum\nolimits_{j\in\mathbf{N}} E_{ij}(t) + \sum\nolimits_{i\in\mathbf{N}} E_i(t), \tag{6.3}$$

where $E_{ij}(t) = \epsilon_{ij}(t) \sum_{s\in\mathbf{N}} (b_{ij}^{(s)}(t) + d_{ij}^{(s)}(t))$ is the energy required to forward tasks and return results on link (i, j) during slot t, and $E_i(t) = \epsilon_i \sum_{s\in\mathbf{N}} f_i^{(s)}(t)$ is the energy required to execute tasks at device i during the time slot.

Consider the stochastic nature of wireless channels, traffic arrivals and device mobility. We propose to minimize the overall time-average energy consumption of fog computing while stabilizing the queues of all the devices. The problem of interest

can be formulated as

$$\begin{aligned}
\mathbf{P}: & \min_{\mathbb{B},\mathbb{D},\mathbb{F}} \overline{E(t)} \\
& \text{s.t. } \mathbf{C1}: \ \textstyle\sum_{s\in\mathbf{N}} f_i^{(s)}(t) \le F_i(t),\ \forall i \in \mathbf{N}, t \in \mathbb{T} \\
& \quad \mathbf{C2}: \ \textstyle\sum_{s\in\mathbf{N}} [b_{ij}^{(s)}(t) + d_{ij}^{(s)}(t)] \le C_{ij}(t) e_{ij}(t) T, \\
& \qquad \forall i, j \in \mathbf{N}, t \in \mathbb{T} \\
& \quad \mathbf{C3}: \ \overline{S_i(t)} \le \overline{C_i(t)},\ \forall i \in \mathbf{N} \\
& \quad \mathbf{C4}: \ f_i^{(s)}(t), b_{ij}^{(s)}(t), d_{ij}^{(s)}(t) \ge 0,\ \forall i, j, s \in \mathbf{N}, t \in \mathbb{T} \\
& \quad \mathbf{C5}: \ \overline{Q_i^{(s)}(t)} < \infty, \overline{D_i^{(s)}(t)} < \infty,\ \forall i, s \in \mathbf{N}
\end{aligned} \tag{6.4}$$

where $b_{ij}^{(s)}(t)$ specifies the size of unprocessed tasks originating from device s and forwarded to device j through device i at slot t, $d_{ij}^{(s)}(t)$ specifies the size of results returned from device i to s through device j at slot t, and $f_i^{(s)}$ is the CPU cycles of device i allocated to the tasks originating from device s at slot t. Let $\mathbb{B} = \{b_{ij}^{(s)}(t) | i, j, s \in \mathbf{N}, t \in \mathbb{T}\}$, $\mathbb{D} = \{d_{ij}^{(s)}(t) | i, j, s \in \mathbf{N}, t \in \mathbb{T}\}$ and $\mathbb{F} = \{f_i^{(s)}(t) | i, s \in \mathbf{N}, t \in \mathbb{T}\}$ collect these variables over all time slots $\mathbb{T} = \{0, 1, \cdots\}$.

Constraint **C1** ensures that the allocated CPU cycles do not exceed the available resources at any device and any time slot. **C2** restrains that the total transmit rate of both tasks and results over a D2D link does not exceed the capacity of the link. **C3** captures the tit-for-tat mechanism to motivate selfish devices to cooperate. **C4** specifies that all the variables to be optimized are non-negative. **C5** requires that all the queues at the devices are stable (i.e., the sizes of the queues must stay finite per slot).

Note that the optimal solution for **P** would require the a-priori knowledge on task variations, channel fluctuations, mobility, and computing resources across all slots. On the other hand, myopic optimization of every slot would substantially increase the energy consumption since variables are coupled in time. In this paper, we decouple the variables between time slots, by converting **P** to a Lyapunov optimization problem which can asymptotically minimize the time-average energy consumption while stabilizing the network (i.e., stabilizing the backlogs of the FIFO queues at all devices).

6.2.2 Distributed Online Operations via Lyapunov Optimization

The Lyapunov optimization is a widely adopted stochastic optimization technique to decouple temporally and spatially coupled variables, derive asymptotically optimal solutions [57], and has been extensively used for smart grid [67], network function virtualization [68], as well as mobile edge computing [64]. The key idea of Lyapunov optimization is to minimize the upper bound of a drift-plus-penalty function per slot, and achieve an $[\mathcal{O}(V), \mathcal{O}(1/V)]$-tradeoff between the drift of Lyapunov function (i.e., the queue lengths) and the penalty (i.e., the time-average energy consumption to be minimized).

A quadratic Lyapunov function $L(t)$ can be defined to measure the stability of the queue, as given by [57]

$$L(t) = \frac{1}{2} \sum\nolimits_{i \in \mathbf{N}} \left\{ Z_i(t)^2 + \sum\nolimits_{s \in \mathbf{N}} \left[Q_i^{(s)}(t)^2 + D_i^{(s)}(t)^2 \right] \right\}. \tag{6.5}$$

The Lyapunov function becomes large, as the queues move towards unstable states. The stability of the queues can be achieved by taking control actions that harness the Lyapunov function per slot t [67].

In (6.5), we define a set of new virtual queues to capture the resource tit-for-tat constraint **C3**. At any device i and time slot t, the backlog of the virtual queue, denoted by $Z_i(t)$, can be defined as [57]

$$Z_i(t+1) = Z_i(t) + S_i(t) - C_i(t), \tag{6.6}$$

where $Z_i(0) = 0$, $\forall i \in \mathbf{N}$. According to [57], the stability of $Z_i(t)$ is the sufficient condition for the satisfaction of **C3**. **C3** can be relaxed by guaranteeing the stability of the virtual queues, i.e., $\overline{Z_i(t)} < \infty$, $\forall i \in \mathbf{N}$.

The backlog of unprocessed tasks $Q_i^{(s)}(t)$ is updated by

$$Q_i^{(s)}(t+1) \leq \max\{Q_i^{(s)}(t) - f_i^{(s)}(t)/\rho_s - \sum_{j \in \mathbf{N}} b_{ij}^{(s)}(t), 0\} + \sum\nolimits_{j \in \mathbf{N}} b_{ji}^{(s)}(t) + A_i^{(s)}(t), \tag{6.7}$$

where $f_i^{(s)}(t)/\rho_s$ is the number of bits that can be processed from the queue at time slot t, and $A_i^{(s)}(t)$ is the number of unprocessed task bits arriving at 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) gives the remaining unprocessed tasks in the queue at the end of time slot t, after part of the tasks have been processed at device i or offloaded to other devices. The other two terms give the new task arrivals offloaded from other devices or generated locally at device i.

The backlog of results $D_i^{(s)}(t)$ is updated by

$$D_i^{(s)}(t+1) \leq \max\{D_i^{(s)}(t) - \textstyle\sum_{j\in\mathbf{N}} d_{ij}^{(s)}(t), 0\} + \sum_{j\in\mathbf{N}} d_{ji}^{(s)}(t) + \xi_s f_i^{(s)}(t)/\rho_s, \; \forall s \neq i, \tag{6.8}$$

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

The inequalities in (6.7) and (6.8) are due to the fact that some devices may not have enough tasks or results to fulfill the rate that they accept. Nevertheless, the difference between the left-hand side (LHS) and the RHS of the inequalities marginalizes as tasks increase, and hence the queues saturate.

A drift-plus-penalty function can be defined to minimize the long-term energy consumption while stabilizing the system, as given by [57]

$$\Delta_V(t) = \Delta(t) + V\mathbb{E}\left[E(t)\right] \tag{6.9}$$

where $\Delta(t) \triangleq \mathbb{E}\left[L(t+1) - L(t)\right]$ is the drift of the Lyapunov function, $\mathbb{E}\left[E(t)\right]$ is the overall energy consumption at slot t, and V is a predefined non-negative coefficient to tune the tradeoff between the queue lengths and energy consumption.

By minimizing the upper bound of $\Delta_V(t)$ per slot t, we can decouple the variables in time, stochastically minimize the time-average energy consumption, and stabilize the queues with asymptotic optimality. The upper bound of $\Delta_V(t)$ is established in the following Lemma.

Lemma 10. *For any queue backlog and actions, $\Delta_V(t)$ is upper bounded by*

$$\begin{aligned}\Delta_V(t) \leq\ & U + V\mathbb{E}\left[E(t)\right] + \sum_{i\in\mathbf{N}} Z_i(t)\mathbb{E}[S_i(t) - C_i(t)] \\ &+ \sum_{i,s\in\mathbf{N}} Q_i^{(s)}(t)\mathbb{E}[A_i^{(s)}(t) - f_i^{(s)}(t)/\rho_s + \sum_{j\in N}(b_{ji}^{(s)}(t) - b_{ij}^{(s)}(t))] \\ &+ \sum_{i,s\in\mathbf{N}} D_i^{(s)}(t)\mathbb{E}[\xi_s f_i^{(s)}(t)/\rho_s + \sum_{j\in N}(d_{ji}^{(s)}(t) - d_{ij}^{(s)}(t)),\end{aligned} \tag{6.10}$$

where

$$\begin{aligned}U =\ & \tfrac{1}{2}\{\textstyle\sum_{i\in\mathbf{N}}[(\xi_{\max}+1)\widehat{F}_i T/\rho_{\min} + A_i^{\max} + 2\sum_{j\in\mathbf{N}} C_{ij}^{\max}T]\}^2 \\ &+\{\textstyle\sum_{i\in\mathbf{N}}\epsilon_i\widehat{F}_i T + \sum_{j\in\mathbf{N}}\epsilon_{ij}^{\max}C_{ij}^{\max}T\}^2.\end{aligned} \tag{6.11}$$

Proof. Taking squares on both sides of (6.7) and (6.8), 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 can obtain

$$\begin{aligned}&[Q_i^{(s)}(t+1)]^2 - [Q_i^{(s)}(t)]^2 \leq [f_i^{(s)}(t)/\rho_s + \textstyle\sum_{j\in\mathbf{N}} b_{ij}^{(s)}(t)]^2 \\ &+[\textstyle\sum_{j\in\mathbf{N}} b_{ji}^{(s)}(t) + A_i^{(s)}(t)]^2 + 2Q_i^{(s)}(t)[\textstyle\sum_{j\in\mathbf{N}}(b_{ji}^{(s)}(t) - b_{ij}^{(s)}(t)) + A_i^{(s)}(t) - f_i^{(s)}(t)/\rho_s],\end{aligned} \tag{6.12}$$

$$\begin{aligned}&[D_i^{(s)}(t+1)]^2 - [D_i^{(s)}(t)]^2 \leq [\textstyle\sum_{j\in\mathbf{N}} d_{ji}^{(s)}(t) + \xi_s f_i^{(s)}(t)/\rho_s]^2 \\ &+[\sum_{j\in\mathbf{N}} d_{ij}^{(s)}(t)]^2 + 2D_i^{(s)}(t)[\xi_s f_i^{(s)}(t)/\rho_s + \sum_{j\in\mathbf{N}}(d_{ji}^{(s)}(t) - d_{ij}^{(s)}(t))].\end{aligned} \tag{6.13}$$

From (6.6), we have

$$Z_i(t+1)^2 - Z_i(t)^2 \leq C_i(t)^2 + S_i(t)^2 + 2Z_i(t)[S_i(t) - C_i(t)]. \tag{6.14}$$

Plugging (6.12)–(6.14) into (6.9), we can achieve (6.10) with U satisfying

$$\begin{aligned}U \geq\ & \tfrac{1}{2}\sum_{i,s\in N}\{\mathbb{E}[(f_i^{(s)}(t)/\rho_s + \textstyle\sum_{j\in\mathbf{N}} b_{ij}^{(s)}(t))^2] + \mathbb{E}[(\sum_{j\in\mathbf{N}} d_{ij}^{(s)}(t))^2] \\ &+\mathbb{E}[(\sum_{j\in\mathbf{N}} b_{ji}^{(s)}(t) + A_i^{(s)}(t))^2] + \mathbb{E}[(\sum_{j\in\mathbf{N}} d_{ji}^{(s)}(t) + \xi_s f_i^{(s)}(t)/\rho_s)^2]\} \\ &+\tfrac{1}{2}\textstyle\sum_{i\in\mathbf{N}}\{\mathbb{E}[C_i(t)^2] + \mathbb{E}[S_i(t)^2]\}.\end{aligned} \tag{6.15}$$

Exploiting the inequality $(\sum_i a_i)^2 \geq \sum_i a_i^2$ for $\forall a_i \geq 0$, (6.11) provides an upper bound for (6.15) by replacing the expectations with their maximums. □

Now, problem **P** can be transformed to minimizing (6.10) at each time slot t, subject to the instantaneous constraints **C1**, **C2** and **C4**. Let $\mathbf{b}(t)$, $\mathbf{d}(t)$ and $\mathbf{f}(t)$

collect $b_{ij}^{(s)}(t)$, $d_{ij}^{(s)}(t)$ and $f_i^{(s)}(t)$, $\forall i, j \in \mathbf{N}$, respectively. By rearranging (6.10), $\mathbf{P}$ can be rewritten as

$$\begin{aligned} \mathbf{P1:} \max_{\mathbf{b}(t),\mathbf{d}(t),\mathbf{f}(t)} & \; g(\mathbf{f}(t)) + \phi(\mathbf{b}(t), \mathbf{d}(t)) \\ \text{s.t. } & \mathbf{C1}, \mathbf{C2} \text{ and } \mathbf{C4}, \end{aligned} \tag{6.16}$$

where

$$\begin{aligned} g(\mathbf{f}(t)) = \textstyle\sum_{i\in\mathbf{N}} \sum_{s\in\mathbf{N}} \{ & f_i^{(s)}(t)[Q_i^{(s)}(t)/\rho_s - D_i^{(s)}(t)\xi_s/\rho_s] \\ & -V\epsilon_i f_i^{(s)}(t) - Z_i(t)[\epsilon_s f_s^{(i)}(t) - \epsilon_i f_i^{(s)}(t)]\}; \end{aligned} \tag{6.17}$$

$$\begin{aligned} &\phi(\mathbf{b}(t), \mathbf{d}(t)) = \textstyle\sum_{i,j,s\in\mathbf{N}} \{-V\epsilon_{ij}(t)(b_{ij}^{(s)}(t) + d_{ij}^{(s)}(t)) \\ &-Q_i^{(s)}(t)[b_{ji}^{(s)}(t) - b_{ij}^{(s)}(t)] - D_i^{(s)}(t)[d_{ji}^{(s)}(t) - d_{ij}^{(s)}(t)] \\ &-Z_i(t)[\epsilon_{sj}(t)(b_{sj}^{(i)}(t) + d_{sj}^{(i)}(t)) - \epsilon_{ij}(t)(b_{ij}^{(s)}(t) + d_{ij}^{(s)}(t))]\}. \end{aligned} \tag{6.18}$$

Both U and $A_i^{(s)}(t)$ are independent of $\mathbf{b}(t)$, $\mathbf{d}(t)$ and $\mathbf{f}(t)$, and suppressed in $\mathbf{P1}$.

Problem $\mathbf{P1}$ can be efficiently solved by separately maximizing $g(\mathbf{f}(t))$ subject to $\mathbf{C1}$ and $\mathbf{C4}$, as reformulated in $\mathbf{P2}$, and $\phi(\mathbf{b}(t), \mathbf{d}(t))$ subject to $\mathbf{C2}$ and $\mathbf{C4}$, as reformulated in $\mathbf{P3}$. This is because $\mathbf{f}(t)$ can be decoupled from $\mathbf{b}(t)$ and $\mathbf{d}(t)$ in both the objective and constraints of $\mathbf{P1}$.

Computing Resource Allocation

From (6.17), the optimization of computing resources can be decoupled from each other. $\mathbf{P2}$ is formulated to optimize $\mathbf{f}_i(t) = \{f_i^{(s)}(t), \forall s \in \mathbf{N}\}$ for each device i, as given by

$$\begin{aligned} \mathbf{P2:} \max_{\mathbf{f}_i(t)} & \textstyle\sum_{s\in\mathbf{N}} \alpha_i^{(s)}(t) f_i^{(s)}(t) \\ \text{s.t. } & \textstyle\sum_{s\in\mathbf{N}} f_i^{(s)}(t) \le F_i(t) \\ & f_i^{(s)}(t) > 0, \forall s \in \mathbf{N} \end{aligned} \tag{6.19}$$

where

$$\alpha_i^{(s)}(t) = -V\epsilon_i + [Q_i^{(s)}(t) - \xi_s D_i^{(s)}(t)]/\rho_s + [Z_i(t) - Z_s(t)]\epsilon_i. \tag{6.20}$$

Apparently, $\mathbf{P2}$ is linear programming in the form of weighted-sum maximization [69], where $\alpha_i^{(s)}(t)$ serves as the weight for $f_i^{(s)}(t)$. The optimal solution can be

readily given by

$$f_i^{(s)}(t) = \begin{cases} F_i(t), \text{ if } s = \arg\max_j \alpha_i^{(j)}(t) \text{ and } \alpha_i^{(s)}(t) > 0; \\ 0, \text{ otherwise.} \end{cases} \tag{6.21}$$

(6.20) and (6.21) indicate that all the available computing resources of device i, $F_i(t)$, are allocated to the queue with the most urgency or the most incentive to be processed.

Offloading and Routing Decisions

P3 is formulated to optimize the decisions of task offloading and result retrieving for each individual D2D link. This is because $\phi(\mathbf{b}(t), \mathbf{d}(t))$ can be decoupled between D2D links, and **C2** specifies the capacity of each individual D2D link. Consider $\mathbf{b}_{ij}(t) = \{b_{ij}^{(s)}(t), \forall s \in \mathbf{N}\}$ and $\mathbf{d}_{ij}(t) = \{d_{ij}^{(s)}(t), \forall s \in \mathbf{N}\}$ for each D2D link (i, j). **P3** can be written as

$$\begin{aligned} \textbf{P3:} \max_{\mathbf{b}_{ij}(t), \mathbf{d}_{ij}(t)} & \textstyle\sum_{s\in\mathbf{N}} \beta_{ij}^{(s)}(t) b_{ij}^{(s)}(t) + \gamma_{ij}^{(s)}(t) d_{ij}^{(s)}(t) \\ \text{s.t.} & \textstyle\sum_{s\in\mathbf{N}} [b_{ij}^{(s)}(t) + d_{ij}^{(s)}(t)] \leq C_{ij}(t) e_{ij}(t) T \\ & b_{ij}^{(s)}(t), d_{ij}^{(s)}(t) \geq 0, \forall s \in \mathbf{N} \end{aligned} \tag{6.22}$$

where

$$\beta_{ij}^{(s)}(t) = -V\epsilon_{ij}(t) + [Q_i^{(s)}(t) - Q_j^{(s)}(t)] + [Z_i(t) - Z_s(t)]\epsilon_{ij}(t); \tag{6.23}$$

$$\gamma_{ij}^{(s)}(t) = -V\epsilon_{ij}(t) + [D_i^{(s)}(t) - D_j^{(s)}(t)] + [Z_i(t) - Z_s(t)]\epsilon_{ij}(t). \tag{6.24}$$

P3 is also a weighted-sum maximization like **P2** [69], and its optimal solution can be obtained by evaluating the weights $\beta_{ij}^{(s)}(t)$ and $\gamma_{ij}^{(s)}(t)$. In specific, if $\beta_{ij}^{(s)}(t) < 0$ and $\gamma_{ij}^{(s)}(t) < 0$, $\forall s \in \mathbf{N}$, device i neither offloads unprocessed tasks originating from device s, nor returns results destined for device s, through device j. If $\max_{s\in\mathbf{N}}\{\beta_{ij}^{(s)}(t)\} > \max_{s\in\mathbf{N}}\{\gamma_{ij}^{(s)}(t)\}$, device i does not return results through device j, i.e., $d_{ij}^{(s)}(t) = 0$, $\forall s \in \mathbf{N}$, and the size of the unprocessed tasks that device

i forwards to device j is given by

$$b_{ij}^{(s)}(t) = \begin{cases} C_{ij}(t)e_{ij}(t)T, \text{ if } s = \arg\max_r \beta_{ij}^{(r)}(t); \\ 0, \text{ otherwise.} \end{cases} \tag{6.25}$$

Otherwise, if $\max_{s\in\mathbf{N}}\{\beta_{ij}^{(s)}(t)\} \leq \max_{s\in\mathbf{N}}\{\gamma_{ij}^{(s)}(t)\}$, device i does not forward unprocessed tasks to device j, i.e., $b_{ij}^{(s)}(t) = 0$, $\forall s \in \mathbf{N}$, and the size of the results that device i returns through device j is given by

$$d_{ij}^{(s)}(t) = \begin{cases} C_{ij}(t)e_{ij}(t)T, \text{ if } s = \arg\max_r \gamma_{ij}^{(r)}(t); \\ 0, \text{ otherwise.} \end{cases} \tag{6.26}$$

In this sense, given a link, the queue which can make the most use of the link at time slot t is activated to transmit unprocessed tasks or results.

Distributed Implementations

Both problems **P2** and **P3** are weighted-sum maximizations [69]. According to (6.21), (6.25) and (6.26), the queues with the maximum weights $\alpha_i^{(s)}(t)$, $\beta_{ij}^{(s)}(t)$ and $\gamma_{ij}^{(s)}(t)$ are selected to be processed, or activated to occupy the link to transmit unprocessed tasks or results, respectively. In this sense, the weights are the cost-effectiveness measures to efficiently and asymptotically minimize the time-average energy consumption of fog computing. The selfish behaviors are discouraged by the cost-effectiveness measures, where a selfish device i would not receive help from another device j, given $\alpha_j^{(i)}(t) < 0$, $\beta_{ji}^{(i)}(t) < 0$ and $\gamma_{ji}^{(i)}(t) < 0$ with the virtual queue $Z_i(t) \gg 0$.

From (6.21), (6.25) and (6.26), we can see that the proposed online optimization is distributed, where the optimal decisions can be made locally at every individual server based on the cost-effectiveness measures. Except for the tit-for-tat incentives $Z_s(t)$, the cost-effectiveness measures only require the local knowledge that a node can acquire from itself and its immediate neighboring nodes. Particularly, device i can acquire the queue backlogs of its immediate neighbor j, i.e., $Q_j^{(s)}(t)$ and $D_j^{(s)}(t)$, and the capacity $C_{ij}(t)$ and cost $\epsilon_{ij}(t)$ of the link between device i itself and device

j. This information can be instantly obtained as part of the process of the neighbor discovery protocol (NDP) [70].

Remark: *The proposed optimization of fog computing has a sub-quadratic computational complexity of less than* $\mathcal{O}(N^2)$.

Proof. At every time slot, each node i selects one queue out of its totally N task queues for local processing based on (6.21), with the computational complexity of $\mathcal{O}(N)$. The node also selects one of the $2N$ task and result queues for each of its K_i radio links for task offloading or result delivery based on (6.25) and (6.26), with the computational complexity of $\mathcal{O}(NK_i)$. Therefore, the overall computational complexity of the node is $\mathcal{O}(NK_i) \leq \mathcal{O}(N^2)$ per time slot, since $K_i \leq N$. This is much lower than the complexity of $\mathcal{O}(N^3)$ per time slot in the state of the art [25]. Moreover, given the K_i neighbors, the complexity of the proposed approach, $\mathcal{O}(NK_i)$, only grows linearly to the network scale N at any node i. The scalability of the proposed algorithm is substantially improved, as compared to the state of the art [25]. □

Calculation of Tit-for-tat Incentives

Recall that $E_i^j(t)$ is the energy that device i consumes to help device j's tasks at time slot t, $i \neq j$, and $Z_s(t)$ is the tit-for-tat incentive of device s and indicates how much energy the device can spend helping others at time slot t. $Z_s(t) = \sum_{\tau=0}^{t-1} \sum_{j \neq s} E_j^s(\tau) - \sum_{\tau=0}^{t-1} \sum_{j \neq s} E_s^j(\tau)$ is designed to be the difference between the total energy that the other devices have consumed to help device s by time slot $(t-1)$, i.e., $\sum_{\tau=0}^{t-1} \sum_{j \neq s} E_j^s(\tau)$, and the energy that device s has consumed to help others by time slot $(t-1)$, i.e., $\sum_{\tau=0}^{t-1} \sum_{j \neq s} E_s^j(\tau)$. Indicating how much help every device can offer at the upcoming time slot t, $Z_s(t)$, $\forall s$, is needed for every device i to make optimal decisions on task processing and offloading, and result delivery, based on (6.20), (6.23) and (6.24).

To achieve this, every device is designed to broadcast a short message at every time slot, to report how much energy the device consumed to help another device

at the last time slot. For example, device s processed tasks for device j ($j \neq s$) at the last time slot $(t-1)$, and reports the amount of energy it consumed on the tasks together with the identities of itself and device j. The message device s sends is $\{E_s^{(j)}(t-1), s, j\}$. In the absence of multi-hop signalling delays, any device i receives such message from every other device and updates its knowledge on the tit-for-tat incentives of all devices (including itself) for the upcoming time slot t, i.e., $Z_s(t) = Z_s(t-1) + \sum_{j \neq s} E_j^s(t-1) - \sum_{j \neq s} E_s^j(t-1)$, $\forall s \in \mathbf{N}$.

In practice, the message from any device s to device i needs to be forwarded hop-by-hop, and can be significantly delayed. As a consequence, the tit-for-tat incentive $Z_s(t)$ can be outdated. In Section 6.3, we will prove that the optimality loss due to the outdated tit-for-tat incentives is upper bounded, and the asymptotic optimality of our approach can be preserved.

6.2.3 Performance Analysis

Let $\overline{E^*(t)}$ denote the stochastically minimized time-average energy consumption of the entire system using distributed online optimization, and E^{opt} be the energy consumption of $\mathbf{P}$ minimized offline (in a posteriori manner) by omitting causality. The gap between $\overline{E^*(t)}$ and E^{opt} can be shown to converge asymptotically with the growth of V, as revealed in the following theorem.

Theorem 6. *The gap between $\overline{E^*(t)}$ and E^{opt} is within U/V, i.e.,*

$$\overline{E^*(t)} \leq E^{\text{opt}} + U/V. \tag{6.27}$$

Proof. For brevity, the proof is suppressed here. Please refer to [57, Theorem 4.8] for the details. □

From Theorem 6, the proposed optimization of fog computing can asymptotically converge to the offline minimum that violates causality, as V increases. In other words, the proposed approach is asymptotically optimal. We can further show the queue lengths are linear to V in the following theorem.

Theorem 7. *All queues of the proposed distributed fog computing are strongly stable and the time-average backlogs of the queues satisfy*

$$\sum\nolimits_{i,s\in\mathbf{N}} \overline{Q_i^{(s)}(t)} + \overline{D_i^{(s)}(t)} \leq \frac{U}{\varepsilon} + \frac{V(E^{\varepsilon} - E^{\text{opt}})}{\varepsilon}, \tag{6.28}$$

where $\varepsilon > 0$ and $E^{\varepsilon} \geq E^{\text{opt}}$ satisfy the Slater condition [57].

Proof. By referring to [57, Theorem 4.8], there exist $\varepsilon > 0$, $E^{\varepsilon} \geq E^{\text{opt}}$, and the corresponding control policy π^* in the feasible region of the problem of interest $\mathbf{P}$, such that [57, Eqs. (4.61)-(4.64)]

$$\begin{aligned}
&\mathbb{E}[E(t)|\pi^*] = E^{\varepsilon},\\
&\mathbb{E}[S_i(t) - C_i(t)|\pi^*] \leq 0,\\
&\mathbb{E}[A_i^{(s)}(t) - f_i^{(s)}(t)/\rho_s + \textstyle\sum_{j\in\mathbf{N}}(b_{ji}^{(s)}(t) - b_{ij}^{(s)}(t))|\pi^*] \leq -\varepsilon,\\
&\mathbb{E}[\xi_s f_i^{(s)}(t)/\rho_s + \textstyle\sum_{j\in\mathbf{N}}(d_{ji}^{(s)}(t) - d_{ij}^{(s)}(t))|\pi^*] \leq -\varepsilon.
\end{aligned} \tag{6.29}$$

By plugging (6.29) into (6.10), we have

$$\Delta_V(t) \leq U + VE^{\varepsilon} - \varepsilon \textstyle\sum_{i,s\in\mathbf{N}}[Q_i^{(s)}(t) + D_i^{(s)}(t)], \tag{6.30}$$

where the RHS is a constant term ($U+VE^{\varepsilon}$) minus a positive term $\varepsilon \sum_{i,s\in\mathbf{N}}[Q_i^{(s)}(t)+ D_i^{(s)}(t)]$. U is obtained in Lemma 1. As compared to [57, Eq. (3.20)], VE^{ε} is an additional pre-configurable coefficient, accounts for the penalty in the drift-plus-penalty (9), and can asymptotically diminish as V decreases. As a result, as the queue backlogs $Q_i^{(s)}(t)$ and $Q_j^{(s)}(t)$ increase, $\varepsilon \sum_{i,s\in\mathbf{N}}[Q_i^{(s)}(t) + D_i^{(s)}(t)]$ becomes increasingly large. The RHS of (6.30) decreases and becomes negative, and so does the Lyapunov drift-plus-penalty $\Delta_V(t)$. This feature results in the Lyapunov stability and the boundlessness of the optimal solution.

By reorganizing, (6.30) can be rewritten as

$$\varepsilon \sum\nolimits_{i,s\in\mathbf{N}}[Q_i^{(s)}(t) + D_i^{(s)}(t)] \leq U + VE^{\varepsilon} - V\mathbb{E}[E(t)] - \mathbb{E}[L(t+1) - L(t)]. \tag{6.31}$$

Taking expectations on both sides of (6.31) and summing up the telescoping series over $\{0, 1, \cdots, T-1\}$, we have

$$\begin{aligned}
&\frac{1}{T}\lim_{T\to\infty}\sum\nolimits_{t=0}^{T-1}\sum\nolimits_{i,s\in\mathbf{N}}\{\mathbb{E}[Q_i^{(s)}(t)]+\mathbb{E}[D_i^{(s)}(t)]\}\\
&\leq \frac{U}{\varepsilon}+\frac{V(E^{\varepsilon}-E^{\mathrm{opt}})}{\varepsilon}+\frac{\mathbb{E}[L(0)]-\mathbb{E}[L(T)]}{\varepsilon T}\\
&\leq \frac{U}{\varepsilon}+\frac{V(E^{\varepsilon}-E^{\mathrm{opt}})}{\varepsilon},
\end{aligned}$$

where $E^{\mathrm{opt}} \leq \mathbb{E}[E(t)]$ is the offline optimum, and the second inequality is due to the fact that $\mathbb{E}[L(T)] \geq 0$ and $L(0) = 0$. This concludes the proof of Theorem 7. □

Theorems 6 and 7 reveal that there is an $[\mathcal{O}(V), \mathcal{O}(1/V)]$-tradeoff between the queue lengths and optimality gap. According to Little's Theorem [58], the queuing delay is proportional to the total queue lengths in a queuing system. As a result, the tradeoff between the queue lengths and optimality loss can be translated to a tradeoff between the delay and energy consumption in this paper. The tradeoff can be adjusted by configuring the parameter V.

The proposed approach can also be used for some statically partitioned tasks following the parallel dependency model [35, Fig. 4(b)], which can be processed independently of each other. This model has also been widely adopted to study fog computing in the literature [11, 12, 16–19, 21, 25, 26]. The size of a statically partitioned task, denoted by D_{task}, does not have to be fixed, and the maximum size is denoted by $D_{\mathrm{task}}^{\max}$. The approach can be readily applied by rounding the allocated computing resources $F_i(t)$ and link schedules $C_{ij}(t)$ to the largest integer numbers of atomic tasks that can be supported, i.e., $f_i^{(s)}(t) = D_{\mathrm{task}}\rho_k\lfloor\frac{F_i(t)}{D_{\mathrm{task}}\rho_k}\rfloor$ in (6.21) and $b_{ij}^{(s)}(t) = D_{\mathrm{task}}\lfloor C_{ij}(t)/D_{\mathrm{task}}\rfloor$ in (6.25). The rounding does not violate the asymptotic optimality in Theorem 6. This is because the optimality loss due to the rounding are $\alpha_i^{(s)}\rho_k D_{\mathrm{task}}^{\max}$ and $\beta_{ij}^{(s)}(t)D_{\mathrm{task}}^{\max}$, since the differences due to the rounding are at most $\rho_k D_{\mathrm{task}} \leq \rho_k D_{\mathrm{task}}^{\max}$ CPU cycles and $D_{\mathrm{task}} \leq D_{\mathrm{task}}^{\max}$ bits. The optimality loss is upper bounded by $\mathcal{O}(1)$ by using (6.28), since $\alpha_i^{(s)}$ and $\beta_{ij}^{(s)}(t)$ are linear to the product of V and the queue backlogs, hence preserving the asymptotic optimality in Theorem 6. The time slot length T can be adequately designed for efficient use

Algorithm 5 The Proposed Distributed Online Optimization under Outdated Network Knowledge

At each device i at the beginning of slot t:
1: Find neighboring devices and establish D2D pairs
2: Acquire $A_i(t)$, $F_i(t)$ and $\epsilon_{ij}(t)$, and queue backlogs, $Q_i^{(s)}(t)$ and $D_i^{(s)}(t)$
3: Exchange this information with the devices in D2D pairs
At each device i:
4: Evaluate the weights $\widehat{\alpha}_i^{(s)}(t)$, $\widehat{\beta}_{ij}^{(s)}(t)$ and $\widehat{\gamma}_{ij}^{(s)}(t)$ by (6.20), (6.23) and (6.24), using the out-of-date backlogs $\widehat{Z}_s^{(i)}(t)$
5: Allocate the computing resources according to (6.21)
6: Schedule the offloading and routing based on (6.25) and (6.26)
Virtual queue update at each device i:
7: Flood the energy consumption for helping others
8: Update the out-of-date backlogs $\widehat{Z}_s^{(i)}(t)$ by (6.35)

of the links. For instance, the slot length can be set as $T = D_{\max}/C_{\min}$, so that at least one task can be transmitted per time slot. $C_{\min}$ is the minimum capacity of an active link in the network.

6.3 Distributed Optimization under Out-of-Date Knowledge

In this section, we consider the tit-for-tat incentives can be outdated due to the multi-hop signalling delays, since the message on how much energy a device consumes to help others at slot t needs to be forwarded hop-by-hop, as dictated in Section 6.2.2. This is distinctively different from existing works under the assumption of instantaneous knowledge on the incentives at a central controller [25, 26].

By referring to [20, 22], the multi-hop signalling delay between devices i and j, τ_{ij}, can be given by

$$\tau_{ij} = \sum_{(i',j')\in\mathbf{P}_{ij}} T_{i'j'}^{\text{prop}} + T_{i'j'}^{\text{tx}} + T_{i'j'}^{\text{syn}} + T_{i'j'}^{\text{que}}, \tag{6.32}$$

where $T_{i'j'}^{\text{prop}}$, $T_{i'j'}^{\text{tx}}$, $T_{i'j'}^{\text{syn}}$ and $T_{i'j'}^{\text{que}}$ are the propagation delay, transmission delay, time synchronization error, and queuing delay over a one-hop link (i', j'), respectively; and $\mathbf{P}_{ij}$ is the route between devices i and j.

In practice, priority is typically given to signalling to be placed head-of-line (HOL)

for transmission. The signalling of interest is also small, only consisting of a few bytes indicating how much energy device i consumed to help another device at the last time slot, and the identities of device i and that device. Therefore, the queuing delay is negligible, i.e., $T^{\text{que}}_{i'j'} \approx 0$. The propagation and transmission delays, $T^{\text{prop}}_{i'j'}$ and $T^{\text{tx}}_{i'j'}$, are also negligible over wireless medium, as compared to the duration of a time slot T. The time synchronization between the devices is maintained by regularly running time synchronization protocols, such as TPSN [61]. The time synchronization error is typically less than 2 μs and negligible, i.e., $T^{\text{syn}}_{i'j'} \ll T$. As a result, $T^{\text{prop}}_{i'j'} + T^{\text{tx}}_{i'j'} + T^{\text{syn}}_{i'j'} + T^{\text{que}}_{i'j'} \leq T$. A device can readily forward the signalling at the next time slot after it has received the signalling:

$$\tau_{ij} \leq n_{ij} T, \tag{6.33}$$

where n_{ij} is the number of hops along the route between the devices. The multi-hop signalling delay can be potentially reduced by meticulous designs of the timing of the signalling, as done in [20, 22, 71]. As proved later, even without meticulous designs of the timing, the proposed approach can asymptotically converge to the global optimality.

Let $\widehat{Z}^{(i)}_s(t)$ denote the outdated tit-for-tat incentive of device s, which device i can obtain based on the outdated signalling received at time slot $(t-1)$. Based on (6.33), $\widehat{Z}^{(i)}_s(t)$ can be given by

$$\widehat{Z}^{(i)}_s(t) = \sum_{j \neq s} \sum_{\tau=0}^{t-n_{ij}} E^s_j(\tau) - \sum_{j \neq s} \sum_{\tau=0}^{t-n_{is}} E^j_s(\tau). \tag{6.34}$$

To evaluate the tolerance of the proposed algorithm against the multi-hop signal delays and the subsequently outdated tit-for-tat incentives, we consider the worst-case scenario where all the tit-for-tat incentives experience the longest possible delays $\delta_{\max}$. $\delta_{\max}$ is the diameter of the stochastic graph $G(t)$. $\delta_{\max} \leq N-1$ in N-node

connected graphs. In the worst-case scenario, (6.34) can be rewritten as

$$\begin{aligned}\widehat{Z}_s^{(i)}(t) &= \sum_{j\neq s}\sum_{\tau=0}^{t-\delta_{\max}} E_j^s(\tau) - \sum_{j\neq s}\sum_{\tau=0}^{t-\delta_{\max}} E_s^j(\tau) \\ &= \widehat{Z}_s^{(i)}(t-1) + \sum_{j\neq s} E_j^s(t-\delta_{\max}) - \sum_{j\neq s} E_s^j(t-\delta_{\max}).\end{aligned} \tag{6.35}$$

By comparing (6.35) to (6.6), we can see $\widehat{Z}_s^{(i)}(t) = Z_s(t-\delta_{\max})$. In the presence of multi-hop signalling delays, the asymptotic optimality of the proposed algorithm can be proved by evaluating the impact of the most severely outdated tit-for-tat incentives $Z_s(t-\delta_{\max})$.

Using the out-of-date incentives $\widehat{Z}_s^{(i)}(t)$, the out-of-date version of the cost-effectiveness measures, denoted by $\widehat{\alpha}_i^{(s)}(t)$, $\widehat{\beta}_{ij}^{(s)}(t)$ and $\widehat{\gamma}_{ij}^{(s)}(t)$, can still be independently evaluated at each device i using (6.20), (6.23) and (6.24). This does not violate the asymptotic optimality of the approach achieved under the assumption of up-to-date knowledge on incentives $Z_s(t)$ at every device, as will be shown in Theorem 8. The optimal decisions are based on the out-of-date cost-effectiveness measures. In particular, the computing resources are allocated to queue $Q_i^{(j)}$ with the maximum $\widehat{\alpha}_i^{(j)}$ according to (6.21), and the queue with the maximum $\widehat{\beta}_{ij}^{(s)}(t)$ and $\widehat{\gamma}_{ij}^{(s)}(t)$ is activated to transmit unprocessed tasks and results over link (i,j) based on (6.25) and (6.26). The proposed distributed optimization of fog computing under out-of-date knowledge is summarized in Algorithm 6.

We proceed to prove that the differences between the out-of-date and up-to-date tit-for-tat incentives are upper bounded at any time slot t, and so are the out-of-date and up-to-date cost-effectiveness measures. In turn, the optimality loss due to out-of-date knowledge is bounded at any time slot.

Lemma 11. *At any slot t, the difference between $Z_s(t)$ and $\widehat{Z}_s^{(i)}(t)$ is upper bounded by*

$$|Z_s(t) - \widehat{Z}_s^{(i)}(t)| \leq \delta_{\max}\theta, \tag{6.36}$$

where $\theta = \sum_{i\in\mathbf{N}}[\epsilon_i \widehat{F}_i + \sum_{j\in\mathbf{N}} \epsilon_{ij}^{\max} C_{ij}^{\max}]T$ is the maximum energy consumption of the network per slot, which depends on the size of the network N, and the upper bounds of the stochastic parameters, such as $\widehat{F}_i$, $\epsilon_{ij}^{\max}$ and $C_{ij}^{\max}$.

Proof. According to (6.6) and (6.35), the out-of-date backlog of a virtual queue at device i satisfies

$$\widehat{Z}_s^{(i)}(t) = Z_s(t - \delta_{\max}). \tag{6.37}$$

The difference of $Z_s(t)$ between consecutive slots t and $t+1$, i.e., $|Z_s(t+1) - Z_s(t)|$ accounts for the difference between the energy that device s consumes to help others and the energy that other devices consume to help device s. The inequality $|Z_s(t+1) - Z_s(t)| \leq \theta$ corresponds to an extreme case where all other devices consume all their energy to help device s at slot t. Then, we can obtain $|Z_s(t) - \widehat{Z}_s^{(i)}(t)| = |Z_s(t) - Z_s(t - \delta_{\max})| = |\sum_{\tau=t-\delta_{\max}}^{t-1} Z_s(\tau+1) - Z_s(\tau)|$. Exploiting the inequality $|a+b| \leq |a| + |b|$, we can prove $|Z_s(t) - \widehat{Z}_s^{(i)}(t)| = |\sum_{\tau=t-\delta_{\max}}^{t-1} Z_s(\tau+1) - Z_s(\tau)| \leq \sum_{\tau=t-\delta_{\max}}^{t-1} |Z_s(\tau+1) - Z_s(\tau)| \leq \delta_{\max}\theta$. □

Lemma 12. *The differences between the instant weights and the weights based on the out-of-date information satisfy*

$$\begin{aligned} &|\widehat{\alpha}_i^{(s)}(t) - \alpha_i^{(s)}(t)| \leq 2\epsilon_i \delta_{\max}\theta, \\ &|\widehat{\beta}_{ij}^{(s)}(t) - \beta_{ij}^{(s)}(t)| \leq 2\epsilon_{ij} \delta_{\max}\theta, \\ &|\widehat{\gamma}_{ij}^{(s)}(t) - \gamma_{ij}^{(s)}(t)| \leq 2\epsilon_{ij} \delta_{\max}\theta. \end{aligned} \tag{6.38}$$

Proof. Based on (6.20), we have $|\widehat{\alpha}_i^{(s)}(t) - \alpha_i^{(s)}(t)| \leq \epsilon_i\{|\widehat{Z}_i^{(i)}(t) - \widehat{Z}_s^{(i)}(t)| - |Z_i(t) - Z_s(t)|\}$. Exploiting the inequality $|a| - |b| \leq |a-b| \leq |a| + |b|$, we have

$$\begin{aligned} &|\widehat{Z}_i^{(i)}(t) - \widehat{Z}_s^{(i)}(t)| - |Z_i(t) - Z_s(t)| \leq |(\widehat{Z}_i^{(i)}(t) - Z_i(t)) - (\widehat{Z}_s^{(i)}(t) - Z_s(t))| \\ &\leq |\widehat{Z}_i^{(i)}(t) - Z_i(t)| + |\widehat{Z}_s^{(i)}(t) - Z_s(t)|. \end{aligned}$$

Substituting (6.36) into the RHS of the last inequality, we can prove that $|\widehat{\alpha}_i^{(s)}(t) - \alpha_i^{(s)}(t)| \leq 2\epsilon_i\delta_{\max}\theta$. Likewise, we can obtain $|\widehat{\beta}_{ij}^{(s)}(t) - \beta_{ij}^{(s)}(t)| \leq 2\epsilon_{ij}\delta_{\max}\theta$ and $|\widehat{\gamma}_{ij}^{(s)}(t) - \gamma_{ij}^{(s)}(t)| \leq 2\epsilon_{ij}\delta_{\max}\theta$, based on (6.23) and (6.24), respectively. □

We can show that the optimality loss of **P1**, stemming from the out-of-date knowledge, is bounded, as revealed in the following theorem.

Theorem 8. *The optimality loss of the solution for **P1**, resulting from the out-of-*

date knowledge on $Z_s(t)$, is within a constant C, i.e.,

$$g(\mathbf{f}^*(t)) + \phi(\mathbf{b}^*(t), \mathbf{d}^*(t)) - g(\mathbf{f}(t)) - \phi(\mathbf{b}(t), \mathbf{d}(t)) \leq C$$

*where $\{\mathbf{f}^*(t), \mathbf{b}^*(t), \mathbf{d}^*(t)\}$ is the solution to **P1** in the presence of instantaneous knowledge, $\{\mathbf{f}(t), \mathbf{b}(t), \mathbf{d}(t)\}$ is the solution under out-of-date knowledge, and $C = 4\delta_{\max}\theta[\sum_{i\in\mathbf{N}} \epsilon_i \widehat{F}_i + \sum_{i,j\in\mathbf{N}} \epsilon_{ij} C_{ij}^{\max}]T$.*

Proof. We first consider the optimality loss of **P2**, i.e., $g(\mathbf{f}^*(t)) - g(\mathbf{f}(t))$. From (6.17), we can obtain

$$\begin{aligned} &g(\mathbf{f}(t)) = \textstyle\sum_{i,s\in\mathbf{N}} \alpha_i^{(s)}(t) f_i^{(s)}(t) \\ &= \textstyle\sum_{i,s\in\mathbf{N}} \{\widehat{\alpha}_i^{(s)}(t) f_i^{(s)}(t) + [\alpha_i^{(s)}(t) - \widehat{\alpha}_i^{(s)}(t)] f_i^{(s)}(t)\}. \end{aligned} \tag{6.39}$$

Since $\mathbf{f}(t)$ is the optimal solution under $\widehat{\alpha}_i^{(s)}(t)$, and therefore, we can also obtain

$$\begin{aligned} &\textstyle\sum_{i,s\in\mathbf{N}} \widehat{\alpha}_i^{(s)}(t) f_i^{(s)}(t) \geq \sum_{i,s\in\mathbf{N}} \widehat{\alpha}_i^{(s)}(t) f_i^{(s)*}(t) \\ &= \textstyle\sum_{i,s\in\mathbf{N}} \{\alpha_i^{(s)}(t) f_i^{(s)*}(t) + [\widehat{\alpha}_i^{(s)}(t) - \alpha_i^{(s)}(t)] f_i^{(s)*}(t)\} \\ &= g(\mathbf{f}^*(t)) + \textstyle\sum_{i,s\in\mathbf{N}} [\hat{\alpha_i}^{(s)}(t) - \alpha_i^{(s)}(t)] f_i^{(s)*}(t), \end{aligned} \tag{6.40}$$

where the second equality is due to the fact that $g(\mathbf{f}^*(t)) = \sum_{i,s\in\mathbf{N}} \alpha_i^{(s)}(t) f_i^{(s)*}(t)$. Subtracting (6.40) from (6.39) and then substituting (6.38), we can obtain

$$\begin{aligned} &g(\mathbf{f}^*(t)) - g(\mathbf{f}(t)) \leq \textstyle\sum_{i,s\in\mathbf{N}} [\widehat{\alpha_i}^{(s)}(t) - \alpha_i^{(s)}(t)](f_i^{(s)}(t) \\ &+ f_i^{(s)*}(t)) \leq 4\delta_{\max}\theta \textstyle\sum_{i\in\mathbf{N}} \epsilon_i \widehat{F}_i T. \end{aligned} \tag{6.41}$$

Likewise, we can consider the optimality loss of **P3** and prove

$$\phi(\mathbf{b}^*(t), \mathbf{d}^*(t)) - \phi(\mathbf{b}(t), \mathbf{d}(t)) \leq 4\delta_{\max}\theta \sum\nolimits_{i,j\in\mathbf{N}} \epsilon_{ij} C_{ij}^{\max} T. \tag{6.42}$$

Adding up (6.41) and (6.42), we finally prove the theorem. □

From Theorems 6, 7, and 8, it can be readily established that the optimality

loss, resulting from the out-of-date knowledge $\widehat{Z}_s^{(i)}(t)$, is within $\frac{U+C}{V}$, i.e.,

$$\overline{E^*(t)} \leq E^{\text{opt}} + \frac{U+C}{V}. \tag{6.43}$$

The time-average backlogs of the queues satisfy

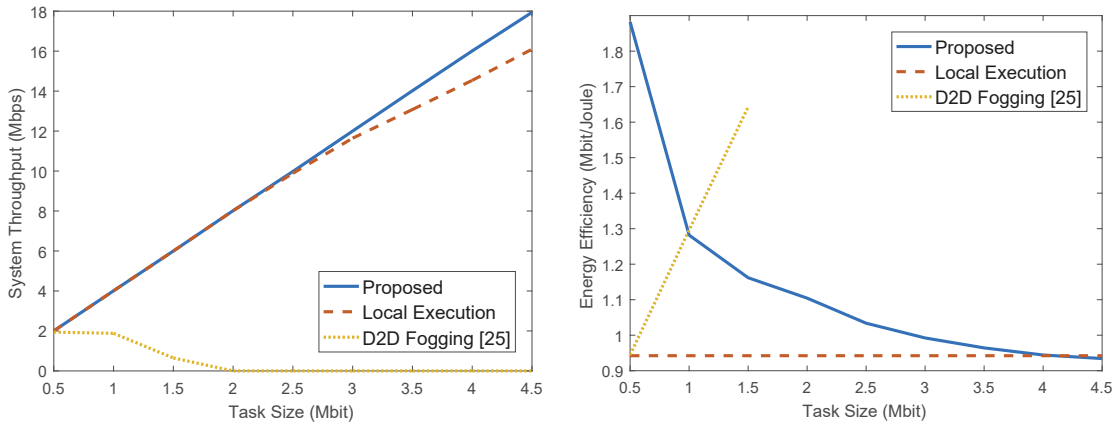
$$\sum\nolimits_{i,s\in\mathbf{N}} \overline{Q_i^{(s)}(t)} + \overline{D_i^{(s)}(t)} \leq \frac{U+C}{\varepsilon} + \frac{V(E^{\varepsilon} - E^{\text{opt}})}{\varepsilon}. \tag{6.44}$$

That is to say, the distributed optimization under out-of-date knowledge is still asymptotically optimal, with an $[\mathcal{O}(V), \mathcal{O}(1/V)]$-tradeoff.

6.4 Simulation Result and Analysis

In this section, we evaluate the proposed distributed online optimization of fog computing with $N = 20$ selfish devices in the presence of non-negligible multi-hop propagation delays in a custom-designed Matlab platform. Our Matlab simulator is focused on the allocation of physical resources for computing and transmission (in CPU cycles and bits), since any allocations of the so-called virtual resources would have to be translated to physical resource requirements and implemented physically [35]. The Matlab simulator allows for fair comparisons of the proposed approach to the state of the art [25] which was evaluated on a custom-designed platform. Particularly, we assume that all tasks are of the same type, and every device is equipped with a single processor, as assumed in [25]. The key step of the state of the art is [25, Eq. 18], which can be reliably solved by using the Matlab optimization toolbox.

We run 10000 slots simulations for each data point, with the slot duration $T =$ 1 sec. The CPU capacity of the mobile devices $\widehat{F}_i$ is uniformly distributed from 1 to 4 GHz with $\epsilon_i = 1/\widehat{F}_i$ Joule/cycle, the background task $\delta_i(t)$ uniformly and randomly varies from 0 to 40%, and the D2D transmission power is 0.2 W for each link. These simulation parameters are drawn from the dataset [25] for fair comparisons with the state of the art. The propagation delay is linear to the transmission hops, i.e., 0.2 sec/hop, and the connectivity ratio among devices p_c is set to be 0.25. The tasks



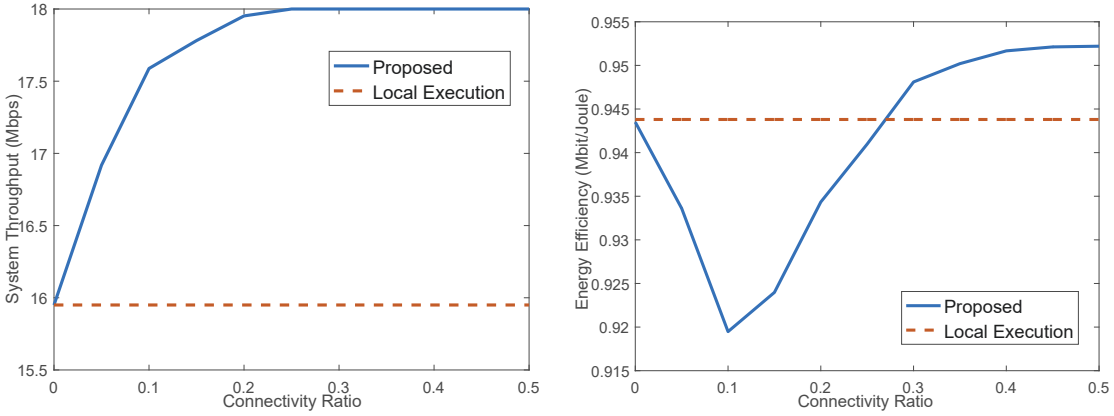
(a) System throughput (b) Energy efficiency

Figure 6.3 : The comparing of the stabilized system throughput and energy efficiency between the proposed approach and the existing benchmarks, as $A_i(t)$ increases from 0.5 to 4.5 Mbits.

arrive at devices following a binomial distribution with the probability 0.2, where $A_i(t)$ is uniformly distributed in $[1, 8]$ Mbits, $\rho_i \in [1000, 3000]$ cycles/bit, and $\xi_i = 1$.

For comparison purpose, we also simulate two benchmark approaches: (a) local execution, where the devices buffer and execute all the tasks locally; and (b) D2D Fogging [25], where the devices can offload tasks to direct neighbors which immediately execute the tasks and then return the results all within a time slot.

Fig. 6.3 compares the stabilized system throughput and energy efficiency between the proposed approach and the existing benchmarks, where the task size $A_i(t)$ increases from 0.5 to 4.5 Mbits and $V = 70$. In Fig. 6.3(a), we can see that the proposed approach and local execution process all arrived tasks and therefore achieve the same throughput when the tasks are small, i.e., $A_i(t) < 2.5$ Mbits. However, as the task size increases, the tasks cannot be all processed locally, and the growth of the system throughput slows down under local execution. In contrast, the proposed approach is still able to process all tasks through multi-hop offloading. In Fig. 6.3(b), we can see that the proposed approach provides higher energy efficiency (up to 50% energy saving) than local execution and D2D Fogging when $A_i(t) = 0.5$ Mbits, since much more tasks can be accomplished through fog computing over multiple hops. However, the gap of the proposed approach decreases with



(a) System throughput (b) Energy efficiency

Figure 6.4 : The stabilized system throughput and energy efficiency of the proposed approach versus connectivity ratio, where $V = 70$ and $A_i(t) = 4.5$ Mbits.

the growth of the task sizes, due to the increasing requirement of offloading tasks further away and subsequently the increasing demand on energy.

In Fig. 6.3, we can see that traffic reshaping is important for the schemes that need to complete the offloading and processing of tasks, and the return of results within a slot, for example, D2D Fogging [25]. Despite the fact that D2D Fogging is susceptible to the task size and its throughput declines with the growth of the task size, it can provide higher energy efficiency for small tasks than the proposed approach. This is because the increase of task size can increasingly result in long processing delays, hence violating the requirement of completing a task within a slot and leading to the decrease of throughput. In this case, the devices that experience good link conditions and abundant computing resources are likely to satisfy the requirement and therefore offload tasks through energy-efficient transmission, thereby leading to the increased energy efficiency of D2D Fogging in Fig. 6.3(b).

Fig. 6.4 evaluates the stabilized system throughput and energy efficiency of the proposed approach, as the connectivity ratio p_c increases, where $V = 70$ and $A_i(t) = 4.5$ Mbits. In Fig. 6.4(a), we can see that the proposed approach can provide 2Mbps more throughput than the local execution, as the connectivity improves. Particularly, when $p_c \leq 0.1$, tasks are prone to be offloaded over multiple hops and processed there under the proposed approach, increasing the throughput

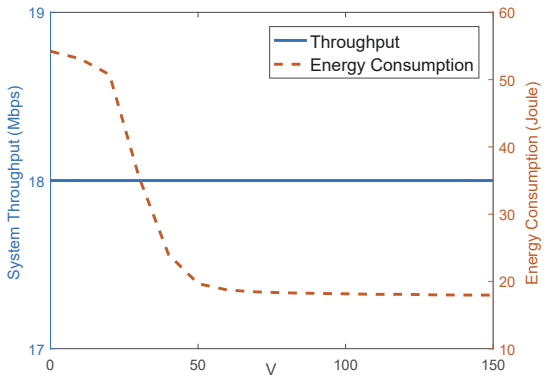
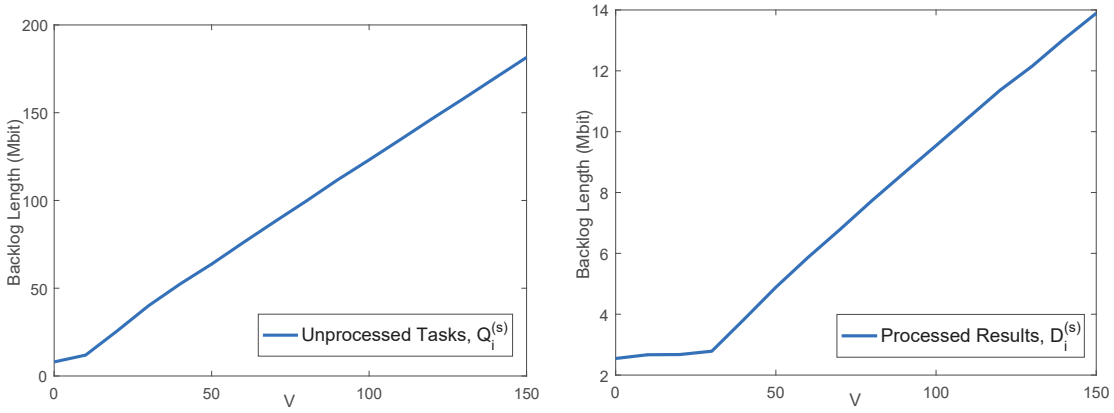


Figure 6.5 : The stabilized system throughput and energy consumption versus V.

and preventing unbounded growth of the queues, as shown in Fig. 6.4(a). In contrast, the throughput of the local execution is limited by the computing capability of individual devices, and the queues of some devices can grow unrestrainedly. The additional energy consumption of offloading in Fig. 6.4(b) contributes to the increase of throughput and the stability of the queues, as compared to the local execution.

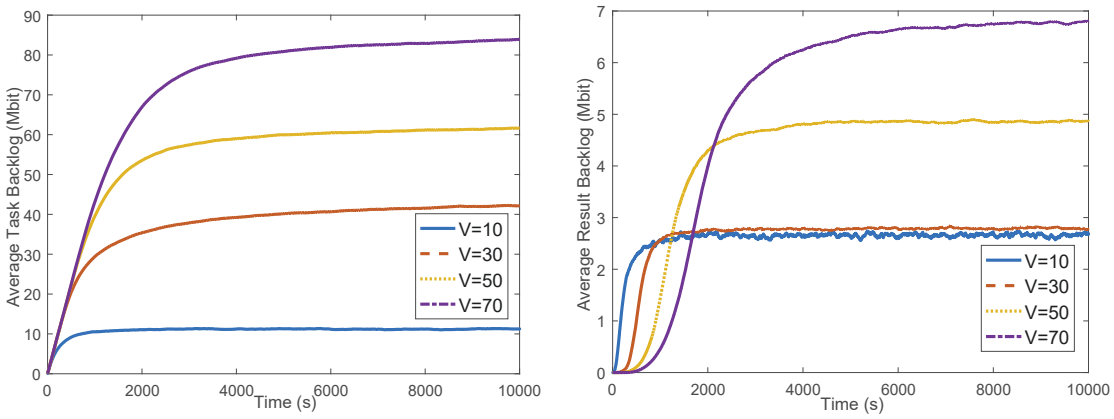
Fig. 6.5 plots the stabilized system throughput and energy consumption of the proposed approach, as V increases from 0 to 150. As shown in the figure, once stabilized, the system throughput is always equal to the task arrival of all the devices, i.e., $0.2 \times 4.5 \times 20 = 18$ Mbps. On the other hand, the stabilized energy consumption of the proposed distributed fog computing decreases with V, due to the increasing priority on the minimization of energy consumption in $\Delta_V(t)$. However, the decline of energy consumption slows down and the energy consumption becomes stable when $V > 50$. This is the energy required to stabilize all queues in the network.

Fig. 6.6 shows the average backlogs of tasks and results, as V increases from 0 to 150. The backlogs continue increasing with V for both types of task queues. However, the average backlog of the result queues is much shorter than that of the task queues. Moreover, the increase of the average backlog of results is quite slow when $V \leq 30$. This is because the increase of the result queues relies on the processing of the task queues, as can be seen in (6.8). Figs. 6.5 and 6.6 validate the tradeoff between the optimality gap and queue lengths established in Theorems 6 and 7, and corroborate the bounded optimality loss in Theorem 8.



(a) The task queues (b) The result queues

Figure 6.6 : The average backlog of the task and result queues versus V.



(a) The task queues (b) The result queues

Figure 6.7 : The change of the average backlog over time.

Fig. 6.7 shows the change of the average backlog of the queues in time, where t increases from 0 to 10000. We can see that both the backlogs of the task and result queues stabilize, and the average backlogs (i.e., the delay of the queues to be stabilized) increase with V, which is consistent with Fig. 6.6. Particularly, in Fig. 6.7(b), the time delay before the backlogs of the result queues start increasing also increases with V. This is because the increasing emphasis on the minimization of energy consumption requires the difference between the task and result queues to be enlarged for processing.

Fig. 6.8 shows the change of the backlog of virtual queues of 4 devices, $Z_s(t)$,

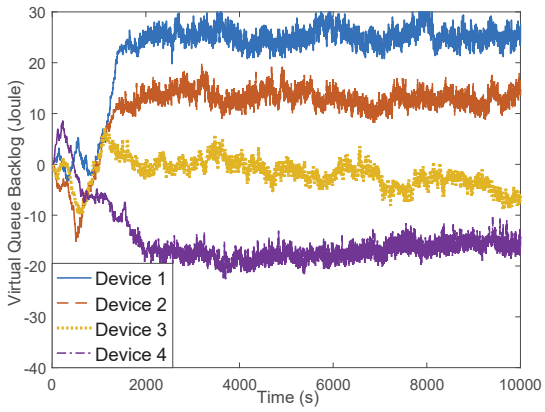


Figure 6.8 : The change of the backlog of virtual queues over time.

$s = 1, \cdots, 4$, as t elapses, where $V = 70$. The backlogs of the virtual queues fluctuate drastically when $t \leq 2000$, and become stabilized when $t > 2000$. Particularly, the backlogs stabilize to different values. This difference is due to the heterogeneity of the devices (here, $\widehat{F}_i$ and ρ_i are randomly initialized), i.e., the devices with higher computing capacities are more likely to help execute other devices' tasks, and therefore achieve lower backlogs of their virtual queues. The stabilized virtual queue backlogs at the devices validate the effectiveness of the proposed tit-for-tat incentives, i.e., a selfish device cannot request services from others without offering (otherwise, the virtual queue backlogs would keep increasing and cannot be stabilized).

Chapter 7

Collaborative Regions for Large-scale Fog Computing

This chapter presents a new fully distributed optimization of fog computing to capture the inhomogeneity and asymptotically minimize the time-average cost in a large-scale network. The contributions can be summarized as follows.

- We optimize the cost-effectiveness measures of the servers. As a result, the time-average cost of fog computing is asymptotically minimized over infinite time, while the servers only need to schedule at each time slot. The asymptotically optimal decisions of a server are made in the absence of the a-priori knowledge on future task arrivals and the network knowledge beyond the immediate neighbors of the server.

- We optimize the placeholders which build up the backlogs of the servers beyond the vicinity of the point of capture and hence prevent tasks from being offloaded to those servers. Specifically, the optimal substructure is uncovered in the optimal placeholders. The celebrated Bellman-Ford algorithm which exploits the optimal substructure is extended to optimize the placeholders in a distributed fashion. Without invalidating the optimal cost-effectiveness measures, the placeholders confine task offloading in the vicinity (i.e., the collaborative region) of the servers which generate task processing requests.

- Corroborated by extensive simulations, the collaborative regions are able to capture the inhomogeneity of fog computing and speed up stabilizing networks. The average size of the regions can quickly stabilize with the growth of the network scale, as is preferable in large-scale networks. As a result, we are able to reduce the number of queues maintained per server, simplify the

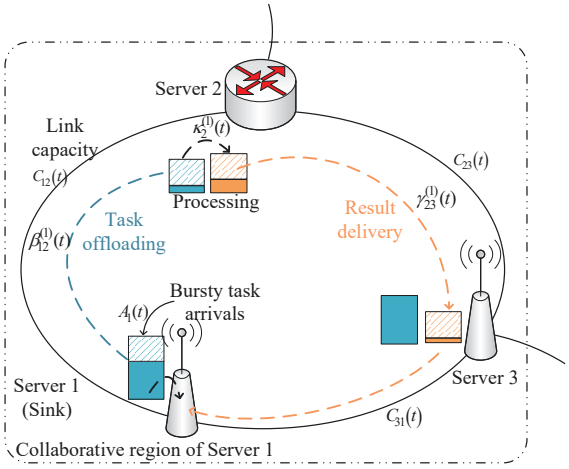


Figure 7.1 : An example of three servers to illustrate the offloading decision-making over a link at time t. The lengths of the queues, in which server 1 buffers the unprocessed tasks of itself, server 2 and server 3, are $Q_1^{(1)}(t) = 59$Mbits, $Q_1^{(2)}(t) = 21$Mbits and $Q_1^{(3)}(t) = 18$Mbits. The lengths of the queues, in which server 2 buffers the unprocessed tasks of server 1, itself, and server 3, are $Q_2^{(1)}(t) = 13$Mbits, $Q_2^{(2)}(t) = 32$Mbits and $Q_2^{(3)}(t) = 19$Mbits, respectively. Consider the link between servers 1 and 2. The price of offloading between servers 1 and 2 is $\zeta_{12}(t) = 0.5$.

queue management, and prevent excessive offloading and queuing delays while preserving the asymptotic optimality of fog computing cost.

Fig. 7.1 provides an example of three servers to illustrate the offloading decision-making over a link at time t. Given the queue lengths, server 1 can decide it is the most cost-effective to offload its own tasks to server 2, because the cost-effectiveness can be measured to be $\beta_{12}^{(1)}(t) = \epsilon \times [Q_1^{(1)}(t) - Q_2^{(1)}(t)] - \zeta_{12}(t) = 0.42$. Likewise, $\beta_{12}^{(2)}(t) = -0.72$, $\beta_{12}^{(3)}(t) = -0.52$, $\beta_{21}^{(1)}(t) = -1.42$, $\beta_{21}^{(2)}(t) = -0.28$ and $\beta_{21}^{(3)}(t) = -0.48$. Here, $\beta_{ij}^{(s)}(t)$ measures the cost-effectiveness for server i to offload the tasks of server s to server j. ϵ is a configurable parameter which can adjust the asymptotic optimality of the approach. In this example, $\epsilon = 0.02$. Such decision-making processes also take place independently over the other links in the meantime. The details on the cost-effectiveness measures are provided in this paper.

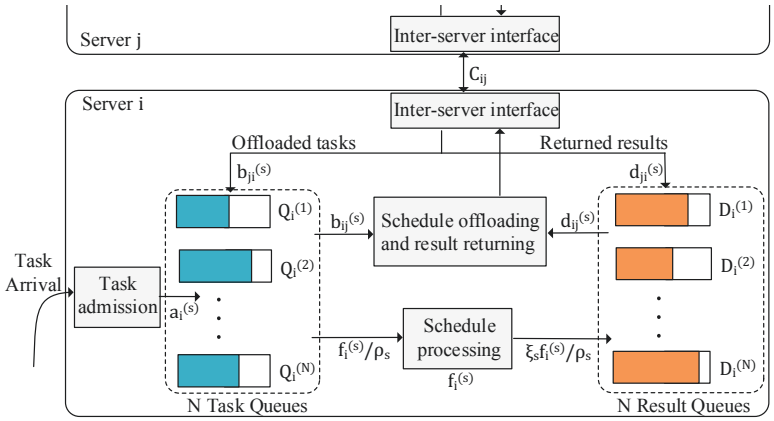


Figure 7.2 : An illustration on the proposed admission, processing and offloading of tasks, and the return of results by having all N servers collaborate.

7.1 System Model

The network of interest consists of N servers. $\mathbf{N} = \{1, \cdots, N\}$ collects the indices for the servers. The network operates in a slotted frame structure with slot length T. Tasks that arrive at a server can be either locally executed, or offloaded to other servers via links with finite bandwidths. Different from most existing works [11, 12, 16, 17, 19, 25, 26, 64], a task can be offloaded multiple hops away, and once executed, the result of the task is returned to the server admitting the task. Fig. 7.2 illustrates the admission, processing and offloading of unprocessed tasks, and the return of results in the case that all N servers collaborate. The notations used in this paper are summarized in Table 7.1.

At slot t, the computational task arriving at server i can be modeled by a triplet $(A_i(t), \rho_i A_i(t), \xi_i A_i(t))$, where $A_i(t)$ is the size of the task (in bits), and requires $\rho_i A_i(t)$ CPU cycles to process, and the result is of $\xi_i A_i(t)$ bits. ρ_i is the number of CPU cycles for processing a task bit, where $\rho_{\min} \leq \rho_i \leq \rho_{\max}$. ξ_i is the ratio of a result to an unprocessed task in terms of size. A task can be divided into subtasks to be processed in parallel at multiple servers. Server i can admit part of the task, denoted by $a_i(t)$, and

$$0 \leq a_i(t) \leq A_i(t) \leq A_i^{\max}, \forall i \tag{7.1}$$

where $A_i^{\max}$ is the maximum task arrival per slot at server i.

Table 7.1 : Definitions of Notations

Notation	**Definition**
$\mathbf{N}$	The set of servers
$\mathbf{N}_i$	The set of immediate neighbors of server i
T	The slot length
$A_i(t)$	The size of tasks generated by server i at slot t
ρ_i	The CPU cycles for processing a bit of task from server i
ξ_i	The ratio of results to unprocessed tasks of server i in size
$a_i(t)$	The size of tasks admitted by server i at slot t
$C_{ij}(t)$	The capacity of link (i,j) during slot t
$\zeta_{ij}(t)$	The cost per bit on link (i,j) during slot t
$\widehat{F}_i$	The computing capacity of server i
$F_i(t)$	The available computing resources of server i at slot t
$\zeta_i(t)$	The computing cost of server i per CPU cycle at slot t
$Q_i^{(s)}(t)$	The task queue backlog at server i for server s at slot t
$D_i^{(s)}(t)$	The result queue backlog at server i for server s at slot t
$b_{ij}^{(s)}(t)$	The tasks of server s offloaded from server i to j at slot t
$d_{ij}^{(s)}(t)$	The results for server s returned via server i to j at slot t
$f_i^{(s)}(t)$	The resources allocated to the tasks in $Q_i^{(s)}$ at slot t
α_i	The income of admitting a task bit at server i
$\Phi(\mathbf{x}^t)$	The instantaneous cost of the system at slot t
$\overline{X(t)}$	The long-term time-average of any process $X(t)$
ϵ	The stepsize of the stochastic subgradient method
$Q_{\max}^{(s)}$	The upper bounds for the queues $\{Q_i^{(s)}, \forall i\}$
$D_{\max}^{(s)}$	The upper bounds for the queues $\{D_i^{(s)}, \forall i\}$
$Q_{i,0}^{(s)}$	The placeholder for the queue $Q_i^{(s)}$
$D_{i,0}^{(s)}$	The placeholder for the queue $D_i^{(s)}$
$Q_{i,\max}^{(s)}$	The upper bound for the queue $Q_i^{(s)}$
$D_{i,\max}^{(s)}$	The upper bound for the queue $D_i^{(s)}$
$\mathcal{R}_s$	The collaborative region for server s
$\widehat{\mathcal{Q}}_0(h)$	The placeholder values for all queues by up to h iterations
$\widehat{\mathcal{Q}}_{\max}(h)$	The upper bound values for all queues by up to h iterations

Let $\mathbf{E}$ collect all the links between the servers. Assume that the capacity of each link remains unchanged within a slot, but may change between slots due to random background traffic, e.g., video and data. The capacity of the link $(i, j) \in \mathbf{E}$ during slot t, denoted by $C_{ij}(t)$ with $0 < C_{ij}(t) \leq C_{ij}^{\max}$, accounts for bi-directional transmission between servers i and j. The links between the servers are assumed to be error-free, since the wired links are typically reliable and require only simple channel coding with substantially lower complexity than computationally demanding tasks in fog computing. (Different from the wireless channels in Chapter 4, there is no interference between the links.) Our approach can be readily extended to less reliable links, where encoding is employed for the transmissions over the links, as will be discussed in Section 7.5. Let $\zeta_{ij}(t)$ denote the cost (e.g., energy consumption) per bit for transmissions from server i to j at time slot t.

Let $\widehat{F}_i$ (in CPU cycles per second) denote the computational capability of server i, and $\delta_i(t)$ denote the percentage of background tasks of the server during slot t. As a result, the available computational capacity of the server is $F_i(t) = [1 - \delta_i(t)]\widehat{F}_i T$. Let $\zeta_i(t)$ denote the computing cost at server i per CPU cycle.

As shown in Fig. 7.2, it is clear that each server needs to maintain N queues for unprocessed tasks admitted by all N servers (including the server itself) and N queues for results destined for the servers. The use of $2N$ separate queues has to date been necessary to allow all N servers to collaborate and return results to where admitted. In this paper, we propose to relax this requirement of $2N$ queues by constructing the collaborative region of every server in a distributed fashion, adapting to inhomogeneous connectivity and task arrivals.

Each of the queues is scheduled on a FIFO basis. As per slot t, $Q_i^{(s)}(t)$ denotes the queue backlog of server i for unprocessed tasks originating from server s; and $D_i^{(s)}(t)$ denotes the queue backlog of server i for results destined for server s. $\mathcal{Q}(t) = \{Q_i^{(s)}(t), D_i^{(s)}(t), \forall i, s\}$ collects all the queue backlogs. Per slot t, $b_{ij}^{(s)}(t)$ denotes the size of unprocessed tasks admitted at server s and forwarded to server j through server i, and $d_{ij}^{(s)}(t)$ denotes the size of results returned from server i to s through

server j. It is easy to establish a causality constraint, as given by

$$\sum\nolimits_{s\in\mathbf{N}} b_{ij}^{(s)}(t) + d_{ij}^{(s)}(t) + b_{ji}^{(s)}(t) + d_{ji}^{(s)}(t) \leq C_{ij}(t),\ \forall(i,j); \tag{7.2a}$$

$$b_{ij}^{(s)}(t) \geq 0,\ d_{ij}^{(s)}(t) \geq 0,\ \forall i,j,s. \tag{7.2b}$$

Let $f_i^{(s)}(t)$ denote the CPU cycles of server i allocated at the slot to the tasks originating from server s. We have

$$\sum\nolimits_{s\in\mathbf{N}} f_i^{(s)}(t) \leq F_i(t),\ \forall i; \tag{7.3a}$$

$$f_i^{(s)}(t) \geq 0,\ \forall i,s. \tag{7.3b}$$

As a result, the backlog of unprocessed tasks $Q_i^{(s)}(t)$ can be updated by

$$Q_i^{(s)}(t+1) = \max\{Q_i^{(s)}(t) - f_i^{(s)}(t)/\rho_s - \sum_{j\in\mathbf{N}} b_{ij}^{(s)}(t), 0\} + \sum_{j\in\mathbf{N}} b_{ji}^{(s)}(t) + a_i^{(s)}(t), \tag{7.4}$$

where $f_i^{(s)}(t)/\rho_s$ is the number of bits that can be processed at slot t from the queue of server i buffering unprocessed tasks originating from server s, and $a_i^{(s)}(t)$ is the number of unprocessed task bits admitted to the queue at the slot. $a_i^{(i)}(t) = a_i(t)$, and $a_i^{(s)}(t) = 0,\ \forall s \neq i$. The first term on the RHS of (7.4) captures unprocessed tasks in the queue at the end of slot t, after part of the tasks are processed at server i or offloaded to other servers. The second and the third terms account for new tasks offloaded from other servers to or admitted at server i.

The backlog of results $D_i^{(s)}(t)$ can be updated by

$$D_i^{(s)}(t+1) = \max\{D_i^{(s)}(t) - \sum_{j\in\mathbf{N}} d_{ij}^{(s)}(t), 0\} + \sum_{j\in\mathbf{N}} d_{ji}^{(s)}(t) + \xi_s f_i^{(s)}(t)/\rho_s,\ \forall s \neq i, \tag{7.5}$$

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

7.2 Distributed Online Control of Fog Computing

We first propose the asymptotically optimal distributed online approach for task admission, processing and routing, where each server is equipped with $2N$ queues to allow all servers to collaborate and ensure the return of results. The approach lays the foundation for our novel adaptive formation of collaborative regions in large-scale inhomogeneous networks, as will be articulated in Section 7.3.

7.2.1 Distributed Online Optimization

We take cost as the unified measure to quantify the performance of fog computing. The instantaneous cost of the system defines the difference between the cost of task processing and the income of task admission per slot t, as given by

$$\Phi(\mathbf{x}^t) = (\sum\nolimits_{i,j\in\mathbf{N}} c_{ij}(t) + \sum\nolimits_{i\in\mathbf{N}} c_i(t)) - \sum\nolimits_{i\in\mathbf{N}} \alpha_i a_i(t) \tag{7.6}$$

where $\mathbf{x}^t = \{b_{ij}^{(s)}(t), d_{ij}^{(s)}(t), f_i^{(s)}(t), a_i(t), \forall i, j, s\}$, and $\alpha_i > 0$ is the income of admitting a task bit at server i. $c_{ij}(t) = \zeta_{ij}(t)\sum_{s\in\mathbf{N}}(b_{ij}^{(s)}(t) + d_{ij}^{(s)}(t))$ and $c_i(t) = \zeta_i(t)\sum_{s\in\mathbf{N}} f_i^{(s)}(t)$ denote the routing cost over link (i, j) and the processing cost at server i, respectively, at slot t.

We define the network is stable if and only if the following is met [57]

$$\overline{Q_i^{(s)}(t)} < \infty, \overline{D_i^{(s)}(t)} < \infty, \forall i, s \tag{7.7}$$

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

Consider the stochastic nature of traffic arrivals and server availability. We minimize the time-average cost of the system while preserving the stability. The problem of interest can be formulated as

$$\begin{aligned} \Phi^* = \min_{\mathbb{X}}\ & \overline{\Phi(\mathbf{x}^t)} \\ \text{s.t. } & (7.1), (7.2), (7.3), (7.4), (7.5), (7.7), \forall t. \end{aligned} \tag{7.8}$$

where $\mathbb{X} = \{\mathbf{x}^t, \forall t\}$ collects all the variables in regards to the admission, routing, and processing across all time slots.

Note that the optimal solution to (7.8) would require the a-priori knowledge on task arrivals, link capacities, and computing resources across the entire network over infinite time, which violates causality. On the other hand, myopic optimization per time slot would compromise the optimality in the long term due to the couplings of variables in (7.4), (7.5) and (7.7). By taking a stochastic subgradient method [72, 73], the variables in $\mathbb{X}$ can be decoupled between time slots for solving (7.8) in an asymptotically optimal manner, as stated in the following theorem.

Theorem 9. *Problem (7.8) can be reformulated to a per-slot optimization problem, as given by*

$$\begin{aligned} &\max_{\mathbf{x}^t} g(\mathbf{a}^t) + \mu(\mathbf{f}^t) + \eta(\mathbf{b}^t, \mathbf{d}^t) \\ &\textit{s.t. (7.1), (7.2), (7.3).} \end{aligned} \tag{7.9}$$

where

$$g(\mathbf{a}^t) = \textstyle\sum_{i\in\mathbf{N}}[\alpha_i - \epsilon Q_i^{(i)}(t)]a_i(t); \tag{7.10a}$$

$$\mu(\mathbf{f}^t) = \textstyle\sum_{i,s\in\mathbf{N}}[\frac{\epsilon[Q_i^{(s)}(t)-\xi_s D_i^{(s)}(t)]}{\rho_s} - \zeta_i(t)]f_i^{(s)}(t); \tag{7.10b}$$

$$\begin{aligned} &\eta(\mathbf{b}^t, \mathbf{d}^t) = \textstyle\sum_{i,j,s\in\mathbf{N}} \epsilon[Q_i^{(s)}(t)(b_{ij}^{(s)}(t) - b_{ji}^{(s)}(t)) \\ &+D_i^{(s)}(t)(d_{ij}^{(s)}(t) - d_{ji}^{(s)}(t))] - \zeta_{ij}(t)(b_{ij}^{(s)}(t) + d_{ij}^{(s)}(t)), \end{aligned} \tag{7.10c}$$

and ϵ is an appropriate stepsize of the stochastic subgradient method and also accounts for the optimality bound of $\widetilde{\Phi}^*$ in (7.8), as will be shown in Theorem 10.

Proof. By combining (7.4), (7.5) and (7.7), it can be shown that the admission, processing and offloading of tasks, as well as the routing of results, must satisfy the following necessary conditions of queue stability; see [57, Lemma 2.1].

$$\begin{aligned} &\overline{a_i^{(s)}(t) - f_i^{(s)}(t)/\rho_s + \textstyle\sum_{j\in\mathbf{N}}(b_{ji}^{(s)}(t) - b_{ij}^{(s)}(t))} \leq 0,\ \forall i, s; \\ &\overline{\xi_s f_i^{(s)}(t)/\rho_s + \textstyle\sum_{j\in\mathbf{N}}(d_{ji}^{(s)}(t) - d_{ij}^{(s)}(t))} \leq 0,\ \forall i, s. \end{aligned} \tag{7.11}$$

As a result, (7.8) can be reformulated as

$$\begin{aligned} \widetilde{\Phi}^* = \min_{\mathbb{X}} \ & \overline{\Phi(\mathbf{x}^t)} \\ \text{s.t. } & (7.1),\ (7.2),\ (7.3),\ (7.11),\ \forall t. \end{aligned} \tag{7.12}$$

Problem (7.12) eliminates the time coupling among the variables in (7.8). Concatenate the system random parameters into a vector $\omega^t = \{A_i(t), C_{ij}(t), F_i(t), \forall i, j \in \mathbf{N}\}$. It is reasonable to assume that ω^t is i.i.d. across all time slots, since tasks are independently admitted at a large number of servers, and the availability of the servers can be affected by independent background tasks. As a result, the time-averages (7.11) can be replaced with the expectations per slot over i.i.d. random parameters, as given by

$$\mathbb{E}[a_i^{(s)}(t) - f_i^{(s)}(t)/\rho_s + \sum\nolimits_{j\in\mathbf{N}}(b_{ji}^{(s)}(t) - b_{ij}^{(s)}(t))] \leq 0,\ \forall i, s; \tag{7.13a}$$

$$\mathbb{E}[\xi_s f_i^{(s)}(t)/\rho_s + \sum_{j\in\mathbf{N}}(d_{ji}^{(s)}(t) - d_{ij}^{(s)}(t))] \leq 0,\ \forall i, s. \tag{7.13b}$$

By also replacing the time-average objective with the expectation, we can rewrite (7.12) as

$$\begin{aligned} \widetilde{\Phi}^* = \min_{\mathbb{X}} \ & \mathbb{E}[\Phi(\mathbf{x}^t; \omega^t)] \\ \text{s.t. } & (7.1),\ (7.2),\ (7.3),\ (7.13),\ \forall t. \end{aligned} \tag{7.14}$$

Since (7.14) is a relaxation of (7.8) with $\widetilde{\Phi}^* \leq \Phi^*$, the optimality bound of Φ^* will be derived later, provided that the solution to (7.14) is also feasible for (7.8); see Theorem 10.

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 (7.14) associated with (7.13a) and (7.13b), respectively. The Lagrangian of (7.14) can be given by [40]

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

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

$$\begin{aligned}\mathcal{L}^t(\mathbf{x}^t, \boldsymbol{\lambda}) &= \Phi(\mathbf{x}^t) + \textstyle\sum_{i,s\in\mathbf{N}} \lambda_{i,1}^s \mathbb{E}[a_i^{(s)}(t) - f_i^{(s)}(t)/\rho_s \\ &+ \textstyle\sum_{j\in\mathbf{N}}(b_{ji}^{(s)}(t) - b_{ij}^{(s)}(t))] + \textstyle\sum_{i,s\in\mathbf{N}} \lambda_{i,2}^s \mathbb{E}[\xi_s f_i^{(s)}(t)/\rho_s \\ &+ \textstyle\sum_{j\in\mathbf{N}}(d_{ji}^{(s)}(t) - d_{ij}^{(s)}(t))].\end{aligned} \tag{7.15}$$

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

$$\begin{aligned}\mathcal{D}(\boldsymbol{\lambda}) = &\min_{\mathbb{X}} \mathcal{L}(\mathbb{X}, \boldsymbol{\lambda}) \\ &\text{s.t. (7.1), (7.2), (7.3), } \forall t.\end{aligned} \tag{7.16}$$

The dual problem can be solved by using a stochastic subgradient method that updates the primal variables $\mathbf{x}^t$ and the stochastic estimates of $\widetilde{\boldsymbol{\lambda}}^t$ slot by slot, till convergence to the optimum [72]. (7.16) can be decoupled between slots. At each slot t, $\mathbf{x}^t$ can be updated by

$$\begin{aligned}\mathbf{x}^t = &\arg\min_{\mathbf{x}^t} \mathcal{L}^t(\mathbf{x}^t, \widetilde{\boldsymbol{\lambda}}^t) \\ &\text{s.t. (7.1), (7.2), (7.3).}\end{aligned} \tag{7.17}$$

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

$$\begin{aligned}\widetilde{\lambda}_{i,1}^s(t+1) = &\max\{\widetilde{\lambda}_{i,1}^s(t) + \epsilon[a_i^{(s)}(t) - f_i^{(s)}(t)/\rho_s \\ &+ \sum\nolimits_{j\in\mathbf{N}}(b_{ji}^{(s)}(t) - b_{ij}^{(s)}(t))], 0\},\end{aligned} \tag{7.18a}$$

$$\begin{aligned}\widetilde{\lambda}_{i,2}^s(t+1) = &\max\{\widetilde{\lambda}_{i,2}^s(t) + \epsilon[\xi_s f_i^{(s)}(t)/\rho_s \\ &+ \sum\nolimits_{j\in\mathbf{N}}(d_{ji}^{(s)}(t) - d_{ij}^{(s)}(t))], 0\}.\end{aligned} \tag{7.18b}$$

With $Q_i^{(s)}(0) = 0$ and $D_i^{(s)}(0) = 0$, by comparing (7.4) and (7.5) with (7.18a) and (7.18b), respectively, the update of dual multipliers per slot can be enclosed in the natural update of queue backlogs, i.e., $\widetilde{\lambda}_{i,1}^s(t) = \epsilon Q_i^{(s)}(t)$ and $\widetilde{\lambda}_{i,2}^s(t) = \epsilon D_i^{(s)}(t)$. As a result, with respect to $\mathbf{x}^t$, (7.8) can be reformulated into (7.9). □

Note that $\mathbf{a}^t = \{a_i(t), \forall i\}$, $\mathbf{f}^t = \{f_i^{(s)}(t), \forall i, s\}$, and $\mathbf{b}^t/\mathbf{d}^t = \{b_{ij}^{(s)}(t)/d_{ij}^{(s)}(t), \forall i, j, s\}$ are decoupled from each other in both the objective and constraints of (7.9). Also, $a_i(t)$ can be decoupled from $a_j(t)$, and $f_i^{(s)}(t)$ can be decoupled from $f_j^{(s)}(t)$, $\forall i \neq j$. $\eta(\mathbf{b}^t, \mathbf{d}^t)$ can be decoupled between links, while (7.2) specifies the capacity for each link. Let $\tilde{\mathbf{b}}_{ij}(t) = \{b_{ij}^{(s)}(t), b_{ji}^{(s)}(t), \forall s\}$ and $\tilde{\mathbf{d}}_{ij}(t) = \{d_{ij}^{(s)}(t), d_{ji}^{(s)}(t), \forall s\}$ for each link (i, j). As a result, (7.9) can be efficiently solved by separately pursuing:

$$\max_{a_i(t)}[\alpha_i - \epsilon Q_i^{(i)}(t)]a_i(t), \text{ s.t. (7.1)}; \tag{7.19a}$$

$$\max_{\mathbf{f}_i(t)} \sum\nolimits_{s\in\mathbf{N}} \kappa_i^{(s)}(t) f_i^{(s)}(t), \text{ s.t. (7.3)}; \tag{7.19b}$$

$$\max_{\tilde{\mathbf{b}}_{ij}(t), \tilde{\mathbf{d}}_{ij}(t)} \eta_{ij}(\tilde{\mathbf{b}}_{ij}(t), \tilde{\mathbf{d}}_{ij}(t)), \text{ s.t. (7.2)}, \tag{7.19c}$$

where $\kappa_i^{(s)}(t) = \epsilon[Q_i^{(s)}(t) - \xi_s D_i^{(s)}(t)]/\rho_s - \zeta_i(t)$ and $\eta_{ij}(\tilde{\mathbf{b}}_{ij}(t), \tilde{\mathbf{d}}_{ij}(t)) = \sum_{s\in\mathbf{N}} \beta_{ij}^{(s)}(t) b_{ij}^{(s)}(t) + \beta_{ji}^{(s)}(t) b_{ji}^{(s)}(t) + \gamma_{ij}^{(s)}(t) d_{ij}^{(s)}(t) + \gamma_{ji}^{(s)}(t) d_{ji}^{(s)}(t)$ can be obtained by decoupling (7.10c) between links, with $\beta_{ij}^{(s)}(t) = \epsilon[Q_i^{(s)}(t) - Q_j^{(s)}(t)] - \zeta_{ij}(t)$ and $\gamma_{ij}^{(s)}(t) = \epsilon[D_i^{(s)}(t) - D_j^{(s)}(t)] - \zeta_{ij}(t)$.

Corollary 2. *(7.19a), (7.19b) and (7.19c) are linear programming (LP) of weighted-sum maximization [69]. The optimal solutions to (11a) and (11b) are*

$$a_i(t) = \begin{cases} A_i(t), \textit{ if } Q_i^{(i)}(t) < \alpha_i/\epsilon; \\ 0, \textit{ otherwise;} \end{cases} \tag{7.20}$$

$$f_i^{(s)}(t) = \begin{cases} F_i(t), \textit{ if } s = \arg\max_j \kappa_i^{(j)}(t) \textit{ and } \kappa_i^{(s)}(t) > 0; \\ 0, \textit{ otherwise.} \end{cases} \tag{7.21}$$

The optimal solution to (7.19c) can be obtained by evaluating the weights $\beta_{ij}^{(s)}(t)$, $\beta_{ji}^{(s)}(t)$, $\gamma_{ij}^{(s)}(t)$ and $\gamma_{ji}^{(s)}(t)$ at servers i and j. If $\max\{\beta_{ij}^{(s)}(t), \gamma_{ij}^{(s)}(t)\} < 0$ or $\max\{\beta_{ij}^{(s)}(t), \gamma_{ij}^{(s)}(t)\} < \max\{\beta_{ji}^{(s)}(t), \gamma_{ji}^{(s)}(t)\}$, server i remains idle at slot t, i.e., neither offloading tasks nor returning results on link (i, j). If $\max_s\{\beta_{ij}^{(s)}(t)\} > \max_s\{\gamma_{ij}^{(s)}(t)\}$, server i does not return results, and instead forwards unprocessed tasks to server j with the

Algorithm 6 The Proposed Distributed Online Optimization of Fog Computing

At each server i at every time slot t:
1: Acquire $A_i(t)$, $F_i(t)$ and $\zeta_{ij}(t)$
2: Observe the queues of its own and immediate neighbors
3: Admit task according to (7.20)
4: Allocate the computing resources by (7.21)
5: Schedule the offloading and routing based on (7.22)
6: Update $Q_i^{(s)}(t)$ and $D_i^{(s)}(t)$ according to (7.4) and (7.5)

task size as specified by

$$b_{ij}^{(s)}(t) = \begin{cases} C_{ij}(t), \text{ if } s = \arg\max_r \beta_{ij}^{(r)}(t); \\ 0, \text{ otherwise.} \end{cases} \tag{7.22a}$$

If $\max_s\{\beta_{ij}^{(s)}(t)\} \leq \max_s\{\gamma_{ij}^{(s)}(t)\}$, *server* i *does not forward tasks to server* j, *and instead returns results through server* j *with the result size as specified by*

$$d_{ij}^{(s)}(t) = \begin{cases} C_{ij}(t), \text{ if } s = \arg\max_r \gamma_{ij}^{(r)}(t); \\ 0, \text{ otherwise.} \end{cases} \tag{7.22b}$$

From (7.20)–(7.22), we see that the optimal decisions of task admission, processing and forwarding, and result delivery can be made locally at every individual server, based on the weights which only depend on the local information of the server and its immediate neighbors. In this sense, the weights are the optimal *cost-effective measures* which can be used to efficiently achieve the asymptotically optimal cost of fog computing. We also see from (7.21) that all the available computing resources of server i, $F_i(t)$, are allocated to the queue in the most urgency to be processed. From (7.22), given a link, the server and the queue which can make the most use of the link at time slot t are activated to transmit unprocessed tasks or results. Algorithm 6 can be established to optimize fog computing based on (7.20)–(7.22).

7.2.2 Asymptotic Optimality

Let $\widetilde{\Phi}^*(\mathbf{x}^t)$ denote the long-term time-average cost of fog computing achieved by Algorithm 6, and Φ^* be the offline optimum (minimized in a posteriori manner violating causality). Relying on Lyapunov optimization techniques [57], we can formally establish that $\widetilde{\Phi}^*(\mathbf{x}^t)$ is asymptotically optimal.

Theorem 10. *The gap between* $\widetilde{\Phi}^*(\mathbf{x}^t)$ *and* Φ^* *satisfies*

$$\widetilde{\Phi}^*(\mathbf{x}^t) - \Phi^* \leq \mathcal{O}(\epsilon), \tag{7.23}$$

where ϵ *is the stepsize of the stochastic subgradient method.*

Proof. Taking squares on both sides of (7.4) and (7.5), 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

$$\begin{aligned}[Q_i^{(s)}(t+1)]^2 &\leq [Q_i^{(s)}(t)]^2 + 2Q_i^{(s)}(t)[\sum_{j\in\mathbf{N}}(b_{ji}^{(s)}(t) - b_{ij}^{(s)}(t)) + a_i^{(s)}(t) - f_i^{(s)}(t)/\rho_s] \\ &+[f_i^{(s)}(t)/\rho_s + \sum_{j\in\mathbf{N}} b_{ij}^{(s)}(t)]^2 + [\sum_{j\in\mathbf{N}} b_{ji}^{(s)}(t) + a_i^{(s)}(t)]^2,\end{aligned} \tag{7.24a}$$

$$\begin{aligned}[D_i^{(s)}(t+1)]^2 &\leq [D_i^{(s)}(t)]^2 + 2D_i^{(s)}(t)[\sum_{j\in\mathbf{N}}(d_{ji}^{(s)}(t) - d_{ij}^{(s)}(t)) \\ &+\tfrac{\xi_s}{\rho_s} f_i^{(s)}(t)] + [\sum_{j\in\mathbf{N}} d_{ij}^{(s)}(t)]^2 + [\sum_{j\in\mathbf{N}} d_{ji}^{(s)}(t) + \tfrac{\xi_s}{\rho_s} f_i^{(s)}(t)]^2.\end{aligned} \tag{7.24b}$$

Considering a standard quadratic Lyapunov function $\mathfrak{L}(t) = \frac{1}{2}\sum_{i,s\in\mathbf{N}}[Q_i^{(s)}(t)^2 + D_i^{(s)}(t)^2]$ [57], the drift $\Delta\mathfrak{L}(t)$ readily follows that

$$\begin{aligned}&\Delta\mathfrak{L}(t) = \mathfrak{L}(t+1) - \mathfrak{L}(t) \\ &\leq U + \sum_{i,s\in\mathbf{N}} D_i^{(s)}(t)[\sum_{j\in\mathbf{N}}(d_{ji}^{(s)}(t) - d_{ij}^{(s)}(t)) + \tfrac{\xi_s}{\rho_s} f_i^{(s)}(t)] \\ &+ \sum_{i,s\in\mathbf{N}} Q_i^{(s)}(t)[\sum_{j\in\mathbf{N}}(b_{ji}^{(s)}(t) - b_{ij}^{(s)}(t)) + a_i^{(s)}(t) - f_i^{(s)}(t)/\rho_s],\end{aligned} \tag{7.25}$$

where $U = \frac{1}{2}\{\sum_{i\in\mathbf{N}}[(\xi_i+1)\widehat{F}_i T/\rho_{\min} + 2\sum_{j\in\mathbf{N}} C_{ij}^{\max} T + A_i^{\max}]\}^2$ 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 (7.24a) and (7.24b).

Taking expectations over ω^t and adding $\frac{1}{\epsilon}\mathbb{E}[\Phi(\mathbf{x}^t)]$ on both sides ($\mathbf{x}^t$ is the optimal policy by solving (7.17)), we arrive at

$$\begin{aligned}
&\mathbb{E}[\Delta\mathfrak{L}(t)] + \tfrac{1}{\epsilon}\mathbb{E}[\Phi(\mathbf{x}^t)] \leq U + \tfrac{1}{\epsilon}\mathbb{E}\{\Phi(\mathbf{x}^t) \\
&+\epsilon \sum_{i,s\in\mathbf{N}} D_i^{(s)}(t)[\textstyle\sum_{j\in\mathbf{N}}(d_{ji}^{(s)}(t) - d_{ij}^{(s)}(t)) + \frac{\xi_s}{\rho_s} f_i^{(s)}(t)] \\
&+\epsilon \sum_{i,s\in\mathbf{N}} Q_i^{(s)}(t)[\sum_{j\in\mathbf{N}}(b_{ji}^{(s)}(t) - b_{ij}^{(s)}(t)) + a_i^{(s)}(t) - \tfrac{f_i^{(s)}(t)}{\rho_s}]\} \\
&= U + \tfrac{1}{\epsilon}\mathbb{E}[\mathcal{L}^t(\mathbf{x}^t(\epsilon\mathcal{Q}(t)), \epsilon\mathcal{Q}(t))] \\
&= U + \tfrac{1}{\epsilon} D(\epsilon\mathcal{Q}(t)) \leq U + \tfrac{1}{\epsilon}\Phi^*
\end{aligned} \tag{7.26}$$

where $\mathcal{L}^t(\mathbf{x}^t, \boldsymbol{\lambda})$ is defined in (7.15); $\mathbf{x}^t(\epsilon\mathcal{Q}(t))$ is the optimal primal variable as given by (7.17) (hence, $\mathbb{E}[\mathcal{L}^t(\mathbf{x}^t(\epsilon\mathcal{Q}(t)), \epsilon\mathcal{Q}(t))] = D(\epsilon\mathcal{Q}(t))$); and the last inequality in (7.26) is due to the weak duality [40].

Summing up all the telescoping series over time slots $\{0, 1, \cdots, T-1\}$, (7.26) leads to

$$\mathbb{E}[\mathfrak{L}(T)] - \mathfrak{L}(0) + \frac{1}{\epsilon}\sum\nolimits_{t=0}^{T-1}\mathbb{E}[\Phi(\mathbf{x}^t)] \leq UT + \frac{T}{\epsilon}\Phi^*. \tag{7.27}$$

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

$$\widetilde{\Phi}^*(\mathbf{x}^t) = \frac{1}{T}\lim_{T\to\infty}\sum\nolimits_{t=0}^{T-1}\mathbb{E}[\Phi(\mathbf{x}^t)] \leq \Phi^* + \epsilon U. \tag{7.28}$$

This concludes the proof. □

Theorem 11. *The backlogs of all the queues in the system are strictly bounded, i.e., $Q_i^{(s)}(t) \leq Q_{\max}^{(s)}$ and $D_i^{(s)}(t) \leq D_{\max}^{(s)}$, $\forall i, s$, at each time slot t,*

$$Q_{\max}^{(s)} = \frac{\alpha_s}{\epsilon} + A_s^{\max} + \theta_s; \tag{7.29a}$$

$$D_{\max}^{(s)} = \frac{\alpha_s/\epsilon + A_s^{\max}}{\xi_s} + \frac{(1+\xi_s)\theta_s}{\xi_s} + \frac{\xi_s\widehat{F}_s}{\rho_s}; \tag{7.29b}$$

where $\theta_s = \sum_{j\in\mathbf{N}_s} C_{js}^{\max}$ is the maximum task arrival at server s per slot via offloading, and $\mathbf{N}_s$ collects the immediate neighbors of server s.

Proof. We first prove (7.29a) through mathematical induction. Note that tasks in

$Q_i^{(s)}$ originate from $Q_s^{(s)}$, and server i only offloads tasks to server j when $Q_i^{(s)} > Q_j^{(s)}$ according to (7.22a). We can conclude that $Q_{s,\max}^{(s)} \geq Q_{i,\max}^{(s)}$, $i \neq s$, where $Q_{i,\max}^{(s)}$ denotes the instantaneous upper bound of $Q_i^{(s)}$ per slot t. Thus, we only need to confirm $Q_s^{(s)}(t) \leq \frac{\alpha_s}{\epsilon} + A_s^{\max} + \theta_s$ to prove (7.29a). The upper bound holds at slot 0 due to the fact that $Q_s^{(s)}(0) = 0$, $\forall s$. Suppose that the upper bound holds at slot t. If $Q_s^{(s)}(t) \geq \frac{\alpha_s}{\epsilon}$, $a_i(t) = 0$ according to (7.20) and we have $Q_s^{(s)}(t+1) \leq Q_s^{(s)}(t) + \theta_s$. Otherwise, if $Q_s^{(s)}(t) < \frac{\alpha_s}{\epsilon}$, the difference between $Q_s^{(s)}(t)$ and $Q_s^{(s)}(t+1)$ is less than $A_s^{\max} + \theta_s$ according to (7.1) and (7.2), i.e., $Q_s^{(s)}(t+1) \leq \frac{\alpha_s}{\epsilon} + A_s^{\max} + \theta_s$. As a result, the upper bound also holds at slot $(t+1)$. This concludes the proof of (7.29a).

Given the upper bound in (7.29a), we proceed to prove (7.29b). Since (7.29b) holds for $t = 0$, we assume that it also holds at slot t. From (7.21), if $D_i^{(s)}(t) \geq Q_{\max}^{(s)}/\xi_s$, no task is processed due to that $\kappa_i^{(s)}(t) < 0$, i.e., $D_i^{(s)}(t+1) \leq D_i^{(s)}(t) + \theta_s$. On the other hand, if $D_i^{(s)}(t) < Q_{\max}^{(s)}/\xi_s$, the result of processed tasks at slot t cannot exceed $\xi_s \widehat{F}_i/\rho_s$, leading to $D_i^{(s)}(t+1) \leq Q_{\max}^{(s)}/\xi_s + \xi_s \widehat{F}_i/\rho_s + \theta_s$, i.e., (7.29b) also holds at slot $t+1$. Using the upper bound in (7.29a), we can prove (7.29b). □

Theorems 10 and 11 reveal that the time-average cost (7.14) converges to within an optimality bound of $\mathcal{O}(\epsilon)$, which diminishes as $\epsilon \to 0$. The typical $[\mathcal{O}(1/\epsilon), \mathcal{O}(\epsilon)]$-tradeoff of stochastic optimization holds; i.e., an $\mathcal{O}(1/\epsilon)$ queue length allows for an $\mathcal{O}(\epsilon)$ close-to-optimal cost.

The theorems confirm the effectiveness of the proposed approach, and also provide an efficient means for comparison studies between the proposed distributed online approach and the globally optimal centralized offline scheme. This is because it is computationally prohibitive to simulate the optimal centralized offline scheme which has an infinite number of variables to optimize, including the task offloading and result delivery decisions at each of the infinite number of time slots.

7.3 Collaborative Region for Large-scale Inhomogeneous Networks

Another critical challenge addressed in this paper is to extrapolate fog computing in large-scale inhomogeneous networks, while preserving the asymptotic optimality stated in Theorem 10. Particularly, we propose to put placeholder in each of the total $2N^2$ queues, and optimize the placeholders and the instantaneous upper bounds of the queues in a distributed manner. A queue is enabled (i.e., the server equipped with the queue participates in the fog computing for another server indicated by the queue), if and only if the placeholder is shorter than its upper bound. Details are provided in the rest of this section.

7.3.1 Adaptive Collaborative Region

According to the placeholder analysis [74], the placeholder satisfies the following property.

Property 4. There exists a non-negative queue backlog vector $\mathcal{Q}_0 = \{Q_{i,0}^{(s)}, D_{i,0}^{(s)}, \forall i, s\}$, as such that: if $\mathcal{Q}(0) \succeq \mathcal{Q}_0$, we have $\mathcal{Q}(t) \succeq \mathcal{Q}_0$ for all $t \geq 0$.

Corollary 3. *$\mathcal{Q}_0$ satisfying Property 4 can be given by*

$$D_{i,0}^{(s)} = \begin{cases} 0, \text{ if } i = s; \\ \max\{\min_j\{D_{j,0}^{(s)} + w_{ij}\}, 0\}, \text{ otherwise;} \end{cases} \tag{7.30a}$$

$$Q_{i,0}^{(s)} = \max\{\min_j\{\xi_s D_{i,0}^{(s)} + \varphi_i^{(s)}, Q_{j,0}^{(s)} + w_{ij}\}, 0\}, \tag{7.30b}$$

where $w_{ij} = \zeta_{ij}^{\min}/\epsilon - C_{ij}^{\max}$; $\varphi_i^{(s)} = \rho_s \zeta_i^{\min}/\epsilon - \widehat{F}_i/\rho_s$; and $\zeta_{ij}^{\min}$ and $\zeta_i^{\min}$ denote the lower bounds of $\zeta_{ij}(t)$ and $\zeta_i(t)$, respectively.

Proof. We note that $D_{s,0}^{(s)} = 0$ for the sink server s, and $D_{i,0}^{(s)} = 0$ satisfies Property 4 since $D_i^{(s)}(t) \geq 0$. Then, we only need to show that $D_{i,0}^{(s)} = \min_j\{D_{j,0}^{(s)} + w_{ij}\}$ satisfies Property 4 to prove (7.30a). Exploiting (7.22b), we can prove this by mathematical induction, as follows.

Suppose that $D_i^{(s)}(t) \geq \min_j\{D_{j,0}^{(s)} + w_{ij}\}$ at slot t. If $D_i^{(s)}(t) \leq \min_j\{D_{j,0}^{(s)} + \zeta_{ij}^{\min}/\epsilon\}$, $\gamma_{ij}^{(s)}(t) \leq 0$ for all neighboring servers, i.e., $D_i^{(s)}(t+1) \geq D_i^{(s)}(t)$, since no results are routed away from $D_i^{(s)}$. Otherwise, $D_i^{(s)}(t) = D_{j_0,0}^{(s)} + \zeta_{ij_0}^{\min}/\epsilon + \delta$, where $j_0 = \arg\min\{D_{j,0}^{(s)} + \zeta_{ij}^{\min}/\epsilon\}$ and $\delta > 0$. $D_i^{(s)}$ is correlated in time by (7.5), i.e., $D_i^{(s)}(t+1)$ is less likely to take small values with the increase of $D_i^{(s)}(t)$. As a result, we only evaluate small values of δ for $D_i^{(s)}(t)$ to attain the worst-case scenario for $D_i^{(s)}(t+1)$. Only $\gamma_{ij_0}^{(s)}(t) > 0$ for small δ, and the routed results cannot exceed $C_{ij_0}^{\max}$. As a result, $D_i^{(s)}(t+1) > D_{j_0,0}^{(s)} + \zeta_{ij_0}^{\min}/\epsilon - C_{ij_0}^{\max} \geq \min_j\{D_{j,0}^{(s)} + w_{ij}\}$. This concludes the proof of (7.30a).

Likewise, we can prove that $Q_{i,0}^{(s)} = \min_j\{Q_{j,0}^{(s)} + w_{ij}\}$ satisfies Property 4. Then we proceed to show that $Q_{i,0}^{(s)} = \xi_s D_{i,0}^{(s)} + \varphi_i^{(s)}$ satisfies Property 4 to prove (7.30b). We assume that it also holds at slot t. If $Q_i^{(s)}(t) \leq \xi_s D_{i,0}^{(s)} + \rho_s \zeta_i^{\min}/\epsilon$, we have $\kappa_i^{(s)}(t) < 0$ and no task in the queue is processed according to (7.21), i.e., $Q_i^{(s)}(t+1) \geq Q_i^{(s)}(t)$. On the other hand, if $Q_i^{(s)}(t) > \xi_s D_{i,0}^{(s)} + \rho_s \zeta_i^{\min}/\epsilon$, at most $\widehat{F}_i/\rho_s$ tasks can be processed, i.e., $Q_i^{(s)}(t+1) > Q_i^{(s)}(t) - \widehat{F}_i/\rho_s > \xi_s D_{i,0}^{(s)} + \varphi_i^{(s)}$. This concludes the proof of (7.30b). □

The placeholder $\mathcal{Q}_0$, satisfying Property 4, is neither offloaded nor processed, and is placed into the queues to accelerate stabilizing the network. On the other hand, according to the proofs of Theorems 10 and 11, the placeholder does not compromise the asymptotic optimality of Algorithm 6 under the condition that N servers collaborate.

We replace $\mathcal{Q}(t)$ with $\widetilde{\mathcal{Q}}(t)$ in Algorithm 6, i.e., configure the placeholder $\mathcal{Q}_0$ at slot 0, as given by

$$\widetilde{Q}_i^{(s)}(t) = Q_i^{(s)}(t) + Q_{i,0}^{(s)}, \forall i, s; \tag{7.31a}$$

$$\widetilde{D}_i^{(s)}(t) = D_i^{(s)}(t) + D_{i,0}^{(s)}, \forall i, s. \tag{7.31b}$$

We also develop tighter upper bounds for the instantaneous queue backlogs than (7.29), as stated in the following Corollary.

Corollary 4. *The instantaneous upper bounds of $Q_i^{(s)}(t)$ and $D_i^{(s)}(t)$ per slot t,*

denoted by $\mathcal{Q}_{\max} = \{Q_{i,\max}^{(s)}, D_{i,\max}^{(s)}, \forall i, s\}$*, can be given by*

$$\begin{aligned} Q_{i,\max}^{(s)} &= \begin{cases} Q_{\max}^{(s)}, \textit{ if } i = s; \\ \min\{\max_j\{Q_{j,\max}^{(s)} - w_{ij}\}, Q_{\max}^{(s)}\}, \textit{ otherwise;} \end{cases} \\ D_{i,\max}^{(s)} &= \min\{\max_j\{\tfrac{Q_{i,\max}^{(s)}}{\xi_s} - \varphi'^{(s)}_i, D_{j,\max}^{(s)} - w_{ij}\}, D_{\max}^{(s)}\}, \end{aligned} \tag{7.32}$$

where $\varphi'^{(s)}_i = \frac{\rho_s \zeta_i^{\min}}{\epsilon \xi_s} - \xi_s \widehat{F}_i / \rho_s$.

Proof. The proof is similar to that of Corollary 3 in Appendix D, and suppressed for brevity. □

Given the placeholder $\mathcal{Q}_0$ in Corollary 3 and the tight upper bounds $\mathcal{Q}_{\max}$ in Corollary 4, the collaborative region $\mathcal{R}_s$, within which servers maintain two queues for server s, i.e., $Q_i^{(s)}$ and $D_i^{(s)}$, can be determined by

$$\mathcal{R}_s = \{i | Q_{i,0}^{(s)} < Q_{i,\max}^{(s)}, \text{ and } D_{i,0}^{(s)} < D_{i,\max}^{(s)}\}, \tag{7.33}$$

This is because, if its placeholder exceeds the corresponding upper bound, by no means can the queue be scheduled for processing or offloading. In other words, the servers that are outside $\mathcal{R}_s$ do not need to maintain queues for server s, and do not participate in the fog computing for server s. The complexity of the proposed approach can be reduced from $\mathcal{O}(N)$ to the average size of the collaborative regions.

However, (7.30) and (7.32) are recursively presented and non-trivial to solve, due to their dependency in the calculation of $\mathcal{Q}_0$ and $\mathcal{Q}_{\max}$ between neighbouring servers. In particular, $Q_{i,0}^{(s)}$, $D_{i,0}^{(s)}$, $Q_{i,\max}^{(s)}$ and $D_{i,\max}^{(s)}$ require explicit knowledge on those of neighboring servers j, except that $D_{s,0}^{(s)} = 0$ for (7.30) and $Q_{s,\max}^{(s)} = Q_{\max}^{(s)}$ for (7.32) are known in prior.

7.3.2 Distributed Formation of Adaptive Collaborative Region

We discover that both (7.30) and (7.32) have the optimal substructure of the Bellman equations [46]. For example, (7.30a) can be regarded as an extended shortest path problem, where $D_{i,0}^{(s)}$ denotes the distance from server s to i defined by the

distances of its immediate neighbors, and the distance of each server is restricted to be non-negative due to Property 4. By exploiting the optimal substructure, or more specifically, the Bellman-Ford algorithm for shortest path problems [75], the solution for $\mathcal{Q}_0$ and $\mathcal{Q}_{\max}$ can be constructed based on the solutions for their subproblems.

Let $\widehat{\mathcal{Q}}_0(h) = \{\widehat{Q}_{i,0}^{(s)}(h), \widehat{D}_{i,0}^{(s)}(h), \forall i, s\}$ and $\widehat{\mathcal{Q}}_{\max}(h) = \{\widehat{Q}_{i,\max}^{(s)}(h), \widehat{D}_{i,\max}^{(s)}(h), \forall i, s\}$ collect the placeholders and the instantaneous upper bounds of the queues of all servers up to h hops away from server s, respectively. Here, a hop can involve either processing, i.e., $\varphi_i^{(s)}$ and $\varphi'^{(s)}_i$, or offloading w_{ij}, as captured in (7.30) and (7.32). $\widehat{\mathcal{Q}}_0(h)$ and $\widehat{\mathcal{Q}}_{\max}(h)$ can be effectively constructed from $\widehat{\mathcal{Q}}_0(h-1)$ and $\widehat{\mathcal{Q}}_{\max}(h-1)$. $\widehat{D}_{i,0}^{(s)}(h) = \widehat{D}_{i,0}^{(s)}(h-1)$ unless it can be further reduced and minimized by allowing one of its immediate neighbors, $j \in \mathbf{N}_i$, to pass on $\widehat{D}_{j,0}^{(s)}(h-1)$ through link (i,j) with the weight w_{ij}. To this end, we can restructure (7.30a) as

$$\widehat{D}_{i,0}^{(s)}(h) = \max\{\min_j\{\widehat{D}_{i,0}^{(s)}(h-1), \widehat{D}_{j,0}^{(s)}(h-1) + w_{ij}\}, 0\}. \tag{7.34a}$$

Similarly, from (7.30b) and (7.32), $\widehat{Q}_{i,0}^{(s)}(h)$, $\widehat{D}_{i,\max}^{(s)}(h)$ and $\widehat{Q}_{i,\max}^{(s)}(h)$ can be updated by

$$\widehat{Q}_{i,0}^{(s)}(h) = \max\left\{\min_j\left\{\begin{array}{l}\widehat{Q}_{i,0}^{(s)}(h-1), \widehat{Q}_{j,0}^{(s)}(h-1) + w_{ij}, \\ \xi_s \widehat{D}_{i,0}^{(s)}(h-1) + \varphi_i^{(s)}\end{array}\right\}, 0\right\}; \tag{7.34b}$$

$$\widehat{D}_{i,\max}^{(s)}(h) = \min\left\{\max_j\left\{\begin{array}{l}\widehat{D}_{i,\max}^{(s)}(h-1), \\ \widehat{D}_{j,\max}^{(s)}(h-1) - w_{ij}, \\ \frac{\widehat{Q}_{i,\max}^{(s)}(h-1)}{\xi_s} - \varphi'^{(s)}_i\end{array}\right\}, D_{\max}^{(s)}\right\}; \tag{7.34c}$$

$$\widehat{Q}_{i,\max}^{(s)}(h) = \min\{\max_j\{\widehat{Q}_{i,\max}^{(s)}(h-1), \widehat{Q}_{j,\max}^{(s)}(h-1) - w_{ij}\}, Q_{\max}^{(s)}\}. \tag{7.34d}$$

Exploiting the optimal substructure, (7.34) provides an efficient means to recursively compute $\widehat{\mathcal{Q}}_0$ and $\widehat{\mathcal{Q}}_{\max}$. The computation of (7.30), (7.32) and (7.33) can be implemented in a distributed fashion, as summarized in Algorithm 7. This is because the iterations of (7.34) only require the knowledge of a server and its immediate neighbors. Specifically, the algorithm starts by setting $\widehat{\mathcal{Q}}_0(0) = \infty$ and $\widehat{\mathcal{Q}}_{\max}(0) = -\infty$ except that $\widehat{D}_{s,0}^{(s)}(0) = 0$ and $\widehat{Q}_{s,\max}^{(s)}(0) = Q_{\max}^{(s)}$ based on the a-priori knowledge in (7.30) and (7.32). Server i updates $\widehat{Q}_{i,0}^{(s)}(h)$, $\widehat{D}_{i,0}^{(s)}(h)$, $\widehat{Q}_{i,\max}^{(s)}(h)$ and $\widehat{D}_{i,\max}^{(s)}(h)$ by collecting information on $\widehat{Q}_{j,0}^{(s)}(h-1)$, $\widehat{D}_{j,0}^{(s)}(h-1)$, $\widehat{Q}_{j,\max}^{(s)}(h-1)$

Algorithm 7 Online Formation of Collaborative Regions

Initialize $\widehat{\mathcal{Q}}_0(0) = \infty$ and $\widehat{\mathcal{Q}}_{\max}(0) = -\infty$ except that $\widehat{D}^{(s)}_{s,0}(0) = 0$ and $\widehat{Q}^{(s)}_{s,\max}(0) = Q^{(s)}_{\max}$
At each server i:
repeat
  Observe $\widehat{\mathcal{Q}}_0(h-1)$ and $\widehat{\mathcal{Q}}_{\max}(h-1)$ of its own and one-hop neighbors
  Update $\widehat{Q}^{(s)}_{i,0}(h)$, $\widehat{D}^{(s)}_{i,0}(h)$, $\widehat{Q}^{(s)}_{i,\max}(h)$, and $\widehat{D}^{(s)}_{i,\max}(h)$ by (7.34)
until $\widehat{\mathcal{Q}}_0(h) = \widehat{\mathcal{Q}}_0(h-1)$ or $\widehat{\mathcal{Q}}_{\max}(h) = \widehat{\mathcal{Q}}_{\max}(h-1)$
Set $Q^{(s)}_{i,0} = \widehat{Q}^{(s)}_{i,0}(h)$, $D^{(s)}_{i,0} = \widehat{D}^{(s)}_{i,0}(h)$, $Q^{(s)}_{i,\max} = \widehat{Q}^{(s)}_{i,\max}(h)$, and $D^{(s)}_{i,\max} = \widehat{D}^{(s)}_{i,\max}(h)$
Only maintain the queues satisfying (7.33)
Place the placeholders $Q^{(s)}_{i,0}$ and $D^{(s)}_{i,0}$ into the queues

and $\widehat{D}^{(s)}_{j,\max}(h-1)$ from its immediate neighbours $j \in \mathbf{N}_i$ and running (7.34) until convergence. Upon the convergence, we have $Q^{(s)}_{i,0} = \widehat{Q}^{(s)}_{i,0}(h)$, $D^{(s)}_{i,0} = \widehat{D}^{(s)}_{i,0}(h)$, $Q^{(s)}_{i,\max} = \widehat{Q}^{(s)}_{i,\max}(h)$, and $D^{(s)}_{i,\max} = \widehat{D}^{(s)}_{i,\max}(h)$. Then, server i can spontaneously form the collaborative region according to (7.33), and decide not to maintain the queues $Q^{(s)}_i$ and $D^{(s)}_i$ for server s, if $Q^{(s)}_{i,0} \geq Q^{(s)}_{i,\max}$ or $D^{(s)}_{i,0} \geq D^{(s)}_{i,\max}$.

The time-complexity of Algorithm 7 depends on the complexity of updating (7.34) per iteration and the number of iterations till convergence. As per the h-th iteration, a server needs to evaluate all the $(h-1)$-th iteration results of its own and immediate neighbors in $\mathcal{O}(N)$ time. The iterations converge in $(2N-1)$ for positive weights w_{ij}, $\varphi^{(s)}_i$ and $\varphi'^{(s)}_i$. This is because, in a network of N servers, the maximum length of offloading paths is $(2N-1)$ hops, consisting of 1 hop for processing and up to $2(N-1)$ hops for offloading tasks and returning results. Involving more hops would lead to circles along an offloading path, i.e., tasks are offloaded twice through a server, which results in unnecessary costs and does not contribute to the calculation of (7.34). As a result, the time-complexity of Algorithm 7 is $\mathcal{O}(N^2)$ per server.

7.4 Simulation Result and Analysis

In this section, we run extensive simulations to evaluate the proposed distributed approach due to the fact that fog computing is an emerging area and widely agreed benchmark suites have yet to be established. Our simulation platform is a network

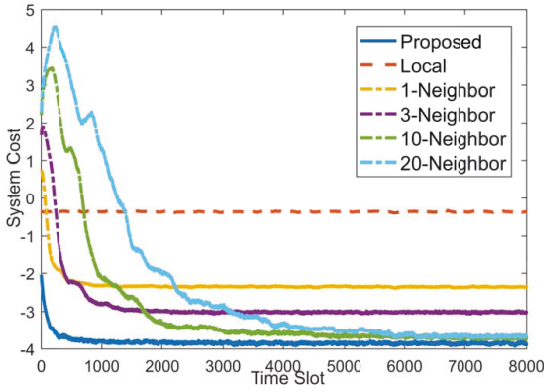


Figure 7.3 : The change of the system cost over time, where $1/\epsilon$=50.

of $N = 200$ servers. Each server is connected to 3 servers; unless otherwise specified. The income of admitting a task bit $\alpha_i = 1$ for each server i. 10000 slots are run for each data point, with the slot duration $T = 1$ sec. The computational capacity of the servers $\widehat{F}_i$ is uniformly distributed from 1 to 20 GHz with $\zeta_i = 1/\widehat{F}_i$ Joule/cycle, and the background task $\delta_i(t)$ varies randomly and uniformly from 0 to 40%. The tasks arrive at a subset of servers, where the percentage of servers with task arrivals is 0.2; unless otherwise specified. $A_i(t)$ is uniformly distributed in $[1, 8]$ Mbits, ρ_i is uniformly distributed in $[1000, 3000]$ cycles/bit, and $\xi_i = 1$.

For comparison purpose, we also simulate two benchmark approaches: (a) local execution (Local), where the servers buffer and execute tasks locally; and (b) fixed offloading scheme (Fixed offloading) which is an enhanced version of the state-of-the-art developed in [25], where "K-neighbor" indicates a server offloads tasks to its K nearest neighbors. Specifically, the state-of-the-art scheme of [25] is centralized and task offloading is confined within a single hop and tasks must be offloaded, processed, and returned all within a single time slot. There is no buffering of offloaded tasks, and tasks that cannot be instantly admitted due to limited resources have to be dropped. The capacity and cost-effectiveness of fog computing would be compromised. For fair comparisons, we have extended and enhanced the scheme of [25] to the Fixed offloading scheme by enabling the buffering of offloaded tasks.

Fig. 7.3 shows the change of the system cost of Proposed, Local and Fixed offloading over time, where t increases from 0 to 8000. We can see that the system

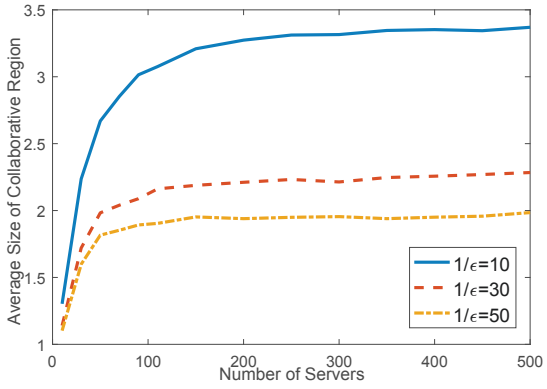
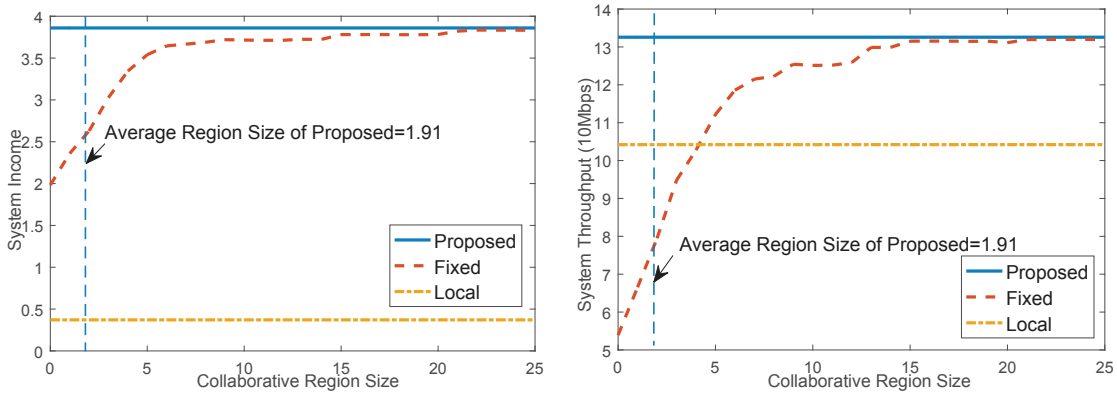


Figure 7.4 : The average collaborative region size, as N increases from 10 to 500.

costs of Proposed and Fixed offloading stabilize over time, and Proposed stabilizes faster and achieves the minimum cost. Recall that the system cost in (7.6) defines the difference between the costs of task processing and the income of task admission. A negative system cost is a positive income (or gain) of the system, and is labelled as "system income" in the rest of the figures. Proposed is able to achieve the minimum cost after stabilization. Fixed offloading can decrease the system cost by enlarging the collaborative region size, but also requires an increasingly long time for the network to stabilize. This is because enlarging the collaborative region would involve more servers in fog computing, and hence increase the implementation cost and delay convergence, as discussed in Section 7.3. The overall cost of Local does not change over time, and is much higher than the stabilized costs of Proposed and Fixed offloading.

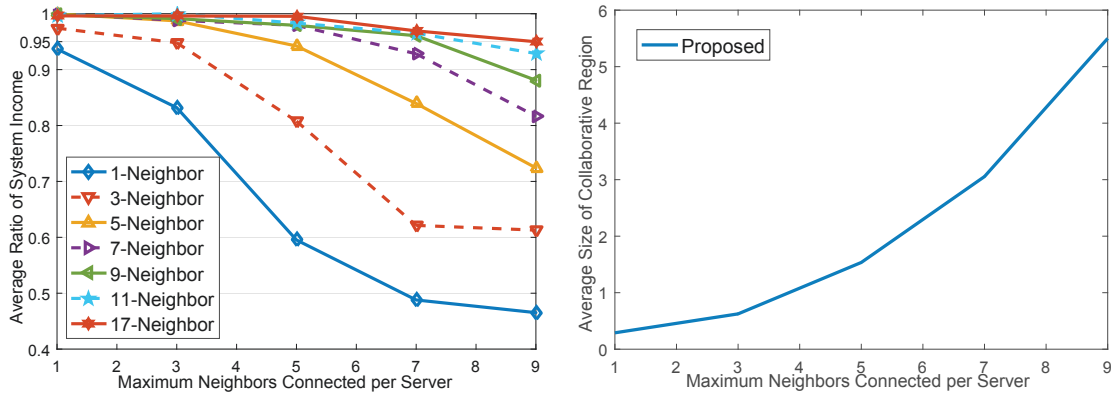
Fig. 7.4 plots the average size of the collaborative region of Proposed, as N increases from 10 to 500. The average region size first increases rapidly with the network size when $N \leq 200$ and then stabilizes. This indicates that the implementation overhead of Proposed does not keep increasing with the network size, as is preferable in large-scale networks. We also see that increasing $1/\epsilon$ can help decrease the average region size, since increasing $1/\epsilon$ can reduce the instantaneous upper bounds and increase the placeholders in (7.34).

Fig. 7.5 shows the system income and the throughput of Fixed offloading as the collaborative region size increases from 0 to 25. We can see that the system



(a) The system income (b) The system throughput

Figure 7.5 : The system income and throughput versus collaborative region size, where $1/\epsilon = 50$.

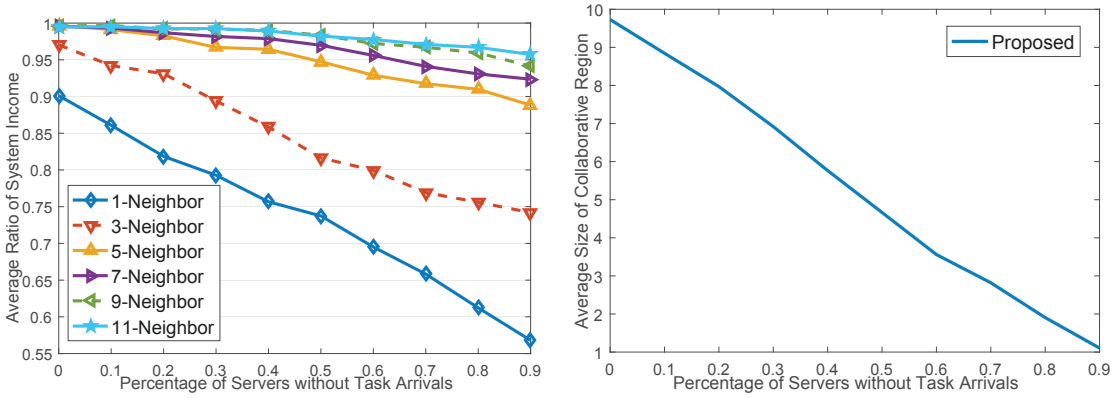


(a) The ratio of income between Fixed offloading and Proposed (b) The average region size

Figure 7.6 : The ratio of income between Fixed offloading and Proposed and the average region size versus maximum connected servers per node, where $1/\epsilon = 50$.

income and throughput increase monotonically with the region size, and approach to that of Proposed when the region size is larger than 20. This indicates that the proposed regions do not compromise the asymptotic optimality achieved by having all N servers collaborate, as presented in Section 7.2. On the other hand, the average region size is 1.91 when $1/\epsilon = 50$ and $N = 200$, as shown in Fig. 7.4, and the system income (or throughput) of Proposed is 35% (or 40%) higher than that of Fixed offloading under the same region size.

Fig. 7.6(a) plots the ratio of system income between Fixed offloading and Pro-



(a) The ratio of income between Fixed offloading and Proposed

(b) The average region size

Figure 7.7 : The ratio of income between Fixed offloading and Proposed and the average region size versus the percentage of servers without task arrivals, where $1/\epsilon = 50$.

posed, where the number of neighbors per server is uniformly distributed from one to the maximum number of neighbors per server that ranges from 1 to 9, as indicated by the x-axis. This captures the increasing inhomogeneity of the network in terms of connectivity. We can see that the ratio of system income monotonically decreases with the increasing connectivity of the servers. Also, the network requires increasingly large collaborative regions to maintain the ratio, as compared with Proposed.

Fig. 7.6(b) shows the collaborative region of Proposed. The average collaborative region of Proposed increases with the maximum number of neighbors connected per server, as the result of the growing average connectivity per server. Moreover, the collaborative region of Proposed is much smaller than that of Fixed offloading. For example, Fixed offloading involves 17 neighbors to achieve 95% ratio of Proposed that collaborates average 5.8 neighbors when the maximum number of neighbors per server is 9. This indicates that Proposed can achieve higher income in smaller collaborative regions than Fixed offloading.

Fig. 7.7(a) plots the ratio of system income between Fixed offloading and Proposed, where the percentage of servers without task arrivals increases from 0 to 0.9. This captures the increasing inhomogeneity of the network in terms of task arrivals.

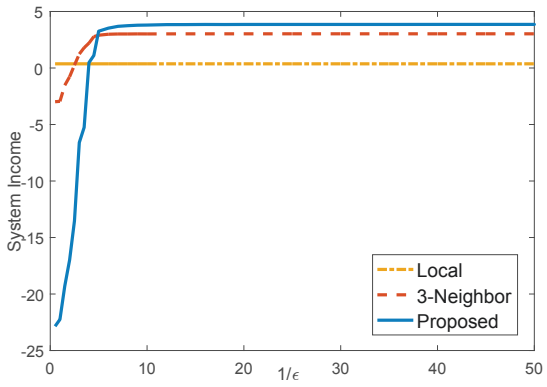


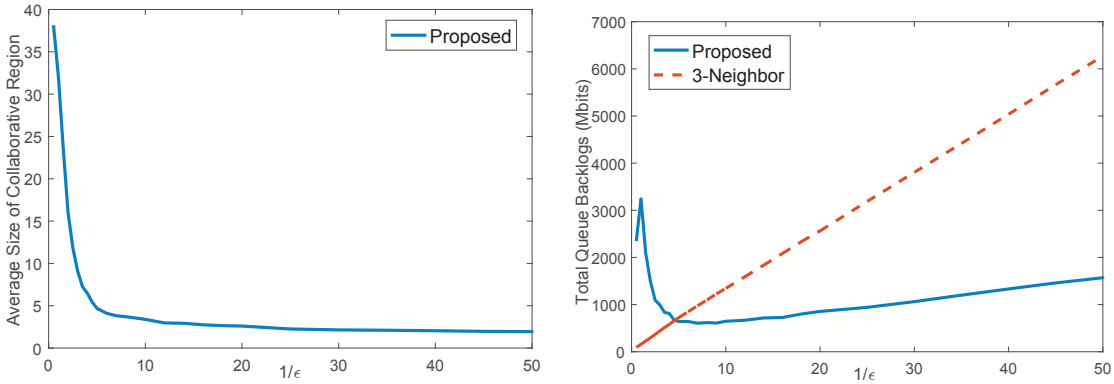
Figure 7.8 : The stabilized system income versus $1/\epsilon$.

Similar to Fig. 7.6(a), the ratio monotonically decreases with the increasing percentage of servers without task arrivals, and Fixed offloading requires increasingly large collaborative regions with the growth of such servers.

Fig. 7.7(b) shows the average collaborative region size of Proposed. We can see that the region size of Proposed decreases almost linearly with the percentage of servers without task arrivals, since Proposed can decrease the region size by tightening their upper bounds. As a result, Proposed can achieve higher income in smaller regions than Fixed offloading. For example, Fixed offloading involves 11 neighbors to achieve 95% ratio of Proposed that collaborates on average 1.2 neighbors when the percentage of servers without task arrivals is 0.9.

Fig. 7.8 plots the stabilized system income of Proposed, 3-Neighbor Fixed offloading (which can be regarded as only offloading tasks to immediate neighbors in the network with an average connectivity of 3 neighbors) and Local, as $1/\epsilon$ increases from 0 to 50. As dictated in Theorem 10, in this figure, the system incomes of Proposed and 3-Neighbor Fixed offloading first increase with $1/\epsilon$ and then stabilize when $1/\epsilon \geq 7$. Proposed can generate more income than 3-Neighbor Fixed offloading after stabilized, as also shown in Fig. 7.5.

It is worth mentioning that when $1/\epsilon$ is small, Proposed can incur a higher cost than Fixed offloading. This is because, with larger collaborative regions, as shown in Fig. 7.9(a), Proposed can admit more tasks by involving more servers to process according to (7.21) and (7.22). By restricting offloading within the 3 nearest neigh-



(a) Region size (b) Total queue backlogs

Figure 7.9 : The total queue backlogs and average region size versus $1/\epsilon$.

bors, 3-Neighbor Fixed offloading can reduce the energy consumption and hence the system cost, but achieve much less overall throughput than Proposed. Fig. 7.9(b) shows the total queue backlogs of Proposed and 3-Neighbor Fixed offloading as $1/\epsilon$ increases from 0 to 50. The backlogs of Proposed first increases when $1/\epsilon \leq 1$, then sharply drops, and finally slowly increases when $1/\epsilon \geq 5$. On the other hand, the backlogs of 3-Neighbor Fixed offloading continue increasing with $1/\epsilon$. This is because the collaborative region size sharply decreases with $1/\epsilon$ when $1/\epsilon \leq 5$, and then stabilizes, as shown in Fig. 7.9(a).

7.5 Extensions to Unreliable Networks

In this section, we discuss the extensions of the proposed distributed online approach to unreliable fog computing networks, where distributed computing with network coding or complicated channel coding is required in case of server failures or unreliable links.

1) Extension to distributed computing with network coding: Distributed computing using coding theory [76] can exploit the excess of servers to perform tasks with redundancy, and therefore is beneficial in case of server failure. The proposed approach has the potential to accommodate distributed computing using network coding. Specifically, the servers can encode admitted tasks to multiple subtasks with enriched redundancy (e.g., using the coding schemes in [76]). The subtasks can be

separately offloaded and processed, and the results of the subtasks can be returned, in the same way as described in Algorithm 1. The servers which have encoded the tasks into subtasks can retrieve the results of the subtasks and recover the result of the task. The only additional requirement is that each server needs to meticulously select the subtasks to be offloaded at every time slot (as opposed to operating in a first-in-first-out manner, as done in Algorithm 1), so that the subtasks of the same task can be prevented from being offloaded to the same neighboring server.

2)Extension of the proposed approach to unreliable links: The proposed approach can be extended to unreliable links, where complicated encoding and decoding are required for transmissions over every link. The computing resource allocation of a server now depends on not only the tasks to be processed, but also the tasks and results to be received and transmitted, at each time slot. The constraint on the computing resource allocation, i.e., (7.3a), needs to be rewritten as

$$\sum\nolimits_{s\in\mathbf{N}} f_i^{(s)}(t) + \sum\nolimits_{s,j\in\mathbf{N}} [(b_{ij}^{(s)}(t) + d_{ij}^{(s)}(t))\rho_e + (b_{ji}^{(s)}(t) + d_{ji}^{(s)}(t))\rho_d] \leq F_i(t),\ \forall i, \tag{7.35}$$

where ρ_e and ρ_d denote the number of CPU cycles for encoding and decoding per data bit. We recall $\zeta_{ij}(t)$ is the transmission price between servers i and j, $b_{ij}^{(s)}(t)$ is the size of tasks admitted at server s and forwarded to server j through server i, $d_{ij}^{(s)}(t)$ is the size of results returned from server i to server s through server j, and $f_i^{(s)}(t)$ is the CPU cycles that server i allocates to the tasks originating from server s, at time slot t.

The newly added terms $\sum_{s,j\in\mathbf{N}}[(b_{ij}^{(s)}(t)+d_{ij}^{(s)}(t))\rho_e+(b_{ji}^{(s)}(t)+d_{ji}^{(s)}(t))\rho_d]$ in (7.35) do not affect the decoupling of the variables between time slots using the stochastic subgradient method; in other words, (7.8) can still be reformulated into (7.9) per slot t. However, the new terms do prevent further decoupling of the decisions of each server on task offloading, task processing and result delivery, and the decisions between servers in (7.9). Problem (7.9) can only be decoupled to (7.19a) for task

admission, and (7.36) for the joint decisions of adjacent servers, as given by

$$\begin{aligned} &\max_{\mathbf{f}^t,\mathbf{b}^t,\mathbf{d}^t} \mu(\mathbf{f}^t) + \eta(\mathbf{b}^t, \mathbf{d}^t) \\ &\text{s.t. (7.2), (7.35) and (7.3b),} \end{aligned} \tag{7.36}$$

which is an LP problem and can be solved efficiently by standard LP solvers in a centralized manner, provided the instant global view of the network on computing and transmission resources is available.

We note that (7.36) can have the potential to be further decoupled for distributed online implementations in practice. One reason is that decoding typically starts after the whole packet has been received and de-interleaved. This can be a delay of one or multiple time slots. In (7.35), $b_{ji}^{(s)}(t)$ and $d_{ji}^{(s)}(t)$ are replaced by $b_{ji}^{(s)}(t-\tau)$ and $d_{ji}^{(s)}(t-\tau)$ which are constant at the moment of the optimization, i.e., slot t. $\tau \geq 1$. This decouples $b_{ji}^{(s)}(t)$ and $d_{ji}^{(s)}(t)$ from the rest of variables in (7.35). The other reason is that, in many cases, the wired link between a pair of servers have directional connections with independent capacities in different link directions. For example, standard 100BASE-BX fibers use different wavelengths in the two link directions, and there is no crosstalk between the link directions. Let $\vec{C}_{ij}(t)$ denote the capacity of the directional link from server i to j. The constraint on the link capacity (7.2a) can be updated as

$$\sum_{s\in\mathbf{N}} b_{ij}^{(s)}(t) + d_{ij}^{(s)}(t) \leq \vec{C}_{ij}(t). \tag{7.37}$$

As a result, (7.36) can be decoupled between servers, as given by

$$\begin{aligned} &\max_{\mathbf{f}_i^t,\mathbf{b}_{ij}^t,\mathbf{d}_{ij}^t} \sum_s \kappa_i^{(s)}(t) f_i^{(s)}(t) + \sum_{s,j} [\beta_{ij}^{(s)}(t) b_{ij}^{(s)}(t) + \gamma_{ij}^{(s)}(t) d_{ij}^{(s)}(t)] \\ &\text{s.t. (7.37), (7.2b), (7.35) and (7.3b).} \end{aligned} \tag{7.38}$$

Clearly, (7.38) is also LP, and can be efficiently solved at individual servers by evaluating the weights $\kappa_i^{(s)}(t)$, $\beta_{ij}^{(s)}(t)$ and $\gamma_{ij}^{(s)}(t)$ in a distributed manner. Specifically, in the descending order of the weights, the corresponding variables take the respective most significant values, one after another, until server i runs out of computing resources.

Chapter 8

Conclusion and Future Work

This thesis illustrated the potential use cases of fog computing in future networks, and discussed the key challenges of designing an efficient and scalable resource management approach for fog computing. The proposed five new approaches can address the challenges to achieve both efficiency and scalability of fog computing, as summarized in the following.

1. In Chapter 3, we take into account user quality of experience through user utility and total system utility functions. We propose a semi-distributed heuristic offloading decision algorithm, which ensures that mobile users with better utility are offloaded, while others perform local executions. Essentially, the complexity of the proposed algorithm is $O(K^3)$ in comparison to the $O(2^K)$ complexity of the optimal. Simulation results show that the proposed algorithm performs very close to the optimal solution.

2. In Chapter 4, we formulated task admission and resource allocation to minimize the total energy consumption of fog computing while guaranteeing the latency requirements of devices. This problem was reformulated as an integer programming problem by pre-admitting resource-restrained devices. A quantized DP algorithm was proposed to solve the integer programming problem at a polynomial complexity $\mathcal{O}(NK^2/\varepsilon)$. We also meticulously designed the quantization interval of energy saving to achieve the asymptotic optimality of the proposed algorithm with an $[\mathcal{O}(\varepsilon), \mathcal{O}(1/\varepsilon)]$-tradeoff between the optimality loss and time-complexity. Simulation results corroborate that, superior in efficiency and stability, the proposed scheme is able to save energy indistinguishably close to the maximum energy saving.

3. In Chapter 5, Lyapunov optimization techniques were taken to generate asymptotically optimal offloading schedules for MEC under partial network knowledge. Particularly, we decomposed the Lyapunov optimization problem into a knapsack problem which can be solved for asymptotically optimal schedules, given up-to-date network knowledge. We proceeded to evaluate the solution for the knapsack problem under out-of-date knowledge, and prove that the solution, encapsulating the asymptotically optimal schedules, preserves the asymptotic optimality. It can be used for devices to self-nominate for feedback and facilitate scheduling. Simulations show that the proposed approach is able to reduce feedbacks by over 98% without compromising the asymptotic optimality, e.g., only require 60 out of 5000 devices to feed back per slot.

4. In Chapter 6, the processing and offloading of tasks, and the return of results were jointly optimized for multi-hop wireless fog computing in the presence of out-of-date signaling resulting from multi-hop propagation. Lyapunov optimization techniques were exploited to asymptotically minimize the time-average energy consumption with an $[\mathcal{O}(V), \mathcal{O}(1/V)]$-tradeoff between the optimality gap and delay. The impact of out-of-date signaling on the asymptotic optimality was proved to be bounded, hence preserving the asymptotic optimality of the proposed approach. Simulations show that our approach is able to reduce the time-average energy consumption by 50% and accommodate larger tasks by engaging wireless devices hops away to collaborate.

5. In Chapter 7, we proposed the new distributed online optimization of collaborative regions adapting to network connectivity and task variations of fog computing. The stochastic subgradient methods were exploited to asymptotically minimize the time-average cost and stabilize queues. The placeholders and instantaneous upper bounds of all the queues were derived with the optimal substructure of the Bellman equations. By comparing the placeholders and bounds, the collaborative regions of fog computing are autonomously formed adaptively, without violating the asymptotic optimality. Simulations show that our scheme reduces the average region size to only 3.2 out of total

500 servers, and achieves 43% higher system income than counterparts with persistent collaborative regions.

The proposed approach are developed under the assumptions that all tasks are of the same type, and can either be divisible or statically partitioned under the parallel dependency model [35, Fig. 4(b)]. The assumptions hold in many application scenarios and have been extensively adopted in the literature[11, 12, 16–19, 21, 25, 26]. Yet, there are other scenarios where there are multiple different types of tasks, and the tasks can follow the general dependency model [35, Fig. 4(c)] and need to be processed in particular orders, such as those modeled by directed acyclic graphs (DAG) [77], as discussed in Section 2.2. Non-trivial efforts would be required to extend the proposed approach to those scenarios, such as the automated installation of virtual instances (at different devices) for different task types and optimal offloading of dependent DAG tasks. These will be the focus of our future research.

Bibliography

[1] O. Consortium *et al.*, "Openfog reference architecture for fog computing," *Architecture Working Group*, 2017.

[2] Y. Zhang, J. Ren, J. Liu, C. Xu, H. Guo, and Y. Liu, "A survey on emerging computing paradigms for big data," *Chinese J. Electron.*, vol. 26, no. 1, pp. 1–12, 2017.

[3] J. Ren, Y. Zhang, K. Zhang, and X. Shen, "Exploiting mobile crowdsourcing for pervasive cloud services: challenges and solutions," *IEEE Commun. Mag.*, vol. 53, no. 3, pp. 98–105, March 2015.

[4] 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.

[5] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, September 2013.

[6] W. Zhang, Y. Wen, and D. O. Wu, "Collaborative task execution in mobile cloud computing under a stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 14, no. 1, pp. 81–93, Jan 2015.

[7] Y.-H. Kao, B. Krishnamachari, M.-R. Ra, and F. Bai, "Hermes: Latency optimal task assignment for resource-constrained mobile computing," in *2015 IEEE INFOCOM*, April 2015, pp. 1894–1902.

[8] M.-R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan, "Odessa: enabling interactive perception applications on mobile devices," in *Mobisys*. ACM, 2011, pp. 43–56.

[9] Z. Cheng, P. Li, J. Wang, and S. Guo, "Just-in-time code offloading for wearable computing," *IEEE Trans. Emerg. Topics Comput.*, vol. 3, no. 1, pp. 74–83, March 2015.

[10] C. You, K. Huang, H. Chae, and B. H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, 2017.

[11] 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.

[12] 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.

[13] 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.

[14] K. Wang, K. Yang, and C. Magurawalage, "Joint energy minimization and resource allocation in c-ran with mobile cloud," *IEEE Trans. Cloud Comput.*, 2016.

[15] 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.

[16] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, April 2015.

[17] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 974–983, April 2015.

[18] 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, April 2017.

[19] 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.*, vol. 66, no. 6, pp. 2603–2616, June 2018.

[20] B. P. Rimal, D. P. Van, and M. Maier, "Mobile-edge computing versus centralized cloud computing over a converged fiwi access network," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 3, pp. 498–513, Sept 2017.

[21] H. Shah-Mansouri and V. W. S. Wong, "Hierarchical fog-cloud computing for iot systems: A computation offloading game," *IEEE Internet Things J.*, pp. 1–1, 2018.

[22] B. P. Rimal, D. P. Van, and M. Maier, "Cloudlet enhanced fiber-wireless access networks for mobile-edge computing," *IEEE Trans. Wireless Commun.*, vol. 16, no. 6, pp. 3601–3618, June 2017.

[23] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, "Multiobjective optimization for computation offloading in fog computing," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 283–294, Feb 2018.

[24] L. Liu, Z. Chang, and X. Guo, "Socially aware dynamic computation offloading scheme for fog computing system with energy harvesting devices," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1869–1879, June 2018.

[25] 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, Dec 2016.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] M. Mukherjee, L. Shu, and D. Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 1826–1857, thirdquarter 2018.

[31] P. Zhang, J. K. Liu, F. R. Yu, M. Sookhak, M. H. Au, and X. Luo, "A survey on access control in fog computing," *IEEE Communications Mag.*, vol. 56, no. 2, pp. 144–149, Feb 2018.

[32] "Mobile-edge computing introductory technical white paper," *White Paper, Mobile-edge Computing (MEC) industry initiative*, 2014.

[33] R. K. Naha, S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gao, Y. Xiang, and R. Ranjan, "Fog computing: Survey of trends, architectures, requirements, and research directions," *IEEE Access*, vol. 6, pp. 47 980–48 009, 2018.

[34] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, Jan 2017.

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

[36] X. Lin, Y. Wang, Q. Xie, and M. Pedram, "Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment," *IEEE Trans. Services Comput.*, vol. 8, no. 2, pp. 175–186, March 2015.

[37] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, 2016.

[38] A. Goldsmith, *Wireless communications*. Cambridge university press, 2005.

[39] Y. Pochet and L. A. Wolsey, *Production planning by mixed integer programming*. Springer Science & Business Media, 2006.

[40] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[41] J.-L. Goffin, Z.-Q. Luo, and Y. Ye, "Complexity analysis of an interior cutting plane method for convex feasibility problems," *SIAM Journal on Optimization*, vol. 6, no. 3, pp. 638–652, 1996.

[42] S. FUJISHIGE, "Submodular functions and optimization," *Annals of Discrete Mathematics*, 1991.

[43] E. U. T. R. Access, "Further advancements for E-UTRA physical layer aspects," 3GPP TR 36.814, Tech. Rep., 2010.

[44] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *IEEE Symp. on Computers and Communications (ISCC)*, 2012, pp. 59–66.

[45] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *Mobisys*. ACM, 2010, pp. 49–62.

[46] 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.

[47] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001.

[48] N. Megiddo, "Linear programming in linear time when the dimension is fixed," *J. ACM*, vol. 31, no. 1, pp. 114–127, Jan. 1984.

[49] T. B. L. E. L. R. Ramjee, "Generalized proportional fair scheduling in third generation wireless data networks," in *2006 IEEE INFOCOM*, 2006.

[50] X. Lyu and H. Tian, "Adaptive receding horizon offloading strategy under dynamic environment," *IEEE Commun. Lett.*, vol. 20, no. 5, pp. 878–881, May 2016.

[51] Y. L. Lee, T. C. Chuah, J. Loo, and A. Vinel, "Recent advances in radio resource management for heterogeneous lte/lte-a networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 2142–2180, Fourthquarter 2014.

[52] H. V. Poor and S. Verdu, "Probability of error in mmse multiuser detection," *IEEE Trans. Inf. Theory*, vol. 43, no. 3, pp. 858–871, May 1997.

[53] R. S. Garfinkel and G. L. Nemhauser, *Integer programming*, vol. 4.

[54] 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.

[55] X. Chen, W. Ni, X. Wang, and Y. Sun, "Optimal quality-of-service scheduling for energy-harvesting powered wireless communications," *IEEE Trans. Wireless Commun.*, vol. 15, no. 5, pp. 3269–3280, May 2016.

[56] S. Lin, W. Ni, H. Tian, and R. P. Liu, "An evolutionary game theoretic framework for femtocell radio resource management," *IEEE Trans. Wireless Commun.*, vol. 14, no. 11, pp. 6365–6376, Nov 2015.

[57] 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.

[58] A. O. Allen, *Probability, statistics, and queueing theory.* Academic Press, 2014.

[59] D. Pisinger, "Algorithms for knapsack problems," 1995.

[60] I. S. Committee *et al.*, "Precision clock synchronization protocol for networked measurement and control systems," *IEEE Std*, vol. 1588, 2004.

[61] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, ser. SenSys '03. ACM, 2003, pp. 138–149.

[62] J. Hill and D. Culler, "A wireless embedded sensor architecture for system-level optimization," *Technical report*, 2001.

[63] I. Demirkol, C. Ersoy, and F. Alagoz, "Mac protocols for wireless sensor networks: a survey," *IEEE Commun. Mag.*, vol. 44, no. 4, pp. 115–121, April 2006.

[64] 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.

[65] 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.

[66] L. Suresh, M. Canini, S. Schmid, and A. Feldmann, "C3: Cutting tail latency in cloud data stores via adaptive replica selection," in *NSDI*. Oakland, CA: USENIX Association, 2015, pp. 513–527.

[67] 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.

[68] 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.

[69] G. Dantzig, *Linear programming and extensions.* Princeton University Press, 2016.

[70] W. Sun, Z. Yang, X. Zhang, and Y. Liu, "Energy-efficient neighbor discovery in mobile ad hoc and wireless sensor networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1448–1459, Third 2014.

[71] B. P. Rimal, D. P. Van, and M. Maier, "Mobile edge computing empowered fiber-wireless access networks in the 5g era," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 192–200, February 2017.

[72] 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.

[73] S. Boyd, L. Xiao, and A. Mutapcic, "Subgradient methods," *Stanford University*, 2009.

[74] 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.

[75] D. B. West *et al.*, *Introduction to graph theory.* Prentice hall Upper Saddle River, 2001, vol. 2.

[76] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. Inf. Theory*, 2017.

[77] B. P. Rimal and M. Maier, "Workflow scheduling in multi-tenant cloud computing environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 1, pp. 290–304, Jan 2017.