RESOURCE MANAGEMENT FOR COST-EFFECTIVE CLOUD AND EDGE SYSTEMS

By

ZHE YU

Bachelor of Engineering in Communications Engineering
University of Science and Technology Beijing
Beijing, China
2014

Master of Science in Electrical Engineering
Vanderbilt University
Nashville, Tennessee
2016

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
December, 2022

RESOURCE MANAGEMENT FOR COST-EFFECTIVE CLOUD AND EDGE SYSTEMS

Dissertation Approved:

Dr. Guoliang Fan

Dissertation Advisor

Dr. Martin Hagan

Dr. Sabit Ekin

Dr. Chenang Liu

ACKNOWLEDGMENTS

First and foremost I am extremely grateful to my advisor, Prof. Guoliang Fan for his invaluable advice, continuous support, and patience during my PhD study. His immense knowledge and plentiful experience have encouraged me in all the time of my academic research and daily life. I would like to thank my doctoral committee members, Prof. Martin Hagan, Prof. Sabit Ekin and Prof. Chenang Liu for their academic support and helpful guidance during my PhD study.

I would also like to thank Dr. Cheng Hao, Prof. Chris Hutchens, Prof. Gary Yen, Prof. James West, Prof. Jerzy Krasinski, Le Zhou, Nate Lannan, Prof. Qi Cheng, Prof. Teague Keith, Prof. Tyler Ley, Xiaowei Chen, Prof. Yuanxiong Guo, Prof. Gong Yanming and Zonghao Huang. It is their kind help and support that have made my PhD study and life in the Oklahoma State University a wonderful time. Finally, I would like to express my gratitude to my parents and my girlfriend. Without their tremendous understanding and encouragement in the past few years, it would be impossible for me to complete my PhD study.

Acknowledgments reflect the views of the author and are not endorsed by committee members or Oklahoma State University.

Name: ZHE YU

Date of Degree: DECEMBER, 2022

Title of Study: RESOURCE MANAGEMENT FOR COST-EFFECTIVE CLOUD AND EDGE SYSTEMS

Major Field: ELECTRICAL ENGINEERING

Abstract: With the booming of Internet-based and cloud/edge computing applications and services, datacenters hosting these services have become ubiquitous in every sector of our economy which leads to tremendous research opportunities. Specifically, in cloud computing, all data are gathered and processed in centralized cloud datacenters whereas in edge computing, the frontier of data and services is pushed away from the centralized cloud to the edge of the network. By fusing edge computing with cloud computing, the Internet companies and end users can benefit from their respective merits, abundant computation and storage resources from cloud computing, and the data-gathering potential of edge computing. However, resource management in cloud and edge systems is complicated and challenging due to the large scale of cloud datacenters, diverse interconnected resource types, unpredictable generated workloads, and a range of performance objectives. It necessitates the systematic modeling of cloud and edge systems to achieve desired performance objectives.

This dissertation presents a holistic system modeling and novel solution methodology to effectively solve the optimization problems formulated in three cloud and edge architectures: 1) *cloud computing in colocation datacenters*; 2) *cloud computing in geographically distributed datacenters*; 3) *UAV-enabled mobile edge computing*. First, we study resource management with the goal of overall cost minimization in the context of cloud computing systems. A cooperative game is formulated to model the scenario where a multi-tenant colocation datacenter collectively procures electricity in the wholesale electricity market. Then, a two-stage stochastic programming is formulated to model the scenario where geographically distributed datacenters dispatch workload and procure electricity in the multi-timescale electricity markets. Last, we extend our focus on joint task offloading and resource management with the goal of overall cost minimization in the context of edge computing systems, where edge nodes with computing capabilities are deployed in proximity to end users. A nonconvex optimization problem is formulated in the UAV-enabled mobile edge computing system with the goal of minimizing both energy consumption for computation and task offloading and system response delay. Furthermore, a novel hybrid algorithm that unifies differential evolution and successive convex approximation is proposed to efficiently solve the problem with improved performance.

This dissertation addresses several fundamental issues related to resource management in cloud and edge computing systems that will further in-depth investigations to improve cost-effective performance. The advanced modeling and efficient algorithms developed in this research enable the system operator to make optimal and strategic decisions in resource allocation and task offloading for cost savings.

TABLE OF CONTENTS

Chapter Page

Chapter Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

1.1 Motivation and Background

1.1.1 Cloud Computing and Edge Computing

The last decade has witnessed the emergence of cloud computing as a paradigm of computing, which provides computation, communication, and storage resources installed in remote large datacenters. Datacenters are buildings where multiple servers and communication gear are colocated because of their common environmental requirements and physical security needs, and for ease of maintenance [15]. Cloud computing utilizes wireless Internet to access and store data instead of storing data in users' own computer hard drives, which it is referred as local computing. Cloud computing has been the driving force for the rapid growth of many Internet companies such as Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform and IBM Cloud, who continue to invest heavily in offering new tools to customers, such as security and privacy, big data & analytics, networking, artificial intelligence (AI) & machine learning and IoT [2].

In recent years, as IoT devices become more pervasive and incorporate more processing power, an enormous amount of data is being generated on the outer edge of computing networks. Traditionally, data generated by IoT devices are uploaded to the central cloud servers for processing, and processed results are sent back to end devices located on the edge of network. However, it takes relatively long time for data to travel back and forth between IoT devices and core cloud datacenters, which imposes tremendous pressure on bandwidth.

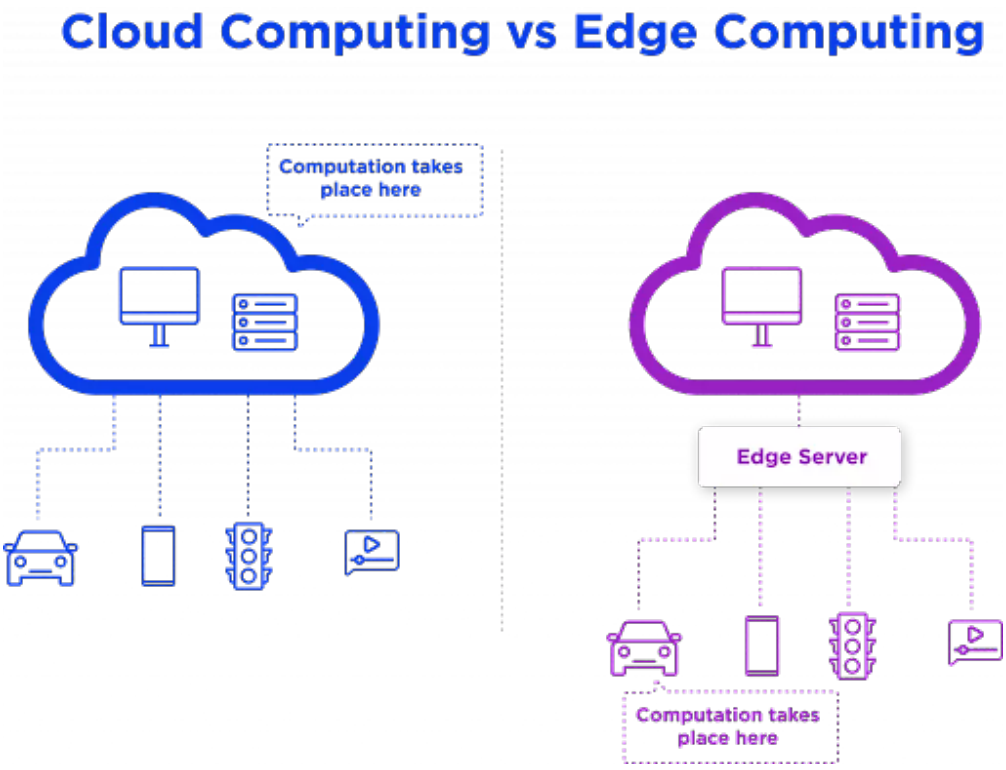


Figure 1: Cloud computing vs edge computing [6].

The combination of long distance and high volume traffic can slow the network down to a crawl. Therefore, edge computing has emerged as a solution to resolve the network latency issue by pushing the frontier of data and services away from centralized cloud to the edge of the network, thereby enabling data analytics and functional operation in the proximity to the data sources. Fortunately, cloud computing and edge computing are not exclusive to each other. The combination of edge computing and cloud computing enables Internet companies to take respective advantage of storage capacity and processing capability of the cloud datacenters and the data-gathering potential of edge computing [3].

1.1.2 Datacenter Energy Management

Datacenters, as a vital carrier for hosting Internet-based cloud computing applications and services, are especially notorious for their intensive energy consumption. A single datacenter can take as much electricity as a medium-size town [39]. According to a recent report [96] issued by the Lawrence Berkeley National Laboratory, datacenters in the U.S. consumed

an estimate 70 billion kWh of electricity in 2014, representing 1.8% of total U.S. electricity consumption and costing U.S. businesses more than 10 billion in annual electricity bills, and their total electricity consumption is estimated to be 73 billion kWh in 2020. Consequentially, cutting down electricity cost has become an urgent concern for datacenter operators. Moreover, it is imperative to advocate renewable energy uses from the perspective of sustainable environment since utilization of fossil fuels causes the surge of greenhouse gas emissions globally and further climate changes.

In view of massive energy consumption and global greenhouse gas footprint brought by datacenters for supporting cloud/edge computing services, our goal with this dissertation is to find efficient and effective ways to manage resource uses for cloud and edge systems such that optimal energy utilization can be achieved. In the context of cloud and edge systems, resource management [53] refers to the process of allocating computation, networking, storage and power resources efficiently and effectively to a set of applications while jointly satisfy the performance objectives of the applications, cloud providers and end users.

In terms of energy management for cloud and edge systems, substantial efforts have been made both from demand side and supply side when datacenter operators aim at reducing the growing energy consumption and electricity bills. On one hand, energy-efficient hardware facilities such as IT servers, energy storage devices and network switches and software mechanisms such as virtualization and dynamic CPU speed scaling (computation resources), dynamic capacity provisioning (server management) and geographical load balancing (workload management) have been developed to optimize the electricity cost from the demand side. On the other hand, from the supply side, datacenters can utilize the temporal diversity of electricity prices by shifting their delay-tolerant workload to off-peak time periods and using energy storage devices and on-site renewable generators to save electricity expenditures. In addition, datacenters can purchase electricity in retail markets beforehand or participate in multi-timescale electricity markets.

Given the significant power consumption and deregulation of electricity market, another

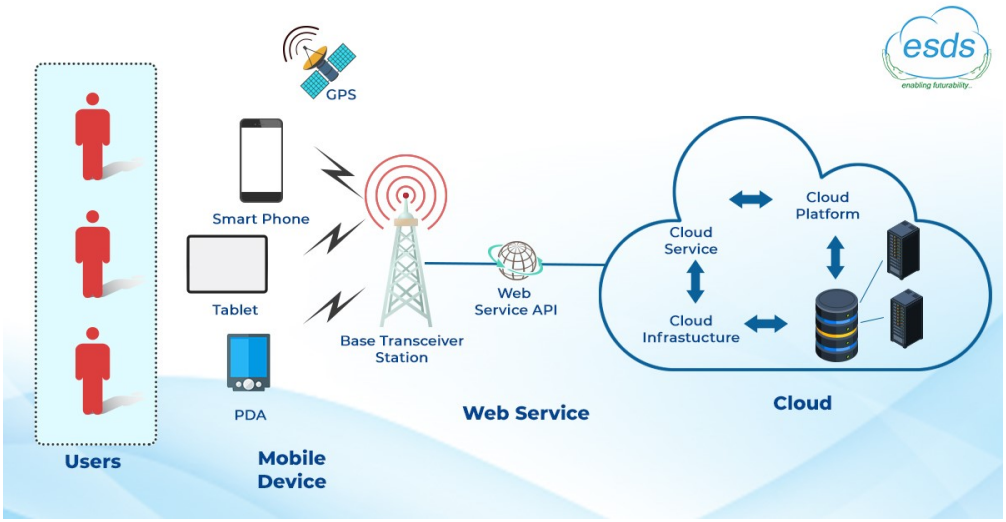
promising opportunity to reduce datacenter energy cost is emerging: datacenters can directly participate in electricity market to meet their power demand. While it is typical for consumers to buy electricity from local utility companies, some independent system operators (ISOs), such as Electric Reliability Council of Texas (ERCOT) [4] and California ISO [1], have recently developed a market that allows consumers to purchase electricity directly from power suppliers by actively participating in the electricity market. Indeed, datacenter operators like Google have been granted the authority to trade in the wholesale electricity market for the purpose of managing their own energy cost [9]. *The key advantage for datacenters to procure electricity from the wholesale electricity market instead of a local utility company is that they can avoid the insurance premiums, service charges, and mark-up included by utilities in retail rates* [37].

1.1.3 Mobile Cloud Computing and Mobile Edge Computing

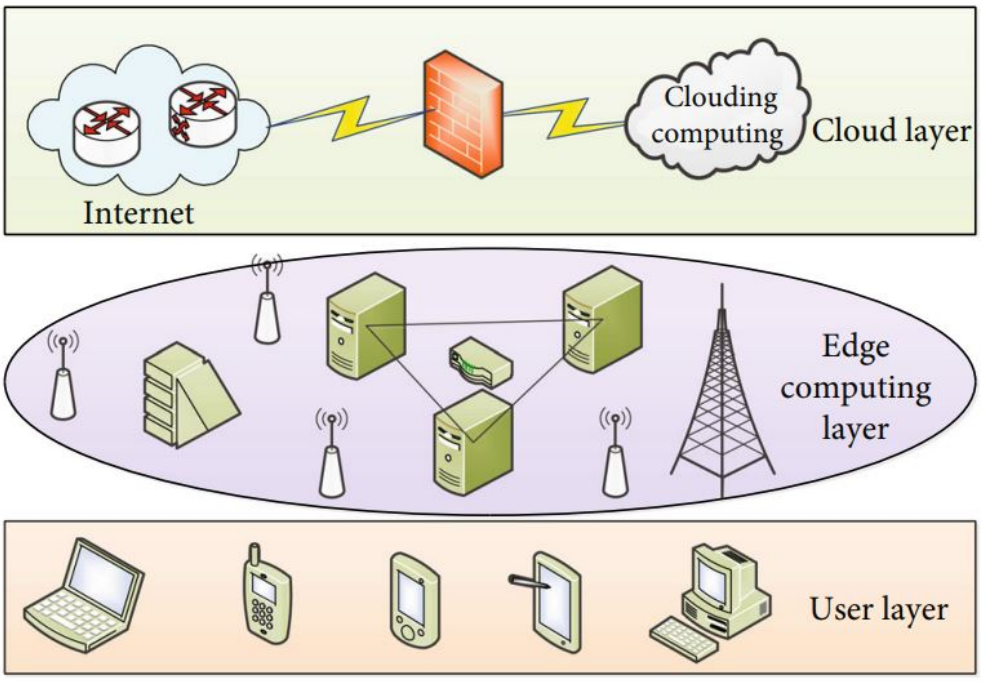
In the meanwhile, explosive growth of mobile devices (e.g., smartphones, tablets) have been seen in the past few years. Mobile computing functions include accessing the Internet through browsers, supporting multiple software applications with a core operating system, and sending and receiving different types of data. The ubiquity of mobile devices is accompanied by the increasing demand for computation and storage of intensive mobile applications such as image processing, computer vision, face recognition and natural language processing (NLP). In general, mobile computing is consumer-facing while cloud computing is business-facing. However, many of these mobile applications involve heavy computation and storage demands, whose performances are essentially limited by the deficiency of CPU speed, memory size and battery life. Therefore, It is intuitive for mobile users to resort to resourceful cloud for rapid execution and abundant storage, which leads to a new research area called mobile cloud computing (MCC) [32] as a combination of cloud computing and mobile computing. In mobile cloud computing, mobile users can run their computing services in remote resourceful datacenters by offloading their computation tasks via wireless and wired connections. Gen-

erally, it is a two-tier network infrastructure, which consists of a local layer of mobile devices and a cloud layer of remote datacenters. Two types of offloading schemes including binary offloading and partial offloading are considered in line with the MCC framework. On the one hand, mobile users can choose to offload all their computation tasks to remote cloud without any local computing while on the other hand, mobile users can pre-execute part of their computation jobs and then choose to offload the rest to the remote cloud for further processing.

Indeed, an inherent limitation resides in MCC, i.e., long propagation distance from end users to the remote cloud servers. Therefore, the round trip transmission time for task uploading and downloading actually becomes the bottleneck for enhancing the user experience by reducing system response delay although task computation delay can be largely shortened due to task offloading. Excessively long latency experienced by end users necessitate new network paradigm known as mobile edge computing (MEC) [123], micro datacenters (cloudlets) for mobile computing, where a capillary distribution of cloud computing are deployed in the vicinity of the mobile users, which largely reduces the transmission delay between mobile users and edge clouds. Applications with low latency tolerance, such as augmented reality (AR) & virtual reality (VR), video streaming and online gaming, can deploy their services on the edge hosts at a cost, to achieve lower latency and better user experience. Generally, it is a three-tier network infrastructure: 1) *Cloud layer* consists of cloud datacenters with large-scale servers located far from some mobile users. The cloud computational capacity is much higher than that of base stations or access points but the communication delay is high as well due to the long distance between base stations or access points and the cloud; 2) *Edge layer* consists of base stations or access points equipped with computational capacity and power resources higher than mobile devices, which can be widely deployed in the proximity to mobile users. Mobile users can choose to offload (part of) their computational tasks for processing at base stations or access points with short computation delay; 3) *User layer* consists of mobile users who possess mobile devices with computational tasks to perform.



(a) Mobile cloud computing [5]



(b) Mobile edge computing [46]

Figure 2: Examples of mobile cloud computing and mobile edge computing.

Communication delay can be avoided if those computational tasks are processed locally in their mobile devices. However, the computational capabilities of mobile devices are largely limited compared to the edge node servers and cloud servers. Mobile users can decide to offload (part of) their computational tasks to base stations or access points or further to the remote cloud to reduce the computation delay. Follow the same offloading schemes

mentioned in MCC framework, an extra middle layer of MEC framework can be utilized to pre-process offloaded computation tasks from mobile users with less transmission delay. If needed, resulting tasks can be further offloaded to the remote powerful cloud datacenters for execution.

1.1.4 UAV-Enabled Mobile Edge Computing

By moving resources to the network edge, close to where the data is being generated and acted upon, MEC can bring many benefits to users, such as lower service latency, reduced network congestion, and better service quality. Meanwhile, resource management becomes a key problem in MEC due to the much limited resources compared to remote clouds and the tight coupling of communication and computing. There have been substantial research on MEC resource management with the goal of optimizing system latency [115, 118, 25, 87], energy consumption [90, 136, 124] and overall cost of system latency and/or energy consumption [27, 24, 23, 134]. However, all of these studies assume wired or dedicated wireless connections with sufficient bandwidth among distributed edge resources deployed in a fixed fashion. Particularly, the existing MEC techniques are not applicable to the situation where the number of mobile users increases explosively or the network facilities are sparely distributed [47]. In view of this insufficiency, wireless networks enabled by unmanned aerial vehicles (UAVs) have recently been proposed as a promising solution to improve the connectivity of ground IoT devices.

UAVs, especially low-cost quadcopters, are undergoing an explosive growth and major regulation relaxation nowadays and have been widely used in civilian domains, such as traffic monitoring [77], public safety [126], search and rescue [95], and reconnaissance over disaster rescue and recovery [72]. UAVs not only provide extended coverage over wide geographical areas, but also possess unique characteristics like fast deployment, easy programmablity, and high scalability. Various payloads such as IoT sensors (including cameras), miniaturized base stations and embedded computing modules can be mounted on UAVs to enable

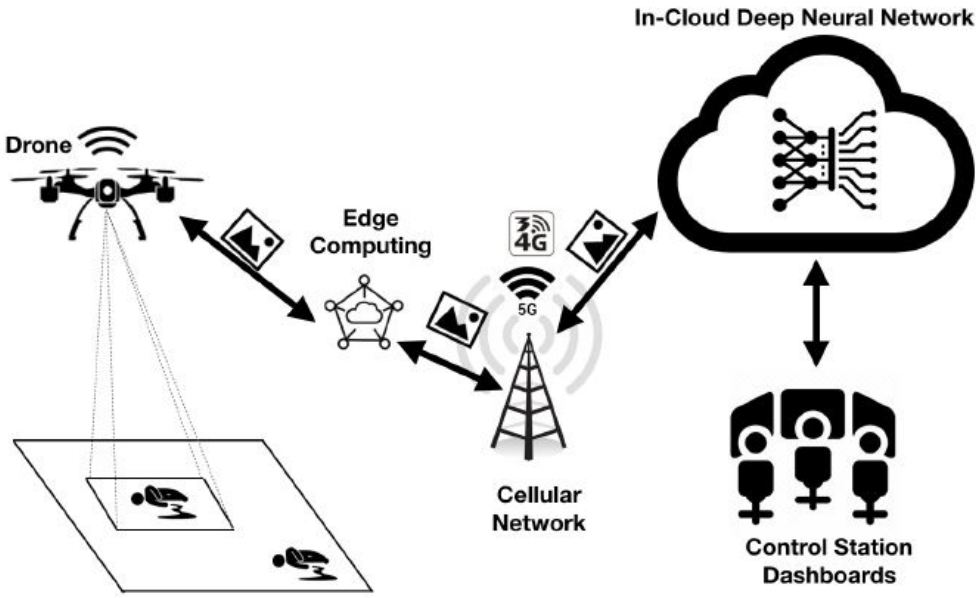


Figure 3: UAV-enabled mobile edge computing example [60].

different sensing, communication, and computing tasks [18, 75]. In particular, reliable and cost-effective wireless communication solutions for multitudes of real-world scenarios can be offered by UAVs if properly deployed and operated [75]. UAVs can act as wireless relays or aerial base stations for improving connectivity and extending coverage of ground wireless devices since the high altitude of UAV enables wireless devices to effectively establish line-of-sight (LoS) communication links thus mitigating the potential signal blockage and shadowing.

1.2 Summary of Contribution

Through this dissertation, we study the resource management for cost-effective cloud and edge systems. The results and future works are presented in Chapters III, IV, V, VI and VII. In particular, this dissertation makes the following contributions.

1.2.1 Energy Management in Colocation Datacenters

In Chapter III, we investigate how colocation datacenter can effectively reduce its energy cost when it participates in the wholesale electricity market via cooperative power procurement.

In summary, this chapter makes the following contributions:

- We study the problem of energy cost minimization for multi-tenant at a colocation datacenter when they participate in the wholesale electricity market via cooperative power procurement.
- We apply cooperative game theory to model the cooperative electricity procurement process of tenants as a cooperative game, and show the cost saving benefits of aggregation.
- We propose a cost allocation scheme based on the marginal contribution of each tenant to the total expected cost is proposed to fairly distribute the aggregation benefits among the participating tenants.
- We conduct numerical experiments based on real-world traces to illustrate the benefits of aggregation compared to noncooperative power procurement.

1.2.2 Resource Management in Geographically Distributed Datacenters

In Chapter IV, we investigate how geographically distributed datacenters can effectively reduce its overall cost when they participate in the multi-timescale electricity market. In summary, this chapter makes the following contributions:

- We study the optimal energy procurement for datacenter operator who manages geographically distributed datacenters when it participates in the multi-timescale electricity markets.
- We jointly optimize electricity procurement in the multi-timescale electricity markets, workload routing decisions, IT server allocation decisions, thermal energy storage charge and discharge decisions such that the total cost of datacenters when serving both interactive and batch workload can be minimized.

- We formulate it as a two-stage stochastic optimization problem by considering random scenarios and then reformulate as its deterministic equivalent problem.
- We conduct a case study based on real-world traces to validate the effectiveness of our proposed stochastic approach.

1.2.3 Joint Task Offloading and Resource Allocation in UAV-Enabled Mobile Edge Computing

In Chapter V, we investigate how a UAV can be properly deployed to facilitate the MEC service provisioning to a set of IoT devices in regions where existing ECs cannot be accessible to IoT devices due to terrestrial signal blockage or shadowing. In summary, this chapter makes the following contributions:

- We propose a novel UAV-enabled MEC system where a UAV is deployed to facilitate the provisioning of MEC services to IoT devices that cannot directly access ECs on the ground due to terrestrial signal blockage and shadowing.
- Considering the stringent quality-of-service requirement of MEC services and limited battery size of UAV, we formulate the joint IoT task offloading and UAV placement under the proposed system as an optimization problem with the goal of minimizing the service delay of IoT devices and maximizing the energy efficiency of UAV.
- Given the non-convexity of the formulated optimization problem, we reformulate it into tractable one using successive convex approximation, and then develop an efficient algorithm to find the sub-optimal approximate solutions to the problem.
- We conduct extensive simulations to evaluate the performance of our proposed collaborative UAV-EC scheme. Numerical experiments demonstrate that our proposed collaborative UAV-EC offloading scheme largely outperforms baseline schemes that solely rely on UAV or ECs for MEC in IoT.

1.2.4 Joint Differential Evolution and Successive Convex Approximation in UAV-enabled Mobile Edge Computing

In Chapter VI, we propose an innovative algorithm by jointly applying DE and SCA to solve for the nonconvex optimization problems raised in UAV-enabled Mobile Edge Computing. In summary, this chapter makes the following contributions:

- We propose an innovative method to jointly consider DE and SCA in view of their own advantages and disadvantages when solely applied to solve for nonconvex optimization problems.
- To better combine with the SCA-based algorithm, we further propose to only select the feasible parts of the best DE solution of the original problem. After this screening procedure, the feasible parts will be used to initialize the SCA-based algorithm.
- We conduct extensive simulations to evaluate the performance of our proposed Screened DE-SCA method in a UAV-enabled MEC system. Numerical experiments demonstrate that our proposed Screened DE-SCA method largely outperforms baseline methods that solely rely on DE or SCA and the DE-SCA method.

1.3 Dissertation Organization

This dissertation is organized as follows. Related works are reviewed in Chapter II. Energy cost optimization for multi-tenant colocation cloud systems is investigated in Chapter III. In Chapter IV, we study resource management for geographically distributed cloud systems. In Chapter V, we study the joint task offloading and resource allocation in a UAV-enabled mobile edge computing system. In Chapter VI, we propose a novel algorithm that combines the DE and SCA to solve for the optimization problem incurred in UAV-enabled mobile edge computing system. Finally, conclusions and future works are given in Chapter VII.

CHAPTER II

RELATED WORKS

In this chapter, we provide an overview of recent research works on resource management in cloud and edge systems aiming to achieve a cost-effective system. On one hand, datacenters can manage energy individually by taking advantage of on-site renewable generations and energy storage systems and optimizing the energy trade in multi-timescale electricity markets. On the other hand, datacenter operator can split incoming workloads among geographically distributed datacenters by utilizing cheap local electricity prices. However, It is challenging to optimize datacenter energy management since workload arrivals, on-site renewable generations and real-time electricity prices are highly uncertain, and thus we resort to either stochastic programming by assuming known probability distributions or robust programming by assuming known bounds of parameters to solve analytically. Besides, it is also challenging to jointly optimize task offloading and resource allocation in cloud and edge systems since they are often inter-dependent, and binary task offloading schemes will result in a mixed integer linear programming (MILP), which is generally NP-hard to solve.

The rest of this chapter is organized as follows. In Section 2.1, we review energy management in cloud and edge systems. Next, task offloading and resource allocation in cloud and edge systems and UAV-enabled MEC using DE methods, SCA methods and alternative methods are reviewed in Section 2.2. Last, a brief review on cooperative game theory is given in Section 2.3.

2.1 Energy Management in Cloud and Edge Systems

Over the past decade, datacenter energy management has received lots of attention and therefore multiple research works have been proposed to reduce the electricity bill of datacenters in cloud and edge systems. Energy management approaches can be categorized into three aspects: demand side energy management, supply side energy management and joint demand and supply side energy management

2.1.1 Demand Side Energy Management

From the demand side, in terms of engineering approaches, energy-efficient servers, storage devices and network switches and advanced cooling have been designed to improve the energy efficiency. On the other hand, in terms of algorithmic approaches, *dynamic capacity provisioning* [41, 65, 137, 66] is proposed to reduce energy cost by dynamically turning off unused servers. *Dynamic speed scaling* [110, 99, 22] is developed to reducing energy consumption by adapting the processing speed to the current load. *Geographical load balancing* [68, 64, 52] is put forward to exploit the spatial diversity of electricity prices to minimize the energy cost of geographically distributed datacenters by dynamically routing the user requests to regional markets with lower electricity prices.

2.1.2 Supply Side Energy Management

From the supply side of datacenters, exploiting the temporal diversity of electricity prices by shifting the delay-tolerant workload to off-peak time periods and using battery to charge from the gird when electricity price is lower and discharge when electricity price is higher is investigated in [106, 43, 122, 44, 45]. On-site renewable energy generated from solar panels and wind turbines can also be utilized to reduce the electricity expenditures [69, 21, 35]. Besides, datacenters can purchase electricity from the *retail electricity market* with a fixed electricity price by signing bilateral contracts beforehand [104, 70]. Participate in the *multi-timescale electricity market* to exploit the uncertainty of electricity prices has been

investigated in [127, 83, 84, 36].

2.1.3 Joint Demand and Supply Energy Management

Joint considering Energy management of datacenters both in demand side and supply side has not been explored in-depth in prior research works. To name a few, [38] investigated the unified energy portfolio optimization of geographically distributed datacenters by considering various energy procurement options such as retailer, day-ahead market and real-time market, ancillary service, local renewable power generators, on-site energy storage systems and geographical load balancing and formulated a mixed-integer linear optimization problem. [139] studied the problem of minimizing the energy cost and bandwidth cost of geographically distributed datacenters when participating in the wholesale electricity markets by jointly considering bidding in day-ahead markets and geographical load balancing scheme and formulated a non-convex infinite-dimensional optimization problem. [63] studied the problem of minimizing the energy cost and delay cost of geographically distributed datacenters when participating in the multi-timescale electricity markets by jointly considering on-site renewable generators and geographical load balancing scheme and formulated a two-stage stochastic optimization problem.

Above papers consider datacenters with the same owner participating in different wholesale electricity markets. In Chapter III, we consider the colocation datacenter where independent tenants colocated together at the same place jointly participate in the wholesale electricity market. Therefore, we need to apply game-theoretic methods to model this multi-agent problem instead of optimization approaches in geographically distributed datacenters. Besides, Different from above works, in Chapter IV, we aim at minimizing the total energy cost plus bandwidth cost of geographically distributed datacenters when participating in the multi-timescale electricity markets while meeting the quality-of-service requirements. In the meanwhile, we consider both delay-sensitive and delay-tolerant workload with thermal energy storage, on-site renewable generation and geographical load balancing scheme and

then formulate a two-stage stochastic optimization problem.

2.2 Task Offloading and Resource Allocation in Cloud and Edge Systems

In cloud and edge Systems, task offloading and resource allocation is a promising approach for resource-hungry mobile users, where intensive computation tasks are migrated from mobile users to nearby edge nodes or remote powerful clouds. In what follows, recent research works on mobile cloud computing, typically two-layer network infrastructure, including one mobile user and multiple mobile users scenarios, and mobile edge computing, typically three-layer network infrastructure, are briefly reviewed.

2.2.1 Mobile Cloud Computing Systems

Most previous work studied the task offloading problems in the mobile cloud computing system, which is a two-tier structure with user layer and cloud layer. Mobile users can offload their computation-intensive tasks to the remote cloud by taking advantage of the resourceful cloud servers to enhance user experiences. Most existing research works can be categorized according to whether they consider single user or multiple users scenarios.

On one hand, [109, 28, 30, 49, 61, 14, 58] considered the task offloading problems for a single mobile user. [109] investigated whether computation task should be processed locally or at a remote cloud by considering two constrained optimization problems in terms of optimizing energy consumption of a mobile user. Closed-form solutions are derived to give the optimal condition for both cases. [30] designed a system called MANI to efficiently partition tasks to optimize energy savings by taking advantage of cloud offloading. [58] proposed Hermes to efficiently solve a NP-hard formulated problem where application with dependent tasks needs to minimize its latency while satisfying given resource utilization requirements. Therefore, these papers do not take into account the case where multiple mobile users can compete for limited communication and computation resources when offload their computation-intensive tasks to the core cloud via base stations or access points.

On the other hand, [82, 13, 56, 98, 42] considered the task offloading problems for multiple mobile users. In this case, we need to decide how to optimally allocate communication (e.g., uplink and downlink bandwidth) and computation (e.g., the clock frequency of the CPU chip) resources to each offloaded task while jointly improving the user-perceived delay and energy consumption for mobile users. [82] proposed MAPCloud, a two-tier cloud structure with local computing aiming to improve both the performance and scalability of mobile applications by considering multiple QoS factors such as energy consumption, delay and cost. [98] designed an online task scheduling algorithms based on a pricing mechanism and Lyaponov optimization to address the energy consumption and computation delay trade-off when mobile users offload in the mobile cloud computing system. [42] addressed the task offloading problem to minimize both energy consumption and application completing time under the scenario of multiple mobile users while taking into account the dependency among computational tasks.

2.2.2 Mobile Edge Computing Systems

Recently, mobile edge computing as a promising computing paradigm, which is a three-tier structure including an additional edge layer with low-resourceful servers deployed in the proximity of mobile users in the traditional mobile cloud computing system, has gained more and more interests from researchers. A few research works concerning task offloading has been investigated in the mobile edge computing. [114] devised a online algorithm to address the task offloading problem under the proposed a two-tiered cloud structure including local cloudlet and remote cloud aiming to reduce the energy consumption for mobile devices while satisfying the user SLA requirements. [24] proposed a SDR-AO-ST algorithm to jointly optimize the offloading decisions of all mobile users and the allocation of communication and computation resources aming to minimize the overall cost of all mobile users under a mobile edge computing system with multiple users, one computing AP and one remote cloud. [115] studied how fog nodes can collaborate with each other to help offload tasks from cloud

datacenters to achieve a better quality-of-experience (QoE) of mobile users in a three-layer fog computing architecture.

A few recent papers adopted a game-theoretic approach to analyze the task offloading problems. [26] formulated a decentralized task offloading game under the mobile cloud computing system with one shared AP, and designed an efficient decentralized algorithm to locate Nash equilibrium. [55] formulated a multi access point task offloading game under the mobile cloud computing system with multiple APs and a non-elastic cloud, and designed JPBR algorithm to find Nash equilibrium in this player-specific congestion game. [19] formulated a generalized Nash equilibrium problem under the mobile edge computing system with one shared AP. It utilized queue theory to model the task arrivals and queue response time and considered partial offloading in all three layers including local layer, edge layer and cloud layer.

Resource Management in MEC

Besides, There is a rich literature on resource management in MEC that aims at optimizing system latency [115, 118, 25, 87], energy consumption [90, 136, 124] and overall cost of system latency and/or energy consumption [27, 24, 23, 134]. The trade-off problem is studied in [115] for computing networks with fog node cooperation aiming at minimizing the response time of fog nodes under a given power efficiency constraint. [118] studied the joint service caching and task offloading problem in dense network aiming at minimizing computation latency while keeping the total computation energy consumption low. [25] investigated the MEC task offloading problem in software defined ultra-dense network aiming at minimizing the total task duration under energy budget constraints. [87] investigated a joint communication and computation resource allocation problem under the collaboration of cloud and edge computing for minimizing the system delay of all mobile devices. [90] formulated multi-cell MEC task offloading problem as a joint optimization of radio and computation resources aiming at minimizing the overall users' energy consumption, while meeting latency

constraints. [136] proposed an energy-efficient offloading scheme for MEC in 5G heterogeneous networks by formulating the optimization problem with the objective of minimizing the total system energy consumption. [124] studied the resource allocation problem for a multi-user mobile-edge computing offloading system based on TDMA and OFDMA with the objective to minimize the weighted sum of mobile energy consumption. [27] formulated a multi-user computation offloading game to study the energy-delay trade-off problem in a mobile-edge cloud computing architecture. [24, 23] jointly optimized the offloading decisions of all users and computing access point and resource allocation aiming at minimizing the overall energy cost and the maximum delay among all users. [134] proposed a distributed joint computation offloading and resource allocation optimization scheme in heterogeneous networks with MEC to minimize the overhead of local energy consumption and execution time cost.

UAV-Enabled MEC Networks

Extensive research efforts have been made from the academia to employ UAVs as different kinds of wireless communication platforms [112]. For instance, UAVs equipped with base stations can be flexibly deployed at specific areas to provide reliable uplink and downlink communication for ground users. They can also serve as the mobile relaying nodes to connect two or more distant users [131, 113]. Moreover, UAVs can assist with information dissemination or data collection by flying over the specific areas [74, 133]. However, prior works in the area of the UAV-enabled wireless networks ignore the computing capability provided by UAVs and mainly focus on their communication aspect, and only a very few recent studies [47, 54, 141, 10, 48] start to consider computing with UAVs' on-board resources. [47] investigated joint offloading and trajectory design for a MEC system where a UAV endowed with computing capability is deployed to serve the task offloading of mobile users, aiming at minimizing the sum of the maximum delay among all the users in each time slot. [54] studied the joint optimization of path planning and bit allocation for a MEC system where

a UAV-mounted cloudlet is deployed to provide offloading opportunities to mobile users, aiming at minimizing the mobile energy consumption while satisfying the quality-of-service requirements of offloaded applications. [141] formulated the computation rate maximization problem under both partial and binary task offloading schemes in a UAV-enabled MEC wireless-powered system where the UAV can simultaneously transmit energy and perform computation. However, these works only consider communication and computation interactions between two types of entities where ground mobile users offload the tasks to UAV for computation. Besides, [10] presented a game-theoretic and reinforcement learning framework to study the computation offloading problem in UAV-enabled MEC networks with multiple service providers where UAV-based privately-owned base stations are interacting with terrestrial privately-owned and operator-controlled base stations. [48] considered a UAV-aided MEC system where the cellular-connected UAV is served as a mobile computing server as well as a relay to help the user equipments complete their computing tasks or further offload their tasks to the AP for computing.

2.2.3 DE methods in UAV-Enabled MEC

Benefiting from its ability to search large space of candidate solutions imposed on the original problems, the DE methods have been utilized to explore the best solutions of the optimization problems formulated in UAV-enabled MEC systems.

On one hand, DE method can be used to generate intermediate results in joint algorithms in [73, 11, 34, 50]. In [73], a discrete differential evolution (DDE) algorithm along with ant colony optimization (ACO) algorithm are put forward to jointly optimize the clustering of IoT devices and UAV trajectory in a UAV-enabled MEC system. In [11], an evolutionary trajectory planning algorithm (ETPA) which adopts a DE clustering method is proposed to jointly optimize the overall system energy consumption as well as the path planning of UAVs in a multi-UAV-assisted MEC system. In [34], a multi-objective evolutionary algorithm which adopts DE to update solutions along with the deep deterministic gradient algorithm

is designed to maximize the sum computation rate at all IoT devices while satisfying the energy harvesting constraints and coverage in UAV-aided wireless powered MEC networks. In [50], a trajectory planning algorithm (TPA) that adopts a DE algorithm with variable population sizes is put forward to optimize the system energy consumption via planning the UAV trajectories in a multi-UAV-associated MEC system.

On the other hand, DE method can be directly applied to generate the best solutions in [117, 121, 51, 107]. In [117], a novel DE algorithm with variable population size based on a mutation strategy pool initialized by K-Means is developed to minimize the energy consumption in a UAV-assisted edge data collection system. In [121], a DE-based mechanism is designed to jointly optimize the load-balancing and latency-aware task scheduling incurred in a multi-UAV-enabled MEC system. [51], a differential evolution algorithm with a variable population size (DEVIPS) is developed to minimize the system energy consumption by optimizing the UAV deployment in a UAV-assisted IoT data collection system. In [107], a DE algorithm with an elimination operator is proposed to jointly optimize the deployment of UAVs and task scheduling for all mobile users in a multi-UAV-enabled MEC system.

2.2.4 SCA-based methods in UAV-Enabled MEC

Due to the ability to approximately transform the nonconvex optimization problem formulated in UAV-enabled MEC systems into a solvable convex form, the SCA-based methods have been widely exploited in the literature.

On one hand, SCA technique can be combined with other algorithms to find solutions in [81, 119, 71, 97, 138]. In [81], the Branch and Bound (BnB) method and SCA technique are exploited to jointly optimize the service placement, task scheduling and UAV trajectory subproblems in a UAV-enabled MEC system. In [119], the Dinkelbach's method, Lagrange duality and SCA technique are combined to jointly maximize the weighted computation efficiency subject to the constraints on resource allocation, minimum computation and UAV's mobility in the UAV-assisted MEC networks. In [71], the SCA and block coordinate descent

(BCD) algorithms are utilized to maximize the minimum secure calculation capacity in order to improve the security of communications in dual UAV MEC systems. In [97], a Dinkelbach method adopting simulated annealing and SCA is proposed to jointly optimize the gateway selection and resource allocation involved with space-air-ground IoT networks. In [138], SCA technique and Lagrangian duality method are jointly applied to minimize the total energy consumption by optimizing the computation bits allocation, time slot scheduling, transmit power allocation, and UAV trajectory in a UAV-assisted MEC system.

On the other hand, SCA technique can be directly used to develop SCA-based algorithms in [132, 108, 54]. In [132], SCA-based algorithms are designed to jointly optimize the completion time and energy consumption of UAV as well as its trajectory in the UAV-enabled MEC system. In [108], a SCA-based algorithm is developed to jointly optimize the UAV trajectory subject to the energy harvesting causality and user scheduling constraints in UAV-enabled wireless powered communication networks. In [54], a SCA-based algorithm is presented to jointly optimize the bit allocation for communication and computation as well as the UAV trajectory in the MEC system via a UAV-mounted cloudlet.

2.2.5 Alternative methods in UAV-Enabled MEC

Apart from DE methods and SCA-based methods, there are ample studies using alternative methods to solve related problems formulated in UAV-enabled MEC.

Optimization Methods

Main problems that are studied in UAV-enabled MEC include task scheduling, computation offloading, user association, resource allocation and trajectory planning. There are many optimization methods to tackle these problems. To name a few, in [80], the UMEC method combing the proposed RTSA and submodularity is designed to effectively solve a mixed-integer nonlinear programming problem that is formulated to model the task selection and scheduling for reconnaissance with time-varying priorities conditions in UAV-enabled MEC.

In [116], an algorithm combining the alternative optimization and successive convex programming is developed to solve a nonconvex problem aiming to maximize the uplink common throughput among all ground users over a finite UAV's flight period in UAV-enabled wireless powered communication network. In [100], a learning-based cooperative particle swarm optimization algorithm with a Markov random field-based decomposition strategy is put forward to search for the optimal UAV resource allocation strategy for industrial IoT in UAV-enabled MEC.

Game-theoretic Methods

There are many papers aiming to investigate games that are formulated in UAV-enabled MEC by viewing the mobile users, base stations and edge/cloud infrastructures as autonomous agents that each can have their own utility functions. To name a few, in [135], a selfish game is formulated to model the task offloading and resource competition problem in UAV-assisted multiaccess edge computing system. The Nash equilibrium is proved to be existent and a game-theoretic scheme is proposed to find the optimal solution. In [10], a two-level game model including cooperative game in the upper level and noncooperative subgames in the lower level is formulated to model the payoff maximization problem raised in UAV-enabled MEC with multiple service providers. The mixed-strategy Nash equilibrium is found by combining coalition formation with reinforcement learning. In [120], a dynamic evolutionary game is formulated to study the access competition among groups of UAVs and it is solved by an evolutionary equilibrium. Also, a noncooperative game is formulated to study the bandwidth that is allocated by base stations to the UAVs and it is solved by finding the uniqueness of Nash equilibrium.

2.3 A Brief Review on Cooperative Game Theory

Cooperative game theory [76, 88, 79, 33] is widely applied in many communication and computation optimization problems to analyze the joint actions of multi-agent systems (i.e.,

network nodes) in terms of obtaining collective payoffs by predicting which coalitions will formulate. In this section, we will briefly introduce the fundamental concepts of cooperative game theory including the definition for a cooperative game with transferable utility, the solution concept (i.e., the core) of a cooperative game, two types of cooperative games with nonempty core (i.e., the convex games and balanced games), and widely-used cost allocation methods (i.e., the Shapley value and nucleolus), as they will be utilized in Chapter IV.

2.3.1 Cooperative Game with Transferable Utility

In general, a cooperative game is defined by a pair $(\mathcal{N}, c)$. The first element is the set of players $\mathcal{N} \coloneqq \{1, 2, \ldots, N\}$, indexed by $i \in \mathcal{N}$. Players may form different coalitions $S \subseteq \mathcal{N}$ to obtain a collective utility. The grand coalition $\mathcal{N}$ is the set of all players. Secondly, $c : 2^{\mathcal{N}} \to \mathbb{R}$ is the cost function that assigns a real cost (i.e., the negative of the utility) to each coalition $S \subseteq \mathcal{N}$. Transferable cost implies that the total cost represented by a real number can be divided in any manner among the coalitional members [88].

2.3.2 Imputations and the Core

The cost function of a cooperative game is said to be subadditive if it satisfies the following condition:

$$c(\mathcal{S}) + c(\mathcal{T}) \geq c(\mathcal{S} \cup \mathcal{T}),\ \forall \mathcal{S}, \mathcal{T} \subseteq \mathcal{N},\ S \cap T = \emptyset. \tag{2.3.1}$$

For such cooperative game, it is to the mutual benefit of the players to form the grand coalition $\mathcal{N}$, since by subadditivity the amount received, $c(\mathcal{N})$, is at least as small as the total amount received by any disjoint set of coalitions they could form. Next, we focus on how to fairly split this amount among participating players.

A cost allocation for the coalition $S \subseteq \mathcal{N}$ is a vector $\pi \in \mathbb{R}^N$ whose entry π_i is the cost dispatched to each player i in the coalition S ($\pi_i = 0, i \notin S$). Further, a cost allocation π is said to be *efficient* if $\sum_{i \in \mathcal{N}} \pi_i = c(\mathcal{N})$, i.e., the total amount received by the players should be equal to $c(\mathcal{N})$. A cost allocation π is said to be *individually rational* if $\pi_i \leq c(\{i\})$, i.e.,

no player will be expected to receive more cost than acting individually. A cost allocation π for the grand coalition is said to be an *imputation* if it is both efficient and individually rational. In cooperative game theory [125, 102], the set of imputations for the game $(\mathcal{N}, c)$ is defined as

$$\mathcal{I} = \left\{ \pi \in \mathbb{R}^N : \sum_{i \in \mathcal{N}} \pi_i = c(\mathcal{N}),\ \pi_i \leq c(\{i\}),\ \forall i \in \mathcal{N} \right\}. \tag{2.3.2}$$

Next, we introduce the solution concept of a cooperative game. The *core* for the game $(\mathcal{N}, c)$ is defined as

$$\mathcal{C} = \left\{ \pi \in \mathbb{R}^N : \sum_{i \in \mathcal{N}} \pi_i = c(\mathcal{N}),\ \sum_{i \in S} \pi_i \leq c(S),\ \forall S \subseteq \mathcal{N} \right\}. \tag{2.3.3}$$

The core is a set of imputations such that no coalitions can obtain a cost which is less than the sum of cost assigned by forming the grand coalition. Obviously, if one can locate a cost allocation vector that lies in the core, then the grand coalition is optimal for the cooperative game.

2.3.3 Convex and Balanced Games

The core is always well-defined, but can be empty. However, the convex games and balanced games are two types of cooperative games which guarantee the existence of *nonempty* core [94, 93]. A cooperative game is said to be convex if the cost function satisfies the following condition:

$$c(\mathcal{S}) + c(\mathcal{T}) \geq c(\mathcal{S} \cup \mathcal{T}) + c(\mathcal{S} \cap \mathcal{T}),\ \forall \mathcal{S}, \mathcal{T} \subseteq \mathcal{N}. \tag{2.3.4}$$

This implies the cooperative game has a submodular cost function.

A map $\rho : 2^{\mathcal{N}} \to [0, 1]$ is said to be balanced if for all $i \in \mathcal{N}$,

$$\sum_{S \in 2^{\mathcal{N}}} \rho(S) \mathbf{1}\{i \in S\} = 1, \tag{2.3.5}$$

where $\mathbf{1}\{\cdot\}$ denotes the indicator function. Thus, the balanced map indicates that the sum of weights $\rho(S)$ assigned for each coalition including player i will be equal to 1. Then a

cooperative game is said to be balanced if and only if for any balanced map ρ,

$$\sum_{S \in 2^{\mathcal{N}}} \rho(S) c(S) \geq c(\mathcal{N}). \tag{2.3.6}$$

2.3.4 Shapley Value

The Shapley value [76] as the cost allocation method is a unique mapping ψ that satisfies a series of characteristic axioms such as efficiency, symmetry, dummy and additivity. For a cooperative game $(\mathcal{N}, c)$ with transferable cost, the Shapley value $\psi_i(c)$ that distributes the cost for each player $i \in \mathcal{N}$ is defined as

$$\psi_i(c) = \sum_{S \subseteq \mathcal{N} \setminus \{i\}} \frac{|S|!(N - |S| - 1)!}{N!} \left[c(S \cup \{i\}) - c(S) \right]. \tag{2.3.7}$$

We observe that in (2.3.7), the marginal contribution of each player is represented as $c(S \cup \{i\}) - c(S)$ and the coefficient ahead of the marginal distribution is the probability that the player i randomly joins the coalition S. Thus, the Shapley value can be interpreted as the expected marginal contribution of player i in the grand coalition $\mathcal{N}$ when it joins the coalition S in a random order. It is guaranteed that the Shapley value lies in the core if the game is convex [94].

2.3.5 Nucleolus

The nucleolus [79] is another common cost allocation method. It uniquely exists in a cooperative game and satisfies the efficiency, individually rational, symmetry and dummy properties [88]. Different from axiomatically designing the cost allocation scheme to ensure fairness as in the Shapley value, the nucleolus aims at minimizing the dissatisfaction of the players. The dissatisfaction of a coalition S given an imputation π is measured by the excess. The

definition of excess is given by

$$e(\pi, S) = \sum_{i \in S} \pi_i - c(S). \tag{2.3.8}$$

Since the core is defined as the set of imputations such that $\sum_{i \in S} \pi_i \leq c(S)$ for all coalitions $S \subseteq \mathcal{N}$, it follows that an imputation π is in the core if and only if all its excesses are negative or zero [33]. In order to find the nucleolus, we first need to locate an imputation that minimizes the maximum of the excesses $e(\pi, S)$ over all coalitions S by solving a linear program. After this is done, one may have to solve a second linear programming problem to minimize the next largest excess, and so on. Therefore, in the worst-case, $\mathcal{O}(2^N)$ linear programs need to be solved, which is computationally expensive [89].

CHAPTER III

ENERGY MANAGEMENT IN COLOCATION DATACENTERS

In this chapter, we study how colocation datacenter can effectively reduce its energy cost when participating in the wholesale electricity market via cooperative power procurement. Intuitively, by aggregating workloads across a group of tenants, the overall power demand uncertainty of tenants can be reduced, resulting in less chance of being penalized when participating in the wholesale electricity market. We use cooperative game theory to model the cooperative electricity procurement process of tenants as a cooperative game, and show the cost saving benefits of aggregation. Then, a cost allocation scheme based on the marginal contribution of each tenant to the total expected cost is proposed to fairly distribute the aggregation benefits among the participating tenants. Finally, numerical experiments based on real-world traces are conducted to illustrate the benefits of aggregation compared to noncooperative power procurement.

The rest of this chapter is organized as follows. In Section 3.1, we describe the models for datacenter power consumption and two-settlement electricity market. In Section 3.2, we model the tenant aggregation process as a cooperative game and quantify the benefits of aggregation. Then, the core of the formulated game is shown to be nonempty, and an efficient scheme is proposed to find a cost allocation belonging to the core in Section 3.3. Simulation results based on real-world traces are presented in Section 3.4. Finally, the summary is given in Section 3.5.

3.1 System Model

In this section, we start by introducing the datacenter power consumption model and characterize the uncertainty of power demand for each datacenter. Then, the two-settlement electricity market is described and the expected electricity cost for each tenant when participating in the market individually is derived.

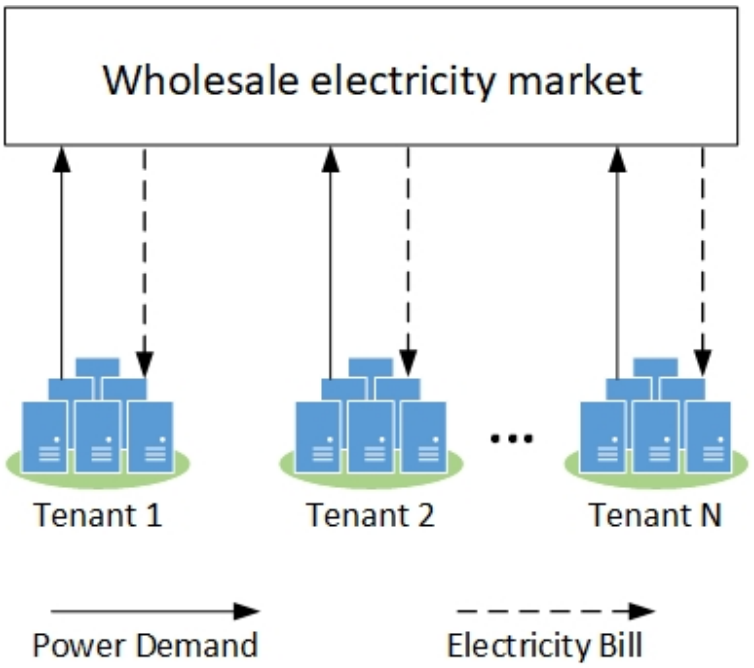
Consider a set $\mathcal{N} \coloneqq \{1, 2, \ldots, N\}$ of independent tenants in a wholesale colocation datacenter where each tenant pays for their own energy consumption. As shown in Fig. III.4(a), each tenant can bid its power demand in the wholesale electricity market, and then pay its electricity bill individually. As shown in Fig. III.4(b), we explore the scenario in which tenants form a coalition under the colocation datacenter operator to collectively bid their aggregated power demand in the wholesale electricity market as a single entity for cost saving. Without loss of generality, in the following of the chapter we restrict our analysis to a specific operating hour.

3.1.1 Datacenter Power Consumption Model

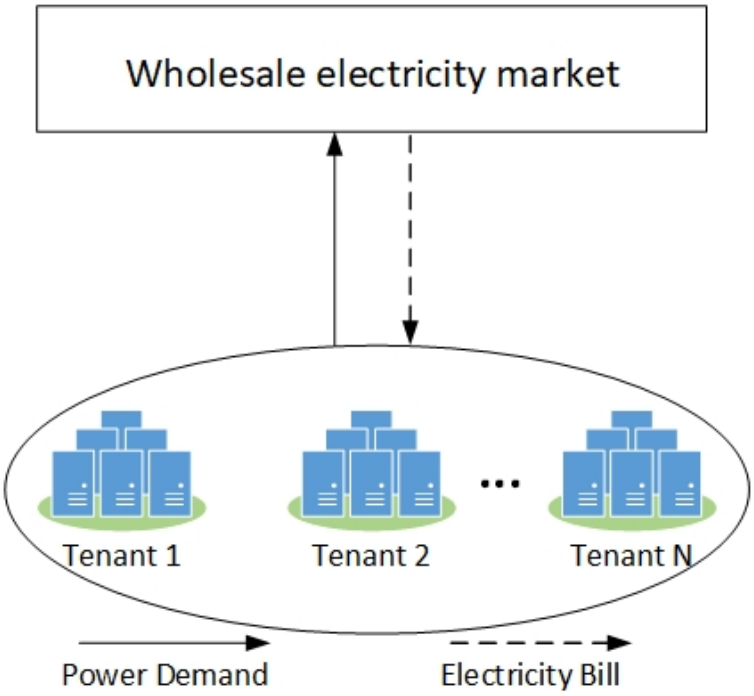
Assume each tenant in the colocation datacenter $i \in \mathcal{N}$ has M_i homogeneous servers whose idle and peak power consumption are P_i^{idle} and P_i^{peak}, respectively. Note that a tenant with heterogeneous servers can be also viewed as several tenants, each having homogeneous servers. Therefore, we focus on the homogeneous case in this work. Users submit their requests (e.g., search queries) to tenants, and tenants process these requests to satisfy the quality-of-service (QoS) requirement as indicated by the service-level agreement (SLA). When tenant i keeps m_i active servers to process the arriving user requests, its IT power consumption can be estimated as [67]

$$P_i = m_i \left[P_i^{\text{idle}} + u_i (P_i^{\text{peak}} - P_i^{\text{idle}}) \right], \tag{3.1.1}$$

where u_i is the average CPU utilization level across all servers at tenant i.



(a) Individual Bidding



(b) Cooperative Bidding

Figure 4: Individual bidding and cooperative bidding in the wholesale electricity market.

We adopt a M/GI/1 Processor Sharing (PS) queue to model the service process at each server [69]. The workload arrival rate at each tenant i, measured in terms of the average

number of arriving user requests per unit time, is assumed to be λ_i. Let μ_i denote the service rate at which user requests are processed by a server at tenant i. Then the average CPU utilization level in tenant i is calculated as $u_i = \lambda_i/(m_i\mu_i)$. Therefore, the power consumption model (3.1.1) can be rewritten as

$$P_i = m_i P_i^{\text{idle}} + \frac{\lambda_i}{\mu_i}\left(P_i^{\text{peak}} - P_i^{\text{idle}}\right). \tag{3.1.2}$$

Since each user request has a QoS requirement, tenants need to turn on enough servers to meet that requirement. Here we use the average response time as the QoS metric. Based on the M/GI/1/PS queuing model, the average response time of user requests given m_i active servers in tenant i is represented as

$$T_i = \frac{1}{\mu_i - \lambda_i/m_i}. \tag{3.1.3}$$

Let $T_i^{\max}$ denote the maximum average response time of user requests that can be tolerated at tenant i. Then to ensure that $T_i \leq T_i^{\max}$, we obtain the following feasible range for the number of active servers at tenant i:

$$\frac{\lambda_i}{\mu_i - 1/T_i^{\max}} \leq m_i \leq M_i. \tag{3.1.4}$$

Here, we relax the constraint that requires m_i to be integer given the fact that tenants usually contain thousands of servers. It is assumed that each tenants turn on the minimal number of active servers without violating their QoS requirement using the dynamic capacity provisioning technique [66, 111]. Therefore the IT power consumption of each tenant i is

$$P_i = \frac{\lambda_i}{\mu_i - 1/T_i^{\max}} P_i^{\text{idle}} + \frac{\lambda_i}{\mu_i}\left(P_i^{\text{peak}} - P_i^{\text{idle}}\right). \tag{3.1.5}$$

In order to incorporate the non-IT (e.g. cooling, lighting) power consumption of tenants, we denote the average power usage effectiveness (PUE) as γ_i, which is defined as the ratio

of the total power consumption to the IT power consumption at tenant i. It follows that the total power demand E_i of tenant i is given by

$$E_i = \theta_i \lambda_i, \tag{3.1.6}$$

where θ_i is a constant defined as

$$\theta_i \coloneqq \gamma_i \left(\frac{P_i^{\text{idle}}}{\mu_i - 1/T_i^{\max}} + \frac{P_i^{\text{peak}} - P_i^{\text{idle}}}{\mu_i} \right). \tag{3.1.7}$$

When tenant i bids in the day-ahead market one day ahead, the user request arrivals for the next day are uncertain, and thus the average workload arrival rate λ_i can be modeled as a random variable whose probability distribution can be empirically estimated from historical data. It follows that the actual tenant power demand $E_i(\lambda_i)$ as a linear function of the average workload arrival rate λ_i is also a random variable

3.1.2 Two-Settlement Electricity Market

Consider a wholesale electricity market managed by an ISO with a two-settlement structure in the region through which the tenants procure power. It consists of a day-ahead forward market and a real-time balancing market. In the day-ahead forward market, participants bid and schedule power transactions for each hour of the following day before the gate closure. After that, the ISO clears the market and calculates the day-ahead market clearing price for each hour as the intersection between the aggregate supply and demand curves. For instance, for California ISO, the day-ahead forward market closes for bids and schedules by 10 AM and clears by 1 PM on the day prior to the operating day. The schedules cleared in the day-ahead market are financially binding. Any deviations between the day-ahead committed schedule and actual power consumption/generation will be settled in the real-time balancing market during the operating day. If the actual consumption is more than or production is less than the committed schedule, the energy shortfall will be purchased in the balancing

market at the negative imbalance price, which is usually higher than the day-ahead price. If the actual consumption is less than or production is more than the committed schedule, the energy surplus will be sold at the positive imbalance price, which is usually lower than the day-ahead price. Therefore, power deviations from day-ahead commitments normally result in penalties for participants.

Specifically, for the considered wholesale market, let $p^d \in \mathbb{R}^+$ be the market clearing price in the day-ahead forward market, $p^- \in \mathbb{R}^+$ be the negative imbalance price for energy shortfall, and $p^+ \in \mathbb{R}^+$ be the positive imbalance price for energy surplus. The tenants are assumed to be price-taking because their energy consumption are often too small to influence the market. The market prices (p^d, p^-, p^+) are not known to the tenants at the time of bidding in the day-ahead market and therefore modeled as random variables with known expected values denoted by μ_p^d, μ_p^-, and μ_p^+, respectively, which can be estimated empirically from historical market data. As explained before, without loss of generality, we assume $\mu_p^+ \leq \mu_p^d \leq \mu_p^-$. Moreover, the market prices (p^d, p^-, p^+) are assumed to be statistically independent of the workload arrival rates $(\lambda_i, \forall i)$.

Suppose that each tenant $i \in \mathcal{N}$ bids a power procurement amount Q_i in the day-ahead market. With the above models and assumptions, it follows that the expected cost of tenant i from participating in the market individually can be calculated as

$$\Phi_i = \mu_p^d Q_i + \mu_p^- \mathbb{E}[(E_i - Q_i)^+] - \mu_p^+ \mathbb{E}[(Q_i - E_i)^+], \tag{3.1.8}$$

where $(x)^+ \coloneqq \max(x, 0)$, $\mu_p^d Q_i$ denotes the day-ahead trading cost, $\mu_p^- \mathbb{E}[(E_i - Q_i)^+]$ denotes the shortfall penalty, and $\mu_p^+ \mathbb{E}[(Q_i - E_i)^+]$ denotes the surplus profit.

3.2 Coalitional Tenant Bidding

In this section, we start by introducing the tenant aggregation model where multiple tenants can form a coalition to bid in the day-ahead market collectively. Then, it can be verified that

by bidding power demand aggregately in the day-ahead market, the total electricity bill can be effectively reduced based on the fact that tenant aggregation can reduce the uncertainty of the total workload arrivals.

3.2.1 Tenant Aggregation as a Cooperative Game

Tenants can form a coalition and bid collectively in the day-ahead market. Any coalition $S \subseteq \mathcal{N}$ represents an agreement among the tenants in S to act as a single entity in the market. The aggregated tenant power demand of a coalition $S \subseteq \mathcal{N}$ is specified by

$$E_S = \sum_{i \in S} E_i. \tag{3.2.1}$$

Further, we denote the cumulative distribution function (CDF) of E_S as

$$F_S(e) = \Pr(E_S \leq e). \tag{3.2.2}$$

The corresponding quantile function is given by

$$F_S^{-1}(\varepsilon) = \inf \{e \in [E_S^{\min}, E_S^{\max}] : \varepsilon \leq F_S(e)\}, \tag{3.2.3}$$

where $E_S^{\min}$ and $E_S^{\max}$ are the lower and upper bounds of the aggregated power demand, which depends on the minimum and maximum workload arrival rates.

Next, we use cooperative game theory [78] to model this cooperation process as a cooperative game $(\mathcal{N}, c)$ with transferable cost since it is under a multi-agent scenario where each tenant tends to minimize its own net cost. In our model, the set of tenants $\mathcal{N}$ is the set of players in the cooperative game. Moreover, we assume each tenant always seeks to minimize its own electricity cost, and then the cost function $c(S)$ associated with every coalition

$S \subseteq \mathcal{N}$ is represented as its minimum expected energy cost calculated as

$$\Phi_S = \mu_p^d Q_S + \mu_p^- \mathbb{E}[(E_S - Q_S)^+] - \mu_p^+ \mathbb{E}[(Q_S - E_S)^+], \tag{3.2.4}$$

$$c(S) = \min_{Q_S \geq 0} \Phi_S, \tag{3.2.5}$$

where Q_S is the bid amount of any coalition S in the day-ahead market, and it is a continuous variable. We assume the market prices for the coalitional bid is the same as that of individual bids. This assumption is acceptable since the tenants are assumed to be relatively small [59] compared to all other consumers participating in the electricity market so that their operations have little impact on the cleared prices of the day-head market or real-time market. Solving (3.2.5) as a news-vendor problem [16, 12], the optimal day-ahead bid and expected cost are given in the following theorem:

Theorem 3.2.1 *The optimal day-ahead bid of any coalition S is given by*

$$Q_S^* = F_S^{-1}(\varepsilon^*), \quad \text{where} \quad \varepsilon^* = \frac{\mu_p^- - \mu_p^d}{\mu_p^- - \mu_p^+}. \tag{3.2.6}$$

The optimal expected cost is given by

$$c(S) = \mu_p^+ \int_0^{\varepsilon^*} F_S^{-1}(\theta)\, \mathrm{d}\theta + \mu_p^- \int_{\varepsilon^*}^1 F_S^{-1}(\theta)\, \mathrm{d}\theta. \tag{3.2.7}$$

Proof. The proof is referred to Appendix. ■

3.2.2 The Benefits of Aggregation

Intuitively, no group of tenants can do worse by joining a coalition than by acting noncooperatively since aggregation can reduce uncertainty. We will prove this by the following theorem:

Theorem 3.2.2 *Given an arbitrary coalition $S \subseteq \mathcal{N}$, let $\{Q_1, Q_2, \ldots, Q_{|S|}\}$ be a set of $|S|$*

individual day-ahead bids. For $Q_S = \sum_{i \in S} Q_i$ *we have:*

$$\Phi_S(Q_S) \leq \sum_{i \in S} \Phi_i(Q_i). \tag{3.2.8}$$

Proof. The proof is referred to Appendix. ■

It is straightforward to see that the expected cost by participating in the market collectively is less than the sum of that by participating in the market individually. That is, the tenants save the expected cost of $\sum_{i \in S} \Phi_i(Q_i) - \Phi_S(Q_S)$ collectively via aggregation. Further, we establish some properties of the cost function associated with every coalition.

Lemma 3.2.1 *The optimal expected cost* $c(S)$ *of any coalition* S *has following properties:*

1. *Positive homogeneity: For any scalar* $\beta \geq 0$, $c(\beta S) = \beta c(S)$.

2. *Subadditivity: For any two disjoint coalitions* S_1 *and* S_2, *if coalition* $S_1 \cup S_2$ *forms, then* $c(S_1 \cup S_2) \leq c(S_1) + c(S_2)$.

Proof. The proof is referred to Appendix. ■

From positive homogeneity, we observe that when the aggregated power demand is scaled, the corresponding value of the optimal expected cost will also be scaled in the same proportion. From subadditivity, we observe that for rational tenants who always try to minimize their cost, they will form a large-size coalition to benefit more from the aggregation. It is straightforward to see in our game that all the tenants will form the grand coalition $\mathcal{N}$ in order to minimize their total expected cost.

3.3 Cost Allocation Mechanism

In the section, we focus on how to find a cost allocation vector π as defined in Section 2.3.2 to split the total expected cost to each tenant in the grand coalition. First, we show that the core of our cooperative game exists and is nonempty by proving it is a balanced game.

Next, we verify that our game is nonconvex, and hence the Shapley value is not applicable to locate the core of our game. Last, we propose a cost allocation scheme based on the marginal contribution of each tenant to the total cost in the grand coalition.

3.3.1 Existence of the Nonempty Core

As shown in Section 2.3, both the convexity and balancedness can guarantee the core of a cooperative game to be nonempty. Since the convexity of a cooperative game is a stronger condition compared to the balancedness, we prove the existence of the core in terms of balancedness by the following theorem:

Theorem 3.3.1 *The cooperative game $(\mathcal{N}, c)$ for tenant aggregation is balanced and has a nonempty core.*

Proof. Given an arbitrary balanced map $\rho : 2^{\mathcal{N}} \to [0, 1]$, by following the concept of the balanced game, we have

$$\begin{aligned}
\sum_{S\in 2^{\mathcal{N}}} \rho(S)c(S) &= \sum_{S\in 2^{\mathcal{N}}} c(\rho(S)S) && (3.3.1)\\
&\geq c\left(\sum_{S\in 2^{\mathcal{N}}} \rho(S)S\right) && (3.3.2)\\
&= c\left(\sum_{S\in 2^{\mathcal{N}}} \rho(S)\left(\bigcup_{i\in\mathcal{N}} \mathbf{1}\{i \in S\}i\right)\right)\\
&= c\left(\bigcup_{i\in\mathcal{N}}\left(\sum_{S\in 2^{\mathcal{N}}} \rho(S)\mathbf{1}\{i \in S\}\right)i\right) && (3.3.3)\\
&= c\left(\bigcup_{i\in\mathcal{N}} i\right) = c(\mathcal{N}),
\end{aligned}$$

where (3.3.1) is because of the positive homogeneity of $c(S)$, (3.3.2) is because of the subadditivity of $c(S)$, and (3.3.3) is derived by the definition of balanced map ρ. Therefore, the cooperative game $(\mathcal{N}, c)$ is balanced and has a nonempty core. ■

3.3.2 Marginal Cost Allocation

Two prominent cost allocation schemes are described in Section 2.3. However, both of them are not applicable to solve our cooperative game. The Shapley value can be guaranteed to lie in the core if the cooperative game is convex. However, as shown through a counterexample in Appendix, our game is not convex. Therefore, the Shapley value does not necessarily belong to the core and hence is not applicable to allocate cost in our game. The nucleolus uniquely exists and can be used as a cost allocation scheme in our game. However, as mentioned before, in the worst-case scenario, $\mathcal{O}(2^N)$ linear programs need to be solved in order to get the cost allocation vector, which is computationally expensive.

Here, we propose a cost allocation scheme based on the marginal contribution of each tenant to the total expected cost when participating in the grand coalition and prove the resulting cost allocation vector is in the core. We define an aggregation level vector $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_N]^T$, where each element $0 \leq \alpha_i \leq 1$ represents the fraction of tenant power demand E_i that participates in the aggregative power procurement. Thus, the weighted power demand of the aggregation with the aggregation level vector $\boldsymbol{\alpha}$ is denoted as

$$E_{\boldsymbol{\alpha},\mathcal{N}} = \sum_{i=1}^{N} \alpha_i E_i, \tag{3.3.4}$$

whose quantile function is represented by $F^{-1}_{\boldsymbol{\alpha},\mathcal{N}}(\varepsilon)$ and defined similar to (3.2.3). Then by applying Theorem 3.2.1, we can obtain the optimal expected cost of the weighted power demand as

$$c_{\boldsymbol{\alpha}}(\mathcal{N}) = \mu_p^+ \int_0^{\varepsilon^*} F^{-1}_{\boldsymbol{\alpha},\mathcal{N}}(\theta)\,\mathrm{d}\theta + \mu_p^- \int_{\varepsilon^*}^{1} F^{-1}_{\boldsymbol{\alpha},\mathcal{N}}(\theta)\,\mathrm{d}\theta. \tag{3.3.5}$$

The positive homogeneity and subadditivity proved in Lemma 3.2.1 can be easily extended to the case where we consider the weighted optimal expected cost $c_{\boldsymbol{\alpha}}(\mathcal{N})$. Further, we show another property as follows:

Lemma 3.3.1 *The weighted optimal expected cost $c_{\boldsymbol{\alpha}}(\mathcal{N})$ of any coalition S is nonincreasing*

over $\boldsymbol{\alpha}$*, i.e., for any two aggregation level vectors, if* $\boldsymbol{\alpha} \succeq \boldsymbol{\alpha}'$[1]*, then* $c_{\boldsymbol{\alpha}}(\mathcal{N}) \leq c_{\boldsymbol{\alpha}'}(\mathcal{N})$.

Proof. Given two aggregation level vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}'$ where $\boldsymbol{\alpha} \succeq \boldsymbol{\alpha}'$, then for any element in the vector $\boldsymbol{\alpha} - \boldsymbol{\alpha}'$, we have $0 \leq \alpha_i - \alpha_i' \leq 1, \forall i \in \mathcal{N}$. Using the subadditivity property, we have

$$c_{\boldsymbol{\alpha}}(\mathcal{N}) \leq c_{\boldsymbol{\alpha}'}(\mathcal{N}) + c_{\boldsymbol{\alpha}-\boldsymbol{\alpha}'}(\mathcal{N}), \tag{3.3.6}$$

which indicates the nonincreasing property. ■

According to Lemma 3.3.1, the optimal expected cost will be achieved when $\boldsymbol{\alpha} = \mathbf{1}$, where $\mathbf{1} \in \mathbb{R}^{N\times 1}$ is an all-one vector. Then it follows that $c_{\boldsymbol{\alpha}}(\mathcal{N})|_{\boldsymbol{\alpha}=\mathbf{1}} = c(\mathcal{N})$.

To distribute the total expected cost $c(\mathcal{N})$ among the tenants in the grand coalition, we compute the expected cost for each tenant i as

$$\pi_i = \frac{\partial c_{\boldsymbol{\alpha}}(\mathcal{N})}{\partial \alpha_i}\Big|_{\boldsymbol{\alpha}=\mathbf{1}}, \ \forall i \in \mathcal{N}. \tag{3.3.7}$$

Indeed, π_i can be decomposed as the multiplication of two terms:

$$\pi_i = \frac{\partial c_{\boldsymbol{\alpha}}(\mathcal{N})}{\partial E_{\boldsymbol{\alpha},\mathcal{N}}}\Big|_{\boldsymbol{\alpha}=\mathbf{1}} \times \frac{\partial E_{\boldsymbol{\alpha},\mathcal{N}}}{\partial \alpha_i}\Big|_{\boldsymbol{\alpha}=\mathbf{1}}, \ \forall i \in \mathcal{N}, \tag{3.3.8}$$

where the second term is exactly the power demand E_i of each tenant. On the other hand, the first term is the partial derivative of the weighted optimal expected cost with respect to the weighted power demand and then evaluating at the full aggregation level, i.e., $\boldsymbol{\alpha} = \mathbf{1}$, which can be considered as the marginal cost assigned to each tenant. Therefore, the multiplication of the marginal cost and power demand gives the distributed cost to each tenant. Further, we prove that the cost allocation vector $\pi = [\pi_1, \ldots, \pi_N]^T$ given in (3.3.7) lies in the core as shown in following theorem:

Theorem 3.3.2 *The resulting cost allocation vector of the proposed cost allocation scheme is fair and lies in the core of our cooperative game.*

[1]The operator $\succeq$ represents component-wise vector comparison.

Proof. The proof is referred to Appendix. ■

The most significant advantage of exploiting this method is its low computational complexity. Compared to using the nucleolus, we only need to calculate $\mathcal{O}(N)$ equations.

3.4 Numerical Experiments

In this section, we first introduce our simulation setup and then conduct trace-driven simulations to show the benefits of tenant aggregation in purchasing power in the wholesale electricity market and the effectiveness of our proposed cost allocation scheme.

3.4.1 Simulation Setup

A set of four independent tenants $\mathcal{N} = \{1, 2, 3, 4\}$ is considered in our simulations. The total number of servers for each tenant is 10,000, 12,500, 15,000 and 17,500, respectively. Assume the idle power and peak power of each server is 150 W and 250 W, respectively. Besides, the average PUEs of all the tenants are set to 1.5. The average service rate of a server in each tenant is set to be 200, 250, 300 and 350 requests per second, respectively. The maximum average response time for each tenant is set to be 100, 80, 60 and 40 ms, respectively. The above simulation parameters are summarized in Table 1.

Table 1: Simulation parameters

	M_i	μ_i (requests/s)	$T_i^{\max}$ (ms)
Tenant 1	10000	200	100
Tenant 2	12500	250	80
Tenant 3	15000	300	60
Tenant 4	17500	350	40

The real-world dataset we use to simulate the workloads is from the Google cluster trace [86]. The selected dataset includes workload information over 29 days (i.e., 696 hours) during May 2011 for a cluster of 12,500 severs. We repeat the original data and extend it to 1008-hour workloads (i.e., 42 days). Then, we randomly choose 4 different 720-hour (i.e., 30 days) portions from the extended dataset as our tenant workloads. Figure III.5(a) shows the CDFs

of the normalized tenant workload arrival rate for four tenants at hour 22. Then we can estimate the power demand of each tenant according to (3.1.6). The CDFs of the power demand for four tenants are depicted in Figure III.5(b).

In our simulations, tenants can purchase power either individually or cooperatively by forming the grand coalition. Moreover, we assume tenants bid their power demand in the day-ahead market for each hour in the following operating day. By default, the expected day-ahead price μ_p^d is set to be 50 cents per kWh, the expected negative imbalance price μ_p^- is set to be 2 dollars per kWh, and the expected positive imbalance price μ_p^+ is set to be 20 cents per kWh in the simulations.

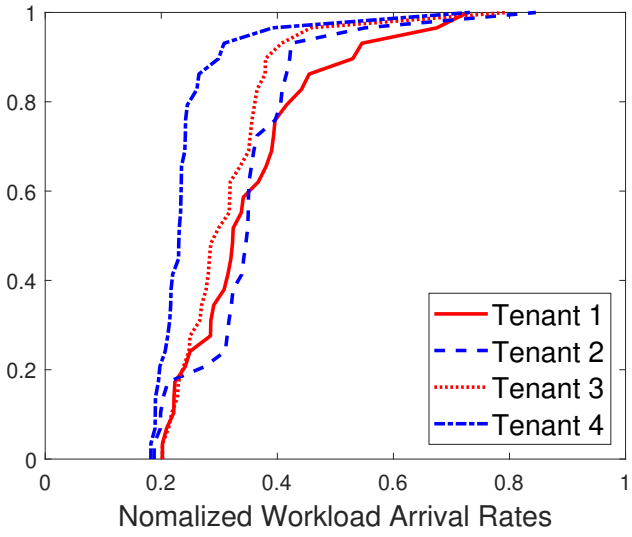
Last, all our simulations are conducted on a desktop computer with an Intel Core i7-4790 3.60GHz CPU and 8GB RAM using MATLAB R2016a.

3.4.2 Experimental Results

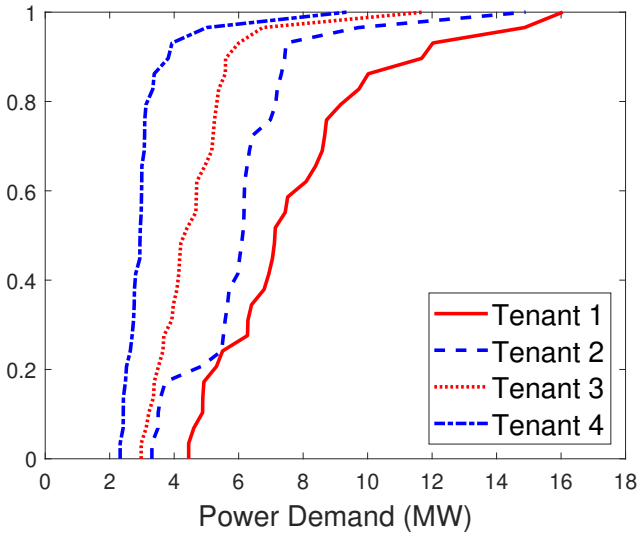
In this section, we simulate and analyze how tenants can benefit from forming the grand coalition to save their electricity cost when purchasing power in the wholesale electricity market. Here, we consider the case where each tenant bids its power demand individually by minimizing its expected energy cost as the baseline scenario for comparison.

Benefits of Aggregation

We first observe the benefits of coalitional bidding in the wholesale market. Based on Theorem 3.2.1, we can calculate the optimal day-ahead bid Q_S^* of any coalition S. Figure 6 shows the resulting optimal day-ahead bidding level of our proposed method and the sum of optimal individual bidding level in the baseline over 24 hours. Figure 7 shows the energy cost comparison of our proposed approach and the baseline. The result of the baseline scenario is obtained by adding up the optimal expected electricity cost of each tenant when they bid in the day-ahead market individually, while the result of the proposed method is obtained by letting tenants form the grand coalition to bid in the day-ahead market cooperatively.



(a)



(b)

Figure 5: CDFs of the normalized tenant workload arrival rates and power demand at hour 22.

It is shown in Figure 7 that the total electricity cost is effectively reduced by cooperative day-ahead bidding, which validates the subadditivity property of our cooperative game given in Lemma 3.2.1. The average hourly cost saving is around 11.03% under the current setting.

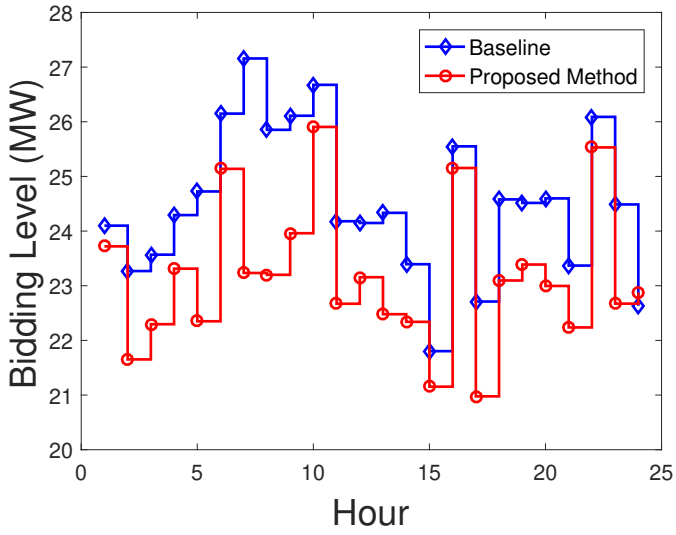


Figure 6: Day-ahead bidding level comparison over 24 hours.

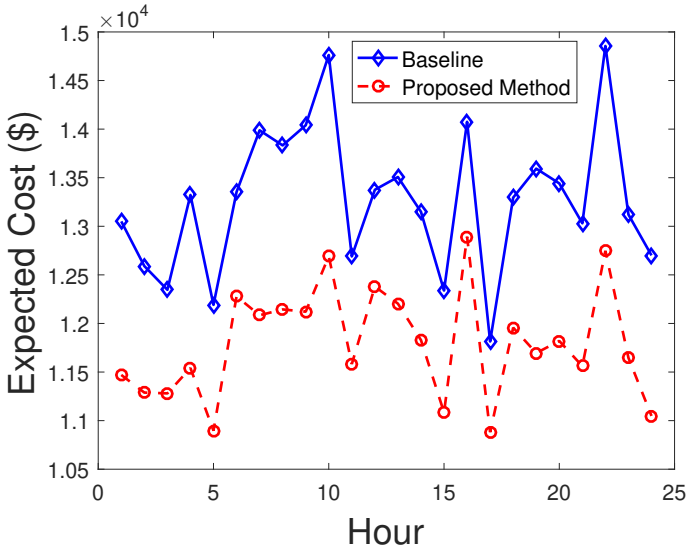


Figure 7: Total expected cost comparison over 24 hours.

Cost Allocation

Next we focus on how to fairly distribute the total energy cost after coalitional bidding among each participating tenant using our proposed cost allocation method. We split the total expected cost based on the marginal contribution of each tenant in the grand coalition by applying the proposed cost allocation scheme in Section 3.3.2. Figure 8 presents the cost allocation to each tenant at hour 22. The height of each bar (yellow bar plus blue

bar) denotes the expected cost of each tenant when it bids individually in the day-ahead market at hour 22. The height of yellow bar shows the reduced cost of each tenant after coalitional day-ahead bidding. The cost saving percentage of each tenant over 24 hours in a day is given in Figure 9. It can be observed that our proposed allocation method can always ensure positive cost reductions for each tenant and the cost saving amount of each tenant is different, depending on its contribution to the aggregation benefits.

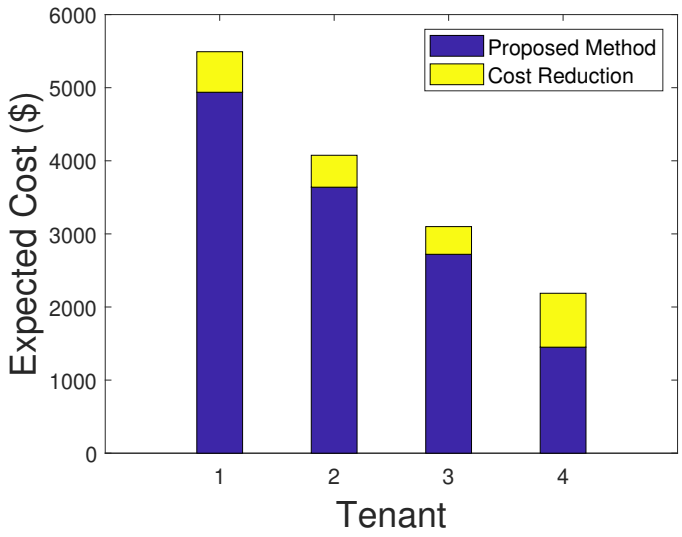


Figure 8: Cost allocation of each tenant at hour 22 under the current setting.

Table 2 presents the noncooperative and coalitional electricity cost of each coalition at hour 22. The last column gives the corresponding excesses $e(\pi, S)$ defined in (2.3.8). From row 1 to row 14, the calculated excesses are all negative which satisfies the condition of subgroup rationality, i.e., $\sum_{i\in S} \pi_i \leq c(S)$. The last row indicates that our cost allocation is efficient since $\sum_{i\in\mathcal{N}} \pi_i = c(\mathcal{N})$. It verifies that our proposed cost allocation lies in the core of the cooperative game since both subgroup rationality and efficiency conditions are satisfied.

Impact of Price Penalty Ratio

Now we present how market prices affect the cost saving and the day-ahead bid of each tenant when they form the grand coalition. According to Theorem 3.2.1, the optimal day-ahead bid depends on the quantile function where the percentile ε^* is decided by expected

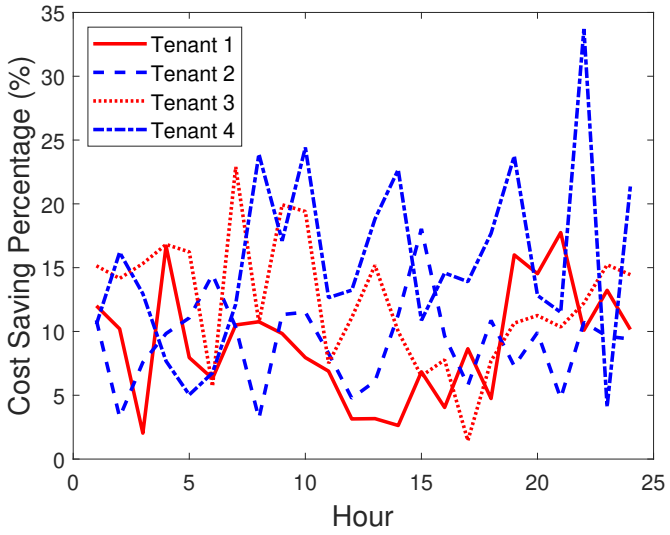
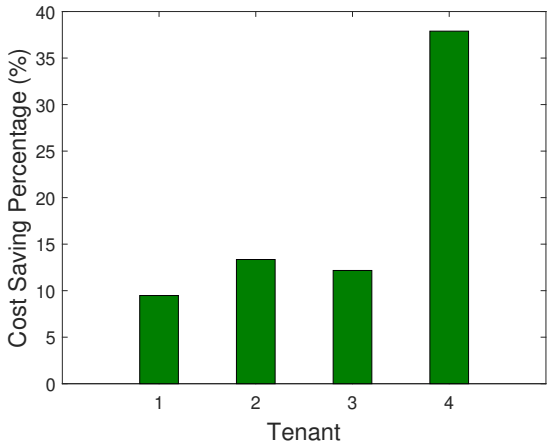


Figure 9: Individual cost saving percentage of each tenant after coalitional day-ahead bidding over 24 hours.

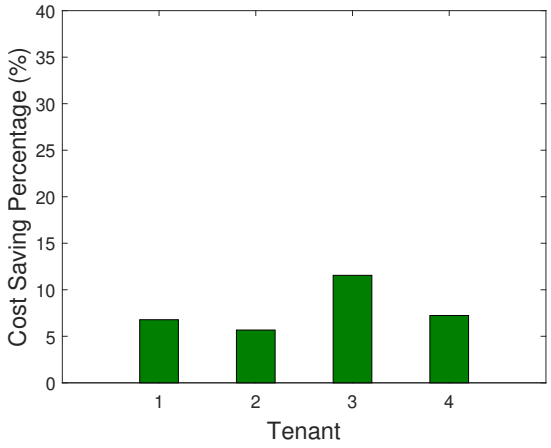
Table 2: Cost comparison for all coalitions of four tenants at hour 22

	S	$c(S)$	$\sum_{i\in S}\pi_i$	$\sum_{i\in S}\pi_i - c(S)$
1	$\{1\}$	5492.2	4937.7	−554.5
2	$\{2\}$	4075.6	3639.0	−436.6
3	$\{3\}$	3099.8	2720.8	−379.0
4	$\{4\}$	2187.8	1450.4	−737.4
5	$\{1,2\}$	8809.2	8576.7	−232.5
6	$\{1,3\}$	8132.5	7658.5	−474.0
7	$\{1,4\}$	7018.5	6388.1	−630.4
8	$\{2,3\}$	6748.7	6359.8	−388.9
9	$\{2,4\}$	5941.8	5089.4	−852.4
10	$\{3,4\}$	4966.8	4171.2	−795.6
11	$\{1,2,3\}$	11498.0	11297.5	−200.5
12	$\{1,2,4\}$	10300.0	10027.1	−272.9
13	$\{1,3,4\}$	9574.0	9108.9	−465.1
14	$\{2,3,4\}$	8454.0	7810.2	−643.8
15	$\{1,2,3,4\}$	12747.9	12747.9	0

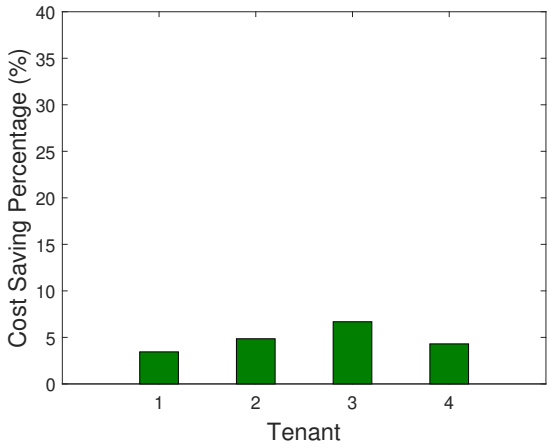
electricity prices μ_p^d, μ_p^- and μ_p^+. For the simplicity of presentation, we set the expected positive imbalance price μ_p^+ to be 0 but it does not affect our analytical analysis beforehand. It follows that the percentile ε^* will reduce to $1 - \mu_p^d/\mu_p^-$. Under this simplification, we introduce the price penalty ratio ω defined as μ_p^d/μ_p^- [16]. In order to obtain different price penalty ratios, we fix the expected day-ahead price μ_p^d as a constant and adjust the expected



(a) $\omega = 0.25$



(b) $\omega = 0.50$



(c) $\omega = 0.75$

Figure 10: Cost saving percentage of each tenant at hour 22 when the price penalty ratio ω is 0.25, 0.50 and 0.75, respectively.

negative imbalance price μ_p^- to different values.

Figure 10 depicts the cost saving percentage of each tenant at hour 22 when the price penalty ratio ω is 0.25, 0.50 and 0.75, respectively. Further, the percentage of the average cost saving of each tenant over 24 hours is listed in Table 3. We can observe that the percentage of the average cost saving decreases when the price penalty ratio ω increases. This is intuitive since we have less chance to reduce cost through aggregation when the penalty price is lower. Indeed, when the expected negative penalty price is the same as the expected day-ahead electricity price, there is no need for aggregation since one could always buy any shortfall from the real-time market without penalty.

Table 3: The percentage of the average cost saving of each tenant under different price penalty ratios

	$\omega = 0.25$	$\omega = 0.50$	$\omega = 0.75$
Tenant 1	10.11%	6.23%	3.08%
Tenant 2	10.17%	7.05%	3.78%
Tenant 3	14.67%	8.15%	4.98%
Tenant 4	17.12%	8.23%	4.90%

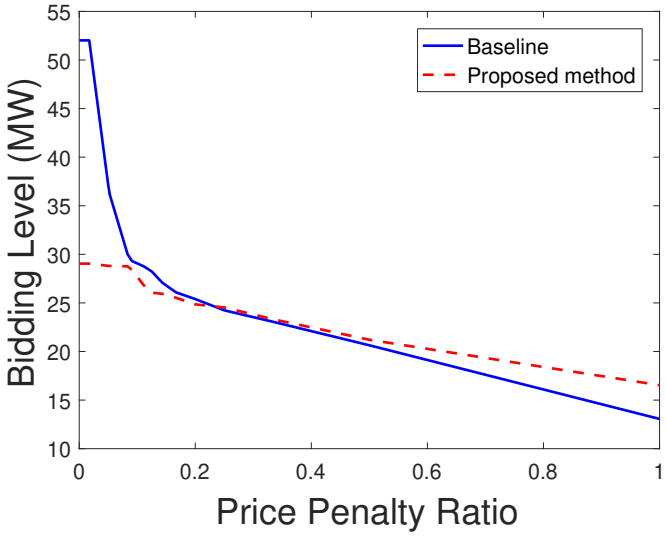


Figure 11: Day-ahead bidding level comparison under price penalty ratios from 0 to 1.

Figure 11 shows the changes of day-ahead bidding level of the baseline and the proposed method under different price penalty ratios. It can be observed that under both cases,

the day-ahead bidding level decreases as the price penalty ratio increases. The reason is that when the price penalty ratio is near 0, tenants behave more conservatively since the expected negative imbalance price is much higher than the expected day-ahead price. In order to avoid high penalty for energy shortfall, they tend to bid more power amount to lower the possible mismatch between committed power supply in the day-ahead market and realized power demand in the real-time market. On the other hand, when the price penalty ratio is approaching 1, tenants can buy any shortfall in the real-time market without penalty and therefore tend to bid less. Moreover, the change rate of bidding level of our proposed method with respect to the price penalty ratio is smaller than that of the baseline. This is due to the fact that the proposed method has a smaller power demand uncertainty and therefore is less sensitive to the price penalty ratio.

3.5 Summary

In this chapter, we have proposed a new approach to minimize the electricity cost for tenants participating in the wholesale electricity market. The electricity cost can be effectively reduced by bidding in the day-ahead market collectively since aggregation can reduce the uncertainty of power demand. We model this aggregation process as a cooperative game and present a cost allocation mechanism based on the marginal contribution of each tenant to the total expected cost to fairly distribute the optimal expected cost to each tenant within the grand coalition. Finally, simulations based on real-world traces verify the effectiveness of our proposed cost saving method.

CHAPTER IV

RESOURCE MANAGEMENT IN GEOGRAPHICALLY DISTRIBUTED DATACENTERS

In this chapter, we investigate the optimal energy procurement for datacenter operator who manages geographically distributed datacenters when participates in the multi-timescale electricity markets. Given the uncertain interactive workload demand, renewable generation and real-time electricity prices, we jointly optimize electricity procurement in the multi-timescale electricity markets, workload routing decisions, IT server allocation decisions, thermal energy storage charge and discharge decisions such that the total cost of datacenters when serving both interactive and batch workload can be minimized. We formulate it as a two-stage stochastic optimization problem by considering random scenarios and then reformulate as its deterministic equivalent problem. Finally, a case study based on real-world traces is presented to validate the effectiveness of our proposed stochastic approach.

The rest of this chapter is organized as follows. In Section 4.1, we present the system model including models of workload, datacenter power consumption, electricity market, renewable energy, thermal energy storage and cost. In Section 4.2, we present the formulation of two-stage stochastic optimization problem. In Section 4.3, we reformulate the two-stage stochastic optimization problem as its deterministic equivalent problem. Extensive numerical simulations based on real-world traces are presented in Section 4.4. Finally, the summary is given in Section 4.5.

4.1 System Modeling

As illustrated in Fig. 12, we consider a cloud service provider (CSP) who operates N geographically located datacenters, procuring electricity by participating in two-timescale electricity markets: long-term market and real-time market. Each datacenter $i \in \mathcal{N} = \{1, 2, \cdots, N\}$ is equipped with on-site renewable generators and thermal energy storage. We consider a discrete-time model whose time periods denoted by $t \in \mathcal{T} = \{0, 1, 2, \cdots, T-1\}$ match the timescale at which the capacity provisioning and scheduling decisions can be updated. The duration of each time period t varies from 15 minutes to one hour.

4.1.1 Workload Model

Assume there is a set of sources $\mathcal{J}$ representing aggregate workload demand which can be routed to a set of geographically deployed datacenters $\mathcal{N}$ through geographical load balancing (GLB). Each source $j \in \mathcal{J} = \{1, 2, \cdots, J\}$ can be interpreted as collective workload demand from a group of local users. Moreover, the workload that datacenters are dealing with can be classified into two classes: delay-sensitive workload and delay-tolerant workload.

Delay-sensitive Workload

Delay-sensitive workload is also known as interactive workload such as web searches and emails. It requires real-time processing and is constrained by the quality of service (QoS) requirements. For interactive workload, there is a maximum response time as indicated by the service-level agreement (SLA).

Denoted by $\lambda_{i,j,t}$ the amount of interactive workload dispatched from source $j \in \mathcal{J}$ to datacenter $i \in \mathcal{N}$ at time period t, we can express the total interactive workload received from datacenter i at time period t as $\lambda_{i,t} = \sum_{j \in \mathcal{J}} \lambda_{i,j,t}$ and total interactive workload generated from each source j at time period t as $L_{j,t} = \sum_{i \in \mathcal{N}} \lambda_{i,j,t}$. In reality, interactive workload demand is highly uncertain and therefore can be modeled as random variables.

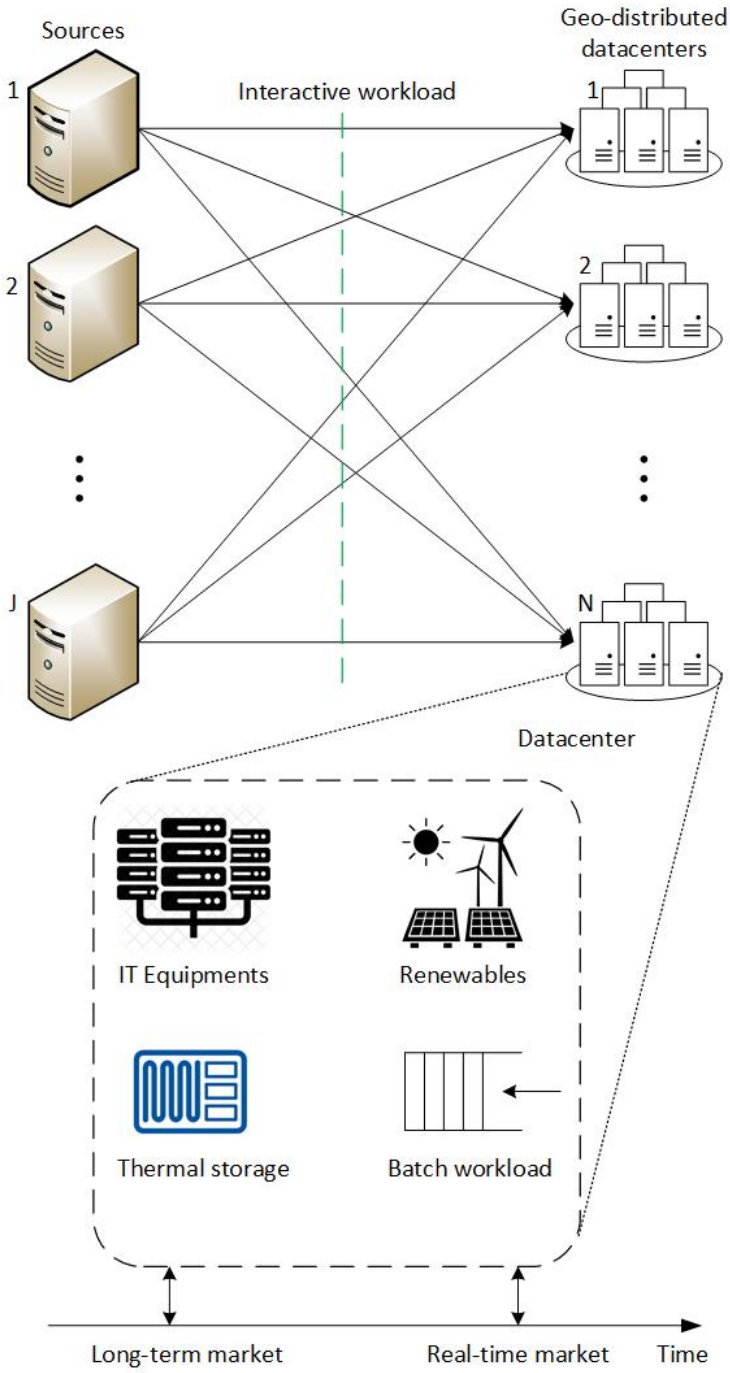


Figure 12: Geographically distributed datacenters participate in two-timescale electricity markets.

Delay-tolerant Workload

Delay-tolerant workload is also known as batch workload such as anti-virus software scanning and scientific simulations. Usually it can be scheduled at any time as long as it is completed

by the designated deadline, i.e., there is a maximum completion time. We denote by H_i, f_i and MP_i the batch workload demand in datacenter i, IT resources provided by a single server in datacenter i at one time period to process batch workload and maximum parallelization in datacenter i, respectively. Here, we assume that the batch workload demand is known beforehand.

4.1.2 Datacenter Power Consumption Model

At each time period t, assume each datacenter i has $0 \leq m_{i,t} \leq M_i$ homogeneous servers that are turned on, where M_i is the total number of servers in datacenter i. Usually there are more than thousands of servers in a single datacenter, therefore we relax the $m_{i,t}$ as real numbers lying in the interval $[0, M_i]$.

Since there are two types of workload, at each time period t, we assume datacenter i assign $a_{i,t}$ and $b_{i,t}$ servers to process interactive workload and batch workload, respectively. It follows that the total number of active servers at time period t in datacenter i can be returned as $m_{i,t} = a_{i,t} + b_{i,t}$. Note that the interactive workload is routed from source $j \in \mathcal{J}$ to datacenter $i \in \mathcal{N}$ through the GLB scheme while batch workload is assumed to be generated from the datacenters themselves. We have the following two constraints for batch workload in datacenter i at time period t:

$$\sum_{t=0}^{T-1} b_{i,t} f_i = H_i, \tag{4.1.1}$$

$$0 \leq b_{i,t} \leq MP_i. \tag{4.1.2}$$

Constraint (4.1.1) indicates that the total batch workload in datacenter i has to be completed within the whole T time periods. Constraint (4.1.2) indicates that the number of allocated servers to process batch workload in datacenter i at each time period t cannot exceed the maximum parallelization MP_i.

It is known that [67] the average IT power consumption of a single server in datacenter

i can be estimated as

$$p_i(u_i) = P_i^{\text{idle}} + u_i \left(P_i^{\text{peak}} - P_i^{\text{idle}} \right), \tag{4.1.3}$$

where P_i^{peak}, P_i^{idle} and u_i denotes its power consumption at idle status, power consumption at fully utilized status and average CPU utilization level, respectively. From this affine relationship in (4.1.3), u_i equals 0 indicates that the server consumes its idle power P_i^{idle} while u_i equals 1 indicates that the server consumes its peak power P_i^{peak}.

M/GI/1 Processor Sharing (PS) queue is adopted here to model the service process at each server [69]. Therefore, the average CPU utilization level can be represented as $\lambda_{i,t}/(a_{i,t}\mu_i)$ where μ_i is the average server service rate at which user requests are processed by a server in datacenter i. Therefore, the total IT power consumption of datacenter i at time period t when $m_{i,t}$ servers are kept active is given by

$$\begin{aligned} E_{i,t} &= a_{i,t} \left[P_i^{\text{idle}} + \frac{\lambda_{i,t}}{a_{i,t}\mu_i} \left(P_i^{\text{peak}} - P_i^{\text{idle}} \right) \right] + b_{i,t} P_i^{\text{peak}} \\ &= a_{i,t} P_i^{\text{idle}} + \frac{\lambda_{i,t}}{\mu_i} \left(P_i^{\text{peak}} - P_i^{\text{idle}} \right) + b_{i,t} P_i^{\text{peak}}. \end{aligned} \tag{4.1.4}$$

Note that we assume batch workload is processed at full utilized CPU level (i.e., $u_i = 1$).

Also, the queuing delay of user requests given $a_{i,t}$ active servers and workload arrival rates $\lambda_{i,t}$ in datacenter i at time period t is represented as

$$g_i(a_{i,t}, \lambda_{i,t}, t) = \frac{1}{\mu_i - \lambda_{i,t}/a_{i,t}}. \tag{4.1.5}$$

Let $T_i^{\max}$ denote the maximum average response time of user requests that can be tolerated at datacenter i. To ensure that $g_i(a_{i,t}, \lambda_{i,t}, t) \leq T_i^{\max}$, we obtain the following constraint for $a_{i,t}$:

$$a_{i,t} \geq \frac{\lambda_{i,t}}{\mu_i - 1/T_i^{\max}}. \tag{4.1.6}$$

Besides, there is non-IT power consumption (e.g., cooling) associated with each datacenter and the average power usage effectiveness (PUE) which is defined as the ratio of the total

power consumption to the IT power consumption at each datacenter is adopted to capture its nature. Denote the average PUE in datacenter i as γ_i, and therefore the corresponding total power consumption of datacenter i at time period t can be represented as $\gamma_i E_{i,t}$.

4.1.3 Electricity Market Model

In the chapter, we consider datacenters participating in two-timescale electricity markets. Assume each geographically distributed datacenter is associated with a local long-time (forward) market and a real-time (spot) market.

Long-term Market

Long-term market is also known as retail market. Datacenters can sign contracts with market operator ahead of time (e.g., weekly/monthly/quarterly/yearly) at a fixed electricity price for certain amount of committed electricity in the future time periods. We denote the long-term electricity price for datacenter i at time period t as $p^l_{i,t}$ and the long-term electricity procurement as $q^l_{i,t}$. When signing contracts, the interactive and batch workload demand, renewable generation and real-time electricity prices are all uncertain and therefore long-term electricity procurement can be interpreted as first-stage decisions for datacenters.

Real-time Market

Different from the long-term market, real-time market serves the instantaneous needs for purchasing electricity in the operating time. Usually it follows a pay-as-you-go fashion and has higher real-time electricity prices than long-term electricity prices. In this chapter, we consider the case where datacenters have to procure electricity in the real-time market whenever there is an energy shortage in the operating day to make up the electricity deficit. We denote the real-time electricity price for datacenter i at time period t as $p^r_{i,t}$ and the real-time electricity procurement as $q^r_{i,t}$.

Different from the long-term market with a fixed electricity price, the electricity price

in real-time market is uncertain since it is cleared on the real-time basis and the involving buyers and sellers can not know the exact value beforehand. Thus, we model the real-time electricity price $p^r_{i,t}$ for datacenter i at time period t as random variables.

4.1.4 Renewable Energy Model

Assume each datacenter is equipped with renewable generators such as wind turbines and/or solar panels. Due to the intermittent nature of the wind and solar renewable energy, the renewable generation of datacenter i at time period t can be modeled as random variables $w_{i,t}$. Without loss of generality, we assume the operating cost of renewable generation is zero.

4.1.5 Thermal Energy Storage Model

For each datacenter i, we denote the thermal energy storage capacity as $S_i^{\max}$, the energy level at time period t as $S_{i,t}$, the charged energy into thermal energy storage system at time period t as $s^+_{i,t}$, and the discharged energy out of the thermal energy storage system at time period t as $s^-_{i,t}$. In practice, thermal energy storage system will undergo conversion loss during charging and discharging processes. We denote the round-trip efficiency during charging and discharging processes as $\eta^+_i < 1$ and $\eta^-_i < 1$, respectively. Then, we derive the following dynamic equation for energy level at adjacent time periods in datacenter i as

$$S_{i,t+1} = S_{i,t} + \eta^+_i s^+_{i,t} - s^-_{i,t}/\eta^-_i, \ \forall t \in [1, T-1]. \tag{4.1.7}$$

Moreover, there exists upper limits on the charge and discharge rate. Denote the maximum charge rate and discharge rate by $s^+_{i,\max}$ and $s^-_{i,\max}$, respectively, and therefore we have the

following constraints:

$$0 \leq s_{i,t}^{+} \leq s_{i,\max}^{+}, \tag{4.1.8}$$

$$0 \leq s_{i,t}^{-} \leq s_{i,\max}^{-}. \tag{4.1.9}$$

Also, at each time period t in datacenter i, the energy level of the thermal energy system is bounded by the maximum capacity. That is,

$$0 \leq S_{i,t} \leq S_i^{\max}, \ \forall t \in [1, T]. \tag{4.1.10}$$

The initial energy level at datacenter i is assumed to be $S_{i,0} \in [0, S_i^{\max}]$.

Without loss of generality, at each time period t in datacenter i, we assume the discharged energy from the thermal energy storage is less than the cooling power consumption, i.e., $s_{i,t}^{-} \leq (\gamma_i - 1)E_{i,t}$. That is to say, the thermal energy cannot be utilized to power the servers.

4.1.6 Cost Model

The total cost when operating geographically distributed datacenters consists of two parts: bandwidth cost and energy cost. Bandwidth cost is the monetary cost incurred due to the communication demand from the interactive workload routed between sources and datacenters. Energy cost is due to the electricity procurement in both long-term markets and real-time markets.

Bandwidth Cost

In this chapter, we apply a linear bandwidth cost model to represent the bandwidth cost between the sources and datacenters. Bandwidth cost when routing interactive workload $\lambda_{i,j,t}$ from source j to datacenter i at time period t is given by

$$\mathcal{B}_{i,j,t} = y_{i,j}\lambda_{i,j,t}, \tag{4.1.11}$$

where $y_{i,j}$ is the unit cost associated with each interactive workload $\lambda_{i,j,t}$.

Energy Cost

Energy cost consists of two parts, long-term energy procurement cost and real-time energy procurement cost. Long-term and real-time energy procurement cost of datacenter i at time period t can be represented as $p^l_{i,t}q^l_{i,t}$ and $p^r_{i,t}q^r_{i,t}$, respectively. It follows that the total energy cost of datacenter i at time period t is given by

$$\mathcal{E}_{i,t} = p^l_{i,t}q^l_{i,t} + p^r_{i,t}q^r_{i,t}. \tag{4.1.12}$$

4.2 Two-Stage Stochastic Formulation

In this chapter, we are interested in minimizing the total cost over a large time horizon incurred when geographically distributed datacenters participating in the two-timescale electricity markets. Therefore, this optimization problem can be stated as follows: given the uncertain interactive workload demand $\mathbf{L}$, renewable generation $\mathbf{w}$ and real-time electricity prices $\mathbf{p^r}$, jointly optimize electricity procurement $\mathbf{q^l}$ and $\mathbf{q^r}$ in the two-timescale electricity markets, workload routing decisions $\boldsymbol{\lambda}$, IT server allocation decisions $\mathbf{a}$ and $\mathbf{b}$, and thermal energy storage charge/discharge decisions $\mathbf{s}^+$ and $\mathbf{s}^-$ such that the total cost of datacenters when serving both interactive and batch workload can be minimized. All the decision variables are continuous. It can be formulated as the following two-stage stochastic optimization:

$$\mathbf{2SP}: \min_{\mathbf{q^l}} \sum_{t=0}^{T-1}\sum_{i=1}^{N} p^l_{i,t}q^l_{i,t} + \mathbb{E}_\xi[\mathcal{Q}(\mathbf{q^l}, \xi)] \tag{4.2.1a}$$

s.t.

$$q^l_{i,t} \geq 0,\ \forall i, t \tag{4.2.1b}$$

where

$$\mathcal{Q}(\mathbf{q}^{\mathbf{l}},\xi) \;=\; \min_{\substack{\mathbf{a},\mathbf{b},\boldsymbol{\lambda},\\ \mathbf{q}^{\mathbf{r}},\mathbf{s}^{+},\mathbf{s}^{-}}} \sum_{t=0}^{T-1}\sum_{i=1}^{N}\sum_{j=1}^{J} y_{i,j}\lambda_{i,j,t}(\xi) + \sum_{t=0}^{T-1}\sum_{i=1}^{N} p_{i,t}^{r}(\xi)q_{i,t}^{r}(\xi) \tag{4.2.2a}$$

s.t.

$$\sum_{i\in\mathcal{N}} \lambda_{i,j,t}(\xi) = L_{j,t}(\xi),\ \forall j,t \tag{4.2.2b}$$

$$a_{i,t}(\xi) \geq \frac{\lambda_{i,t}(\xi)}{\mu_i - 1/T_i^{\max}},\ \forall i,t \tag{4.2.2c}$$

$$\sum_{t=0}^{T-1} b_{i,t}(\xi) f_i = H_i,\ \forall i \tag{4.2.2d}$$

$$0 \leq b_{i,t}(\xi) \leq MP_i,\ \forall i,t \tag{4.2.2e}$$

$$a_{i,t}(\xi) + b_{i,t}(\xi) \leq M_i,\ \forall i,t \tag{4.2.2f}$$

$$S_{i,1}(\xi) = S_{i,0} + \eta_i^{+} s_{i,0}^{+}(\xi) - s_{i,0}^{-}(\xi)/\eta_i^{-},\ \forall i \tag{4.2.2g}$$

$$S_{i,t+1}(\xi) = S_{i,t}(\xi) + \eta_i^{+} s_{i,t}^{+}(\xi) - s_{i,t}^{-}(\xi)/\eta_i^{-}, \forall i, t \in [1, T-1] \tag{4.2.2h}$$

$$s_{i,t}^{-}(\xi) \leq (\gamma_i - 1)E_{i,t}(\xi),\ \forall i,t \tag{4.2.2i}$$

$$q_{i,t}^{l} + w_{i,t}(\xi) + s_{i,t}^{-}(\xi) + q_{i,t}^{r}(\xi) \geq \gamma_i E_{i,t}(\xi) + s_{i,t}^{+}(\xi), \forall i,t \tag{4.2.2j}$$

$$0 \leq s_{i,t}^{+}(\xi) \leq s_{i,\max}^{+},\ \forall i,t \tag{4.2.2k}$$

$$0 \leq s_{i,t}^{-}(\xi) \leq s_{i,\max}^{-},\ \forall i,t \tag{4.2.2l}$$

$$0 \leq S_{i,t}(\xi) \leq S_i^{\max},\ \forall i, t \in [1,T] \tag{4.2.2m}$$

$$\lambda_{i,j,t}(\xi) \geq 0,\ \forall i,j,t \tag{4.2.2n}$$

$$a_{i,t}(\xi), q_{i,t}^{r}(\xi) \geq 0,\ \forall i,t \tag{4.2.2o}$$

where ξ represents random scenarios, constraint (4.2.2j) describes the power supply-demand relationship in datacenter i at time period t, and IT power consumption $E_{i,t}(\xi)$ for datacenter i at time period t is given by replacing with $a_{i,t}(\xi)$, $b_{i,t}(\xi)$ and $\lambda_{i,t}(\xi)$ in (4.1.4).

4.3 Solution Methodology

The proposed two-stage stochastic programming (**2SP**) can be reformulated as a deterministic equivalent problem (**DEP**) with its finite number of scenarios assumed with known discrete probability distributions so that the expectations can be expressed as the finite summations and each constrain in the second stage is given separately for each scenario. Further, we apply Monte Carlo simulations to reduce the scenario set to a manageable size. Assuming that all our random parameters are independent and identically distributed (i.i.d.), we can generate a sample containing K scenarios ξ^1, ξ^2, ..., ξ^K of the random parameters with equal probability $1/K$. Then, the expectation function $\mathbb{E}_\xi[\mathcal{Q}(\mathbf{q}^\mathbf{l}, \xi)]$ can be approximated by the sample average

$$\hat{\mathcal{Q}}(\mathbf{q}^\mathbf{l}, \xi) = \frac{1}{K} \sum_{k=1}^{K} \mathcal{Q}(\mathbf{q}^\mathbf{l}, \xi^k). \tag{4.3.1}$$

Next, let $\mathbf{x}$ denote the first-stage decision variables $\mathbf{q}^\mathbf{l}$, and $\mathbf{z}(\xi)$ denote the second-stage decision variables $\mathbf{a}(\xi)$, $\mathbf{b}(\xi)$, $\boldsymbol{\lambda}(\xi)$, $\mathbf{q}^\mathbf{r}(\xi)$, $\mathbf{s}^+(\xi)$ and $\mathbf{s}^-(\xi)$. Then, with properly chosen matrices $\mathbf{c}$, $\mathbf{q}(\xi)$, $\mathbf{T}(\xi)$, $\mathbf{W}(\xi)$ and $\mathbf{h}(\xi)$, the corresponding deterministic equivalent problem can be reformulated as following matrix form:

$$\mathbf{DEP}: \min_{\mathbf{x}} \mathbf{c}^\top \mathbf{x} + \frac{1}{K} \sum_{k=1}^{K} \mathcal{Q}(\mathbf{x}, \xi^k) \tag{4.3.2a}$$

s.t.

$$\mathbf{x} \geq \mathbf{0} \tag{4.3.2b}$$

where for each scenario $\xi^k, k = 1, \ldots, K$

$$\mathcal{Q}(\mathbf{x}, \xi^k) = \min_{\mathbf{z}(\xi)} \mathbf{q}(\xi)^\top \mathbf{z}(\xi) \tag{4.3.2c}$$

s.t.

$$\mathbf{T}(\xi)\mathbf{x} + \mathbf{W}(\xi)\mathbf{z}(\xi) = \mathbf{h}(\xi) \tag{4.3.2d}$$

$$\mathbf{z}(\xi) \geq \mathbf{0} \tag{4.3.2e}$$

4.4 Case Study

In this section, we present the results of the proposed stochastic approach based on real-world traces. We compare the performance of our proposed stochastic approach with the deterministic approach. All the experiments are implemented in MATLAB R2018a using GUROBI 8.0.1 on a desktop computer with an Intel Core i7-4790 3.60GHz CPU and 16GB RAM.

4.4.1 Numerical Settings

In what follows, we present the parameter settings of datacenters and workload, stochastic parameters and thermal energy storage.

Datacenters and Workload

In this chapter, the number of geographically distributed datacenters N, sources J and time periods T are set to be 4, 4 and 24, respectively. For each datacenter i, server idle power consumption P_i^{idle}, server peak power consumption P_i^{peak} and PUE γ_i are to be 150W, 250W and 1.2, respectively. Assume all the workload can be fully processed by half of the total datacenters and therefore we set the number of servers M_i for each datacenter i to be 8000. The maximum parallelization MP_i for datacenter i is set to be 20% of the total number of servers M_i. The unit cost $y_{i,j}$ associated with each interactive workload $\lambda_{i,j,t}$ is set proportionally to the distance between each datacenter and source. The average server service rate μ_i, maximum average response time $T_i^{\max}$, local batch workload demand H_i and provided IT resources f_i for processing batch workload for each datacenter i are summarized in Table 4.

Stochastic Parameters

The nominal value of renewable power generation $\overline{w}_{i,t}$ for each datacenter i at time period t is taken from the NREL National Wind Technology Center (M2) [7], where coefficients

Table 4: Datacenters and workload parameters

Datacenters	μ_i (requests/s)	$T_i^{\max}$ (ms)	H_i	f_i
1	100	20	50000	10
2	80	25	48000	12
3	100	20	60000	15
4	75	30	55000	10

are appropriately scaled. The nominal value of interactive workload demand $\overline{L}_{j,t}$ for each source j at time period t is taken from the Google cluster trace [86], where coefficients are appropriately scaled. For the electricity market parameter settings, the long-term electricity price $p^l_{i,t}$ and the nominal value of real-time electricity price $\overline{p}^r_{i,t}$ for each datacenter i at time period t are taken from [8], where coefficients are appropriately scaled.

Thermal Energy Storage

In this chapter, we assume each datacenter is equipped with the same thermal energy storage device. The initial energy level $S_{i,0}$ is assumed to be 10kWh while the maximum capacity $S_i^{\max}$ is assumed to be 100kWh for datacenter i. Round-trip efficiency during charging η_i^+ and discharging η_i^- for datacenter i are assumed to be 0.9. Maximum charge $s^+_{i,t}$ and discharge $s^-_{i,t}$ rate are assumed to be 10kW for each datacenter i at time period t.

4.4.2 Results and Discussions

In what follows, we first describe the chosen benchmark (i.e., deterministic approach), and then apply it to contrast our proposed stochastic approach in terms of the optimality when reduce the total operating cost.

Deterministic Approach

Assume that the expected values of interactive workload demand $\overline{L}_{j,t}$ for each source j at time period t and renewable generation $\overline{w}_{i,t}$ and real-time electricity prices $\overline{p}^r_{i,t}$ for each datacenter i at time period t can be accurately estimated before the operating period. Then,

we replace the random parameters $\mathbf{L}$, $\mathbf{w}$ and $\mathbf{p^r}$ by their corresponding expected values in the second stage of **2SP**. Therefore, the two-stage stochastic optimization problem reduces to a simple linear programming (LP) without any uncertainties, which can be easily solved. The deterministic approach can be stated as following two steps:

- Given the expected values of stochastic parameters $\overline{L}_{j,t}$, $\overline{w}_{i,t}$ and $\overline{p}^r_{i,t}$, solve **2SP** for optimal first-stage long-term electricity procurement decisions $q^{l(*)}_{i,t}$.

- Fix first-stage decisions at $q^{l(*)}_{i,t}$ and apply Monte Carlo simulations as illustrated in Section 4.3. Solve **2SP** with respect to each generated scenario $\xi^k, k = 1, 2, \ldots, K$ to obtain the second-stage cost and return the total operating cost C as

$$C = \sum_{t=0}^{T-1} \sum_{i=1}^{N} p^l_{i,t} q^{l(*)}_{i,t} + \frac{1}{K} \sum_{k=1}^{K} \mathcal{Q}(\mathbf{q^{l(*)}}, \xi^k). \tag{4.4.1}$$

Comparison with Stochastic Approach

In this part, we compare the performance of our proposed stochastic approach against the benchmark deterministic approach. We assume symmetrical deviations for each stochastic parameter and set them to be 80% of their nominal values. For instance, the renewable generation $w_{i,t}$ for each datacenter i at time period t can take values in the interval $[0.2\overline{w}_{i,t}, 1.8\overline{w}_{i,t}]$. Three sets of 1000 randomly sampled renewable generation $w_{i,t}$ scenarios are generated for each datacenter i at time period t, one following a uniform distribution in the interval $[0.2\overline{w}_{i,t}, 1.8\overline{w}_{i,t}]$ and the other two following normal distribution and Laplacian distribution with mean $\overline{w}_{i,t}$ and standard deviation $0.8\overline{w}_{i,t}/\sqrt{3}$. Samples from normal distribution falling outside of the region $[0.2\overline{w}_{i,t}, 1.8\overline{w}_{i,t}]$ are truncated. In the same way, we generate another two three sets of 1000 randomly sampled interactive workload demand $\overline{L}_{j,t}$ for each source j at time period t and real-time electricity price $\overline{p}^r_{i,t}$ for each datacenter i at time period t.

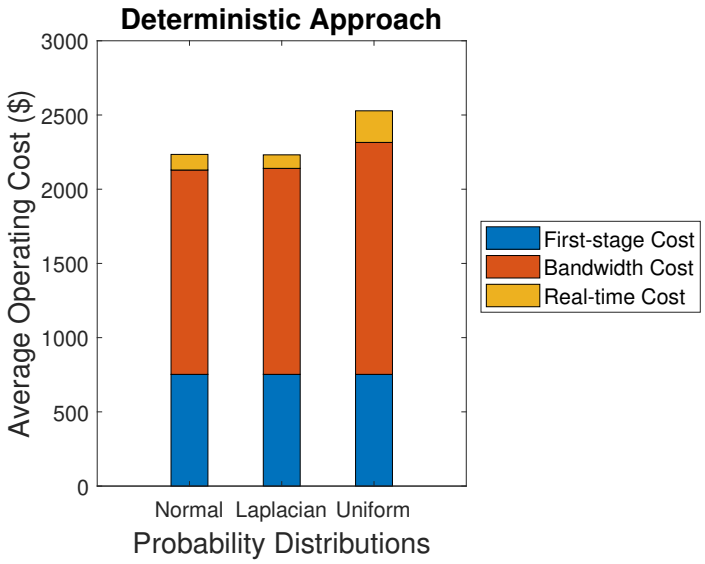
From Fig. 13, we observe that the Laplacian probability distribution renders the lowest operating cost among the chosen three probability distributions both for the deterministic approach and stochastic approach. Specific amount of each cost for both approaches, including first-stage cost, bandwidth cost, real-time cost and total cost, are summarized in Table 5. We observe that the first-stage cost is higher for deterministic approach since it behaves more conservative by only considering the expected values of stochastic parameters while for the stochastic approach, all the randomly sampled scenarios are taken into account. Moreover, the stochastic approach outperforms deterministic approach by 2.74% on average in terms of optimality by reducing the average total operating cost.

Table 5: Comparison of datacenter operating cost with the stochastic approach and the deterministic approach under three different probability distributions

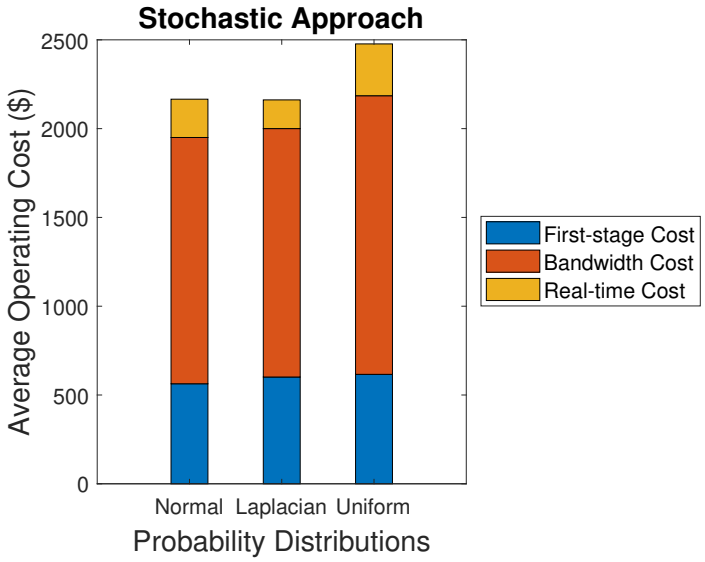
Approaches	Cost ($)	Normal	Laplacian	Uniform
Deterministic	First-stage	752.58	752.58	752.58
	Bandwidth	1376.73	1387.83	1562.66
	Real-time	105.18	91.13	212.94
	Total	2234.49	2231.54	2528.18
Stochastic	First-stage	563.32	601.31	616.27
	Bandwidth	1386.77	1398.94	1569.00
	Real-time	215.71	161.57	291.83
	Total	2165.80	2161.82	2477.10

4.5 Summary

In this chapter, we investigate the optimal energy procurement for datacenter operator who manages geographically distributed datacenters when participating in the multi-timescale electricity markets. A two-stage stochastic optimization problem is developed with the objective to minimize the total operating cost spanning the whole operating period considering the uncertainties of renewable generation, interactive workload demand and real-time electricity prices. Furthermore, we reformulate the two-stage stochastic optimization problem into its deterministic equivalent problem and perform Monte Carlo simulations to efficiently solve it. Finally, extensive simulations are conducted based on real-world traces to verify



(a) Deterministic Approach



(b) Stochastic Approach

Figure 13: Average operating cost of deterministic and stochastic approach under normal, Laplacian and uniform probability distributions.

the correctness of our proposed stochastic approach. The deterministic approach is adopted to contrast with our proposed stochastic approach in terms of optimality by reducing the total operating cost. Under three different probability distributions, the stochastic approach outperforms the deterministic approach by 2.74% on average.

CHAPTER V

JOINT TASK OFFLOADING AND RESOURCE ALLOCATION IN UAV-ENABLED MOBILE EDGE COMPUTING

MEC is an emerging technology to support resource-intensive yet delay-sensitive applications using small cloud-computing platforms deployed at the mobile network edges. However, the existing MEC techniques are not applicable to the situation where the number of mobile users increases explosively or the network facilities are sparely distributed. In view of this insufficiency, UAVs have been employed to improve the connectivity of ground IoT devices due to their high altitude. In this chapter, we propose an innovative UAV-enabled MEC system involving the interactions among IoT devices, UAV and edge clouds (ECs). The system deploys and operates a UAV properly to facilitate the MEC service provisioning to a set of IoT devices in regions where existing ECs cannot be accessible to IoT devices due to terrestrial signal blockage or shadowing. The UAV and ECs in the system collaboratively provide MEC services to the IoT devices. For optimal service provisioning in this system, we formulate an optimization problem aiming at minimizing the weighted sum of the service delay of all IoT devices and UAV energy consumption by jointly optimizing UAV position, communication and computing resource allocation, and task splitting decisions. However, the resulting optimization problem is highly non-convex and thus difficult to solve optimally. To tackle this problem, we develop an efficient algorithm based on successive convex approximation to obtain sub-optimal solutions. Numerical experiments demonstrate that our proposed collaborative UAV-EC offloading scheme largely outperforms baseline schemes that solely rely on UAV or edge clouds for MEC in IoT.

The rest of this chapter is organized as follows. In Section 5.1, we describe the system model and then formulate the optimal IoT task offloading processes as a non-convex optimization problem. In Section 5.2, we reformulate the original problem as an approximated convex optimization problem and then solve it by means of successive convex approximation. Simulation results based on real-world traces are presented in Section 5.3. Finally, the summary is given in Section 5.4.

5.1 System Model and Problem Formulation

In this section, we first introduce the system model for UAV-enabled MEC system. After that, we formulate an optimization problem to model the optimal UAV-enabled IoT task offloading process.

5.1.1 System Model

In this chapter, we consider the UAV-enabled MEC system as depicted in Fig. 14, which consists of a set of ground mobile users[1] (MUs) $i \in \mathcal{N} = \{1, 2, \ldots, N\}$, a UAV, and a set of ground ECs $j \in \mathcal{J} = \{1, 2, \ldots, J\}$. The UAV is deployed to facilitate the MEC service provisioning for ground MUs who cannot establish wireless communication with nearby cellular base stations or WiFi access points due to signal blockage and shadowing. In this scenario, ground-to-air (G2A) uplink communication is from MUs to the UAV while air-to-ground (A2G) downlink communication is from the UAV to ECs, which form a 3D wireless communication network.

We assume the UAV is equipped with certain CCS resources but subject to the size, weight, and power (SWAP) limitations. The ECs are composed of ground MEC servers co-located with cellular base stations or WiFi access points that have more CCS resources compared to the UAV and MUs. Each MU i has periodical computation-intensive tasks to perform, which are modeled as a triplet $\mathcal{W}_i = \langle L_i, C_i, \lambda_i \rangle$, where L_i (in bits) denotes the

[1]Note that we use mobile users and IoT devices interchangeably in this chapter.

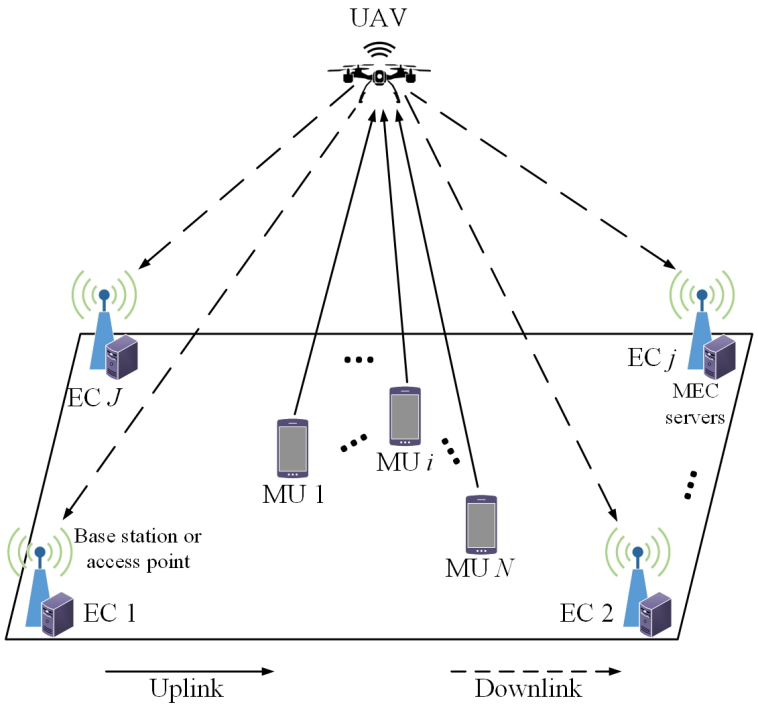


Figure 14: Illustration of an exemplary UAV-enabled MEC system with N MUs, J ECs, and a UAV.

input data size for processing the task, C_i (in CPU cycles/bit) denotes the number of CPU cycles required to process 1-bit of task data and λ_i (in unit of #task per second) denotes the arrival rate of tasks.

In this chapter, we use the 3D Cartesian coordinate system to represent the locations of MUs, the UAV and ECs. The position of the UAV is denoted by $Q^{\mathrm{UAV}} = (x^{\mathrm{UAV}}, y^{\mathrm{UAV}}, H)$, where the height H is assumed to be fixed while the horizontal coordinates x^{UAV} and y^{UAV} affect the channel gain during data communication processes and need to be optimized in our problem. Besides, we assume the positions of MU i and EC j are fixed in our model, which are denoted as $Q_i^{\mathrm{MU}} = (x_i^{\mathrm{MU}}, y_i^{\mathrm{MU}}, 0)$ and $Q_j^{\mathrm{EC}} = (x_j^{\mathrm{EC}}, y_j^{\mathrm{EC}}, 0)$, respectively.

Communication Model

In the UAV-enabled network, the LoS links are much more dominant than other channel impairments such as shadowing or small-scale fading due to the high altitude of the UAV. Therefore, the uplink channel gain from MU i to the UAV can be described by the free-space

path loss model:

$$h_i^{\mathrm{UL}} \triangleq \alpha_0 (d_i^{\mathrm{UL}})^{-2} = \frac{\alpha_0}{\left\| Q_i^{\mathrm{MU}} - Q^{\mathrm{UAV}} \right\|^2}, \tag{5.1.1}$$

where α_0 represents the received power at the reference distance of 1 m for a transmission power of 1 W, d_i^{UL} denotes the uplink distance from MU i to the UAV, and $\|\cdot\|$ denotes the Euclidean norm of a vector. Similarly, the downlink channel gain from the UAV to EC j can be described as

$$h_j^{\mathrm{DL}} \triangleq \alpha_0 (d_j^{\mathrm{DL}})^{-2} = \frac{\alpha_0}{\left\| Q^{\mathrm{UAV}} - Q_j^{\mathrm{EC}} \right\|^2}, \tag{5.1.2}$$

where d_j^{DL} denotes the downlink distance from the UAV to the EC j.

We assume the FDMA protocol for bandwidth sharing among MUs during the task offloading process. According to Shannon's capacity, the achievable uplink transmission data rate (in bps) from MU i to the UAV can be expressed as

$$R_i^{\mathrm{UL}} = B_i^{\mathrm{UL}} \log_2 \left(1 + \frac{h_i^{\mathrm{UL}} P_i^{\mathrm{MU}}}{\sigma^2} \right), \tag{5.1.3}$$

where B_i^{UL}, P_i^{MU} and σ^2 represent the assigned bandwidth to MU i, transmit power of MU i and the noise power at the UAV, respectively. For simplicity, we assume the noise power is the same at UAV and ECs [131]. However, it can be easily extended to the case when they are different. Similarly, the downlink transmission data rate (in bps) from the UAV to EC j can be computed as

$$R_j^{\mathrm{DL}} = B_j^{\mathrm{DL}} \log_2 \left(1 + \frac{h_j^{\mathrm{DL}} P_{\mathrm{TX}}^{\mathrm{UAV}}}{\sigma^2} \right), \tag{5.1.4}$$

where B_j^{DL} and $P_{\mathrm{TX}}^{\mathrm{UAV}}$ represent the per-device bandwidth[2] preassigned to EC j and transmit power of the UAV, respectively.

[2]We assume that each MU is assigned a certain bandwidth beforehand when they communicate with ECs via the UAV.

Delay Analysis

In our model, we assume that MUs do not perform local computing due to their limited computation capacities. In contrast, tasks will be first offloaded to the UAV, and then the UAV will determine the portion of tasks that are processed locally or further offloaded to ECs on the ground. Note that the decision time to split a task is very short compared to the entire communication and computation latency, and therefore can be neglected. Besides, the output data size of the computation results is often very small compared to the input data size in many computation-intensive applications such as face recognition and video analysis. Thus, the time needed to send the computation results back to MUs can be ignored as well.

In what follows, we will describe the four key components of the total delay for the offloading process: 1) G2A uplink transmission delay from MUs to the UAV; 2) computation delay at the UAV; 3) A2G downlink transmission delay from the UAV to the ECs; 4) computation delay at the ECs.

G2A uplink transmission delay from MUs to the UAV: As mentioned before, all the tasks will be offloaded to the UAV first via G2A links without any local computation. Therefore, the G2A transmission delay from MU i to the UAV is computed as the ratio of task input data size and the associated uplink transmission data rate:

$$t_i^{\mathrm{G2A}} = \frac{L_i}{R_i^{\mathrm{UL}}}. \tag{5.1.5}$$

Computation delay at the UAV: The UAV will decide the portion of the received tasks that will be processed locally at the UAV or further offloaded to the ground ECs for processing. Denote $\{\beta_{ij} \in [0,1], i \in \mathcal{N}, j \in \mathcal{J}\}$ and $\{\beta_{i0} \in [0,1], i \in \mathcal{N}\}$ as the portion of received tasks from MU i to be processed at EC j and the UAV, respectively. Then the computation delay at the UAV side to process the offloaded tasks from MU i can be calculated as

$$t_i^{\mathrm{UAV}} = \frac{\beta_{i0} L_i C_i}{f_i^{\mathrm{UAV}}}, \tag{5.1.6}$$

where f_i^{UAV} (in CPU cycles/s) is the computation resource that the UAV allocates to MU i. Note that β_{i0} equals 0 means that no computation will be executed at the UAV side while β_{i0} equals 1 indicates that no further offloading will occur from the UAV to ECs.

A2G downlink transmission delay from the UAV to ECs: The UAV may further offload the tasks to more powerful ECs on the ground to reduce the computation latency. Then, the A2G transmission delay from the MU i to EC j via UAV is described as the ratio of offloaded task input data size and the associated downlink transmission data rate:

$$t_{ij}^{\mathrm{A2G}} = \frac{\beta_{ij} L_i}{R_j^{\mathrm{DL}}}. \tag{5.1.7}$$

Computation delay at ECs: After receiving the offloaded task data from the UAV, ECs can start the computation process. Therefore, the computation delay at the EC side to process the offloaded task from the MU i to EC j via UAV is

$$t_{ij}^{\mathrm{EC}} = \frac{\beta_{ij} L_i C_i}{f_{ij}^{\mathrm{EC}}}, \tag{5.1.8}$$

where f_{ij}^{EC} (in CPU cycles/s) is the computation resource that EC j allocates to MU i.

UAV Energy Consumption Analysis

To ensure service availability, it is important to manage the energy consumption of the UAV due to its limited battery size. In this chapter, we focus on computation and transmission energy consumption of UAV, and ignore the hovering power since it is independent of our decisions.

Computation Energy Consumption: Similar to [105], we model the power consumption of the CPU in UAV as $\kappa(f_i^{\mathrm{UAV}})^3$, where κ denotes the effective switched capacitance depending on the CPU architecture. It follows that the corresponding energy consumption of UAV when processing tasks offloaded from MU i is given by the product of the power

level and computation time:

$$E_i^{\mathrm{CP}} = \kappa (f_i^{\mathrm{UAV}})^3 t_i^{\mathrm{UAV}} = \kappa \beta_{i0} L_i C_i (f_i^{\mathrm{UAV}})^2. \tag{5.1.9}$$

Transmission Energy Consumption: The transmission energy consumption of the UAV when receiving the task input data via the G2A uplink transmission channels from MU i is given by

$$E_i^{\mathrm{RX}} = P_{\mathrm{RX}}^{\mathrm{UAV}} t_i^{\mathrm{G2A}} = \frac{L_i P_{\mathrm{RX}}^{\mathrm{UAV}}}{R_i^{\mathrm{UL}}}, \tag{5.1.10}$$

where $P_{\mathrm{RX}}^{\mathrm{UAV}}$ is the receiving power of UAV. Besides, the transmission energy consumption of the UAV when offloading the task input data of MU i via the A2G downlink transmission channels to EC j is given by

$$E_{ij}^{\mathrm{TX}} = P_{\mathrm{TX}}^{\mathrm{UAV}} t_{ij}^{\mathrm{A2G}} = \frac{\beta_{ij} L_i P_{\mathrm{TX}}^{\mathrm{UAV}}}{R_j^{\mathrm{DL}}}. \tag{5.1.11}$$

Therefore, the total energy consumption of the UAV when serving the task offloading and computation of MU i is given by

$$E_i^{\mathrm{UAV}} = \lambda_i (E_i^{\mathrm{CP}} + E_i^{\mathrm{RX}} + \sum_{j \in \mathcal{J}} E_{ij}^{\mathrm{TX}}). \tag{5.1.12}$$

5.1.2 Problem Formulation

In this chapter, we are interested in minimizing the total energy consumption of the UAV when serving the computation and communication needs of the MUs and the total service delay of all MUs. To define the service delay of each MU, we make the following assumptions: (i) the UAV cannot partition a task until receiving its entire input data to ensure the accuracy of task splitting; (ii) the UAV and ECs cannot start the processing of tasks until the end of the transmission between MUs and the UAV or the UAV and ECs to ensure the reliability of the computation results; and (iii) the computation at the UAV can proceed simultaneously

with the transmission of the tasks to each EC since the communication and computation modules are often separated at the UAV. Based on the above assumptions, the service delay of MU i can be represented as

$$T_i = t_i^{\mathrm{G2A}} + \max_{j \in \mathcal{J}} \{t_i^{\mathrm{UAV}}, t_{ij}^{\mathrm{A2G}} + t_{ij}^{\mathrm{EC}}\}. \tag{5.1.13}$$

Our problem becomes jointly optimizing the UAV position Q^{UAV}, G2A uplink communication resource allocation B_i^{UL}, task partition variables β_{i0} and β_{ij} and computation resource allocation of the UAV f_i^{UAV} and ECs f_{ij}^{EC} with the goal of minimizing the weighted sum of total energy consumption of UAV and total service delay of all MUs. All the decision variables are continuous. It can be formulated as the following optimization problem:

$$\min_{\substack{Q^{\mathrm{UAV}}, B_i^{\mathrm{UL}}, \beta_{i0}, \\ \beta_{ij}, f_i^{\mathrm{UAV}}, f_{ij}^{\mathrm{EC}}}} \quad \sum_{i \in \mathcal{N}} E_i^{\mathrm{UAV}} + \rho \sum_{i \in \mathcal{N}} T_i \tag{5.1.14a}$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{N}} B_i^{\mathrm{UL}} \leq B^{\mathrm{UL}} \tag{5.1.14b}$$

$$\beta_{i0} + \sum_{j \in \mathcal{J}} \beta_{ij} = 1, \ \forall i \tag{5.1.14c}$$

$$\sum_{i \in \mathcal{N}} f_i^{\mathrm{UAV}} \leq F^{\mathrm{UAV}} \tag{5.1.14d}$$

$$\sum_{i \in \mathcal{N}} f_{ij}^{\mathrm{EC}} \leq F_j^{\mathrm{EC}}, \ \forall j \tag{5.1.14e}$$

$$0 \leq \beta_{ij} \leq 1, \ \forall i, j \tag{5.1.14f}$$

$$0 \leq \beta_{i0} \leq 1, \ \forall i \tag{5.1.14g}$$

$$B_i^{\mathrm{UL}}, f_i^{\mathrm{UAV}} \geq 0, \ \forall i \tag{5.1.14h}$$

$$f_{ij}^{\mathrm{EC}} \geq 0, \ \forall i, j \tag{5.1.14i}$$

where $\rho > 0$ is a parameter defining the relative weight of energy and delay, (5.1.14b), (5.1.14d), (5.1.14e), (5.1.14h) and (5.1.14i) ensure that the allocated resources for uplink

bandwidth, UAV and EC CPU frequencies are non-negative and no more than their limits while (5.1.14c), (5.1.14f) and (5.1.14g) constrain that the offloading tasks of MUs are completely processed by UAV and ECs, and the values of partition variables are between 0 and 1.

5.2 Solution Methodology

Problem (5.1.14) is hard to solve due to the non-convexity of the objective function and constraints. In what follows, we will first linearize the maximum term in (5.1.13) by leveraging auxiliary variables and reformulate the original optimization problem into a tractable one. Then, we develop a SCA-based algorithm to transform the non-convex objective function and constraints into suitable convex approximants to iteratively solve the resulting optimization problem.

5.2.1 Problem Reformulation

We first define an auxiliary variable for each MU i as $z_i \triangleq \max_{j\in\mathcal{J}}\{t_i^{\mathrm{UAV}}, t_{ij}^{\mathrm{A2G}} + t_{ij}^{\mathrm{EC}}\}$. Then, we linearize the service delay term in (5.1.14a) using z_i and reformulate the original optimization problem into the following:

$$\min_{\substack{z_i, Q^{\mathrm{UAV}}, B_i^{\mathrm{UL}}, \\ \beta_{i0}, \beta_{ij}, f_i^{\mathrm{UAV}}, f_{ij}^{\mathrm{EC}}}} \quad \sum_{i\in\mathcal{N}} E_i^{\mathrm{UAV}} + \rho \sum_{i\in\mathcal{N}} (t_i^{\mathrm{G2A}} + z_i) \tag{5.2.1a}$$

$$\text{s.t.} \quad z_i \geq t_i^{\mathrm{UAV}}, \ \forall i \tag{5.2.1b}$$

$$z_i \geq t_{ij}^{\mathrm{A2G}} + t_{ij}^{\mathrm{EC}}, \ \forall i, j \tag{5.2.1c}$$

$$(5.1.14\mathrm{b}) - (5.1.14\mathrm{i}) \tag{5.2.1d}$$

However, the reformulated optimization problem is still difficult to solve due to the non-convex objective function (5.2.1a) and non-convex constraints (5.2.1b) and (5.2.1c). Note that both the uplink and downlink transmission data rate functions (5.1.3) and (5.1.4) are

non-convex with respect to the UAV position Q^{UAV}.

5.2.2 Successive Convex Approximation

In this subsection, we will show how to build the convex approximation for the non-convex objective function and non-convex constraints in the reformulated problem (5.2.1) while preserving the local first-order behavior of the original non-convex problem and solve the resulting problem iteratively to obtain sub-optimal solutions by means of SCA. Before we develop the SCA-based algorithm, we first present the background of SCA.

Background of SCA

Consider the following optimization problem:

$$\mathcal{P}: \min_{\boldsymbol{x}} \quad U(\boldsymbol{x}) \tag{5.2.2a}$$

$$\text{s.t.} \quad g_l(\boldsymbol{x}) \leq 0, \ \forall l = 1, \ldots, m \tag{5.2.2b}$$

$$\boldsymbol{x} \in \mathcal{K}, \tag{5.2.2c}$$

where the objective function $U : \mathcal{K} \to \mathbb{R}$ is smooth (possibly non-convex) and $g_l : \mathcal{K} \to \mathbb{R}$ is smooth (possibly non-convex), for all $l = 1, \ldots, m$; the feasible set is denoted as $\mathcal{X}$. A widely used method for solving this specific problem is SCA (also known as majorization minimization) where at each iteration, a convex approximation of original problem is solved via replacing the non-convex objective function and constraints by suitable convex approximants. The convex approximation of original problem can be stated as follows: given $\boldsymbol{x}^k \in \mathcal{X}$,

$$\mathcal{P}_{\boldsymbol{x}^k}: \min_{\boldsymbol{x}} \quad \tilde{U}(\boldsymbol{x}; \boldsymbol{x}^k) \tag{5.2.3a}$$

$$\text{s.t.} \quad \tilde{g}_l(\boldsymbol{x}; \boldsymbol{x}^k) \leq 0, \ \forall l = 1, \ldots, m \tag{5.2.3b}$$

$$\boldsymbol{x} \in \mathcal{K}, \tag{5.2.3c}$$

where $\tilde{U}(\boldsymbol{x};\boldsymbol{x}^k)$ and $\tilde{g}_l(\boldsymbol{x};\boldsymbol{x}^k)$ represent the approximants of $U(\boldsymbol{x})$ and $g_l(\boldsymbol{x})$ at current iterate $\boldsymbol{x}^k$, respectively; the feasible set is denoted as $\mathcal{X}(\boldsymbol{x}^k)$. More specifically, we consider the SCA method presented in Algorithm 1. It is assumed that at each iteration, some original functions $U(\boldsymbol{x})$ and $g_l(\boldsymbol{x})$ are approximated by their upper bounds where the same first-order behavior is preserved [85]. It is well known that a stationary point of the problem $\mathcal{P}$ is also local minimum [91].

Algorithm 1 SCA algorithm for problem $\mathcal{P}$

Find a feasible solution $\boldsymbol{x} \in \mathcal{X}$ in $\mathcal{P}$, choose a step size $\theta \in (0, 1]$ and set $k = 0$.
Repeat

1. Compute $\hat{\boldsymbol{x}}(\boldsymbol{x}^k)$, the solution of $\mathcal{P}_{\boldsymbol{x}^k}$;
2. Set $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \theta(\hat{\boldsymbol{x}}(\boldsymbol{x}^k) - \boldsymbol{x}^k)$;
3. Set $k \leftarrow k + 1$.

Until some convergence criterion is met.

SCA-based algorithm

The authors in [91] proposes a framework that unifies several existing SCA-based algorithms to solve the problem $\mathcal{P}$ in a parallel and distributed fashion. It also offers much flexibility in the choice of the convex approximation functions, and the objective function U need not be an upper bound of itself at any feasible point. Multiple examples are summarized to find the candidate approximants $\tilde{g}_l(\boldsymbol{x})$ and $\tilde{U}(\boldsymbol{x})$ while necessary assumptions are satisfied to develop the SCA-based algorithm. We first present the assumptions and examples that we will utilize to approximate the non-convex terms in our problem as follows:

Assumption 1: The key assumptions on the choice of the approximated function $\tilde{g}_l : \mathcal{K} \times \mathcal{X} \to \mathbb{R}$ are given as follows:

A1) $\tilde{g}_l(\bullet;\boldsymbol{y})$ is convex on $\mathcal{K}$ for all $\boldsymbol{y} \in \mathcal{X}$;

A2) *Upper-bound*: $g_l(\boldsymbol{x}) \leq \tilde{g}_l(\boldsymbol{x};\boldsymbol{y}),\ \forall \boldsymbol{x} \in \mathcal{K}, \boldsymbol{y} \in \mathcal{X}$;

A3) *Function value consistency*: $\tilde{g}_l(\boldsymbol{y};\boldsymbol{y}) = g_l(\boldsymbol{y})$, for all $\boldsymbol{y} \in \mathcal{X}$;

A4) $\tilde{g}_l(\bullet;\bullet)$ is continuous on $\mathcal{K} \times \mathcal{X}$;

A5) $\nabla_{\boldsymbol{x}} \tilde{g}_l(\bullet;\bullet)$ is continuous on $\mathcal{K} \times \mathcal{X}$;

A6) *Gradient consistency*: $\nabla_{\boldsymbol{x}} \tilde{g}_l(\boldsymbol{y};\boldsymbol{y}) = \nabla_{\boldsymbol{x}} g_l(\boldsymbol{y})$, for all $\boldsymbol{y} \in \mathcal{X}$;

where $\nabla_{\boldsymbol{x}} \tilde{g}_l(\boldsymbol{y};\boldsymbol{y})$ denotes the partial gradient of function $\tilde{g}_l$ with respect to the argument $\boldsymbol{x}$ evaluated at $(\boldsymbol{y};\boldsymbol{y})$.

Assumption 2: The key assumptions on the choice of the approximated function $\tilde{U}$: $\mathcal{K} \times \mathcal{X} \to \mathbb{R}$ are given as follows:

B1) $\tilde{U}(\bullet;\boldsymbol{y})$ is uniformly strongly convex on $\mathcal{K}$ with constant $\mu > 0$, i.e., for all $\boldsymbol{x}, \boldsymbol{z} \in \mathcal{K}$ and $\boldsymbol{y} \in \mathcal{X}$:

$$(\boldsymbol{x} - \boldsymbol{z})^\top (\nabla_{\boldsymbol{x}} \tilde{U}(\boldsymbol{x};\boldsymbol{y}) - \nabla_{\boldsymbol{x}} \tilde{U}(\boldsymbol{z};\boldsymbol{y})) \geq \mu \|\boldsymbol{x} - \boldsymbol{z}\|^2 .$$

B2) *Gradient consistency*: $\nabla_{\boldsymbol{x}} \tilde{U}(\boldsymbol{y};\boldsymbol{y}) = \nabla_{\boldsymbol{x}} U(\boldsymbol{y})$, for all $\boldsymbol{y} \in \mathcal{X}$;

B3) $\nabla_{\boldsymbol{x}} \tilde{U}(\bullet;\bullet)$ is continuous on $\mathcal{K} \times \mathcal{X}$;

where $\nabla_{\boldsymbol{x}} \tilde{U}(\boldsymbol{u};\boldsymbol{v})$ denotes the partial gradient of function $\tilde{U}$ with respect to the argument $\boldsymbol{x}$ evaluated at $(\boldsymbol{u};\boldsymbol{v})$. Note that A1) and B1) make the problem $\mathcal{P}_{\boldsymbol{x}^k}$ strongly convex while A2) and A3) guarantee the iterate feasibility that $\boldsymbol{x}^k \in \mathcal{X}(\boldsymbol{x}^k) \subseteq \mathcal{X}$.

Example 1–Approximation of $g_l(\boldsymbol{x})$: (Example 3 in [91]) Suppose that g_l has a Difference of Convex (DC) structure, i.e., $g_l(\boldsymbol{x}) = g_j^+(\boldsymbol{x}) - g_j^-(\boldsymbol{x})$ with both g_l^+ and g_l^- being convex and continuously differentiable. By linearizing the concave part g_l^-, we obtain the convex upper approximation of g_l as follows: for all $\boldsymbol{x} \in \mathcal{K}$ and $\boldsymbol{y} \in \mathcal{X}$,

$$\tilde{g}_l(\boldsymbol{x};\boldsymbol{y}) \triangleq g_l^+(\boldsymbol{x}) - g_l^-(\boldsymbol{y}) - \nabla_{\boldsymbol{x}} g_l^-(\boldsymbol{y})^\top (\boldsymbol{x} - \boldsymbol{y}) \geq g_j(\boldsymbol{x}). \tag{5.2.4}$$

Example 2–Approximation of $g_l(\boldsymbol{x})$: (Example 4 in [91]) Suppose that $g_l(\boldsymbol{x})$ has a Product of Functions (PF) structure, i.e., $g_l(\boldsymbol{x}) = f_1(\boldsymbol{x}) f_2(\boldsymbol{x})$ with both f_1 and f_2 being convex and

non-negative. Observe that $g_l(\boldsymbol{x})$ can be rewritten as a function with DC structure:

$$g_l(\boldsymbol{x}) = \frac{1}{2}(f_1(\boldsymbol{x}) + f_2(\boldsymbol{x}))^2 - \frac{1}{2}(f_1^2(\boldsymbol{x}) + f_2^2(\boldsymbol{x})). \tag{5.2.5}$$

Then, the convex upper approximation of g_l can be obtained by linearizing the concave part in (5.2.5): for any $\boldsymbol{y} \in \mathcal{X}$,

$$\begin{aligned}\tilde{g}_l(\boldsymbol{x};\boldsymbol{y}) \triangleq\ & \frac{1}{2}(f_1(\boldsymbol{x}) + f_2(\boldsymbol{x}))^2 - \frac{1}{2}(f_1^2(\boldsymbol{y}) + f_2^2(\boldsymbol{y})) \\ & - f_1(\boldsymbol{y})f_1'(\boldsymbol{y})(\boldsymbol{x}-\boldsymbol{y}) - f_2(\boldsymbol{y})f_2'(\boldsymbol{y})(\boldsymbol{x}-\boldsymbol{y}) \geq g_l(\boldsymbol{x}).\end{aligned} \tag{5.2.6}$$

Example 3–Approximation of $U(\boldsymbol{x})$: (Example 8 in [91]) Suppose that $U(\boldsymbol{x})$ has a PF structure, i.e., $U(\boldsymbol{x}) = h_1(\boldsymbol{x})h_2(\boldsymbol{x})$ with both h_1 and h_2 being convex and non-negative. For any $\boldsymbol{y} \in \mathcal{X}$, a convex approximation of $U(\boldsymbol{x})$ is given by

$$\begin{aligned}\tilde{U}(\boldsymbol{x};\boldsymbol{y}) =\ & h_1(\boldsymbol{x})h_2(\boldsymbol{y}) + h_1(\boldsymbol{y})h_2(\boldsymbol{x}) \\ & + \frac{\tau}{2}(\boldsymbol{x}-\boldsymbol{y})^\top \boldsymbol{H}(\boldsymbol{y})(\boldsymbol{x}-\boldsymbol{y}),\end{aligned} \tag{5.2.7}$$

where $\tau > 0$ is a positive constant, and $\boldsymbol{H}(\boldsymbol{y})$ is a uniformly positive definite matrix.

Then, we transform the non-convex constraints and non-convex objective function in the reformulated problem (5.2.1) into suitable approximants by following the above examples. For constraint (5.2.1b), we observe that the non-convex term t_i^{UAV} can be written as the product of convex and non-negative functions[3]

$$t_i^{\mathrm{UAV}} = L_i C_i g_l(\beta_{i0}, f_i^{\mathrm{UAV}}) = L_i C_i f_1(\beta_{i0}) f_2(f_i^{\mathrm{UAV}}), \tag{5.2.8}$$

where $f_1(\beta_{i0}) = \beta_{i0}$ and $f_2(f_i^{\mathrm{UAV}}) = 1/f_i^{\mathrm{UAV}}$. Then, given a feasible solution $\beta_{i0}(k)$ and $f_i^{\mathrm{UAV}}(k)$ for the kth iteration of SCA-based algorithm, we derive a convex upper approxima-

[3]Without loss of generality, we factorize the constants (L_i, C_i, etc.) out of the term since they will not affect the convexity.

tion of t_i^{UAV} by using Example 2 as

$$
\begin{aligned}
&t_i^{\mathrm{UAV}} \leq \tilde{t}_i^{\mathrm{UAV}}(\beta_{i0}, f_i^{\mathrm{UAV}}; \beta_{i0}(k), f_i^{\mathrm{UAV}}(k)) \triangleq \\
&L_i C_i \left[\frac{1}{2} \left(\left(\beta_{i0} + \frac{1}{f_i^{\mathrm{UAV}}} \right)^2 - (\beta_{i0}(k))^2 - \left(\frac{1}{f_i^{\mathrm{UAV}}(k)} \right)^2 \right) - \right. \\
&\left. (\beta_{i0}(k)(\beta_{i0} - \beta_{i0}(k))) + \left(\frac{1}{f_i^{\mathrm{UAV}}(k)} \right)^3 \left(\frac{1}{f_i^{\mathrm{UAV}}} - \frac{1}{f_i^{\mathrm{UAV}}(k)} \right) \right].
\end{aligned} \tag{5.2.9}
$$

For constraint (5.2.1c), t_{ij}^{A2G} can be written as the product of L_i, β_{ij} and $1/R_j^{\mathrm{DL}}$. However, $1/R_j^{\mathrm{DL}}$ is a non-convex function with respect to the UAV location Q^{UAV}, and therefore Example 2 cannot be directly applied to derive a convex upper approximation. To tackle the non-convexity, we replace it by non-negative auxiliary variables $\{\phi_j\}_{j \in \mathcal{J}}$. Then, the non-convex term t_{ij}^{A2G} can be written as the product of convex and non-negative functions

$$
t_{ij}^{\mathrm{A2G}} = L_i g_l(\beta_{ij}, \phi_j) = L_i f_1(\beta_{ij}) f_2(\phi_j), \tag{5.2.10}
$$

where $f_1(\beta_{ij}) = \beta_{ij}$ and $f_2(\phi_j) = 1/\phi_j$ in (5.2.10). Similarly, the non-convex term t_{ij}^{EC} in (5.2.1c) can be written as the product of convex and non-negative functions

$$
t_{ij}^{\mathrm{EC}} = L_i C_i g_l(\beta_{ij}, f_{ij}^{\mathrm{EC}}) = L_i C_i f_1(\beta_{ij}) f_2(f_{ij}^{\mathrm{EC}}), \tag{5.2.11}
$$

where $f_1(\beta_{ij}) = \beta_{ij}$ and $f_2(f_{ij}^{\mathrm{EC}}) = 1/f_{ij}^{\mathrm{EC}}$ in (5.2.11). Then, given a feasible solution $\beta_{ij}(k)$, $\phi_j(k)$ and $f_{ij}^{\mathrm{EC}}(k)$ for the kth iteration of SCA-based algorithm, we derive convex upper approximation of t_{ij}^{A2G} and t_{ij}^{EC} by using Example 2 as

$$
\begin{aligned}
&t_{ij}^{\mathrm{A2G}} \leq \tilde{t}_{ij}^{\mathrm{A2G}}(\beta_{ij}, \phi_j; \beta_{ij}(k), \phi_j(k)) \triangleq \\
&L_i \left[\frac{1}{2} \left(\left(\beta_{ij} + \frac{1}{\phi_j} \right)^2 - (\beta_{ij}(k))^2 - \left(\frac{1}{\phi_j(k)} \right)^2 \right) \right. \\
&\left. - (\beta_{ij}(k)(\beta_{ij} - \beta_{ij}(k))) + \left(\frac{1}{\phi_j(k)} \right)^3 \left(\frac{1}{\phi_j} - \frac{1}{\phi_j(k)} \right) \right],
\end{aligned} \tag{5.2.12}
$$

and

$$t_{ij}^{\mathrm{EC}} \leq \tilde{t}_{ij}^{\mathrm{EC}}(\beta_{ij}, f_{ij}^{\mathrm{EC}}; \beta_{ij}(k), f_{ij}^{\mathrm{EC}}(k)) \triangleq$$
$$L_i C_i \left[\frac{1}{2} \left(\left(\beta_{ij} + \frac{1}{f_{ij}^{\mathrm{EC}}} \right)^2 - (\beta_{ij}(k))^2 - \left(\frac{1}{f_{ij}^{\mathrm{EC}}(k)} \right)^2 \right) - \right.$$
$$\left. (\beta_{ij}(k)(\beta_{ij} - \beta_{ij}(k))) + \left(\frac{1}{f_{ij}^{\mathrm{EC}}(k)} \right)^3 \left(\frac{1}{f_{ij}^{\mathrm{EC}}} - \frac{1}{f_{ij}^{\mathrm{EC}}(k)} \right) \right]. \tag{5.2.13}$$

By defining $\overline{R}_i^{\mathrm{UL}} \triangleq \log_2(1 + \frac{h_i^{\mathrm{UL}} P_i^{\mathrm{MU}}}{\sigma^2})$, we replace $1/\overline{R}_i^{\mathrm{UL}}$ in t_i^{G2A} by non-negative auxiliary variables $\{\gamma_i\}_{i \in \mathcal{N}}$ since it is a non-convex function with respect to the UAV location Q^{UAV}. Then, the non-convex terms in objective function (5.2.1a) can be written as the product of convex and non-negative functions

$$E_i^{\mathrm{CP}} = \kappa L_i C_i h_1(\beta_{i0}) h_2(f_i^{\mathrm{UAV}}) \tag{5.2.14}$$
$$E_{ij}^{\mathrm{TX}} = L_i P_{\mathrm{TX}}^{\mathrm{UAV}} h_1(\beta_{ij}) h_3(\phi_j) \tag{5.2.15}$$
$$t_i^{\mathrm{G2A}} = L_i h_3(B_i^{\mathrm{UL}}) h_3(\gamma_i), \tag{5.2.16}$$
$$E_i^{\mathrm{RX}} = P_{\mathrm{RX}}^{\mathrm{UAV}} t_i^{\mathrm{G2A}} = P_{\mathrm{RX}}^{\mathrm{UAV}} L_i h_3(B_i^{\mathrm{UL}}) h_3(\gamma_i), \tag{5.2.17}$$

where $h_1(\beta_{i0}) = \beta_{i0}$ and $h_2(f_i^{\mathrm{UAV}}) = (f_i^{\mathrm{UAV}})^2$ in (5.2.14), $h_1(\beta_{ij}) = \beta_{ij}$ and $h_3(\phi_j) = 1/\phi_j$ in (5.2.15), while $h_3(B_i^{\mathrm{UL}}) = 1/B_i^{\mathrm{UL}}$ and $h_3(\gamma_i) = 1/\gamma_i$ in (5.2.16). Then, given a feasible solution $\beta_{i0}(k)$, $\beta_{ij}(k)$, $\phi_j(k)$, $\gamma_i(k)$, $B_i^{\mathrm{UL}}(k)$ and $f_i^{\mathrm{UAV}}(k)$ for the kth iteration of SCA-based algorithm, we derive convex approximation of E_i^{CP}, E_{ij}^{TX}, t_i^{G2A} and E_i^{RX} by using Example

3 as

$$\tilde{E}_i^{\mathrm{CP}}(\beta_{i0}, f_i^{\mathrm{UAV}}; \beta_{i0}(k), f_i^{\mathrm{UAV}}(k)) \triangleq$$
$$\kappa L_i C_i \left(\beta_{i0} (f_i^{\mathrm{UAV}}(k))^2 + \beta_{i0}(k)(f_i^{\mathrm{UAV}})^2 \right)$$
$$+ \frac{\tau_{\beta_{i0}}}{2}(\beta_{i0} - \beta_{i0}(k))^2 + \frac{\tau_{f_i^{\mathrm{UAV}}}}{2}(f_i^{\mathrm{UAV}} - f_i^{\mathrm{UAV}}(k))^2, \tag{5.2.18}$$
$$\tilde{E}_{ij}^{\mathrm{TX}}(\beta_{ij}, \phi_j; \beta_{ij}(k), \phi_j(k)) \triangleq L_i P_{\mathrm{TX}}^{\mathrm{UAV}} \left(\frac{\beta_{ij}}{\phi_j(k)} + \frac{\beta_{ij}(k)}{\phi_j} \right)$$
$$+ \frac{\tau_{\beta_{ij}}}{2}(\beta_{ij} - \beta_{ij}(k))^2 + \frac{\tau_{\phi_j}}{2}(\phi_j - \phi_j(k))^2, \tag{5.2.19}$$
$$\tilde{t}_i^{\mathrm{G2A}}(B_i^{\mathrm{UL}}, \gamma_i; B_i^{\mathrm{UL}}(k), \gamma_i(k)) \triangleq L_i \left(\frac{1}{B_i^{\mathrm{UL}}\gamma_i(k)} + \frac{1}{B_i^{\mathrm{UL}}(k)\gamma_i} \right)$$
$$+ \frac{\tau_{B_i^{\mathrm{UL}}}}{2}(B_i^{\mathrm{UL}} - B_i^{\mathrm{UL}}(k))^2 + \frac{\tau_{\gamma_i}}{2}(\gamma_i - \gamma_i(k))^2, \tag{5.2.20}$$

and

$$\tilde{E}_i^{\mathrm{RX}}(B_i^{\mathrm{UL}}, \gamma_i; B_i^{\mathrm{UL}}(k), \gamma_i(k)) \triangleq$$
$$P_{\mathrm{RX}}^{\mathrm{UAV}} \tilde{t}_i^{\mathrm{G2A}}(B_i^{\mathrm{UL}}, \gamma_i; B_i^{\mathrm{UL}}(k), \gamma_i(k)), \tag{5.2.21}$$

where $\tau_{\beta_{i0}}, \tau_{\beta_{ij}}, \tau_{\phi_j}, \tau_{\gamma_i}, \tau_{B_i^{\mathrm{UL}}}, \tau_{f_i^{\mathrm{UAV}}} > 0$. Therefore, the convex surrogate objective function of (5.2.1a) can be denoted as the non-negative weighted sum of convex functions

$$\sum_{i \in \mathcal{N}} \lambda_i (\tilde{E}_i^{\mathrm{CP}} + \tilde{E}_i^{\mathrm{RX}} + \sum_{j \in \mathcal{J}} \tilde{E}_{ij}^{\mathrm{TX}}) + \rho \sum_{i \in \mathcal{N}} (\tilde{t}_i^{\mathrm{G2A}} + z_i), \tag{5.2.22}$$

where the convexity is preserved.

Moreover, as we replace the non-convex data rate functions in both objective function and constraints by the auxiliary variables $\{\phi_j\}_{j \in \mathcal{J}}$ and $\{\gamma_i\}_{i \in \mathcal{N}}$, we obtain equality constraints $\{\phi_j\}_{j \in \mathcal{J}} = 1/R_j^{\mathrm{DL}}$ and $\{\gamma_i\}_{i \in \mathcal{N}} = 1/\overline{R}_i^{\mathrm{UL}}$. To further address the non-convexity, we first relax

them as the following inequalities

$$0 \leq \phi_j \leq R_j^{\mathrm{DL}},\ \forall j \tag{5.2.23}$$

$$0 \leq \gamma_i \leq \overline{R}_i^{\mathrm{UL}},\ \forall i \tag{5.2.24}$$

where the optimality is preserved since at optimal solutions the auxiliary variables will equate their upper bounds. The key observation is that in (5.2.23) and (5.2.24), although R_j^{DL} and $\overline{R}_i^{\mathrm{UL}}$ are not concave with respect to Q^{UAV}, they are convex functions with respect to $\left\|Q^{\mathrm{UAV}} - Q_j^{\mathrm{EC}}\right\|^2$ and $\left\|Q_i^{\mathrm{MU}} - Q^{\mathrm{UAV}}\right\|^2$, respectively. Recall that any convex function is globally lower-bounded by its first-order Taylor expansion at any point [17]. Therefore, by taking the first-order Taylor expansion of R_j^{DL} and $\overline{R}_i^{\mathrm{UL}}$ with respect to $\left\|Q^{\mathrm{UAV}} - Q_j^{\mathrm{EC}}\right\|^2$ and $\left\|Q_i^{\mathrm{MU}} - Q^{\mathrm{UAV}}\right\|^2$ respectively, we obtain lower bounds of R_j^{DL} and $\overline{R}_i^{\mathrm{UL}}$ at local point $Q^{\mathrm{UAV}}(k)$ for the kth iteration of SCA-based algorithm as follows:

$$\begin{aligned} R_j^{\mathrm{DL}} &\geq R_{j,LB}^{\mathrm{DL}}(Q^{\mathrm{UAV}}; Q^{\mathrm{UAV}}(k)) \triangleq R_j^{\mathrm{DL}}(Q^{\mathrm{UAV}}(k)) \\ &- \frac{B_j^{\mathrm{DL}}\eta(\left\|Q^{\mathrm{UAV}} - Q_j^{\mathrm{EC}}\right\|^2 - \left\|Q^{\mathrm{UAV}}(k) - Q_j^{\mathrm{EC}}\right\|^2)}{\ln 2(\left\|Q^{\mathrm{UAV}}(k) - Q_j^{\mathrm{EC}}\right\|^2)(\eta + \left\|Q^{\mathrm{UAV}}(k) - Q_j^{\mathrm{EC}}\right\|^2)},\ \forall j \end{aligned} \tag{5.2.25}$$

and

$$\begin{aligned} \overline{R}_i^{\mathrm{UL}} &\geq \overline{R}_{i,LB}^{\mathrm{UL}}(Q^{\mathrm{UAV}}; Q^{\mathrm{UAV}}(k)) \triangleq \overline{R}_i^{\mathrm{UL}}(Q^{\mathrm{UAV}}(k)) \\ &- \frac{\varepsilon_i(\left\|Q_i^{\mathrm{MU}} - Q^{\mathrm{UAV}}\right\|^2 - \left\|Q_i^{\mathrm{MU}} - Q^{\mathrm{UAV}}(k)\right\|^2)}{\ln 2(\|Q_i^{\mathrm{MU}} - Q^{\mathrm{UAV}}(k)\|^2)(\varepsilon_i + \|Q_i^{\mathrm{MU}} - Q^{\mathrm{UAV}}(k)\|^2)},\ \forall i, \end{aligned} \tag{5.2.26}$$

where $\eta \triangleq \alpha_0 P_{\mathrm{TX}}^{\mathrm{UAV}}/\sigma^2$ and $\varepsilon_i \triangleq \alpha_0 P_i^{\mathrm{MU}}/\sigma^2$. Note that both $R_{j,LB}^{\mathrm{DL}}$ and $\overline{R}_{i,LB}^{\mathrm{UL}}$ are concave functions with respect to Q^{UAV}. Then, by replacing R_j^{DL} and $\overline{R}_i^{\mathrm{UL}}$ with their lower bounds, we obtain the approximated convex constraints as

$$0 \leq \phi_j \leq R_{j,LB}^{\mathrm{DL}}(Q^{\mathrm{UAV}}; Q^{\mathrm{UAV}}(k)),\ \forall j \tag{5.2.27}$$

$$0 \leq \gamma_i \leq \overline{R}_{i,LB}^{\mathrm{UL}}(Q^{\mathrm{UAV}}; Q^{\mathrm{UAV}}(k)),\ \forall i. \tag{5.2.28}$$

Finally, we denote the set of decision variables for our optimization problem as $\boldsymbol{\psi} = (z_i, Q^{\mathrm{UAV}}, B_i^{\mathrm{UL}}, \beta_{i0}, \beta_{ij}, f_i^{\mathrm{UAV}}, f_{ij}^{\mathrm{EC}}, \phi_j, \gamma_i)$. The convex approximation of the reformulated problem (5.2.1) with a feasible solution $\boldsymbol{\psi}(k)$ for the kth iteration of SCA-based algorithm is given by

$$\min_{\boldsymbol{\psi}} \quad \sum_{i\in\mathcal{N}} \lambda_i \Bigg(\tilde{E}_i^{\mathrm{CP}}(\boldsymbol{\psi};\boldsymbol{\psi}(k)) + \tilde{E}_i^{\mathrm{RX}}(\boldsymbol{\psi};\boldsymbol{\psi}(k)) + \sum_{j\in\mathcal{J}} \tilde{E}_{ij}^{\mathrm{TX}}(\boldsymbol{\psi};\boldsymbol{\psi}(k)) \Bigg) + \rho \sum_{i\in\mathcal{N}} (\tilde{t}_i^{\mathrm{G2A}}(\boldsymbol{\psi};\boldsymbol{\psi}(k)) + z_i) \tag{5.2.29a}$$

$$\text{s.t.} \quad z_i \geq \tilde{t}_i^{\mathrm{UAV}}(\boldsymbol{\psi};\boldsymbol{\psi}(k)),\ \forall i \tag{5.2.29b}$$

$$z_i \geq \tilde{t}_{ij}^{\mathrm{A2G}}(\boldsymbol{\psi};\boldsymbol{\psi}(k)) + \tilde{t}_{ij}^{\mathrm{EC}}(\boldsymbol{\psi};\boldsymbol{\psi}(k)),\ \forall i,j \tag{5.2.29c}$$

$$0 \leq \phi_j \leq R_{j,LB}^{\mathrm{DL}}(\boldsymbol{\psi};\boldsymbol{\psi}(k)),\ \forall j \tag{5.2.29d}$$

$$0 \leq \gamma_i \leq \overline{R}_{i,LB}^{\mathrm{UL}}(\boldsymbol{\psi};\boldsymbol{\psi}(k)),\ \forall i \tag{5.2.29e}$$

$$(5.1.14\mathrm{b}) - (5.1.14\mathrm{i}), \tag{5.2.29f}$$

which has a unique solution denoted by $\hat{\boldsymbol{\psi}}(\boldsymbol{\psi}(k))$. The above optimization problem (5.2.29) is convex, and the SCA-based algorithm is summarized in Algorithm 2.

Algorithm 2 SCA-based algorithm for problem (5.2.29)

Input: $\boldsymbol{\psi}(0) = (z_i, Q^{\mathrm{UAV}}(0), B_i^{\mathrm{UL}}(0), \beta_{i0}(0), \beta_{ij}(0), f_i^{\mathrm{UAV}}(0), f_{ij}^{\mathrm{EC}}(0), \phi_j(0), \gamma_i(0))$, and $\tau_{\beta_{i0}}, \tau_{\beta_{ij}}, \tau_{\phi_j}, \tau_{\gamma_i}, \tau_{B_i^{\mathrm{UL}}}$, $\tau_{f_i^{\mathrm{UAV}}} > 0$ for $i \in \mathcal{N}$ and $j \in \mathcal{J}$, $\theta(k) \in (0,1]$. Set $k = 0$, $\alpha = 0.5$.

Repeat

1. Compute $\hat{\boldsymbol{\psi}}(\boldsymbol{\psi}(k))$, the solution of (5.2.29);
2. Set $\boldsymbol{\psi}(k+1) = \boldsymbol{\psi}(k) + \theta(k)(\hat{\boldsymbol{\psi}}(\boldsymbol{\psi}(k)) - \boldsymbol{\psi}(k))$, with $\theta(k) = \theta(k-1)(1 - \alpha\theta(k))$;
3. Set $k \leftarrow k+1$.

Until $\boldsymbol{\psi}(k)$ is a stationary solution of (5.1.14).

Output: Q^{UAV}, B_i^{UL}, β_{i0}, β_{ij}, f_i^{UAV} and f_{ij}^{EC}.

Note that a diminishing step-size rule is applied in step 2), which is numerically more efficient than a constant one. The convergence of Algorithm 1 is guaranteed if the step size

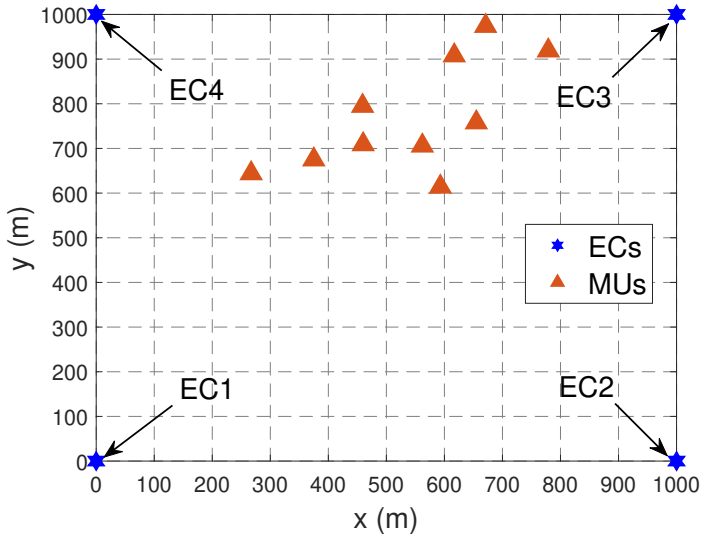


Figure 15: Locations of 10 MUs and 4 ECs in the MEC system.

$\theta(k)$ is chosen so that $\theta(k) \in (0, 1]$, $\theta(k) \to 0$ and $\sum_v \theta(k) = \infty$, then $\boldsymbol{\psi}(k)$ is bounded and at least one of its limit points is stationary [91]. For the termination criterion, it is very convenient to use $\left\|\hat{\boldsymbol{\psi}}(\boldsymbol{\psi}(k)) - \boldsymbol{\psi}(k)\right\|$, which is a measure of stationarity. Thus, a reliable termination rule is to check $\left\|\hat{\boldsymbol{\psi}}(\boldsymbol{\psi}(k)) - \boldsymbol{\psi}(k)\right\| \leq \zeta$, where ζ is the desired accuracy.

5.3 Numerical Experiments

In this section, we validate the effectiveness of our proposed SCA-based algorithm via extensive numerical experiments. All the experiments are implemented in MATLAB R2018a using CVX on a desktop computer with an Intel Core i7-4790 3.60 GHz CPU and 16 GB RAM. The convergence tolerance threshold ζ for the proposed algorithm is set to be 10^{-2}.

5.3.1 Simulation Setup

We consider a UAV-enabled MEC system with 4 ground ECs placed at each vertex and 10 ground MUs that are randomly distributed within a 2D area of 1000×1000 m^2, as illustrated in Fig. 15. The UAV is deployed and operated to facilitate the MEC service provisioning, and the optimal 3D location of UAV can be found using our proposed SCA-based algorithm. The simulation parameter settings are summarized in Table 6 unless otherwise stated.

Table 6: Simulation Parameters

Parameters	Values	Parameters	Values
H	100 m	P_{TX}^{UAV}	1 W
α_0	-50 dB	L_i	$[1, 5]$ Mbits
σ^2	-100 dBm	C_i	$[100, 200]$ CPU cycles/bit
κ	10^{-28} [129, 130]	B^{UL}	10 MHz
λ_i	30 tasks/min	B_j^{DL}	0.5 MHz
P_i^{MU}	0.1 W	F^{UAV}	3 GHz
P_{RX}^{UAV}	0.1 W	F_j^{EC}	$[6, 9]$ GHz
ρ	5		

Our system settings involving the interactions among IoT devices, UAV and ECs are different from prior works. The approaches proposed in their studies are not directly applicable to our settings. Therefore, we consider the following intuitive methods as baselines:

1. *Random UAV location scheme*: the task splitting and resource allocation decisions are optimized while the UAV location is randomly selected without optimization;

2. *UAV-only scheme*: all tasks are offloaded to and processed at the UAV without further offloading to any ECs;

3. *EC-only scheme*: all tasks are first offloaded to the UAV without any computations and further offloaded to ECs for processing;

4. *Fixed UAV-EC scheme*: half of the tasks are processed at the UAV while the other half are processed at ECs.

Note that the UAV is deployed at optimal position for the last three baselines similar to our proposed method. To investigate the importance of UAV location optimization, we name our proposed method as *optimized UAV location scheme* and compare it with the random UAV location scheme. To study the benefits of utilizing computing capacity at both UAV and ECs, we rename our proposed method as *collaborative UAV-EC scheme* and compare it with UAV-only, EC-only and fixed UAV-EC schemes.

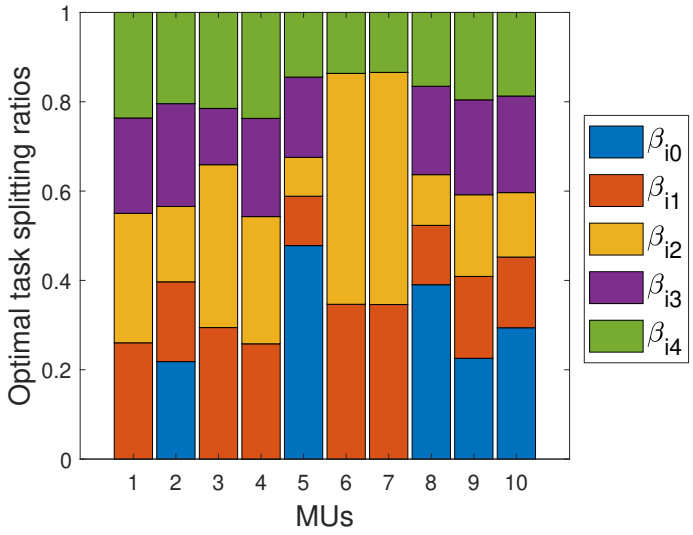


Figure 16: Optimal task splitting ratios of the UAV β_{i0} $(i = 1, 2, \ldots, 10)$ and ECs β_{ij} $(j = 1, 2, 3, 4)$ for MUs.

5.3.2 Experimental Results

In this section, we first simulate and analyze how UAV position and per-device bandwidth in the downlink communication will affect the system cost of studied UAV-enabled MEC system. Then, we compare the performances of our proposed collaborative UAV-EC scheme with UAV-only, EC-only and fixed UAV-EC schemes to verify the effectiveness of our method in reducing the overall system cost as well as the benefits of UAV-EC collaboration. We set the simulation parameters F_j^{EC} $(j = 1, 2, 3, 4)$ and L_i $(i = 1, 2, \ldots, 10)$ to be $[8, 9, 6, 7]$ GHz and $[3, 5, 2, 3, 5, 1, 1, 5, 4, 5]$ Mbits, respectively.

Importance of optimizing the UAV position

In this part, we compare the performances of our proposed optimized UAV location scheme with random UAV location scheme where the location of UAV is randomly assigned without optimization in terms of reducing the system cost. The results are summarized in Table 7. It is shown that under the optimized UAV location scheme, the optimal 3D position $(x_*^{\text{UAV}}, y_*^{\text{UAV}}, H)$ found for the UAV is at (558.11, 724.52, 100) m. The system cost of the optimized UAV location scheme is 20.83, which is the best compared with random selected

Table 7: System cost comparison for optimized UAV location and random UAV location schemes

Optimal UAV location	System cost	
(558.11, 724.52, 100) m	20.83	
Random UAV location	System cost	Cost saving percentage
(500, 500, 100) m	22.19	6.13%
(100, 100, 100) m	24.05	13.39%
(900, 100, 100) m	23.85	12.67%
(900, 900, 100) m	22.44	7.17%
(100, 900, 100) m	23.07	9.71%

UAV locations (at the center or near each EC), and our proposed scheme can achieve high cost saving as 13.39%. The rationale behind the system cost difference is that for our proposed approach, the UAV location is optimized to obtain better channel condition when providing the offloading opportunities for ground MUs while for the random UAV location scheme, the UAV location is randomly assigned beforehand without optimization. Besides, the optimal task splitting ratios of MUs for UAV and ECs are shown in Fig. 16. We observe that for MUs 2, 5, 8, 9 and 10 with large amount of input data size, 32.12% tasks in average are first processed at the UAV (i.e., β_{i0}) to reduce the data size, and then the remaining tasks are distributed to ECs for further processing.

Impact of the per-device bandwidth

In this part, we first study how the per-device bandwidth will affect the optimal task splitting ratios at ECs. As mentioned before, per-device bandwidth is assigned to each MU beforehand, and it plays an important role in affecting the optimal task splitting ratios at ECs and system cost. To proceed, we increase the per-device bandwidth B_1^{DL} assigned to EC1 from 0.5 to 5 MHz while the other three B_2^{DL}, B_3^{DL} and B_4^{DL} remain unchanged. In Fig. 17, we observe that the optimal task splitting ratio β_{11} for MU1 at EC1 is increasing while the other three β_{12}, β_{13} and β_{14} are decreasing. The reason is that as B_1^{DL} increases, the A2G downlink transmission delay t_{11}^{A2G} from the UAV to the EC1 can be reduced, and then more

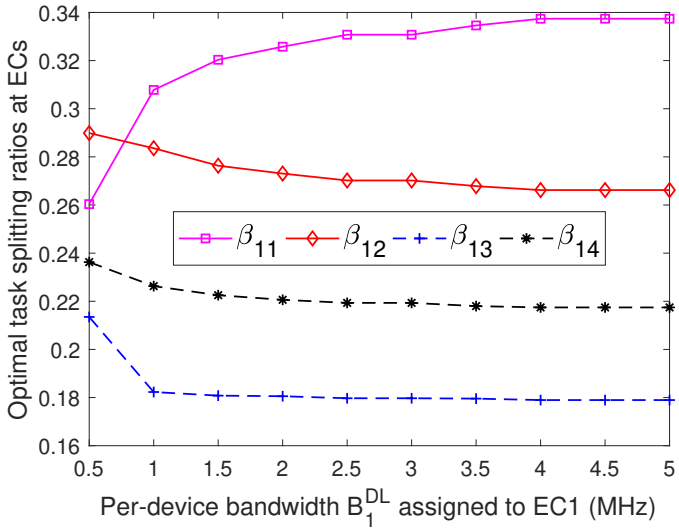


Figure 17: Optimal task splitting ratios at each EC for MU1 as a function of per-device bandwidth B_1^{DL} assigned to EC1.

tasks will be offloaded to EC1 for further processing and therefore the corresponding optimal task splitting ratio grows.

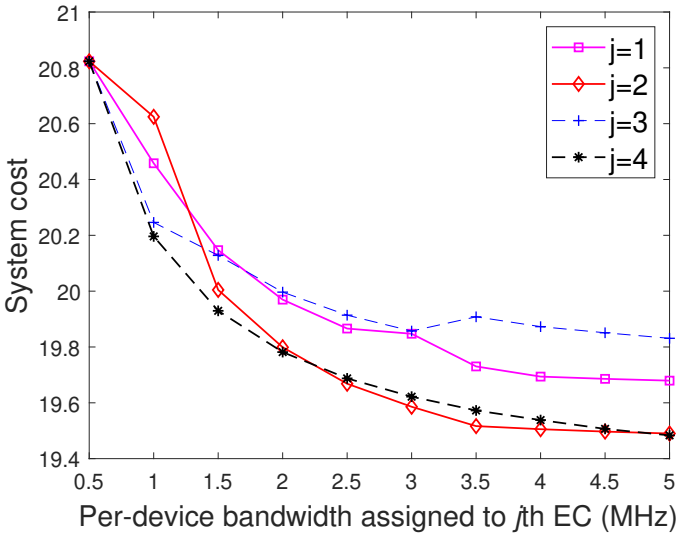


Figure 18: System cost as a function of per-device bandwidth B_j^{DL} assigned to jth EC ($j = 1, 2, 3, 4$) while fixing the others at 0.5 MHz.

Next, we investigate how the per-device bandwidth will affect the system cost. To proceed, we increase the per-device bandwidth B_j^{DL} assigned to jth EC ($j = 1, 2, 3, 4$) from 0.5 to 5 MHz while the other three remain at 0.5 MHz. In Fig. 18, we observe that the system

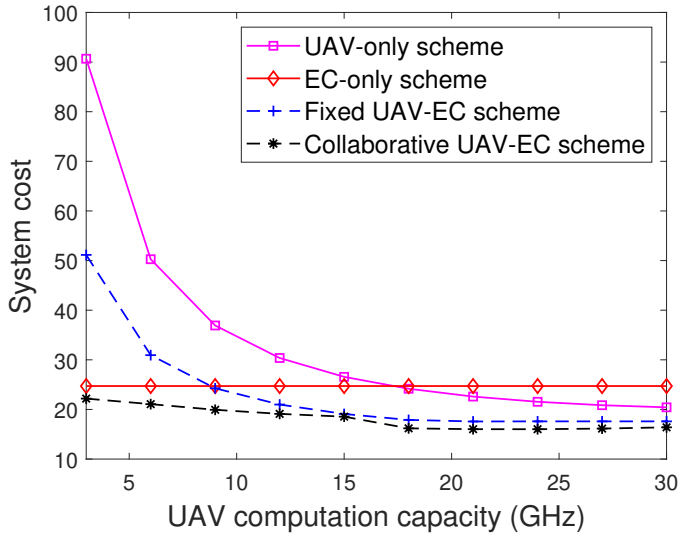


Figure 19: System cost as a function of the UAV computation capacity F^{UAV} under four different offloading schemes.

cost reduces as the per-device bandwidth assigned to jth EC increases. The reason is that as more bandwidth assigned to each MU when tasks are offloaded from the UAV to jth EC, higher downlink transmission data rates can be achieved, and thus the downlink transmission delay and downlink transmission energy consumption of the UAV can be reduced accordingly.

Benefits of UAV-EC Collaboration

In this part, we compare the performances of our proposed collaborative UAV-EC schemes with UAV-only, EC-only and fixed UAV-EC schemes in terms of reducing the system cost. Meanwhile, we investigate how system cost behaves as the UAV computation capacity and UAV transmission power change, respectively. First, we study how system cost behaves as the UAV computation capacity F^{UAV} increases from 3 to 30 GHz. As described in Fig. 19, the system cost of EC-only scheme does not change as F^{UAV} varies since this scheme prescribes that all MUs must offload their tasks to ECs without any computations at the UAV side. We further observe that for the other three offloading schemes, system cost all decreases as F^{UAV} increases since more computation resources are available to reduce the task computation

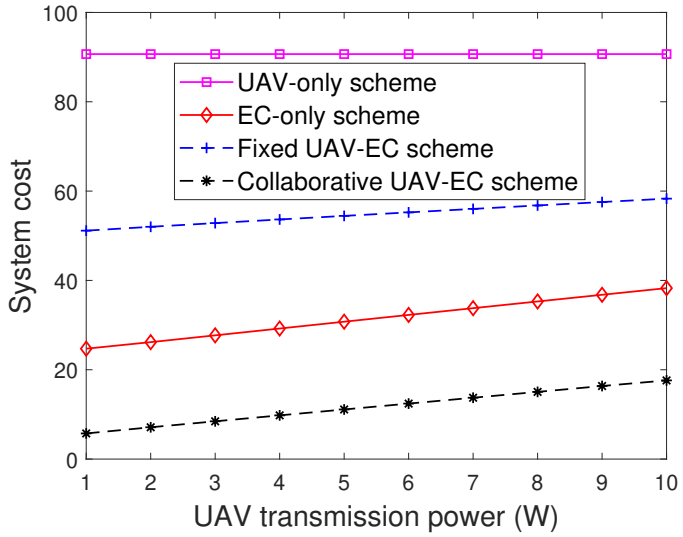


Figure 20: System cost as a function of the UAV transmission power $P_{\mathrm{TX}}^{\mathrm{UAV}}$ under four different offloading schemes.

delay at the UAV side. Then, we investigate how system cost will be affected as the UAV transmission power $P_{\mathrm{TX}}^{\mathrm{UAV}}$ increases from 1 to 10 W. As illustrated in Fig. 20, the system cost of UAV-only scheme remains constant as $P_{\mathrm{TX}}^{\mathrm{UAV}}$ varies since this scheme indicates that all MUs must offload their tasks to the UAV for execution without further offloading to any ECs. We further observe that for the other three offloading schemes, system cost increases as $P_{\mathrm{TX}}^{\mathrm{UAV}}$ increases since the downlink transmission energy consumption of the UAV is an increasing function of $P_{\mathrm{TX}}^{\mathrm{UAV}}$. Under the above two scenarios, we observe that our proposed approach largely outperforms baseline schemes such as UAV-only, EC-only and fixed UAV-EC offloading schemes in terms of reducing the system cost, which verifies the benefits of UAV-EC collaboration in the task offloading processes.

5.4 Summary

In this chapter, we have studied an innovative UAV-enabled MEC system involving the interactions among IoT devices, UAV and ECs. We have proposed to deploy a UAV properly to facilitate the MEC service provisioning to a set of stationary IoT devices in regions where existing ECs cannot be accessible to IoT devices due to terrestrial signal blockage

and shadowing. The UAV and the ECs in our system collaboratively provide MEC services to the IoT devices using aerial-to-ground communications. We have formulated a non-convex optimization problem with the goal of minimizing the weighted sum of the service delay of all IoT devices and UAV energy consumption by jointly optimizing UAV position, communication and computing resource allocation and task splitting decisions. We have developed a SCA-based algorithm to tackle the non-convexity of the original problem by first transforming the original non-convex problem into its approximated convex form and then solve it efficiently. We have also conducted numerical experiments to verify that our proposed collaborative UAV-EC offloading scheme largely outperforms baseline schemes that solely rely on UAV or ECs for MEC in IoT.

CHAPTER VI

JOINT DIFFERENTIAL EVOLUTION AND SUCCESSIVE CONVEX APPROXIMATION IN UAV-ENABLED MOBILE EDGE COMPUTING

UAV-enabled mobile edge computing (MEC) is an emerging technology to support resource-intensive yet delay-sensitive applications with edge clouds (ECs) deployed in the proximity to mobile users and UAVs served as computing base stations in the air. The formulated optimization problems therein are highly nonconvex and thus difficult to solve. To tackle the nonconvexity, the successive convex approximation (SCA) technique has been widely used to solve for the nonconvex optimization problems by transforming the nonconvex objective functions and constraints into suitable convex surrogates. However, the optimal solutions are based on the approximated optimization problem not the original one and they are highly dependent on the feasible solution initialization. Unlike SCA, Differential Evolution (DE) is a global optimization method that iteratively updates the best candidate solutions with respect to the predefined objective functions. DE works well especially in unconstrained optimization problems since it can freely search very large regions of possible solutions without considering the convexity of the original problem. However, when it comes to the constrained optimization problem, DE becomes inefficient to find the feasible and optimal solutions within given time limits. In view of the shortcomings incurred in both DE and SCA, we propose an innovative algorithm by jointly applying DE and SCA (DE-SCA) to solve for the nonconvex optimization problems. However, directly using full DE solutions to initialize the SCA-based algorithm will result in worse objective function values as the DE solutions are often infeasible. Therefore, we further design to screen the feasible parts

from the DE solutions and utilize them to initialize the SCA-based algorithm. In experimental simulations, we consider a system of UAV-enabled MEC where IoT devices, the UAV and ECs interact with each other. The simulation results demonstrate that our proposed Screened DE-SCA algorithm largely outperforms the benchmarks including DE, SCA-based and state-of-the-art algorithms in the UAV-enabled MEC system.

The rest of this chapter is organized as follows. In Section 6.1, we present preliminaries on constrained optimization problems, Differential Evolution and successive convex approximation. In Section 6.2, we describe the concrete framework and algorithm of our proposed method. In Section 6.3, we introduce the system model of a UAV-enabled MEC system and then formulate the optimal IoT task offloading processes as a nonconvex optimization problem. The numerical experiment results based on real-world traces are analyzed in Section 6.4. Finally, the conclusion is summarized in Section 6.5.

6.1 Preliminaries

In this section, we will briefly introduce the fundamental concepts of constrained optimization problems, differential evolution and successive convex approximation.

6.1.1 Constrained Optimization Problems

Without loss of generality, COPs in a minimization sense can be formulated as follows:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & g_j(\boldsymbol{x}) \leq 0, \ \forall j = 1, \ldots, m \qquad (6.1.1a) \\ & h_j(\boldsymbol{x}) = 0, \ \forall j = m+1, \ldots, l \qquad (6.1.1b) \end{aligned}$$

where $f(\boldsymbol{x})$ is the objective function that needs to be minimized over the decision variables $\boldsymbol{x} = [x_1, \ldots, x_n]^\top \in \mathbb{R}^n$ while satisfying the m inequality constraints and $l - m$ equality constraints. Note that the equality constraints (6.1.1b) can also be converted into inequality

constraints with positive tolerance degree δ as

$$|h_j(\boldsymbol{x})| - \delta \leq 0. \tag{6.1.2}$$

For COPs, the degree of constraint violation over the decision variable $\boldsymbol{x}$ can be denoted as

$$G(\boldsymbol{x}) = \sum_{j=1}^{l} G_j(\boldsymbol{x}), \tag{6.1.3}$$

where $G_j(\boldsymbol{x})$ is the degree of constraint violation on the jth constraint and can be calculated as

$$G_j(\boldsymbol{x}) = \begin{cases} \max\{0, g_j(\boldsymbol{x})\}, & j = 1, \ldots, m \\ \max\{0, |h_j(\boldsymbol{x})| - \delta\}, & j = m+1, \ldots, l \end{cases} \tag{6.1.4}$$

Therefore, a candidate solution is called feasible if $G_j(\boldsymbol{x}) = 0$, for all $j = 1, \ldots, l$, i.e., all constraints are satisfied. Optimal solutions must be feasible but feasible solutions are not necessarily optimal.

6.1.2 Differential Evolution

DE belongs to the category of evolutionary algorithms. Different from other genetic algorithms, DE generates offspring by incorporating the difference between genes from randomly chosen parents in the pool. Generally, the main procedure of DE can be described as following stages.

Initialization

DE initializes by selecting a population size of NP, which is the cardinality of the set of decision vectors $P = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{NP}\}$. For ith decision vector $\boldsymbol{x}_i$, the jth decision variable can be randomly generated as follows:

$$\boldsymbol{x}_{i,j} = \boldsymbol{x}_{i,j}^{lb} + rand(0,1) \cdot (\boldsymbol{x}_{i,j}^{ub} - \boldsymbol{x}_{i,j}^{lb}), \tag{6.1.5}$$

where $\boldsymbol{x}_{i,j}^{lb}$ and $\boldsymbol{x}_{i,j}^{ub}$ represents the lower bound and upper bound for decision variable $\boldsymbol{x}_{i,j}$ while $rand(0,1)$ is the uniformly distributed random numbers between 0 and 1.

Mutation

After the initialization of population and decision vectors, the individuals in the population will undergo a mutation stage. In this stage, a mutation operation is applied to generate a mutant decision vector for each original decision vector $\boldsymbol{x}_i^t$ ($i \in \mathcal{S} = \{1, \ldots, NP\}$) at generation t. For simplicity, DE/rand/1 mutation scheme is selected for illustration as follows:

$$\boldsymbol{v}_i^t = \boldsymbol{x}_{r1}^t + F \cdot (\boldsymbol{x}_{r2}^t - \boldsymbol{x}_{r3}^t), \tag{6.1.6}$$

where $\boldsymbol{v}_i^t$ is the ith mutant decision vector at generation t, $F \in (0, 1+]$ is the differential scaling factor and $r1$, $r2$ and $r3$ are randomly generated integers from $\mathcal{S}$ such that $r1 \neq r2 \neq r3 \neq i$. Other commonly used mutation schemes include DE/rand/2, DE/rand-to-best/1, DE/current-to-best/1 and DE/current-to-rand/1 [103].

Crossover

In order to maintain diversity in the population, for the ith individual at stage t, a trial decision vector $\boldsymbol{u}_i^t$ is generated between original decision vector $\boldsymbol{x}_i^t$ and mutant decision vector $\boldsymbol{v}_i^t$. The crossover operation can be described as follows:

$$\boldsymbol{u}_{i,j}^t = \begin{cases} \boldsymbol{v}_{i,j}^t, & \text{if } rand(0,1) < CR \text{ or } j = j0 \\ \boldsymbol{x}_{i,j}^t, & \text{otherwise} \end{cases} \tag{6.1.7}$$

where $\boldsymbol{u}_{i,j}^t$, $\boldsymbol{v}_{i,j}^t$ and $\boldsymbol{x}_{i,j}^t$ represents the jth variable of $\boldsymbol{u}_i^t$, $\boldsymbol{v}_i^t$ and $\boldsymbol{x}_i^t$ at generation t, respectively; $CR \in [0, 1]$ is the crossover rate and $j0$ is a randomly generated integer from $\mathcal{S}$.

Selection

In this stage, the decision vector of the next generation is determined from the current generation by comparing the fitness (i.e., objective function value) of the trial decision vector and original decision vector. It is important to handle constraints when we apply DE to deal with COPs. Classical feasibility rules [31] to handle constraints are summarized as follows:

- Between two feasible decision vectors, the one with smaller objective function value (minimization sense) is selected.
- If one decision vector is feasible and the other one is infeasible, the feasible one is selected.
- Between two infeasible decision vectors, the one with less degree of constraint violation is selected.

Therefore, the selection operation for COPs with positive tolerance ε [101] can be described as follows:

$$\boldsymbol{x}_i^{t+1} = \begin{cases} \boldsymbol{u}_i^t, & \text{if } G(\boldsymbol{x}_i^t) \leq \varepsilon \text{ and } G(\boldsymbol{u}_i^t) \leq \varepsilon \\ & \text{and } f(\boldsymbol{u}_i^t) < f(\boldsymbol{x}_i^t) \\ \boldsymbol{x}_i^t, & \text{if } G(\boldsymbol{x}_i^t) \leq \varepsilon \text{ and } G(\boldsymbol{u}_i^t) \leq \varepsilon \\ & \text{and } f(\boldsymbol{x}_i^t) < f(\boldsymbol{u}_i^t) \\ \boldsymbol{u}_i^t, & \text{if } G(\boldsymbol{x}_i^t) > \varepsilon \text{ and } G(\boldsymbol{u}_i^t) \leq \varepsilon \\ \boldsymbol{x}_i^t, & \text{if } G(\boldsymbol{u}_i^t) > \varepsilon \text{ and } G(\boldsymbol{x}_i^t) \leq \varepsilon \\ \boldsymbol{u}_i^t, & \text{if } G(\boldsymbol{x}_i^t) > G(\boldsymbol{u}_i^t) \geq \varepsilon \\ \boldsymbol{x}_i^t, & \text{if } G(\boldsymbol{u}_i^t) > G(\boldsymbol{x}_i^t) \geq \varepsilon \end{cases} \tag{6.1.8}$$

6.1.3 Successive Convex Approximation

SCA technique has been broadly applied in many areas such as communications, machine learning, networking and signal processing. In what follows, we will review the specific

optimization problem solved by the SCA technique under required assumptions and give some examples of function approximation.

Problem Statement

Consider the optimization problem as follows:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f_0(\boldsymbol{x}) \triangleq h_0(\boldsymbol{x}) + g_0(\boldsymbol{x}) \\ \text{s.t.} \quad & f_j(\boldsymbol{x}) \triangleq h_j(\boldsymbol{x}) + g_j(\boldsymbol{x}) \leq 0, \ \forall j = 1, \ldots, l \qquad (6.1.9a) \\ & \boldsymbol{x} \in \mathcal{X}, \qquad (6.1.9b) \end{aligned}$$

where the function $f_j(\boldsymbol{x})$ is smooth (possibly nonconvex) and $g_j(\boldsymbol{x})$ is convex (possibly nonsmooth), for all $j = 0, \ldots, l$ and the feasible set is denoted as $\mathcal{X}$. A commonly used method for solving this specific problem is SCA (also known as majorization minimization) where under a tight convex restriction of the constraint sets, a locally tight approximation of the original optimization problem is solved iteratively. A tight approximation of the original optimization problem can be stated as follows: given $x^k \in \mathcal{X}$ at iterate k

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \tilde{f}_0(\boldsymbol{x}; \boldsymbol{x}^k) \\ \text{s.t.} \quad & \tilde{f}_j(\boldsymbol{x}; \boldsymbol{x}^k) \leq 0, \ \forall j = 1, \ldots, l \qquad (6.1.10a) \\ & \boldsymbol{x} \in \mathcal{X}^k, \qquad (6.1.10b) \end{aligned}$$

where $\tilde{f}_0(\boldsymbol{x}; \boldsymbol{x}^k)$ and $\tilde{f}_j(\boldsymbol{x}; \boldsymbol{x}^k)$ represent the approximated functions of $\tilde{f}_0(\boldsymbol{x})$ and $\tilde{f}_j(\boldsymbol{x})$, respectively and $\boldsymbol{x} \in \mathcal{X}^k$ is the feasible set at iterate k. More precisely, the SCA method is presented in Algorithm 3. It is well known that a stationary point of the problem (6.1.9) is also local minimum [91]. The key assumptions [85] to validate this algorithm are summarized as follows:

Assumption 1: The approximated functions $\tilde{f}_j(\bullet; \bullet)$ for all $j = 0, \ldots, l$ are assumed to

Algorithm 3 SCA Algorithm for Problem (6.1.9)

Find a feasible solution $\boldsymbol{x}^k \in \mathcal{X}$ in (6.1.9), choose a step size $\gamma \in (0, 1]$ and set $k = 0$.
Repeat

1. Compute $\hat{\boldsymbol{x}}^k$, the solution of (6.1.10);
2. Set $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \gamma(\hat{\boldsymbol{x}}^k - \boldsymbol{x}^k)$;
3. Set $k \leftarrow k + 1$.

Until some convergence criterion is met.

satisfy the following statements:

- $\tilde{f}_j(\boldsymbol{x}; \boldsymbol{y})$ is continuous in $(\boldsymbol{x}, \boldsymbol{y})$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$.
- $\tilde{f}_j(\bullet; \boldsymbol{y})$ is convex for all $\boldsymbol{y} \in \mathcal{X}$.
- $\tilde{f}_j(\boldsymbol{x}; \boldsymbol{y}) = \tilde{h}_j(\boldsymbol{x}; \boldsymbol{y}) + g_j(\boldsymbol{x})$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$.
- *Function value consistency*: $\tilde{h}_j(\boldsymbol{x}; \boldsymbol{x}) = h_j(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{X}$.
- *Gradient consistency*: $\nabla_{\boldsymbol{x}} \tilde{f}_j(\bullet; \boldsymbol{x}) = \nabla_{\boldsymbol{x}} f_j(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{X}$, where $\nabla_{\boldsymbol{x}} \tilde{f}_j(\bullet; \boldsymbol{x})$ denotes the partial gradient of the function $\tilde{f}_j(\bullet; \boldsymbol{x})$ with respect to the argument $\boldsymbol{x}$ evaluated at $(\bullet; \boldsymbol{x})$.
- *Upper bound*: $\tilde{f}_j(\boldsymbol{x}; \boldsymbol{y}) \geq f_j(\boldsymbol{x})$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$.

Therefore, the above assumptions guarantee that the original functions can be approximately replaced by suitable upper-bounding functions where the same first-order behavior can be preserved.

Examples of Function Approximation

In this section, we will briefly introduce some examples of function approximation technique that will be used throughout this chapter.

Example 1–Approximation of $f_0(\boldsymbol{x})$: (Example 8 in [91]) Suppose that $f_0(\boldsymbol{x})$ has a product of functions (PF) structure, i.e., $f_0(\boldsymbol{x}) = p_1(\boldsymbol{x})p_2(\boldsymbol{x})$ with both p_1 and p_2 being positive and

convex. For any $\boldsymbol{y} \in \mathcal{X}$, a convex approximation of $f_0(\boldsymbol{x})$ is given by

$$\begin{aligned}\tilde{f}_0(\boldsymbol{x};\boldsymbol{y}) &= p_1(\boldsymbol{x})p_2(\boldsymbol{y}) + p_1(\boldsymbol{y})p_2(\boldsymbol{x}) \\ &\quad + \frac{\tau}{2}(\boldsymbol{x}-\boldsymbol{y})^\top \boldsymbol{U}(\boldsymbol{y})(\boldsymbol{x}-\boldsymbol{y}),\end{aligned} \tag{6.1.11}$$

where $\tau > 0$ is a positive constant, and $\boldsymbol{U}(\boldsymbol{y})$ is a uniformly positive definite matrix.

Example 2–Approximation of $f_j(\boldsymbol{x})$: (Example 3 in [91]) Suppose that $f_j(\boldsymbol{x})$ has a difference of convex (DC) structure, i.e., $f_j(\boldsymbol{x}) = f_j^+(\boldsymbol{x}) - f_j^-(\boldsymbol{x})$ with both $f_j^+(\boldsymbol{x})$ and $f_j^-(\boldsymbol{x})$ being continuously differentiable and convex. By linearizing the concave part $-f_j^-(\boldsymbol{x})$, we obtain a convex upper approximation of $f_j(\boldsymbol{x})$ as follows: for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$,

$$\tilde{f}_j(\boldsymbol{x};\boldsymbol{y}) \triangleq f_j^+(\boldsymbol{x}) - f_j^-(\boldsymbol{y}) - \nabla_{\boldsymbol{x}} f_j^-(\boldsymbol{y})^\top(\boldsymbol{x}-\boldsymbol{y}) \geq f_j(\boldsymbol{x}). \tag{6.1.12}$$

Example 3–Approximation of $f_j(\boldsymbol{x})$: (Example 4 in [91]) Suppose that $f_j(\boldsymbol{x})$ has a PF structure, i.e., $f_j(\boldsymbol{x}) = q_1(\boldsymbol{x})q_2(\boldsymbol{x})$ with both $q_1(\boldsymbol{x})$ and $q_2(\boldsymbol{x})$ being positive and convex. Observe that $f_j(\boldsymbol{x})$ can be rewritten as a function with a DC structure:

$$f_j(\boldsymbol{x}) = \frac{1}{2}(q_1(\boldsymbol{x}) + q_2(\boldsymbol{x}))^2 - \frac{1}{2}(q_1^2(\boldsymbol{x}) + q_2^2(\boldsymbol{x})). \tag{6.1.13}$$

Then, a convex upper approximation of $f_j(\boldsymbol{x})$ can be obtained by linearizing the concave part in (6.1.13): for any $\boldsymbol{y} \in \mathcal{X}$,

$$\begin{aligned}\tilde{f}_j(\boldsymbol{x};\boldsymbol{y}) &\triangleq \frac{1}{2}(q_1(\boldsymbol{x}) + q_2(\boldsymbol{x}))^2 - \frac{1}{2}(q_1^2(\boldsymbol{y}) + q_2^2(\boldsymbol{y})) \\ &\quad - q_1(\boldsymbol{y})q_1'(\boldsymbol{y})(\boldsymbol{x}-\boldsymbol{y}) - q_2(\boldsymbol{y})q_2'(\boldsymbol{y})(\boldsymbol{x}-\boldsymbol{y}) \geq f_j(\boldsymbol{x}).\end{aligned} \tag{6.1.14}$$

6.2 Screened DE-SCA

Motivated by the pros and cons of DE and SCA, we propose to merge their own advantages in jointly optimizing COPs such that better local optimal solutions of the original problem can

be obtained. The framework and algorithm of Screened DE-SCA are presented in Figure 21 and Algorithm 4, respectively. In Figure 21, the process of finding and screening DE solutions (step 1–7) can be implemented in parallel to the process of problem reformulation (step 1 and step 8–9). In step 10–12, the screened DE solutions from step 7 will be used to initialize the SCA-based algorithm on reformulated convex problem from step 9, and then the optimal solutions can be found.

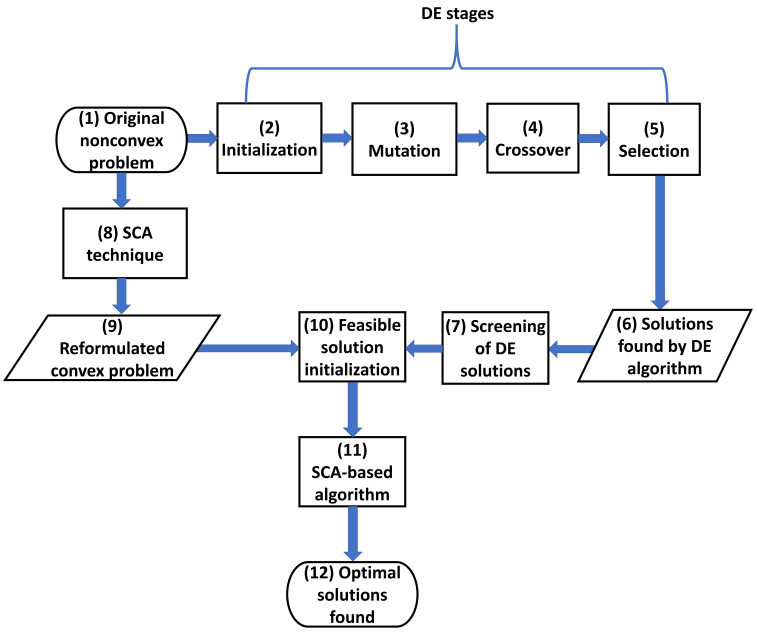


Figure 21: Framework of Screened DE-SCA method. Optimal solutions of the original nonconvex problem can be found by initializing the SCA-based algorithm with partial feasible solutions of DE algorithm applied to the original nonconvex problem.

6.2.1 DE Stages

Our algorithm starts from applying the DE algorithm. Specifically, DE algorithm spans from line 1 to line 6 where stages of initialization, mutation, crossover and selection that are implemented based on equations from (6.1.5) to (6.1.8) are applied to solve the original nonconvex problem (6.1.9). Note that we selected DE/rand/1 as a mutation scheme and ε approximation as a constraint-handling technique [29, 62]. One can also choose different mutation schemes and constraint-handling techniques in this stage. However, it is out of

the scope of our chapter to determine which mutation schemes and constraint-handling techniques work best in our proposed algorithm.

Algorithm 4 Screened DE-SCA Algorithm for Problem (6.1.9)

Input:
DE: NP, $F \in (0, 1+]$, $CR \in [0, 1]$, δ, ε,
$MaxIter$: the maximum DE iterations.
SCA: α, $\gamma(k) \in (0, 1]$, $\tau > 0$.
1: Set $t = 1$;
2: Randomly generate an initial population $P^t = \{\boldsymbol{x}_1^t, \ldots, \boldsymbol{x}_{NP}^t\}$ using equation (6.1.5);
Repeat:
3: Evaluate the objective functions $f_0(\boldsymbol{x}_i^t)$ and degree of constraint violation $G(\boldsymbol{x}_i^t)$ for all $i \in \mathcal{S}$;
4: **for** $i = 1 : NP$ **do**

1) Calculate mutant decision vector $\boldsymbol{v}_i^t$ using equation (6.1.6);

2) **for** $j = 1 : ND$ **do**
/*ND is the number of decision variables*/

i) Obtain trial decision variable $\boldsymbol{u}_{i,j}^t$ using equation (6.1.7);

3) Apply the feasibility rule with tolerance ε according to equation (6.1.8);

5: Set $t \leftarrow t + 1$;
Until $t > MaxIter$
6: Output and store the best individual in P^t as $\boldsymbol{x}_*^{\mathrm{DE}}$;
7: Set $k = 0$;
8: Randomly initialize the SCA-based algorithm with feasible solutions $\boldsymbol{x}^k \in \mathcal{X}$ where the feasible parts of $\boldsymbol{x}_*^{\mathrm{DE}}$ are directly used for assignment;
Repeat:
9: Compute $\hat{\boldsymbol{x}}^k$, the solution of (6.1.10);
10: Set $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \gamma(\hat{\boldsymbol{x}}^k - \boldsymbol{x}^k)$, with $\gamma(k) = \gamma(k-1)(1 - \alpha\gamma(k))$;
11: Set $k \leftarrow k + 1$;
Until $\hat{\boldsymbol{x}}^k$ is a stationary solution of (6.1.10);
12: Output the optimal solution $\boldsymbol{x}_*^{\mathrm{SCA}}$ and objective function value $f_0(\boldsymbol{x}_*^{\mathrm{SCA}})$.

6.2.2 Screened DE Solutions

The DE algorithm will simply terminate if the number of iterations exceeds the pre-defined maximum DE iterations $MaxIter$. Then, we can output and store the best solutions $\boldsymbol{x}_*^{\mathrm{DE}}$ as in line 6. Next, we will need to initialize the SCA-based algorithm and the straightforward

idea is to initialize it with $\boldsymbol{x}_*^{\mathrm{DE}}$. However, direct initialization with full DE solutions can actually result in worse initialization in the SCA-based algorithm. We observe that $\boldsymbol{x}_*^{\mathrm{DE}}$ is infeasible in general as the degree of constraint violation $G(\boldsymbol{x}_*^{\mathrm{DE}})$ cannot be small enough (e.g., $\delta = 10^{-4}$) especially when there are hundreds of constraints. Therefore, it is necessary to go through the screening process of the best DE solutions before we perform the initialization. In real problems, there can be constraints that contain all decision variables as well as constraints that contain only one decision variable. The set of indices of feasible decision variables from the best DE solution can be found as

$$\mathcal{K} = \{z \in \mathcal{Z} | G_j(\boldsymbol{x}_{z,*}^{\mathrm{DE}}) = 0, \forall j = 1, \ldots, l\}, \tag{6.2.1}$$

where $\mathcal{Z} = \{1, \ldots, ND\}$ denotes the set of indices for decision variables; $G_j(\boldsymbol{x}_{z,*}^{\mathrm{DE}})$ denotes the zth decision variable from the best DE solution. In order for the zth decision variable to be considered feasible, we require that all the constraints that contain it must have degree of constraint violation with 0. Therefore, the feasible parts of DE solutions can be denoted as $\boldsymbol{x}_{\mathcal{K},*}^{\mathrm{DE}}$ while the infeasible parts are simply denoted as $\boldsymbol{x}_{\mathcal{Z}\setminus\mathcal{K},*}^{\mathrm{DE}}$.

6.2.3 SCA Stages

While we apply DE algorithm to solve for the original nonconvex problem, we also need to convert the original problem into its approximated convex problem such that the SCA-based algorithm can be applied to find the feasible and optimal solutions of it. Specifically, SCA-based algorithm spans from line 7 to line 12 where the approximated convex problem (6.1.10) is iteratively solved with a random solution $\boldsymbol{x}^0$ as initialization. It is meaningful to only use the feasible parts $\boldsymbol{x}_{\mathcal{K},*}^{\mathrm{DE}}$ of the best DE solution to initialize the SCA-based algorithm in line 8 as the initialization must be feasible over all decision variables $\boldsymbol{x} \in \mathcal{X}$. Note that at line 10, in order to improve the convergence rate of the SCA-based algorithm, a diminishing step-size rule is applied compared to a constant step size in Algorithm 3. To terminate the

SCA-based algorithm by finding the stationary solution of (6.1.10), we can simply check if $\|\hat{\boldsymbol{x}}^k - \boldsymbol{x}^k\| \leq \zeta$ where ζ is the desired algorithm accuracy. Finally in line 12, we output the optimal solution $\boldsymbol{x}_*^{\text{SCA}}$ and plug it into the original objective function to find $f_0(\boldsymbol{x}_*^{\text{SCA}})$.

6.3 UAV-enabled Mobile Edge Computing

In this section, we will briefly introduce the proposed model in [128] used to validate our innovative Screened DE-SCA algorithm. We first present the system model for a UAV-enabled IoT task offloading process. Then, a nonconvex optimization problem is formulated to model this process.

6.3.1 System Model

In this chapter, we consider a UAV-enabled IoT task offloading process as illustrated in Figure 22, which reflects a three-tier network infrastructure: 1) *User layer* consists of a set of ground IoT devices $i \in \mathcal{N} = \{1, \ldots, N\}$ which have periodical computation-intensive tasks to perform; 2) *UAV layer* consists of a UAV that is equipped with CCS resources but is constrained by SWAP; 3) *Edge layer* consists of a set of ground ECs $j \in \mathcal{J} = \{1, \ldots, J\}$. ECs are composed of edge servers co-located with base stations or access points, which are deployed in the proximity of IoT devices.

We assume that the IoT devices do not perform local computing due to their limited computational capacities and thus the generated tasks need to be offloaded to the ECs that have more resources for processing. We consider the scenario that these IoT devices cannot directly communicate with ground ECs due to terrestrial signal blockage and shadowing but a UAV can be deployed to help facilitate the task offloading process via unhindered ground-to-air (G2A) and air-to-ground (A2G) communications due to its high altitude.

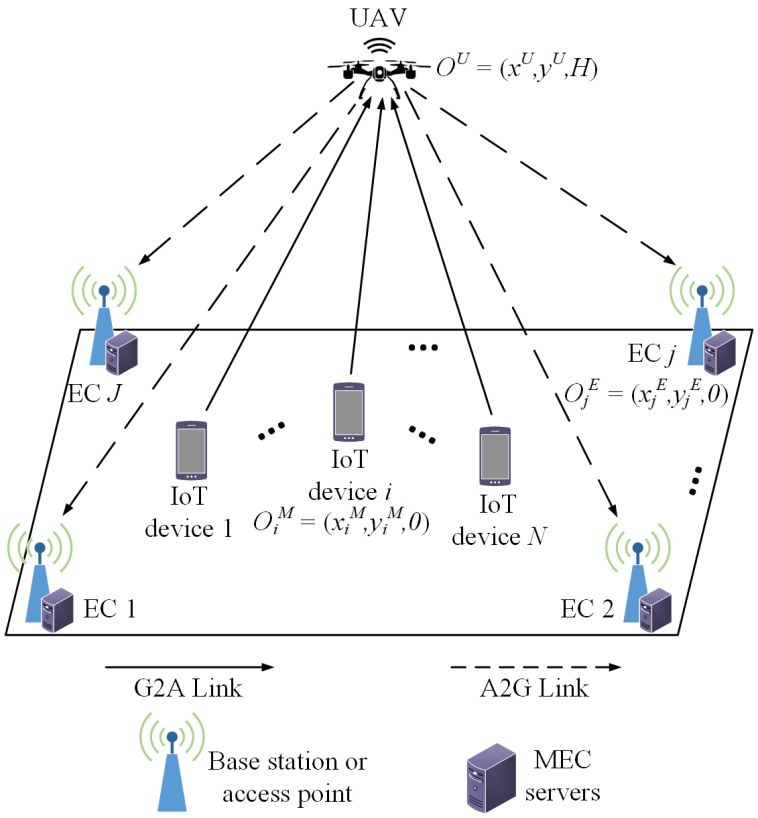


Figure 22: 3D plane of the UAV-enabled MEC system with N IoT devices, J ECs and a UAV.

Task Model

The generated task for each IoT device i can be modeled as a triplet $\mathcal{M}_i = \langle L_i, C_i, \lambda_i \rangle$ where L_i denotes the input data size in bits, C_i denotes the required CPU cycles to process 1 bit of data and λ_i denotes the number of tasks generated per second.

Coordinate Model

Our model can be visualized in a 3D Cartesian coordinate system where the locations of ground IoT devices, UAV and ground ECs can be denoted as $O_i^M = (x_i^M, y_i^M, 0)$, $O^U = (x^U, y^U, H)$ and $O_j^E = (x_j^E, y_j^E, 0)$, respectively. We assume the height H of the UAV is fixed but the horizontal locations x^U and y^U of the UAV need to be optimized in our problem since the deployment of the UAV will affect the channel gain during G2A and A2G communications.

Communication Model

The G2A uplink channel gain from IoT device i to the UAV can be obtained using the free-space path loss model

$$g_i^{ul} \triangleq \frac{\beta_0}{d_{i,ul}^2} = \frac{\beta_0}{\|O_i^M - O^U\|^2}, \tag{6.3.1}$$

where β_0 represents the received power at the reference distance of 1 m for a transmission power of 1 W, $d_{i,ul}$ denotes the distance from IoT device i to the UAV, and $\|\cdot\|$ denotes the Euclidean norm of a vector. Accordingly, the A2G downlink channel gain from the UAV to EC j can be obtained as

$$g_j^{dl} \triangleq \frac{\beta_0}{d_{j,dl}^2} = \frac{\beta_0}{\|O^U - O_j^E\|^2}, \tag{6.3.2}$$

where $d_{j,dl}$ denotes the distance from the UAV to EC j.

Assume a FDMA protocol is applied for bandwidth sharing in our studied model. Then, the achievable uplink transmission data rate in bps from IoT device i to the UAV can be calculated according to Shannon–Hartley theorem as

$$D_i^{ul} = B_i^{ul} \log_2 \left(1 + \frac{g_i^{ul} P_i^M}{\sigma^2}\right), \tag{6.3.3}$$

where B_i^{ul} denotes the allocated bandwidth to IoT device i, P_i^M denotes the transmission power of IoT device i and σ^2 denotes the noise power at the UAV. Without loss of generality, we assume both UAV and ECs have the same noise power [131]. Accordingly, the achievable downlink transmission data rate in bps from the UAV to EC j can be calculated as

$$D_j^{dl} = B_j^{dl} \log_2 \left(1 + \frac{g_j^{dl} P_{TX}^U}{\sigma^2}\right), \tag{6.3.4}$$

where B_j^{dl} denotes the per-device bandwidth pre-allocated to the UAV when it communicates with EC j and P_{TX}^U denotes the transmission power of UAV.

Delay Model

In this model, the total delay incurred during the IoT task offloading process can be divided into: i) G2A uplink transmission delay; ii) computation delay at the UAV; iii) A2G downlink transmission delay; and iv) computation delay at ECs.

G2A Uplink Transmission Delay: As mentioned before, IoT devices are assumed to not perform any local computing due to their limited computational capacities and thus all tasks will be first offloaded to the UAV. The G2A uplink transmission delay from IoT device i can be calculated as

$$t_i^{G2A} = \frac{L_i}{D_i^{ul}}. \tag{6.3.5}$$

Computation Delay at the UAV: After receiving all the tasks from IoT device i, the UAV will determine the task partitioning where the portion of $\alpha_{i0} \in [0, 1]$ will be processed at the UAV while the portion of $\alpha_{ij} \in [0, 1]$ will be further offloaded to EC j for processing. The computation delay at the UAV when processing the offloaded tasks from IoT device i can be calculated as

$$t_i^U = \frac{\alpha_{i0} L_i C_i}{f_i^U}, \tag{6.3.6}$$

where f_i^U denotes the allocated computation capacities in CPU cycles/s from the UAV to IoT device i.

A2G Downlink Transmission Delay: The A2G downlink transmission delay of task offloading from IoT device i to EC j via the UAV can be calculated as

$$t_{ij}^{A2G} = \frac{\alpha_{ij} L_i}{D_j^{dl}}. \tag{6.3.7}$$

Computation Delay at ECs: The computation delay at ECs when processing the tasks offloaded from IoT device i to EC j via the UAV can be calculated as

$$t_{ij}^E = \frac{\alpha_{ij} L_i C_i}{f_{ij}^E}, \tag{6.3.8}$$

where f_{ij}^{E} denotes the allocated computation capacities in CPU cycles/s from EC j to IoT device i.

Finally, the total delay experienced by IoT device i during task offloading can be calculated as

$$T_i = t_i^{G2A} + \max_{j \in \mathcal{J}} \{t_i^U, t_{ij}^{A2G} + t_{ij}^E\}. \tag{6.3.9}$$

Energy Model

In this model, we mainly consider the energy consumed at the UAV side during computation and communication processes since the battery size of the UAV is limited.

Computation Energy Consumption: The power consumption of the CPU in the UAV when processing tasks offloaded from IoT device i can be modeled as $\kappa(f_i^U)^3$ according to [105], where κ denotes the effective switched capacitance based on the CPU architecture. Therefore, the energy consumption of the UAV when processing tasks offloaded from IoT device i can be calculated as

$$E_i^{CP} = \kappa(f_i^U)^3 t_i^U = \kappa \beta_{i0} L_i C_i (f_i^U)^2. \tag{6.3.10}$$

Communication Energy Consumption: The reception energy consumption of the UAV when receiving the task input data from IoT device i can be calculated as

$$E_i^{RX} = P_{RX}^U t_i^{G2A} = \frac{L_i P_{RX}^U}{D_i^{ul}}, \tag{6.3.11}$$

where P_{RX}^U denotes the receiving power of UAV. Similarly, the transmission energy consumption of the UAV when transmitting the task input data of IoT device i from the UAV to EC j can be calculated as

$$E_{ij}^{TX} = P_{TX}^U t_{ij}^{A2G} = \frac{\alpha_{ij} L_i P_{TX}^U}{D_j^{dl}}. \tag{6.3.12}$$

Finally, the total energy consumption of the UAV when serving the computation and

communication needs of IoT device i during task offloading can be calculated as

$$E_i = \lambda_i(E_i^{CP} + E_i^{RX} + \sum_{j\in\mathcal{J}} E_{ij}^{TX}). \tag{6.3.13}$$

6.3.2 Problem Formulation

Based on the system model proposed above, our problem can be stated as follows: with the objective of minimizing the weighted sum of UAV energy consumption and system latency experienced by all IoT devices, we jointly optimize the UAV position O^U, G2A communication resource allocation B_i^{ul}, task partitioning α_{i0} and α_{ij} and computation resource allocation of the UAV f_i^U and ECs f_{ij}^E. All the decision variables are continuous. It can be formulated as the following optimization problem:

$$\min_{\substack{O^U, B_i^{ul}, \alpha_{i0}, \\ \alpha_{ij}, f_i^U, f_{ij}^E}} \quad \sum_{i\in\mathcal{N}} E_i + \varrho \sum_{i\in\mathcal{N}} T_i \tag{6.3.14a}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{N}} B_i^{ul} \leq B^{ul} \tag{6.3.14b}$$

$$\alpha_{i0} + \sum_{j\in\mathcal{J}} \alpha_{ij} = 1,\ \forall i \tag{6.3.14c}$$

$$\sum_{i\in\mathcal{N}} f_i^U \leq F^U \tag{6.3.14d}$$

$$\sum_{i\in\mathcal{N}} f_{ij}^E \leq F_j^E,\ \forall j \tag{6.3.14e}$$

$$0 \leq \alpha_{ij} \leq 1,\ \forall i, j \tag{6.3.14f}$$

$$0 \leq \alpha_{i0} \leq 1,\ \forall i \tag{6.3.14g}$$

$$B_i^{ul}, f_i^U \geq 0,\ \forall i \tag{6.3.14h}$$

$$f_{ij}^E \geq 0,\ \forall i, j \tag{6.3.14i}$$

where ϱ is a positive constant describing the relative weight of UAV energy consumption and system latency; (6.3.14b), (6.3.14d), (6.3.14e), (6.3.14h) and (6.3.14i) guarantee that the

allocated resources for G2A bandwidth, CPU frequencies of UAV and ECs are non-negative and cannot exceed their limits B^{ul}, F^U and $\{F_j^E\}_{j=1}^J$, respectively; (6.3.14c), (6.3.14f) and (6.3.14g) ensure that the offloading tasks of all IoT devices are completely partitioned over the UAV and ECs, and each partition variable is between 0 and 1.

Problem (6.3.14) is highly nonconvex due to the nonconvex objective function (6.3.14a). To tackle the nonconvexity, function approximation methods in Section 6.1.3 are applied to convert them into suitable convex substitutes since all delay terms t_i^{G2A}, t_i^U, t_{ij}^{A2G} and t_{ij}^E and energy terms E_i^{CP}, E_i^{TX} and E_i^{RX} have a PF structure. In this chapter, we will skip the reformulation process of problem (6.3.14) since our focus is mainly on the solution rather than the system modeling. However, interested readers can refer to Section IV in [128] for the detailed derivation.

6.4 Numerical Experiments

In this section, we will validate the effectiveness of our proposed Screened DE-SCA algorithm via extensive numerical experiments. All the experiments are implemented in MATLAB R2019b on a desktop computer with an Intel Core i7-8700 3.20GHz CPU and 32GB RAM.

6.4.1 Simulation Setup

For simplicity, we consider the same simulation settings used in [128] where 4 ground ECs are placed at each vertex and 10 ground IoT devices are randomly distributed in a 2D area of 1×1 km^2 as depicted in Figure 23. The simulation parameters are summarized in Table 8 unless otherwise stated.

In order to validate the effectiveness of our proposed Screened DE-SCA algorithm in terms of achieving the best system cost of problem (6.3.14) in the UAV-enable MEC system, we consider the baseline algorithms as follows:

- DE: Extend the classical DE algorithm to solve (6.3.14) by utilizing the feasibility rules introduced in Section 6.1.2.

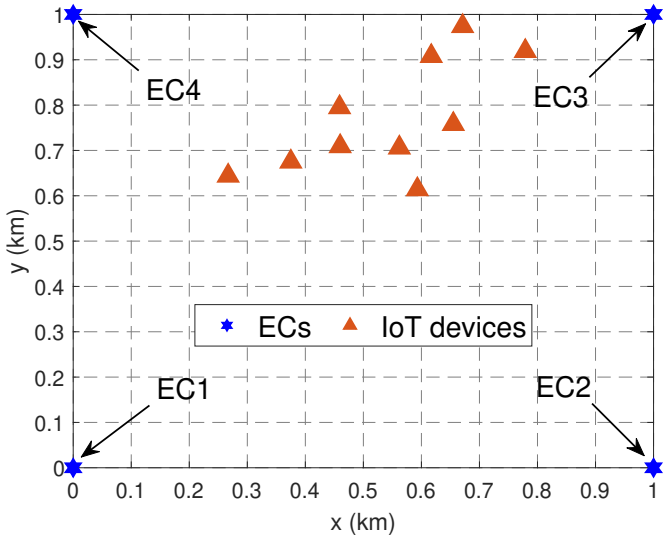


Figure 23: 2D plane of the simulated MEC system with 10 IoT devices and 4 ECs.

- DE with Penalty Function (DE-PF): Add the degree of constraint violation $G(\boldsymbol{x})$ to the objective function as a penalty function when applying DE to solve (6.3.14).

- DE with Initialization Normalization (DE-IN): Force the equality constraints (6.3.14c) to be satisfied by normalizing the value of initialized task partitioning variables when applying DE to solve (6.3.14).

- SCA-based Method: Apply SCA technique to transform problem (6.3.14) into approximated convex problem and then iteratively solve it with random initialization.

- SCA-based Method with Full DE Solutions as Initialization (DE-SCA): Initialize the approximated convex problem of (6.3.14) by full solutions obtained from solving (6.3.14) with DE and then iteratively solve it.

- UMEC: We adapt the task selection and scheduling algorithm proposed in [80] to solve for our nonconvex problem of joint task offloading and resource allocation.

- LCPSO: We adapt the learning-based task sequence and resource allocation algorithm in [100] to solve for our nonconvex problem of joint task offloading and resource allocation.

We would like to clarify that our system setting is different from those in prior works. Therefore, the approaches proposed in prior studies are not directly applicable to our settings. We carefully select two state-of-the-art methods (not DE or SCA-based) UMEC and LCPSO applied in studying the UAV-based MEC systems. Although the problems studied in their papers are different from ours, we are still able to evaluate them by adapting them into our model. To differentiate from the DE-SCA algorithm, our proposed Screened DE-SCA method only uses the partial DE solutions that are feasible to perform initialization of SCA-based method.

Table 8: Simulation Parameters

Parameters	Values	Parameters	Values
DE:			
CR	0.2	F	$[0.2, 0.8]$
$MaxIter$	10000	NP	700
δ	10^{-4}		
SCA:			
α	0.5	γ	0.5
ζ	10^{-2}		
UAV-based MEC:			
C_i	$[100, 200]$ CPU cycles/bit	ρ	5
σ^2	-100 dBm	F_j^E	$[6, 9]$ GHz
β_0	-50 dB	F^U	3 GHz
κ	10^{-28} [129, 130]	H	0.1 km
λ_i	30 tasks/min	L_i	$[1, 5]$ Mbits
P_i^M	0.1 W	B^{ul}	10 MHz
P_{RX}^U	0.1 W	B_j^{dl}	0.5 MHz
P_{TX}^U	1 W		

6.4.2 Experimental Results

In this section, we will first discuss the tradeoff between feasibility and optimality incurred while applying DE to solve for COPs. Then, we compare two different DE variants DE-PF and DE-IN used in handling the constraints with original DE in our problem. Finally, we

compare our proposed Screened DE-SCA method with baseline methods that solely rely on DE or SCA and the DE-SCA method.

Feasibility and Optimality Tradeoff for DE

With current simulation settings of 10 IoT devices and 4 ECs, there are 112 constraints in total including 10 equality constraints. A large number of constraints combining with equality constraints generally generally makes it impossible for the DE algorithm to find the optimal solutions within given time limits. Therefore, it is necessary to relax the constraint requirements by introducing the ε approximation [101] where a large ε represents more tolerance we allow for constraint violation. Table 9 summarizes how optimal system cost of problem (6.3.14) changes as the ε decreases, where $G_{eq}(\boldsymbol{x})$ denotes the degree of constraint violation by the equality constraints (6.3.14d) and η defines the ratio of $G_{eq}(\boldsymbol{x})$ and $G(\boldsymbol{x})$. We

Table 9: System cost vs. ε for DE Algorithm

ε	$G(\boldsymbol{x})$	$G_{eq}(\boldsymbol{x})$	$\eta \triangleq G_{eq}(\boldsymbol{x})/G(\boldsymbol{x})$	System Cost
10	9.98	7.95	79.64%	8.10
5	4.94	4.49	90.76%	18.24
1	0.97	0.45	46.48%	42.04
0.5	0.46	0.20	43.95%	64.63
0.1	0.19	0.10	54.76%	86.87
0.05	0.14	0.06	44.00%	117.19
0.01	0.17	0.08	48.81%	149.81

observe that when $\varepsilon \geq 0.5$, the degree of constraint violation $G(\boldsymbol{x})$ will be strictly less than ε but when $\varepsilon < 0.5$, ε approximation will be violated as $G(\boldsymbol{x})$ are larger than ε. Moreover, the relationship between ε and optimal system cost is plotted in Figure 24. We observe that the system cost generally decreases as the value of ε increases since there are less restrictions on meeting all the constraints. In other words, a tradeoff exists between feasibility and optimality for DE algorithm. The less feasible the solutions are, the better system cost we can achieve. However, all solutions under the given ε values are actually infeasible as $G(\boldsymbol{x})$ are non-zero. Note that in solving real problems, the feasibility of solutions should

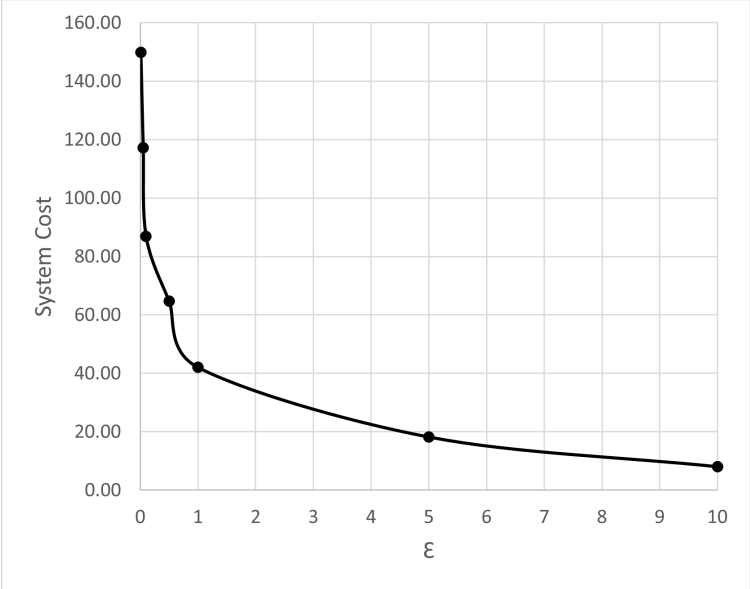


Figure 24: System cost of problem (6.3.14) as a function of ε for DE algorithm.

be firstly satisfied before considering the optimality. Therefore, the DE algorithm with ε approximation can generate a worse system cost if we want the feasibility of our solutions to be better considered.

DE Variants for Constraint Handling

In this section, we will compare the original DE algorithm with its two variants considering different constraint handling techniques in terms of reducing the system cost. For DE-PF method, we obtain the adjusted objective function by augmenting the original objective function with $G(\boldsymbol{x})$ since in a minimization sense, we want both the objective function value and $G(\boldsymbol{x})$ to be as small as possible. According to the value of η in Table 9, we found that around 58.34% on average of the degree of constraint violation is caused by the equality constraints. Therefore, in DE-IN method, we force the equality constraints to be satisfied after random DE initialization by normalization:

$$\alpha'_{i0} = \frac{\alpha_{i0}}{\alpha_{i0} + \sum_{j=1}^{J} \alpha_{ij}}, \ \forall i \tag{6.4.1a}$$

$$\alpha'_{ij} = \frac{\alpha_{ij}}{\alpha_{i0} + \sum_{j=1}^{J} \alpha_{ij}}, \ \forall i, j, \tag{6.4.1b}$$

where we can guarantee that the equality constraints hold for the normalized task partitioning variables α'_{i0} and α'_{ij}. Besides, we can also require that the initialized decision variables for all individuals must satisfy all inequality constraints before we proceed towards mutation stage. However, one can easily get trapped in this initialize-check feasibility-reinitialize endless loop, given the large number of constraints considered in our problem. For simplicity, we set ε to 1 and summarize the system cost comparison results in Table 10. We observe that given $\varepsilon = 1$, DE-IN algorithm has the lowest system cost while DE-PF algorithm has the smallest $G(\boldsymbol{x})$. According to the feasibility rules, if one decision vector is feasible and the other one i infeasible, the feasible one is selected. Therefore, we will select the solutions of DE-PF since it has the smallest degree of constraint violation among all three methods.

Table 10: System cost comparison for different DE algorithms when $\varepsilon = 1$ where DE-PF and DE-IN stand for DE with Penalty Function and DE with Initialization Normalization, respectively

Methods	ε	$G(\boldsymbol{x})$	System Cost
DE	1	0.97	42.04
DE-PF	1	0.84	37.83
DE-IN	1	1.00	22.67

Effectiveness of Screened DE-SCA

In this section, we will demonstrate the effectiveness of our proposed Screened DE-SCA algorithm in terms of reducing the system cost of problem (6.3.14) compared to other baseline methods. We will also verify why our method can lead to better local optimal solutions.

Cost comparison among Screened DE-SCA methods: As discussed in Section 6.4.2, the solutions obtained by DE algorithm are indeed infeasible no matter how we select the value of ε. Therefore, we cannot initialize the SCA-based algorithm with full DE solutions since it requires the initialization to be feasible. In contrast, we will only choose the partial parts $\boldsymbol{x}^{\mathrm{DE}}_{\mathcal{K},*}$ of the solution found by DE to perform the initialization. To validate why DE-PF method is the best one to initialize the SCA-based method, we compare the system cost of Screened DE-SCA, Screened DE-IN-SCA and Screened DE-PF-SCA where the SCA-based

method is initialized by the feasible parts of DE, DE-IN and DE-PF, respectively. As shown in Table 11, the Screened DE-PF-SCA method can achieve the lowest system cost and thus it demonstrates the superiority of DE-PF over DE and DE-IN methods as a means to initialize the SCA-based method.

Table 11: System cost comparison for Screened DE-SCA algorithms where DE, DE-IN and DE-PF are used to initialize the SCA-based method, respectively

Methods	System Cost	Energy Cost	Delay Cost
Screened DE-SCA	39.09	1.89	7.44
Screened DE-IN-SCA	44.26	2.16	8.42
Screened DE-PF-SCA	32.32	1.68	6.13

Initialization feasibility over the iterates: Apart from initializing the SCA-based method with feasible solution at the first iteration, we also require that the solutions to the approximated convex problem to be feasible at each iteration since this algorithm runs in an iterative fashion until a stationary solution is found. In other words, the found solutions at the $(k-1)$th iteration are used to initialize the approximated convex problem at the kth iteration. Table 12 summarizes the initialization feasibility comparison between SCA-based and Screened DE-SCA methods, where "#Iterations" denotes the average number of iterations needed to obtain a stationary solution while $\bar{G}(\boldsymbol{x})$ denotes the average degree of constraint violation over all iterations. We observe that the Screened DE-SCA only takes 11 iterations on average, which is twice faster than the SCA-based method in achieving a stationary solution. Besides, we observe that $\bar{G}(\boldsymbol{x})$ of our method is approximately zero across all iterates, which implies the solutions found at all iterates are feasible. However, $\bar{G}(\boldsymbol{x})$ of 0.77 in SCA-based method indicates that some of the iterative solutions found are indeed infeasible. Therefore, we conclude that because of the initialization feasibility guaranteed at each iterate, our proposed method can generate better local optimal solutions compared to the SCA-based method.

Cost comparison with benchmark methods: The system cost comparison with other benchmark methods are given in Table 13, where the DE-PF method with $\varepsilon = 1$ is selected for

Table 12: Initialization feasibility comparison for SCA-based and Screened DE-SCA algorithms initialized by DE-PF method

Methods	#Iterations	$\bar{G}(\boldsymbol{x})$
SCA-based	23	0.77
Screened DE-SCA	11	10^{-4}

initialization as its system cost is comparable to that of the SCA-based algorithm. Note that all three methods SCA-based, DE-SCA and Screened DE-SCA will generate feasible solutions as the SCA technique can transform the original nonconvex problem into a suitable convex surrogate and then it can be successfully solved by optimization software such as Gurobi and CVX. From this table, we observe that the DE-SCA method does not generate a better system cost as the infeasible solutions of DE-PF violate the feasibility requirements for SCA-based algorithm initialization. However, our proposed Screened DE-SCA method using feasible parts of DE-PF solution for initialization can improve the system cost by 6.21%, 15.77% and 27.35% compared to the SCA-based, UMEC and LCPSO algorithms, respectively, which validates its effectiveness in terms of system cost reduction.

Table 13: System cost comparison for Screened DE-SCA using feasible parts of DE-PF solution for initialization with other benchmark methods

Methods	System Cost	Energy Cost	Delay Cost
DE-SCA	110.76	0.87	21.98
LCPSO [100]	44.49	3.44	8.21
UMEC [80]	38.37	2.33	7.28
SCA-based	34.46	1.59	6.57
Screened DE-SCA	32.32	1.68	6.13

Discussions

In this section, we will discuss the potential difficulties of applying the screened DE-SCA method and the future work.

Time complexity: The total number of decision variables considered in our problem can be calculated as $3N + 2NJ + 2$ where N and J are the number of IoT devices and ECs, respectively. In simulation, we set N to 10 and J to 4 so there are 112 decision variables in

total. For the DE method, the time complexity can be denoted as $\mathcal{O}(MaxIter*NP*(N+J))$ where $MaxIter$ and NP denote the pre-defined maximum DE iterations and population size. Suppose $MaxIter$ and NP remain constant in our experiments, the time complexity can be reduced to $\mathcal{O}(N+J)$, which has linear running time. As for the SCA-based method, the reformulated problem can be solved by the interior-point optimizer such as SeDuMi and the time complexity of this optimizer can be denoted as $\mathcal{O}(ND^3)$ where ND is the number of decision variables $3N + 2NJ + 2$. Therefore, the SCA-based method has polynomial running time, which is "tractable" and "fast" according to Cobham's thesis [40]. In summary, the total time complexity of our proposed method will be $\mathcal{O}(N+J) + \mathcal{O}(ND^3)$, which is polynomial and thus tractable.

Hyperparameters: Both DE and SCA-based methods are dependent on the hyperparameter initialization such as population size, differential scaling factor and crossover rate in DE and step size in SCA-based method. These hyperparameters needs to be tuned by lots of trial and error as there is no theoretic evidence on how to calculate the values of them.

Convex approximation: We shall note that not all nonconvex optimization problems can be approximated by suitable convex forms. In our problem setting, there are mainly two types of nonconvex constraints that can be represented by either difference of functions or product of functions. According to the function approximation examples in Section 6.1.3, they can be successfully converted into convex forms. More candidate nonconvex constraints and objective functions can be found in [91, 92].

Future work: There are still some open questions on how to determine qualified DE solutions to initialize the SCA-based method. First, we can investigate which mutation scheme and constraint-handing technique will jointly work the best in finding the solutions. Second, we can define different metrics to evaluate the quality of the DE solutions even if the DE algorithm cannot converge within time limits. Third, we can design different schemes to screen the best DE solutions such that the initialization of the SCA-based method will lead to better solutions.

6.5 Summary

In this article, we have proposed an innovative Screened DE-SCA algorithm by jointly considering both the classical DE algorithm and SCA-based algorithm. This algorithm can not only overcome the lack of feasibility guarantee in DE algorithm but also the inability of working on original nonconvex problems in SCA-based algorithm. Specifically, this algorithm unifies DE and SCA techniques such that the feasible parts of the solutions of an original nonconvex problem found by DE algorithm can be used to initialize the SCA-based algorithm on its approximated convex surrogate. We have utilized a UAV-enabled MEC system that involves the interactions among IoT devices, UAV and ECs to validate the effectiveness of our proposed algorithm as the formulated problem therein is highly nonconvex. Through extensive experiments, we have verified that our proposed Screened DE-SCA algorithm largely outperforms benchmarks including DE, SCA-based and state-of-the-art algorithms to solve the formulated problem in the UAV-enabled MEC system. In the future, we will extend our proposed algorithm to solve the optimization problems incurred in a UAV-enabled MEC system that contains multiple UAVs.

CHAPTER VII

CONCLUDING REMARKS

7.1 Conclusions

In this dissertation, we have considered resource management for cost-effective cloud and edge systems. For multi-tenant colocation datacenter, we have proposed to use aggregation-based approach to procure power collectively in the wholesale electricity market such that the total energy cost can be minimized. A novel cost allocation scheme based on the marginal contribution of each tenant to the total expected cost have been proposed to fairly distribute the aggregation benefits among participating tenants. For geographically distributed datacenters, we have proposed to jointly optimize electricity bills and bandwidth cost associated with workload management from both demand and supply side when participating in multi-timescale electricity market by utilizing local renewable energy sources and energy storage systems. We have formulated a two-stage stochastic optimization problem by considering random scenarios of interactive workload demand, renewable generation and real-time electricity prices, and then reformulate as its deterministic equivalent problem. Experimental results based on real-world traces have shown that our proposed cost allocation scheme and stochastic optimization algorithm can effectively reduce the overall cost for datacenters participating in electricity markets. We have further studied the task offloading for UAV-enabled mobile edge computing systems with the goal of minimizing the weighted sum of total energy consumption of the UAV when serving the computation and communication needs of the mobile users when they choose to offload tasks and the overall delay of all mobile users. We have also proposed a novel method that combines both DE and SCA-based

algorithms. Through extensive experiments, we have verified that our proposed Screened DE-SCA algorithm largely outperforms the baseline algorithms that solely rely on DE or SCA to solve the formulated problem in the UAV-enabled MEC system.

7.2 Future Works

In Chapters III and IV, we have discussed the resource management with the goal of energy cost minimization in the context of cloud computing systems considering multi-agent colocation datacenter and geographically distributed datacenters, respectively. In Chapters V and VI, we have also discussed how resource management with the goal of overall cost minimization in the context of UAV-enabled mobile edge computing systems and proposed a novel Screened DE-SCA optimization algorithm. Although we have theoretically verified the correctness of our mathematical modeling and proposed algorithms, we could foresee some practical issues in order to fully apply our research work in the industry. Apart from applying optimization and evolutionary computation methods in UAV-enabled MEC, We are further interested in applying game-theoretic and deep reinforcement learning-based methods. Two future research topics including selfish task offloading in mobile edge computing systems with the goal of finding a Nash equilibrium solution to the formulated game, where no mobile users can be better off by changing strategies while the other mobile users fix their strategies, and autonomous task offloading and resource allocation by deep reinforcement learning for UAV-enabled mobile edge computing systems with the goal of minimizing the weighted sum of total energy consumption of the UAV when serving the computation and communication needs of the mobile users when they choose to offload tasks and the overall delay of all mobile users, are introduced as follows.

7.2.1 Practical Issues of Our Research Work

For the research task of energy management in colocation datacenters, it may be difficult for tenants to form a coalition due to privacy and security concerns. They can be reluctant to

disclose their energy demand profile to potential competitors. To tackle this issue, we can designate a trustworthy arbitrator who will collect the energy demand of all participating tenants and then split the cooperative energy cost to each tenant. Second, the realized cost can be very different from the expected cost on a given time interval and may not be distributed in a satisfactory way for each tenant [20]. Therefore, we need to design an alternative cost allocation scheme such that it allows for a satisfactory distribution of the realized cost among the tenants, and the cost allocation lies in the core of the cooperative game of realized cost.

For the research task of joint task offloading and resource allocation in UAV-enabled mobile edge computing, we need to further conduct a sensitivity analysis of the constant ρ in the objective function (5.1.14a). This constant represents the relative weight of UAV energy consumption and service delay. By performing such analysis, we can gain more insights into how the choice of ρ will impact the total cost and whether we should put more emphasis on energy consumption or network latency. Besides, the flying time of the UAV should be taken into account as the UAV battery size can be the bottleneck especially when the considered system is deployed in a large space. Adding the UAV flying time constraints as well as the UAV trajectory planning will further complicate our model but it represents a more realistic scenario.

7.2.2 Selfish Task Offloading in Mobile Edge Computing Systems

We consider a three-tier architecture for mobile offloading scenario as illustrated in Figure 25, which consists of local layer of mobile users, middle layer of computing nodes in the proximity of mobile users, typically characterized by a limited amount of resources, and remote layer of distant cloud servers, which have relatively infinite resources. For this architecture, we consider a decentralized scenario where multiple non-cooperative mobile users share the limited computing resources of nearby computing access points and can selfishly decide how and where to offload (part of) their computational tasks in a three-tier computing

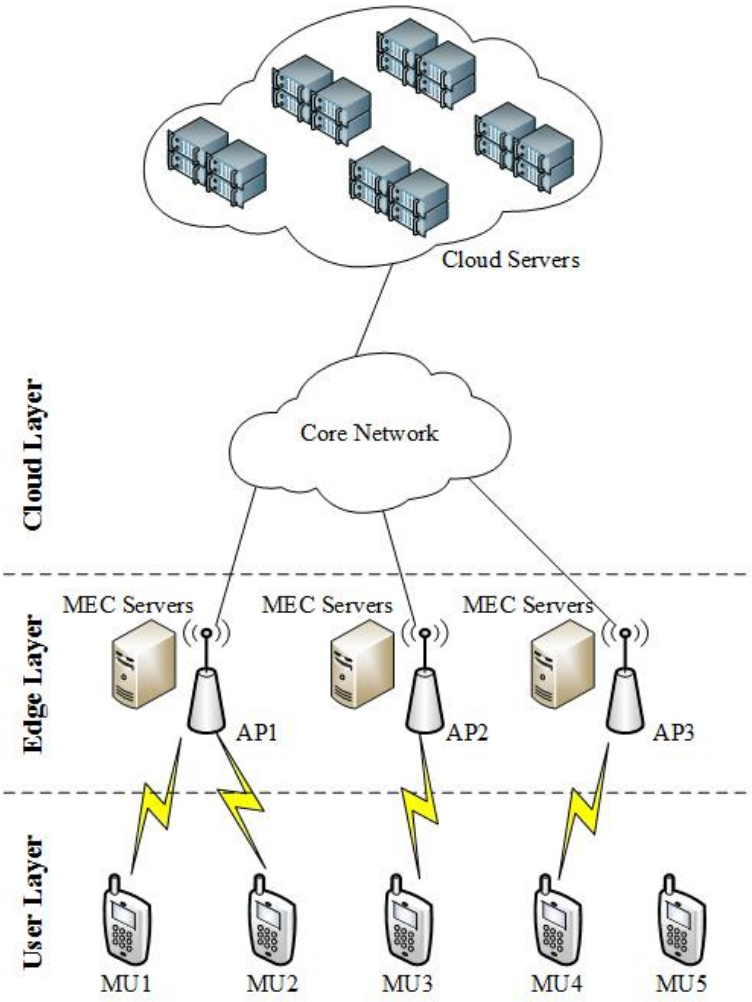


Figure 25: An example of a mobile edge computing system including 5 mobile users, 3 access points and a remote cloud, where both MU1 and MU2 offload tasks via AP1, MU3 offloads tasks via AP2, and MU4 offloads tasks via AP3 while MU5 performs local computing without offloading.

architecture with multiple access points. We assume each mobile user is selfish and rational aiming to minimize their own cost in the mobile edge computing system, which consists of weighted sum of the total delay and energy consumption. A strategic form game defined by players, strategies and cost function is formulated to model this problem. Our goal is to find a Nash equilibrium solution to the formulated game, where no mobile users can be better off by changing strategies while the other mobile users fix their strategies.

7.2.3 Deep Reinforcement Learning for Joint Task Offloading and Resource Allocation in UAV-Enabled Mobile Edge Computing

Reinforcement learning (RL) can be employed in a UAV-enabled MEC to learn the optimal policy in the absence of labelled training data by interacting with MEC environment. Specifically, RL seeks to maximize the expected cumulative reward of the agents in an environment under the optimal policy taken by the agents. Classical RL results are stored in a Q-table that consists of states, actions and rewards. However, it will become incapable when the state space or action space is huge. Thanks to the epoch-making development of deep learning, deep reinforcement learning (DRL) has been widely used to approximate the Q values with the assistance of deep neural networks without explicitly designing the state space.

We will consider the similar UAV-enabled MEC system that is discussed in Sections V and VI. In the meanwhile, the UAV is allowed to move freely in the 3D architecture but subject to the SWAP limitations. Therefore, our problem can be reformulated in the framework of DRL: given current states of UAV battery level, available computation resources and communication bandwidth, we can optimize the actions of UAV trajectory, task offloading ratio and resource allocation such that the reward (negative overall system cost) of the UAV-enabled MEC system can be maximized. However, it is challenging to solve our problem using DRL. On one hand, the constrains in the formulated problem are highly convex, and there is no guarantee that all of them can be strictly satisfied by applying the DRL. The optimal policy can be infeasible in practice. On the other hand, there can be infinite combinations of action when the actions can take on continuous values. This issue can significantly increase the time complexity of the DRL, which makes DRL unable to handle the combinatorial action space.

REFERENCES

[1] *California ISO*, http://www.caiso.com/.

[2] *Cloud computing companies*, https://www.datamation.com/cloud-computing/cloud-computing-companies.html.

[3] *Edge computing vs. cloud computing: What you need to know*, https://www.vxchnge.com/blog/edge-computing-vs-cloud-computing.

[4] *Electric reliability council of texas*, http://www.ercot.com/.

[5] *Exploring mobile cloud computing*, https://www.esds.co.in/blog/exploring-mobile-cloud-computing-esds/#sthash.v1ekjcfd.dpbs.

[6] *Fauna*, https://fauna.com/blog/edge-computing-vs-cloud-computing-whats-the-difference.

[7] *Measurement and instrumentation data center (MIDC)*, http://www.nrel.gov/midc/.

[8] *PJM - markets & operations*, https://www.pjm.com/markets-and-operations.aspx.

[9] *Google gets go-ahead to buy, sell energy*, https://www.cnet.com/news/google-gets-go-ahead-to-buy-sell-energy/, Feb. 2010.

[10] Alia Asheralieva and Dusit Niyato, *Hierarchical game-theoretic and reinforcement learning framework for computational offloading in UAV-enabled mobile edge computing networks with multiple service providers*, IEEE Internet of Things Journal **6** (2019), no. 5, 8753–8769.

[11] Muhammad Asim, Wali Khan Mashwani, Habib Shah, and Samir Brahim Belhaouari, *An evolutionary trajectory planning algorithm for multi-UAV-assisted MEC system*, Soft Computing (2021), 1–14.

[12] Enrique Baeyens, Eilyan Y Bitar, Pramod P Khargonekar, and Kameshwar Poolla, *Coalitional aggregation of wind power*, IEEE Transactions on Power Systems **28** (2013), no. 4, 3774–3784.

[13] Sergio Barbarossa, Stefania Sardellitti, and Paolo Di Lorenzo, *Joint allocation of computation and communication resources in multiuser mobile cloud computing*, Signal Processing Advances in Wireless Communications (SPAWC), 2013 IEEE 14th Workshop on, 2013, pp. 26–30.

[14] Marco V Barbera, Sokol Kosta, Alessandro Mei, and Julinda Stefa, *To offload or not to offload? the bandwidth and energy costs of mobile cloud computing*, IEEE INFOCOM, 2013, pp. 1285–1293.

[15] Luiz André Barroso and Urs Hölzle, *The datacenter as a computer: An introduction to the design of warehouse-scale machines*, Synthesis lectures on computer architecture **4** (2009), no. 1, 1–108.

[16] Eilyan Y Bitar, Ram Rajagopal, Pramod P Khargonekar, Kameshwar Poolla, and Pravin Varaiya, *Bringing wind energy to market*, IEEE Transactions on Power Systems **27** (2012), no. 3, 1225–1235.

[17] Stephen Boyd and Lieven Vandenberghe, *Convex optimization*, Cambridge university press, 2004.

[18] Xianbin Cao, Peng Yang, Mohamed Alzenad, Xing Xi, Dapeng Wu, and Halim Yanikomeroglu, *Airborne communication networks: A survey*, IEEE Journal on Selected Areas in Communications **36** (2018), no. 9, 1907–1926.

[19] Valeria Cardellini, Vittoria De Nitto Personé, Valerio Di Valerio, Francisco Facchinei, Vincenzo Grassi, Francesco Lo Presti, and Veronica Piccialli, *A game-theoretic approach to computation offloading in mobile cloud computing*, Mathematical Programming **157** (2016), no. 2, 421–449.

[20] Pratyush Chakraborty, Enrique Baeyens, Pramod P Khargonekar, and Kameshwar Poolla, *A cooperative game for the realized profit of an aggregation of renewable energy producers*, 2016 IEEE 55th Conference on Decision and Control (CDC), IEEE, 2016, pp. 5805–5812.

[21] Changbing Chen, Bingsheng He, and Xueyan Tang, *Green-aware workload scheduling in geographically distributed data centers*, Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on, IEEE, 2012, pp. 82–89.

[22] Lijun Chen and Na Li, *On the interaction between load balancing and speed scaling*, IEEE Journal on Selected Areas in Communications **33** (2015), no. 12, 2567–2578.

[23] Meng-Hsi Chen, Min Dong, and Ben Liang, *Resource sharing of a computing access point for multi-user mobile cloud offloading with delay constraints*, IEEE Transactions on Mobile Computing **17** (2018), no. 12, 2868–2881.

[24] Meng-Hsi Chen, Ben Liang, and Min Dong, *Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point*, IEEE INFOCOM, 2017, pp. 1–9.

[25] Min Chen and Yixue Hao, *Task offloading for mobile edge computing in software defined ultra-dense network*, IEEE Journal on Selected Areas in Communications **36** (2018), no. 3, 587–597.

[26] Xu Chen, *Decentralized computation offloading game for mobile cloud computing*, IEEE Transactions on Parallel and Distributed Systems **26** (2015), no. 4, 974–983.

[27] Xu Chen, Lei Jiao, Wenzhong Li, and Xiaoming Fu, *Efficient multi-user computation offloading for mobile-edge cloud computing*, IEEE/ACM Transactions on Networking (ToN) **24** (2016), no. 5, 2795–2808.

[28] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti, *Clonecloud: elastic execution between mobile device and cloud*, Proceedings of the sixth conference on Computer systems, ACM, 2011, pp. 301–314.

[29] Carlos A Coello Coello, *Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art*, Computer Methods in Applied Mechanics and Engineering **191** (2002), no. 11-12, 1245–1287.

[30] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl, *Maui: making smartphones last longer with code offload*, Proceedings of the 8th international conference on Mobile systems, applications, and services, ACM, 2010, pp. 49–62.

[31] Kalyanmoy Deb, *An efficient constraint handling method for genetic algorithms*, Computer Methods in Applied Mechanics and Engineering **186** (2000), no. 2-4, 311–338.

[32] Hoang T Dinh, Chonho Lee, Dusit Niyato, and Ping Wang, *A survey of mobile cloud computing: architecture, applications, and approaches*, Wireless communications and mobile computing **13** (2013), no. 18, 1587–1611.

[33] Theo SH Driessen, *Cooperative games, solutions and applications*, vol. 3, Springer Science & Business Media, 2013.

[34] Wanmei Feng, Jie Tang, Nan Zhao, Xiuyin Zhang, Xianbin Wang, and Kai-Kit Wong, *A deep learning-based approach to resource allocation in UAV-aided wireless powered MEC networks*, ICC 2021-IEEE International Conference on Communications, IEEE, 2021, pp. 1–6.

[35] Mahdi Ghamkhari and Hamed Mohsenian-Rad, *Optimal integration of renewable energy resources in data centers with behind-the-meter renewable generator*, Communications (ICC), 2012 IEEE International Conference on, IEEE, 2012, pp. 3340–3344.

[36] Mahdi Ghamkhari, Hamed Mohsenian-Rad, and Adam Wierman, *Optimal risk-aware power procurement for data centers in day-ahead and real-time electricity markets*, Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on, IEEE, 2014, pp. 610–615.

[37] Mahdi Ghamkhari, Adam Wierman, and Hamed Mohsenian-Rad, *Energy portfolio optimization of data centers*, IEEE Transactions on Smart Grid (2016).

[38] ———, *Energy portfolio optimization of data centers*, IEEE Transactions on Smart Grid **8** (2017), no. 4, 1898–1910.

[39] James Glanz, *Power, pollution and the internet*, `https://www.nytimes.com/2012/09/23/technology/data-centers-waste-vast-amounts-of-energy-belying-industry-image.html`, 2012.

[40] Oded Goldreich, *Computational complexity: a conceptual perspective*, ACM Sigact News **39** (2008), no. 3, 35–39.

[41] Brian Guenter, Navendu Jain, and Charles Williams, *Managing cost, performance, and reliability tradeoffs for energy-aware server provisioning*, INFOCOM, 2011 Proceedings IEEE, IEEE, 2011, pp. 1332–1340.

[42] Songtao Guo, Bin Xiao, Yuanyuan Yang, and Yang Yang, *Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing*, IEEE INFOCOM, 2016, pp. 1–9.

[43] Yuanxiong Guo, Zongrui Ding, Yuguang Fang, and Dapeng Wu, *Cutting down electricity cost in internet data centers by using energy storage*, Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE, IEEE, 2011, pp. 1–5.

[44] Yuanxiong Guo and Yuguang Fang, *Electricity cost saving strategy in data centers by using energy storage*, IEEE Transactions on Parallel and Distributed Systems **24** (2013), no. 6, 1149–1160.

[45] Yuanxiong Guo, Yanmin Gong, Yuguang Fang, Pramod P Khargonekar, and Xiaojun Geng, *Energy and network aware workload management for sustainable data centers with thermal storage*, IEEE Transactions on Parallel and Distributed Systems **25** (2014), no. 8, 2030–2042.

[46] Songyue Han, Dawei Ma, Chao Kang, Wei Huang, Chaoying Lin, and Chunyuan Tian, *Optimization of mobile edge computing offloading model for distributed wireless sensor devices*, Journal of Sensors **2022** (2022).

[47] Qiyu Hu, Yunlong Cai, Guanding Yu, Zhijin Qin, Minjian Zhao, and Geoffrey Ye Li, *Joint offloading and trajectory design for UAV-enabled mobile edge computing systems*, IEEE Internet of Things Journal **6** (2018), no. 2, 1879–1892.

[48] Xiaoyan Hu, Kai-Kit Wong, Kun Yang, and Zhongbin Zheng, *UAV-assisted relaying and edge computing: Scheduling and trajectory optimization*, IEEE Transactions on Wireless Communications **18** (2019), no. 10, 4738–4752.

[49] Dong Huang, Ping Wang, and Dusit Niyato, *A dynamic offloading algorithm for mobile computing*, IEEE Transactions on Wireless Communications **11** (2012), no. 6, 1991–1995.

[50] Pei-qiu Huang, Yong Wang, and Ke-zhi Wang, *Energy-efficient trajectory planning for a multi-UAV-assisted mobile edge computing system*, Frontiers of Information Technology & Electronic Engineering **21** (2020), no. 12, 1713–1725.

[51] Pei-Qiu Huang, Yong Wang, Kezhi Wang, and Kun Yang, *Differential evolution with a variable population size for deployment optimization in a UAV-assisted IoT data collection system*, IEEE Transactions on Emerging Topics in Computational Intelligence **4** (2020), no. 3, 324–335.

[52] Mohammad A Islam, Anshul Gandhi, and Shaolei Ren, *Minimizing electricity cost for geo-distributed interactive services with tail latency constraint.*, IGSC, 2016, pp. 1–8.

[53] Brendan Jennings and Rolf Stadler, *Resource management in clouds: Survey and research challenges*, Journal of Network and Systems Management **23** (2015), no. 3, 567–619.

[54] Seongah Jeong, Osvaldo Simeone, and Joonhyuk Kang, *Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning*, IEEE Transactions on Vehicular Technology **67** (2018), no. 3, 2049–2063.

[55] Sladana Josilo and Gyorgy Dan, *A game theoretic analysis of selfish mobile computation offloading*, IEEE INFOCOM, 2017.

[56] Rakpong Kaewpuang, Dusit Niyato, Ping Wang, and Ekram Hossain, *A framework for cooperative resource management in mobile cloud computing*, IEEE Journal on Selected Areas in Communications **31** (2013), no. 12, 2685–2700.

[57] Dileep Kalathil, Chenye Wu, Kameshwar Poolla, and Pravin Varaiya, *The sharing economy for the smart grid*, arXiv preprint arXiv:1608.06990 (2016).

[58] Yi-Hsuan Kao, Bhaskar Krishnamachari, Moo-Ryong Ra, and Fan Bai, *Hermes: Latency optimal task assignment for resource-constrained mobile computing*, IEEE Transactions on Mobile Computing **16** (2017), no. 11, 3056–3069.

[59] Mahdi Kohansal and Hamed Mohsenian-Rad, *Price-maker economic bidding in two-settlement pool-based markets: The case of time-shiftable loads*, IEEE Transactions on Power Systems **31** (2016), no. 1, 695–705.

[60] Anis Koubâa, Adel Ammar, Mahmoud Alahdab, Anas Kanhouch, and Ahmad Taher Azar, *Deepbrain: Experimental evaluation of cloud-based computation offloading and edge computing in the internet-of-drones for deep learning applications*, Sensors **20** (2020), no. 18, 5240.

[61] Karthik Kumar, Jibang Liu, Yung-Hsiang Lu, and Bharat Bhargava, *A survey of computation offloading for mobile systems*, Mobile Networks and Applications **18** (2013), no. 1, 129–140.

[62] Jouni Lampinen, *A constraint handling approach for the differential evolution algorithm*, Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600), vol. 2, IEEE, 2002, pp. 1468–1473.

[63] Tan N Le, Jie Liang, Zhenhua Liu, Ramesh K Sitaraman, Jayakrishnan Nair, and Bong Jun Choi, *Optimal energy procurement for geo-distributed data centers in multi-timescale electricity markets*, ACM SIGMETRICS Performance Evaluation Review **45** (2017), no. 2, 58–63.

[64] Minghong Lin, Zhenhua Liu, Adam Wierman, and Lachlan LH Andrew, *Online algorithms for geographical load balancing*, Green Computing Conference (IGCC), 2012 International, IEEE, 2012, pp. 1–10.

[65] Minghong Lin, Adam Wierman, Lachlan LH Andrew, and Eno Thereska, *Online dynamic capacity provisioning in data centers*, Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on, IEEE, 2011, pp. 1159–1163.

[66] ———, *Dynamic right-sizing for power-proportional data centers*, IEEE/ACM Transactions on Networking (TON) **21** (2013), no. 5, 1378–1391.

[67] Zhenhua Liu, Yuan Chen, Cullen Bash, Adam Wierman, Daniel Gmach, Zhikui Wang, Manish Marwah, and Chris Hyser, *Renewable and cooling aware workload management for sustainable data centers*, ACM SIGMETRICS Performance Evaluation Review, vol. 40, ACM, 2012, pp. 175–186.

[68] Zhenhua Liu, Minghong Lin, Adam Wierman, Steven H Low, and Lachlan LH Andrew, *Geographical load balancing with renewables*, ACM SIGMETRICS Performance Evaluation Review **39** (2011), no. 3, 62–66.

[69] ———, *Greening geographical load balancing*, Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems, ACM, 2011, pp. 233–244.

[70] Zhenhua Liu, Iris Liu, Steven Low, and Adam Wierman, *Pricing data center demand response*, ACM SIGMETRICS Performance Evaluation Review **42** (2014), no. 1, 111–123.

[71] Weidang Lu, Yu Ding, Yuan Gao, Su Hu, Yuan Wu, Nan Zhao, and Yi Gong, *Resource and trajectory optimization for secure communications in dual unmanned aerial vehicle mobile edge computing systems*, IEEE Transactions on Industrial Informatics **18** (2021), no. 4, 2704–2713.

[72] Zhoujia Mao, Can Emre Koksal, and Ness B Shroff, *Near optimal power and rate control of multi-hop sensor networks with energy replenishment: Basic limitations with finite energy and data storage*, IEEE Transactions on Automatic Control **57** (2011), no. 4, 815–829.

[73] Mohamed H Mousa and Mohamed K Hussein, *Efficient UAV-based mobile edge computing using differential evolution and ant colony optimization*, PeerJ Computer Science **8** (2022), e870.

[74] Mohammad Mozaffari, Walid Saad, Mehdi Bennis, and Mérouane Debbah, *Mobile unmanned aerial vehicles (UAVs) for energy-efficient internet of things communications*, IEEE Transactions on Wireless Communications **16** (2017), no. 11, 7574–7589.

[75] Mohammad Mozaffari, Walid Saad, Mehdi Bennis, Young-Han Nam, and Mérouane Debbah, *A tutorial on UAVs for wireless networks: Applications, challenges, and open problems*, IEEE Communications Surveys & Tutorials **21** (2019), no. 3, 2334–2360.

[76] Roger B Myerson, *Game theory: analysis of conflict*, Harvard University (1991).

[77] Michael J Neely, *Intelligent packet dropping for optimal energy-delay tradeoffs in wireless downlinks*, IEEE Transactions on Automatic Control **54** (2009), no. 3, 565–579.

[78] Martin J Osborne and Ariel Rubinstein, *A course in game theory*, MIT press, 1994.

[79] Guillermo Owen, *Game theory*, Emerald Group, 2013.

[80] Zhen Qin, Hai Wang, Zhenhua Wei, Yuben Qu, Fei Xiong, Haipeng Dai, and Tao Wu, *Task selection and scheduling in UAV-enabled MEC for reconnaissance with time-varying priorities*, IEEE Internet of Things Journal **8** (2021), no. 24, 17290–17307.

[81] Yuben Qu, Haipeng Dai, Haichao Wang, Chao Dong, Fan Wu, Song Guo, and Qihui Wu, *Service provisioning for UAV-enabled mobile edge computing*, IEEE Journal on Selected Areas in Communications **39** (2021), no. 11, 3287–3305.

[82] M Reza Rahimi, Nalini Venkatasubramanian, Sharad Mehrotra, and Athanasios V Vasilakos, *Mapcloud: mobile applications on an elastic and scalable 2-tier cloud architecture*, Utility and Cloud Computing (UCC), 2012 IEEE Fifth International Conference on, 2012, pp. 83–90.

[83] Lei Rao, Xue Liu, Le Xie, and Wenyu Liu, *Minimizing electricity cost: optimization of distributed internet data centers in a multi-electricity-market environment*, INFOCOM, 2010 Proceedings IEEE, IEEE, 2010, pp. 1–9.

[84] Lei Rao, Xue Liu, Le Xie, and Zhan Pang, *Hedging against uncertainty: A tale of internet data center operations under smart grid environment*, IEEE Transactions on Smart Grid **2** (2011), no. 3, 555–563.

[85] Meisam Razaviyayn, *Successive convex approximation: Analysis and applications*, Ph.D. thesis, University of Minnesota, 2014.

[86] Charles Reiss, John Wilkes, and Joseph L Hellerstein, *Google cluster-usage traces: format+ schema*, Google Inc., White Paper (2011), 1–14.

[87] Jinke Ren, Guanding Yu, Yinghui He, and Ye Li, *Collaborative cloud and edge computing for latency minimization*, IEEE Transactions on Vehicular Technology **68** (2019), no. 5, 5031–5044.

[88] Walid Saad, Zhu Han, Mérouane Debbah, Are Hjorungnes, and Tamer Basar, *Coalitional game theory for communication networks*, IEEE Signal Processing Magazine **26** (2009), no. 5, 77–97.

[89] Jayaram K Sankaran, *On finding the nucleolus of an n-person cooperative game*, International Journal of Game Theory **19** (1991), no. 4, 329–338.

[90] Stefania Sardellitti, Gesualdo Scutari, and Sergio Barbarossa, *Joint optimization of radio and computational resources for multicell mobile-edge computing*, IEEE Transactions on Signal and Information Processing over Networks **1** (2015), no. 2, 89–103.

[91] Gesualdo Scutari, Francisco Facchinei, and Lorenzo Lampariello, *Parallel and distributed methods for constrained nonconvex optimization—part I: Theory*, IEEE Transactions on Signal Processing **65** (2017), no. 8, 1929–1944.

[92] Gesualdo Scutari, Francisco Facchinei, Lorenzo Lampariello, Stefania Sardellitti, and Peiran Song, *Parallel and distributed methods for constrained nonconvex optimization-part II: Applications in communications and machine learning*, IEEE Transactions on Signal Processing **65** (2016), no. 8, 1945–1960.

[93] Lloyd S Shapley, *On balanced sets and cores*, Naval research logistics quarterly **14** (1967), no. 4, 453–460.

[94] ———, *Cores of convex games*, International journal of game theory **1** (1971), no. 1, 11–26.

[95] Gaurav Sharma, Ravi Mazumdar, and Ness B Shroff, *Delay and capacity trade-offs in mobile ad hoc networks: A global perspective*, IEEE/ACM Transactions on Networking (ToN) **15** (2007), no. 5, 981–992.

[96] Arman Shehabi, Sarah Smith, Dale Sartor, Richard Brown, Magnus Herrlin, Jonathan Koomey, Eric Masanet, Nathaniel Horner, Inês Azevedo, and William Lintner, *United states data center energy usage report*, (2016).

[97] Yongpeng Shi, Yujie Xia, and Ya Gao, *Joint gateway selection and resource allocation for cross-tier communication in space-air-ground integrated IoT networks*, IEEE Access **9** (2020), 4303–4314.

[98] Jian Song, Yong Cui, Minming Li, Jiezhong Qiu, and Rajkumar Buyya, *Energy-traffic tradeoff cooperative offloading for mobile cloud computing*, Quality of Service (IWQoS), 2014 IEEE 22nd International Symposium of, 2014, pp. 284–289.

[99] Rade Stanojevic and Robert Shorten, *Distributed dynamic speed scaling*, INFOCOM, 2010 Proceedings IEEE, IEEE, 2010, pp. 1–5.

[100] Lu Sun, Liangtian Wan, and Xianpeng Wang, *Learning-based resource allocation strategy for industrial IoT in UAV-enabled MEC systems*, IEEE Transactions on Industrial Informatics **17** (2020), no. 7, 5031–5040.

[101] Tetsuyuki Takahama and Setsuko Sakai, *Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites*, 2006 IEEE International Conference on Evolutionary Computation, IEEE, 2006, pp. 1–8.

[102] Stef H Tijs and Theo SH Driessen, *Game theory and cost allocation problems*, Management Science **32** (1986), no. 8, 1015–1028.

[103] Bing-Chuan Wang, Han-Xiong Li, Jia-Peng Li, and Yong Wang, *Composite differential evolution for constrained evolutionary optimization*, IEEE Transactions on Systems, Man, and Cybernetics: Systems **49** (2018), no. 7, 1482–1495.

[104] Cheng Wang, Bhuvan Urgaonkar, George Kesidis, Uday V Shanbhag, and Qian Wang, *A case for virtualizing the electric utility in cloud data centers.*, HotCloud, 2014.

[105] Yanting Wang, Min Sheng, Xijun Wang, Liang Wang, and Jiandong Li, *Mobile-edge computing: Partial computation offloading using dynamic voltage scaling*, IEEE Transactions on Communications **64** (2016), no. 10, 4268–4282.

[106] Yefu Wang, Xiaorui Wang, and Yanwei Zhang, *Leveraging thermal storage to cut the electricity bill for datacenter cooling*, Proceedings of the 4th Workshop on Power-Aware Computing and Systems, ACM, 2011, p. 8.

[107] Yong Wang, Zhi-Yang Ru, Kezhi Wang, and Pei-Qiu Huang, *Joint deployment and task scheduling optimization for large-scale mobile users in multi-UAV-enabled mobile edge computing*, IEEE Transactions on Cybernetics **50** (2019), no. 9, 3984–3997.

[108] Zhen Wang, Wenjun Xu, Dingcheng Yang, and Jiaru Lin, *Joint trajectory optimization and user scheduling for rotary-wing UAV-enabled wireless powered communication networks*, IEEE Access **7** (2019), 181369–181380.

[109] Yonggang Wen, Weiwen Zhang, and Haiyun Luo, *Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones*, IEEE INFOCOM, 2012, pp. 2716–2720.

[110] Adam Wierman, Lachlan LH Andrew, and Ao Tang, *Power-aware speed scaling in processor sharing systems*, INFOCOM 2009, IEEE, IEEE, 2009, pp. 2007–2015.

[111] Qiang Wu, *Making Facebook's software infrastructure more energy efficient with Autoscale*, `https://code.fb.com/production-engineering/making-facebook-s-software-infrastructure-more-energy-efficient-with-autoscale/`.

[112] Qingqing Wu, Liang Liu, and Rui Zhang, *Fundamental trade-offs in communication and trajectory design for UAV-enabled wireless network*, IEEE Wireless Communications **26** (2019), no. 1, 36–44.

[113] Qingqing Wu, Yong Zeng, and Rui Zhang, *Joint trajectory and communication design for multi-UAV enabled wireless networks*, IEEE Transactions on Wireless Communications **17** (2018), no. 3, 2109–2121.

[114] Qiufen Xia, Weifa Liang, Zichuan Xu, and Bingbing Zhou, *Online algorithms for location-aware task offloading in two-tiered mobile cloud environments*, Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on, 2014, pp. 109–116.

[115] Yong Xiao and Marwan Krunz, *QoE and power efficiency tradeoff for fog computing networks with fog node cooperation*, IEEE INFOCOM 2017-IEEE Conference on Computer Communications, IEEE, 2017, pp. 1–9.

[116] Lifeng Xie, Jie Xu, and Rui Zhang, *Throughput maximization for UAV-enabled wireless powered communication networks*, IEEE Internet of Things Journal **6** (2018), no. 2, 1690–1703.

[117] Bin Xu, Lu Zhang, Zipeng Xu, Yichuan Liu, Jinming Chai, Sichong Qin, and Yanfei Sun, *Energy optimization in multi-UAV-assisted edge data collection system*, Computers, Materials & Continua (2021).

[118] Jie Xu, Lixing Chen, and Pan Zhou, *Joint service caching and task offloading for mobile edge computing in dense networks*, IEEE INFOCOM, 2018, pp. 207–215.

[119] Yu Xu, Tiankui Zhang, Yuanwei Liu, Dingcheng Yang, Lin Xiao, and Meixia Tao, *UAV-assisted MEC networks with aerial and ground cooperation*, IEEE Transactions on Wireless Communications **20** (2021), no. 12, 7712–7727.

[120] Shi Yan, Mugen Peng, and Xueyan Cao, *A game theory approach for joint access selection and resource allocation in UAV assisted IoT communication networks*, IEEE Internet of Things Journal **6** (2018), no. 2, 1663–1674.

[121] Lei Yang, Haipeng Yao, Jingjing Wang, Chunxiao Jiang, Abderrahim Benslimane, and Yunjie Liu, *Multi-UAV-enabled load-balance mobile-edge computing for IoT networks*, IEEE Internet of Things Journal **7** (2020), no. 8, 6898–6908.

[122] Yuan Yao, Longbo Huang, Abhihshek Sharma, Leana Golubchik, and Michael Neely, *Data centers power reduction: A two time scale approach for delay tolerant workloads*, INFOCOM, 2012 Proceedings IEEE, IEEE, 2012, pp. 1431–1439.

[123] Shanhe Yi, Cheng Li, and Qun Li, *A survey of fog computing: concepts, applications and issues*, Proceedings of the 2015 workshop on mobile big data, ACM, 2015, pp. 37–42.

[124] Changsheng You, Kaibin Huang, Hyukjin Chae, and Byoung-Hoon Kim, *Energy-efficient resource allocation for mobile-edge computation offloading*, IEEE Transactions on Wireless Communications **16** (2017), no. 3, 1397–1411.

[125] H Peyton Young, *Cost allocation: methods, principles, applications*, North Holland Publishing Co., 1985.

[126] Hao Yu and Michael J Neely, *A new backpressure algorithm for joint rate control and routing with vanishing utility optimality gaps and finite queue lengths*, IEEE/ACM Transactions on Networking (ToN) **26** (2018), no. 4, 1605–1618.

[127] Liang Yu, Tao Jiang, Yang Cao, and Qian Zhang, *Risk-constrained operation for internet data centers in deregulated electricity markets*, IEEE Transactions on Parallel and Distributed Systems **25** (2014), no. 5, 1306–1316.

[128] Zhe Yu, Yanmin Gong, Shimin Gong, and Yuanxiong Guo, *Joint task offloading and resource allocation in UAV-enabled mobile edge computing*, IEEE Internet of Things Journal **7** (2020), no. 4, 3147–3159.

[129] Wanghong Yuan and Klara Nahrstedt, *Energy-efficient soft real-time CPU scheduling for mobile multimedia systems*, ACM SIGOPS Operating Systems Review, vol. 37, 2003, pp. 149–163.

[130] ———, *Energy-efficient CPU scheduling for multimedia applications*, ACM Transactions on Computer Systems (TOCS) **24** (2006), no. 3, 292–331.

[131] Yong Zeng, Rui Zhang, and Teng Joon Lim, *Throughput maximization for UAV-enabled mobile relaying systems*, IEEE Transactions on Communications **64** (2016), no. 12, 4983–4996.

[132] Cheng Zhan, Han Hu, Xiufeng Sui, Zhi Liu, and Dusit Niyato, *Completion time and energy optimization in the UAV-enabled mobile-edge computing system*, IEEE Internet of Things Journal **7** (2020), no. 8, 7808–7822.

[133] Cheng Zhan and Yong Zeng, *Completion time minimization for multi-UAV-enabled data collection*, IEEE Transactions on Wireless Communications **18** (2019), no. 10, 4859–4872.

[134] Jing Zhang, Weiwei Xia, Feng Yan, and Lianfeng Shen, *Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing*, IEEE Access **6** (2018), 19324–19337.

[135] Kaiyuan Zhang, Xiaolin Gui, Dewang Ren, and Defu Li, *Energy–latency tradeoff for computation offloading in UAV-assisted multiaccess edge computing system*, IEEE Internet of Things Journal **8** (2020), no. 8, 6709–6719.

[136] Ke Zhang, Yuming Mao, Supeng Leng, Quanxin Zhao, Longjiang Li, Xin Peng, Li Pan, Sabita Maharjan, and Yan Zhang, *Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks*, IEEE Access **4** (2016), 5896–5907.

[137] Qi Zhang, Mohamed Faten Zhani, Shuo Zhang, Quanyan Zhu, Raouf Boutaba, and Joseph L Hellerstein, *Dynamic energy-aware capacity provisioning for cloud computing environments*, Proceedings of the 9th international conference on Autonomic computing, ACM, 2012, pp. 145–154.

[138] Tiankui Zhang, Yu Xu, Jonathan Loo, Dingcheng Yang, and Lin Xiao, *Joint computation and communication design for UAV-assisted mobile edge computing in IoT*, IEEE Transactions on Industrial Informatics **16** (2019), no. 8, 5505–5516.

[139] Ying Zhang, Lei Deng, Minghua Chen, and Peijian Wang, *Joint bidding and geographical load balancing for datacenters: Is uncertainty a blessing or a curse?*, INFOCOM 2017-IEEE Conference on Computer Communications, IEEE, IEEE, 2017, pp. 1–9.

[140] Yue Zhao, Junjie Qin, Ram Rajagopal, Andrea Goldsmith, and H Vincent Poor, *Wind aggregation via risky power markets*, IEEE Transactions on Power Systems **30** (2015), no. 3, 1571–1581.

[141] Fuhui Zhou, Yongpeng Wu, Rose Qingyang Hu, and Yi Qian, *Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems*, IEEE Journal on Selected Areas in Communications **36** (2018), no. 9, 1927–1941.

APPENDICES

Proof of Theorem 3.2.1

We first rewrite (3.2.5) as below:

$$c(S) = \min_{Q_S} \mu_p^d Q_S + \mu_p^- \int_{Q_S}^{E_S^{\max}} (u - Q_S) f_S(u)\,\mathrm{d}u - \mu_p^+ \int_{E_S^{\min}}^{Q_S} (Q_S - u) f_S(u)\,\mathrm{d}u, \tag{A.1}$$

where f_S is the corresponding probability density function (PDF) of the CDF as defined in (3.2.2). Then by applying the first order optimality condition associated with Leibniz integral rule, we have

$$\mu_p^d - \mu_p^-(1 - F_S(Q_S)) - \mu_p^+ F_S(Q_S) = 0, \tag{A.2}$$

$$Q_S^* = F_S^{-1}(\varepsilon^*), \quad \text{where} \quad \varepsilon^* = \frac{\mu_p^- - \mu_p^d}{\mu_p^- - \mu_p^+}. \tag{A.3}$$

Optimal expected cost is given by direct substitution of Q_S^* into (A.1):

$$\begin{aligned} c(S) &= \mu_p^d Q_S^* + \mu_p^- \int_{Q_S^*}^{E_S^{\max}} (u - Q_S^*) f_S(u)\,\mathrm{d}u - \mu_p^+ \int_{E_S^{\min}}^{Q_S^*} (Q_S^* - u) f_S(u)\,\mathrm{d}u \\ &= \mu_p^d Q_S^* + \mu_p^- \int_{\varepsilon^*}^{1} (F_S^{-1}(\theta) - Q_S^*)\,\mathrm{d}\theta - \mu_p^+ \int_0^{\varepsilon^*} (Q_S^* - F_S^{-1}(\theta))\,\mathrm{d}\theta \\ &= Q_S^* \underbrace{(\mu_p^d - \mu_p^- + \varepsilon^*(\mu_p^- - \mu_p^+))}_{=0} + \mu_p^+ \int_0^{\varepsilon^*} F_S^{-1}(\theta)\,\mathrm{d}\theta + \mu_p^- \int_{\varepsilon^*}^{1} F_S^{-1}(\theta)\,\mathrm{d}\theta. \end{aligned} \tag{A.4}$$

Proof of Theorem 3.2.2

We introduce an ancillary random variable $X_i \coloneqq E_i - Q_i$ and rewrite (3.2.4) in terms of X_i as follows:

$$\Phi_S(Q_S) = \mu_p^d Q_S + \mu_p^- \mathbb{E}\big[\big(\sum_{i\in S} X_i\big)^+\big] - \mu_p^+ \mathbb{E}\big[\big(-\sum_{i\in S} X_i\big)^+\big], \tag{A.5}$$

$$\sum_{i\in S} \Phi_i(Q_i) = \mu_p^d \sum_{i\in S} Q_i + \mu_p^- \mathbb{E}\big[\sum_{i\in S} \big(X_i\big)^+\big] - \mu_p^+ \mathbb{E}\big[\sum_{i\in S} \big(-X_i\big)^+\big]. \tag{A.6}$$

By adopting the equivalent forms of $(x)^+$:

$$(x)^+ \coloneqq \max(x, 0) \coloneqq \frac{x + |x|}{2}, \tag{A.7}$$

we have

$$\begin{aligned}
(\text{A.5}) - (\text{A.6}) &= \mu_p^- \mathbb{E}\left[\frac{\sum_{i\in S} X_i + \big|\sum_{i\in S} X_i\big|}{2} - \sum_{i\in S} \frac{X_i + |X_i|}{2}\right] \\
&\quad - \mu_p^+ \mathbb{E}\left[\frac{\big|\sum_{i\in S} X_i\big| - \sum_{i\in S} X_i}{2} - \sum_{i\in S} \frac{|X_i| - X_i}{2}\right] \\
&= \left(\frac{\mu_p^- - \mu_p^+}{2}\right) \mathbb{E}\big[\big(\big|\sum_{i\in S} X_i\big| - \sum_{i\in S} |X_i|\big)\big] \leq 0.
\end{aligned} \tag{A.8}$$

The above inequality holds according to the triangle inequality, i.e., $\big|\sum_{i\in S} X_i\big| \leq \sum_{i\in S} |X_i|$ and also by assumption, we have $\mu_p^- \geq \mu_p^+$. Therefore, $\Phi_S(Q_S) \leq \sum_{i\in S} \Phi_i(Q_i)$.

Proof of Lemma 3.2.1

First we prove the positive homogeneity. The CDF of the positively scaled E_S is denoted as

$$F_{\beta S}(u) = \Pr(\beta E_S \leq u) = F_{\beta S}\left(\frac{u}{\beta}\right).$$

It follows that the quantile function of $F_{\beta S}(u)$ is given by

$$F_{\beta S}^{-1}(\varepsilon^*) = \beta F_S^{-1}(\varepsilon^*).$$

Using the results from Theorem 3.2.1, we can prove the positive homogeneity as

$$\begin{aligned} c(\beta S) &= \mu_p^+ \int_0^{\varepsilon^*} F_{\beta S}^{-1}(\theta)\,\mathrm{d}\theta + \mu_p^- \int_{\varepsilon^*}^1 F_{\beta S}^{-1}(\theta)\,\mathrm{d}\theta \\ &= \beta\left(\mu_p^+ \int_0^{\varepsilon^*} F_S^{-1}(\theta)\,\mathrm{d}\theta + \mu_p^- \int_{\varepsilon^*}^1 F_S^{-1}(\theta)\,\mathrm{d}\theta\right) \\ &= \beta c(S). \end{aligned} \tag{A.9}$$

Next we prove the subadditivity as

$$\begin{aligned} c(S_1) + c(S_2) &= \min_{Q_{S_1}} \Phi_{S_1}(Q_{S_1}) + \min_{Q_{S_2}} \Phi_{S_2}(Q_{S_2}) \\ &= \Phi_{S_1}(Q_{S_1}^*) + \Phi_{S_2}(Q_{S_2}^*), \end{aligned} \tag{A.10}$$

where $Q_{S_1}^*$ and $Q_{S_2}^*$ are the optimal day-ahead bids of their respective minimization problems. It follows from Theorem 3.2.2 that

$$\begin{aligned} \Phi_{S_1}(Q_{S_1}^*) + \Phi_{S_2}(Q_{S_2}^*) &\geq \Phi_{S_1 \cup S_2}(Q_{S_1}^* + Q_{S_2}^*) \\ &\geq \Phi_{S_1 \cup S_2}(Q_{S_1 \cup S_2}^*) \\ &= c(S_1 \cup S_2), \end{aligned} \tag{A.11}$$

where $Q_{S_1 \cup S_2}^*$ is the optimal solution of the expected cost minimization problem under coalition $S_1 \cup S_2$, while $Q_{S_1}^* + Q_{S_1}^*$ is a feasible solution of the minimization problem, then it follows that $c(S_1 \cup S_2) \leq c(S_1) + c(S_2)$.

Proof of Nonconvex Game

Consider a cooperative game involving three datacenters, indexed by $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$, respectively. We assume the marginal distribution of $\mathcal{A}_1$ and $\mathcal{A}_2$ are given by

$$\mathcal{A}_i = \begin{cases} 2, & \text{w.p. } 0.5 \\ 4, & \text{w.p. } 0.5 \end{cases} \quad \forall i = 1, 2.$$

Further, assume $\mathcal{A}_3$ is perfectly positively correlated to $\mathcal{A}_2$, i.e., $\mathcal{A}_3 = \mathcal{A}_2$. Set the expected day-ahead, negative imbalance and positive imbalance prices as $\mu_p^d = 0.9$, $\mu_p^- = 1.4$ and $\mu_p^+ = 0.4$, respectively. Then based on Theorem 3.2.1, we have:

$$\begin{aligned} \varepsilon^* &= \frac{1.4 - 0.9}{1.4 - 0.4} = 0.5, \\ c(\{1\}) = c(\{2\}) &= c(\{3\}) = 3.2, \\ c(\{1,2\}) &= c(\{1,3\}) = 5.9, \\ c(\{2,3\}) &= 6.4, \\ c(\{1,2,3\}) &= 9.1. \end{aligned}$$

Here, we choose two coalitions as $\mathcal{S} = \{1, 2\}$ and $\mathcal{T} = \{1, 3\}$, and then from the above example, we have:

$$c(\{1,2\}) + c(\{1,3\}) = 10.8 \leq c(\{1,2,3\}) + c(\{1\}) = 12.3,$$

which violates the definition of convex game given in (2.3.4). Therefore, our cooperative game is nonconvex.

Proof of Theorem 3.3.2

Our proof is similar to [140, 57] which focus on different aggregation problems. Here we only give a sketch of the proof process. The basic idea is that we could also use the non-cooperative game theory to model the same problem by allowing power exchange within datacenters as well, and our proposed allocation method can find the Nash equilibrium of the formulated noncooperative game. Since the core of our cooperative game can be shown to be the same as the Nash equilibrium of the corresponding noncooperative game, our proposed cost allocation scheme is guaranteed to find the core of the cooperative game. Details about the proof process can be found in [140, 57].

VITA

Zhe Yu

Candidate for the Degree of

Doctor of Philosophy

Dissertation: RESOURCE MANAGEMENT FOR COST-EFFECTIVE CLOUD AND EDGE SYSTEMS

Major Field: Electrical Engineering

Biographical:

Education:

Completed the requirements for the Doctor of Philosophy in Electrical Engineering at Oklahoma State University, Stillwater, Oklahoma in December, 2022.

Completed the requirements for the Master of Science in Electrical Engineering at Vanderbilt University, Nashville, Tennessee in 2016.

Completed the requirements for the Bachelor of Engineering in Communications Engineering at University of Science and Technology Beijing, Beijing, China in 2014.

Experience:

Visual Computing and Image Processing Lab (VCIPL), Oklahoma State University, 10/2019 – 12/2022

Smart Energy and Networked Systems Laboratory, Oklahoma State University, 08/2016 – 07/2019

Professional Membership:

IEEE Graduate Student Member