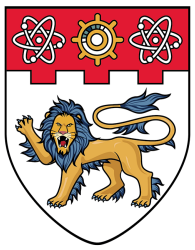


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

Metaverse's Stochastic Optimization for Enhanced Computation and Communication

Wei Chong Ng

School of Computer Science and Engineering (SCSE)

A thesis submitted to the Nanyang Technological University in partial fulfilment of the requirement for the degree of Doctor of Philosophy

2023

Statement of Originality

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

7/5/2024

Date

chong

Wei Chong Ng

Supervisor Declaration Statement

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

7/5/2024

Date

Prof. Dusit Niyato

Authorship Attribution Statement

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

- 3 published journal article
- 1 accepted journal article

Chapter 2:

Chapter 2 is a journal article published as: Xu, M., Ng, W. C., Lim, W. Y. B., Kang, J., Xiong, Z., Niyato, D., and Miao, C, 2022. A full dive into realizing the edge-enabled metaverse: Visions, enabling technologies, and challenges. *IEEE Communications Surveys & Tutorials,* vol. 25, no. 1, pp. 656-700, Firstquarter 2023. DOI: 10.1109/COMST.2022.3221119.
The contributions of the co-authors are as follows:

- Mr Xu and I conducted the literature review and prepared the manuscript draft.
- Prof. Dusit and Prof. Lim provided the initial research direction.
- All authors discussed the results and commented on the manuscript.

Chapter 3:

Chapter 3 is a journal article published as: Ng, W. C., Lim, W. Y. B., Xiong, Z., Niyato, D., Miao, C., Han, Z., and Kim, D. I., 2022. Stochastic Coded Offloading Scheme for Unmanned-Aerial-Vehicle-Assisted Edge Computing. *IEEE Internet of Things Journal*, vol. 10, no. 7, pp. 5626-5643, 1 April1, 2023. DOI: 10.1109/JIOT.2022.3150472.
The contributions of the co-authors are as follows:

- Prof. Dusit and Prof. Lim provided the initial research direction.
- I prepared the manuscript draft, proposed the system model, solved the optimization problem, and implemented the simulations.
- All authors discussed the results and commented on the manuscript.

Chapter 4:

Chapter 4 is a journal article published as: Ng, W. C., Lim, W. Y. B., Xiong, Z., Niyato, D., Poor, H. V., Shen, X. S., and Miao, C., 2023. Resource Optimization for UAV-assisted Wireless Power Charging Enabled Hybrid Coded Edge Computing Network. *IEEE Transactions on Mobile Computing*. DOI: 10.1109/TMC.2023.3246994.
The contributions of the co-authors are as follows:

- Prof. Dusit and Prof. Lim provided the initial research direction.
- I prepared the manuscript draft, proposed the system model, solved the optimization problem, and implemented the simulations.
- All authors discussed the results and commented on the manuscript.

Chapter 5:

Chapter 5 is a journal article accepted as: Ng, W. C., Lim, W. Y. B., Xiong, Z., Niyato, D., Shen, X. S., and Miao, C., 2023. Distributionally Robust Cost Minimized Edge Semantic Intelligence in the Sustainable Metaverse. *IEEE Transactions on Mobile Computing*.
The contributions of the co-authors are as follows:

- Prof. Dusit and Prof. Lim provided the initial research direction.
- I prepared the manuscript draft, proposed the system model, solved the optimization problem, and implemented the simulations.
- All authors discussed the results and commented on the manuscript.

Signed:

Wei Chong Ng
Student
School of Computer Science and Engineering
Nanyang Technological University
Date: 7/5/2024

Acknowledgements

I extend my deepest gratitude to my supervisor, Prof. Dusit Niyato, for his constant support and guidance throughout my Ph.D. study. He will always be there to provide all the help that I need. Without him, this journey will be impossible.

I am also deeply grateful to my graduated seniors, Prof. Wei Yang Bryan Lim and Prof. Zehui Xiong, for their constant motivation as well as for providing the necessary information and direction. I would like to thank all my friends who are helping me along the way and the collaborators who have shared research ideas and knowledge with me.

A special gratitude I give to my wife/friend, Zhou Wuyufei Faye. She is like a lighthouse, always illuminating the path ahead of me. Whenever I face the most challenging times in life, she shines a light to guide me in the right direction. At the same time, she is also a friend, constantly encouraging me and cheering me on, preventing me from getting trapped in negative emotions.

Contents

Abstract

The emergence of the Metaverse, a collective virtual shared space, represents a paradigm shift in how we perceive and interact with digital environments. Beyond mere entertainment, the Metaverse promises transformative applications in education, professional collaboration, healthcare, social interaction, and even in reshaping economic models through virtual commerce and real estate. This thesis offers an in-depth exploration of the technological and computational challenges and solutions pivotal to these applications.

Central to this exploration is the role of Virtual Service Providers. Their challenges, especially in terms of latency reduction and efficient data communication, are dissected, revealing the intricate balance required to maintain real-time interactions in such expansive virtual spaces.

The research introduces the mobile cloud-edge-end collaborative computing paradigm, a cornerstone for the Metaverse's infrastructure. This paradigm promises to revolutionize data processing and delivery, ensuring seamless experiences for users across various applications, from virtual classrooms to immersive therapeutic environments.

A significant portion of the study is dedicated to understanding the Wireless Charging Hybrid Network. The nuances of its optimization, especially when considering stochastic offloading, are presented, highlighting its potential to support the vast array of devices that users might employ within the Metaverse.

Furthermore, the thesis delves into advanced computational models, with a particular emphasis on BERT. The potential of such models in semantic data processing is explored, emphasizing their role in ensuring that data delivery in the Metaverse is not only efficient but also contextually relevant, enhancing user experiences across diverse applications.

Through rigorous analysis, case studies, and a synthesis of current technological trends, this research paints a comprehensive picture of the Metaverse's current state, its potential applications, and its future trajectory. It underscores the importance of continuous innovation, the challenges that lie ahead, and the immense potential the Metaverse holds for reshaping digital interactions in various sectors.

Keywords: **Metaverse, Resource Allocation, Stochastic Integer Programming, Wireless Power Transfer, Computation Offloading, Coded Distributed Computing, Semantic Communication**

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background of Edge Computing

Due to the rapid advancement of Internet of Things (IoT) enabled technologies, the number of wirelessly connected devices is increasing exponentially [5] and generating huge amounts of data [6]. Various computations, such as cloud and edge computing, are developed to lighten the burden on the computation requirement on mobile devices as not all mobile devices have strong computation capability. Devices with weak computation capability will carry out the tasks with longer delays. For example, devices with weak computation will face difficulties when rendering a large 3D model [7]. However, with the help of cloud and edge computing, weak computation devices can offload partial/all tasks to other computation-enabled infrastructures. Devices with strong computation can also perform offloading to reduce energy consumption. The following are the descriptions of cloud and edge computing.

- Cloud computing: Cloud computing provides computing services to either mobile phones or mobile embedded systems. The cloud provider installs the servers in their private infrastructure, and the services are provided to the client via virtual machines [8]. The computation-intensive task can be offloaded to the nearby data centre to bring cloud computing capabilities closer to the users and supports all the users within the same region.

- Edge computing: The increases of the IoT devices cause an unprecedented generation of the huge and heterogeneous amount of data [9]. Cloud computing

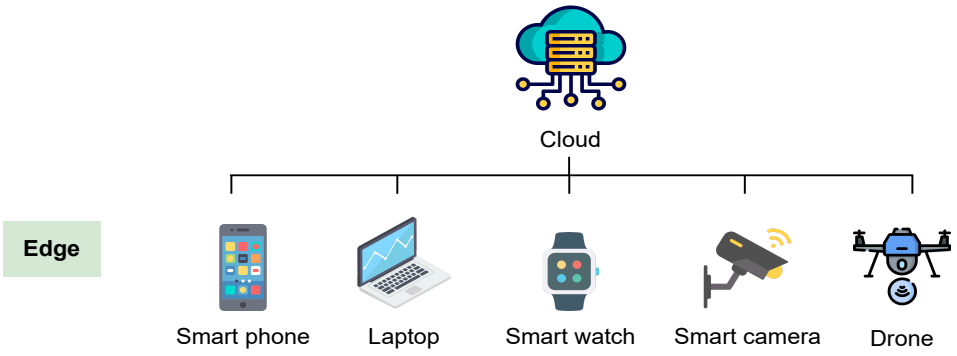


Figure 1.1: Cloud computing and edge computing models.

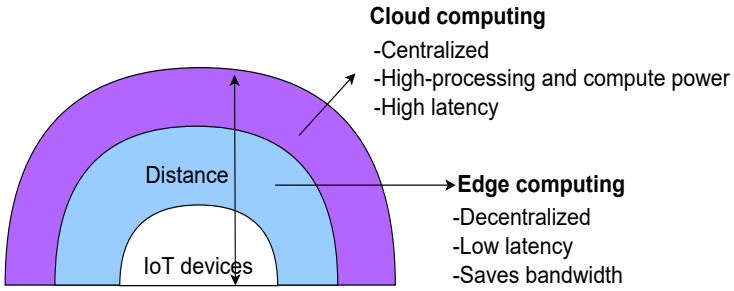


Figure 1.2: The difference between cloud and edge computing.

can no longer support the high demands for real-time applications [10]. Edge computing is proposed to reduce the latency. Edge computing observes the computation capability in the local network and computes the task that could have been carried out in the cloud, reducing latency and improving the decision-making process. The edge computing system architecture allows the computation to occur at the edge of the network where data are generated [11].

Fig. 1.1 shows the cloud computing and edge computing models. Edge computing can either operate as a single computing platform or collaborate with other components such as cloud server [12]. An edge-cloud architecture accelerates the service provided at the edges of the network without jeopardizing the rich functionality of a secure cloud. The difference between cloud and edge computing is shown in Fig.1.2. Edge computing has a lower latency as compared to cloud computing as the edge nodes/devices are much closer to the users. Edge devices are the hardware that drives the application of edge computing. They can store and analyze the data locally to support operations at the edge. Examples of edge devices are smartphones,

augmented reality or virtual reality glasses, and industrial robots. Besides latency, edge computing is beneficial in situations when connectivity is intermittent, or bandwidth is limited due to environmental factors. For instance, oil rigs and ships at sea. Edge computing performs computations on-site, often on the edge device itself, such as water quality sensors on water purifiers in distant villages, and can save data to send to a central location only when connectivity is available. The amount of data that must be sent can be greatly decreased by processing it locally, needing significantly less bandwidth and connectivity time than would otherwise be required.

A few potential edge computing applications in various industries are listed below.

- Video surveillance: Video data is slow and expensive when it is sent to a central server. Edge computing can speed up and reduce the cost of the process by allowing cameras to perform basic video analytics. Then, the camera can transmit the filtered video to a local edge for additional processing.
- Connected vehicles: A vehicle with edge devices may collect data from various sensors and respond to circumstances on the road in real-time. This functionality will be crucial in the development of self-driving cars.
- Monitoring within oil and gas industries: Oil and gas problems can be avoided with the use of edge computing. Because these plants are generally located in rural areas, an edge device is a far better solution than a remote server or cloud. In addition, edge devices can use real-time analytics to monitor the system and shut down computers before a crisis.

1.2 The Metaverse and its Relationship with Edge Computing

The Metaverse is the next generation of the Internet that allows users to experience the Internet of the future and has the potential to transform the way we live, interact, and learn [13]. It connects people around the world in the form of avatars to allow them to interact virtually through the use of virtual reality (VR)/augmented reality (AR)/mixed reality (MR) gadgets. Empowered by digital twins, the Metaverse closely synchronizes the virtual and physical world. For example, users can operate a virtual mechanical arm in the Metaverse, which is synchronized with the physical one located at a remote factory. With the help of VR/AR/MR, a user guide can be referred to by the user to operate the arm safely.

However, creating a virtual universe is a complex, time-consuming, and expensive endeavor, with projected costs reaching into the trillions of dollars. Numerous large

corporations are allocating significant resources to carve out their own niche in the Metaverse, aiming to become akin to the Oracle character in the Matrix films [14]. In the Metaverse, a fully realized digital universe will be created from a simple environment, such as a bedroom, to a complex environment, like a city. The number of virtual objects and avatars might be limited in simpler spaces like a bedroom. However, in a bustling city scene, there could be buildings, cars, dining areas, and people. Software developers designed these digital spaces, objects, and avatars, and they are stored and managed in mega-scale centralized data centers [14]. However, the expected energy consumption of a mega-scale data center is close to 1,380 gigawatt-hours per year, nearly half as much energy as all the data centers in the Netherlands combined [15]. In addition, the Metaverse data centers will need to be in very close proximity to the user to achieve low latency [14] so as to ensure that users do not suffer from cause cybersickness, a condition closely related to motion sickness [16]. To address these challenges, edge computing emerges as a valuable solution for enhancing Metaverse scalability. It enables data processing closer to users, mitigating latency issues. In the subsequent sections, we explore solutions to enhance network latency from both computational and communication perspectives, and it is illustrated in Fig. 1.3.

1.2.1 Computation Offloading

Apart from the advantages mentioned above, most of the edge devices are small in size and have limited battery capacity. Therefore, the concept of computation offloading in mobile edge computing (MEC) can be introduced to leverage the power of the edge devices and speed up the application [17]. By making use of the base stations (BSs), users can offload computation tasks with high bandwidth, low latency, and high QoE [18]. MEC networks comprise multiple edge servers positioned adjacent to base stations. Each edge server can be seen as a small data center established by an operator to meet specific service demands, such as data preprocessing, handling content requests, and offloading tasks.

There are two types of offloading: partial and full offloading. Partial offloading allows the edge device to offload a part of the computing-intensive tasks to other edge devices/servers. Numerous studies on the partial offloading of tasks in MEC networks have been investigated. The authors in [19] proposed a price-based distributed method to manage the offloaded computation tasks where the computational tasks can be arbitrarily divided in bitwise for partial local computing and partial offloading process. The authors in [20] considered the heterogeneous energy harvesting MEC systems with multiple mobile devices and multiple edge servers. While full offloading means

that the task should be offloaded to an edge server as a whole. The authors in [21] studied user mobility and proposed a device-to-device offloading MEC network. The idle and resource-rich devices can serve as an edge node to establish an offloading link with a resource-limited task device. Therefore, in Chapter 3, we adopt a coding technique to partition the task and propose an offloading scheme that considers multiple stochastic uncertainties.

1.2.2 Wireless Energy Transfer

Besides computation offloading, wireless energy transfer is another technique to tackle the issue of limited battery capacity of edge devices [22]. Wireless energy transfer (WET) is an emerging technology that enables the recharging of sensor node batteries without the need for physical wires. WET can be classified into two categories: electromagnetic (EM) radiation and magnetic resonant coupling [23]. EM radiation transfers power through technologies like radio waves, microwaves, and visible light. In contrast, magnetic resonant coupling power transfer operates on the principle of electromagnetic resonance [24].

Comparing the two categories, WET with magnetic resonant coupling power transfer has several advantages, such as resilience to the surrounding environment and the absence of a line-of-sight (LOS) prerequisite [23]. The experiments described in [24] demonstrated that magnetic resonant coupling can transmit 60 watts of power effectively over a 2-meter distance, achieving an impressive 40% efficiency, a feat that cannot be achieved using electromagnetic (EM) radiation. However, in order for wireless energy transfer to function effectively, the charging station needs to be positioned within reasonable proximity to the sensor node to ensure optimal energy transfer efficiency. Therefore, in Chapter 4, we extend the work from Chapter 3 to address the complexity of the optimization while considering the uncertainty of the wireless charging efficiency.

1.2.3 Semantic Communication

Other than the computational aspect, improving the latency in Metaverse applications also depends on enhancing the current communication system, as creating the Metaverse requires systems capable of handling large amounts of data beyond what current 5G networks can manage [25]. In contrast to existing communication technologies, semantic communication systems consider transmission successful when the received information retains its intended meaning throughout the process.

To ensure an optimal immersive experience for users, it is essential to meet strict

requirements for data rates and the timing of data transfer and processing. For example, in a VR scenario, semantic communication involves extracting meaning from data collected and tracked by end devices, such as head movements, arm gestures, or speech [26]. This enables the end device to send only the relevant information to the Metaverse virtual service providers (VSPs), saving bandwidth and reducing computing latency. Simultaneously, the Metaverse VSPs can extract semantic information from videos based on user preferences, disregarding irrelevant details due to bandwidth constraints. This approach alleviates the pressure on the downlink and contributes to efficient data exchange within the Metaverse [26]. Therefore, in Chapter 5, we proposed a semantic cropping module that helps to reduce the size/energy consumption in data transmission. Then, we propose a resource subscription scheme to reduce the operation cost of the VSPs.

1.2.4 Integrated Framework for a Seamless Metaverse Experience

The Metaverse, an immersive and interconnected virtual world, is positioned as the next evolution of the internet. It is essential to integrate multiple technologies and methodologies in order to fully realize its potential. Offloading, wireless energy transfer, and semantic communication can work in concert to improve the Metaverse experience in the following ways:

- **Offloading in the Metaverse**: Real-time interactions are critical in the Metaverse. Lags or delays can ruin the user experience and prevent immersion. Smooth, lag-free interactions are ensured by offloading computational tasks from devices with limited capabilities to more powerful edge or cloud servers. This is particularly important when users interact with complicated digital constructs or participate in complex simulations.

- **Energizing the Metaverse with Wireless Energy Transfer**: The gadgets that let users to explore the Metaverse, like VR/AR glasses or other Internet of Things gadgets, have to be charged at all times. The experience is ruined when these devices are frequently connected to physical chargers. Thanks to wireless energy transfer, users can stay fully immersed in the Metaverse without having to worry about running out of battery life, which can guarantee that devices receive constant power.

- **Semantic Communication: Enhancing Metaverse Interactions**: The Metaverse will produce enormous volumes of data. Raw data transmission can

cause network congestion and latency to rise. This can be made easier by using semantic communication, which focuses on conveying the essence or meaning of data. For example, only changes or interactions relevant to the user's current activity might be sent, rather than all the details of a virtual environment. This guarantees effective bandwidth utilization and improves real-time responsiveness.

As shown in Fig. 1.4, offloading, wireless energy transfer, and semantic communication can all be combined to create a Metaverse where users can immerse themselves seamlessly and continuously. Semantic communication guarantees effective data exchange, wireless energy transfer maintains device power, and offloading guarantees seamless interactions. When combined, these technologies have the potential to create a Metaverse, a digital universe that is a seamless extension of our physical world.

1.3 Motivation and Objectives

The Metaverse, heralded for its potential to revolutionize interactions within virtual environments, hinges on the seamless integration of emerging technologies such as augmented/virtual reality [27], smart cities [28], and digital twins technology. Notably, the city of Seoul's initiative, "Metaverse Seoul," embodies efforts to transcend traditional limitations of time, space, and language to enhance user interaction with public services and overall satisfaction [29]. However, despite these advancements, several significant challenges impede the Metaverse's full potential:

- **Limited Bandwidth and High Latency**: The proliferation of wirelessly connected devices required for real-time applications in the Metaverse places unprecedented demands on the existing wireless communication infrastructure. This limitation is particularly acute in real-time applications that are inherently delay-sensitive.

- **Stochastic Nature of User Demands**: Virtual Service Providers (VSPs) face difficulties in resource allocation due to unpredictable user behavior and variable demand patterns, complicating the consistent delivery of high-quality services.

In response to these challenges, this chapter proposes innovative solutions aimed at reducing service costs and improving communication and computation efficiency. We detail these solutions as follows:

1. **Stochastic Coded Offloading Scheme (SCOS):** Recognizing the potential of IoT devices in the Metaverse for data collection and processing, we leverage edge computing capabilities to enhance real-time responsiveness. Chapter 3 introduces a coded distributed computing approach for computation offloading, incorporating a two-phase stochastic coded offloading scheme that optimizes UAV allocations amid weather uncertainties and computational demand fluctuations.

2. **Wireless Charging Hybrid Network:** To support the continuous operation of AI-driven applications, Chapter 4 discusses a hybrid network model that integrates wireless power transfer with edge computing. This model allows IoT devices to perform local computations effectively, reducing the reliance on central servers and addressing issues related to scalability and energy efficiency.

3. **Semantic Transmission Selection:** In Chapter 5, we introduce semantic cropping techniques to minimize data transmission sizes while preserving essential information. This approach is complemented by a resource allocation framework that dynamically adjusts to user experience feedback and transmission costs, employing a double dutch auction and a Distributed Robust Optimization strategy to mitigate uncertainties and optimize operational efficiency.

1.3.1 Objectives

The main goal of the studies is to help the VSPs to improve the services that are offered to the users.

- Objective 1: To introduce a new semantic cropping module that helps to improve energy efficiency in data transmission.
- Objective 2: To optimize the subscription problem of the VSPs to prevent over- and under-subscription of the resources in the network.
- Objective 3: To show an illustration of how trending technology, such as Coded Distribution Computing (CDC), can be adopted in edge computing.
- Objective 4: To optimize the offloading problem with uncertainty factors, i.e., demand and shortfall, which have not been considered in the existing works.
- Objective 5: To improve the energy limitation in the edge devices, i.e., wireless charging efficiency uncertainty.

- Objective 6: To reduce the computational time of the offloading problem.

1.4 Contributions

The contributions of this thesis are presented as follows:

- In Chapter 3, we introduced the Stochastic Coded Offloading Scheme (SCOS) to help the VSPs improve transmission delays. SCOS is a two-phase optimization scheme that adopts a CDC technique to reduce the total cost of the network. In this chapter, we have achieved objectives 3 and 4.
  - The proposed optimization is split into two phases: Phase One (UAV type allocation) and Phase Two (task allocation).
  - Phase One is able to provide a strategic scenario-based decision that adapts well to the weather conditions in which the current solutions for UAVs are limited.
  - The proposed SCOS can minimize the UAVs' overall costs by optimizing the task allocation. At the same time, it can also minimize all the UAVs' energy consumption. The optimal solution is achieved by considering both the demand and shortfall uncertainty.
  - From the performance evaluation, we use the real data to validate that SCOS is the optimal scheme when the performance is compared with the Expected-Value Formulation (EVF) and random scheme.
- In Chapter 4, we propose a stochastic resource optimization framework for the wireless-powered hybrid coded edge computing network. In this chapter, we have achieved objectives 3, 4, 5, and 6.
  - Presents a novel wireless powered hybrid coded edge computing network that jointly leverages the use of base stations as wireless charging stations and edge servers for coded computation offloading.
  - Propose a stochastic resource optimization solution for efficient resource management amid stochastic uncertainties that can only be realized ex-post.
  - Analyze and reduce the complexity of the two-stage SIP model using Benders' decomposition and sample-average approximation.
- In Chapter 5, we propose a two-phase stochastic semantic resource allocation (SSRA) scheme to minimize the cost of the network and prevent over- and

under-subscription of the resources. In this chapter, we have achieved objectives 1 and 2.

- A semantic cropping module is proposed to reduce the energy consumption in data transmission.
- In phase one of the SSRA scheme, the proposed DRL-driven-double dutch auction mechanism is tractable, dynamic, and adaptive to the stochastic network conditions.
- In phase two of the SSRA scheme, instead of assuming the historical observation data is the true distribution, we construct a confidence set using the reference distribution derived from historical observation data and distance metrics.
- Using our self-collected real-world data, we compare SSRA with other baselines, such as deterministic, random, and stochastic integer programming.

1.5 Organisation of the Thesis

The thesis is organized as follows. Fig. 1.5 shows the relationship among the chapters.

Chapter 1 provides an overview of edge computing, edge-enable Metaverse, research challenges and motivations, and the major contributions of this thesis.

Chapter 2 presents the literature review based on four categories: i) the architecture of the Metaverse, ii) current development, iii) computation in the Metaverse, and iv) communication in the Metaverse.

Chapter 3 proposed a two-phase SCOS to minimize the cost of the network and energy consumption of the UAVs and prevent over and under-subscription of the resources. In the first phase, the appropriate UAVs are allocated to the charging stations amid weather uncertainty. In the second phase, we use the z-stage SIP to optimize the number of computation subtasks offloaded and computed locally while taking into account the computation shortfall and demand uncertainty.

Chapter 4 is extended from Chapter 3 by considering the energy constant of IoT devices that can be recharged with the wireless power transfer derived from base stations. To address the complexity of the SIP problem, which scales with the size of the network, we introduce the efficient computation methods of Benders' decomposition and sample average approximation.

Chapter 5 proposed a semantic cropping module to reduce the size of the data transmitted and reduce energy consumption while maintaining its meaning. Then, to

minimize the cost of the network and prevent over- and under-subscription of the resources, we propose a two-phase stochastic semantic resource allocation SSRA scheme.

Chapter 6 concludes the thesis and presents potential future research directions.

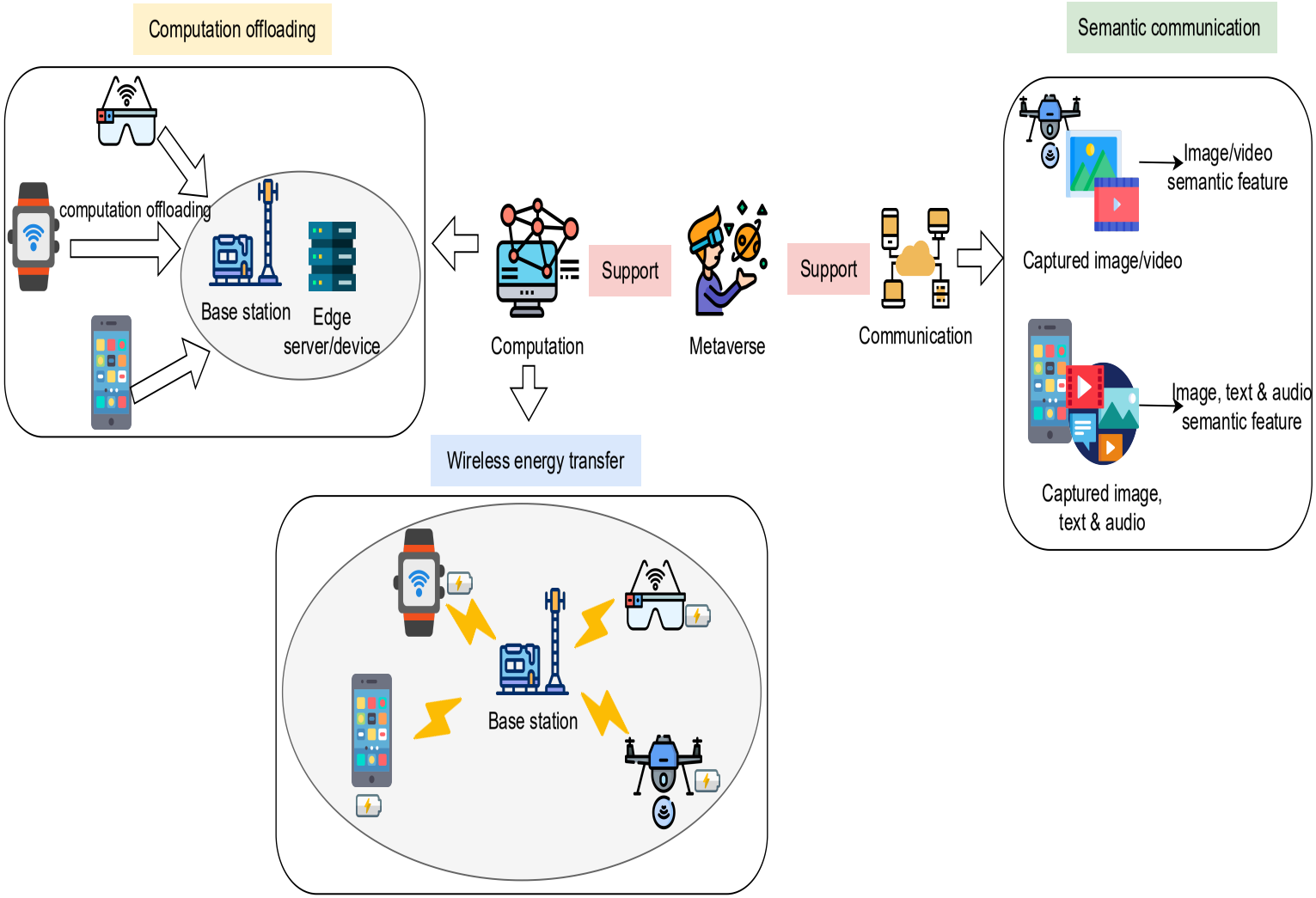


Figure 1.3: An illustration of computational and communication support for the Metaverse.

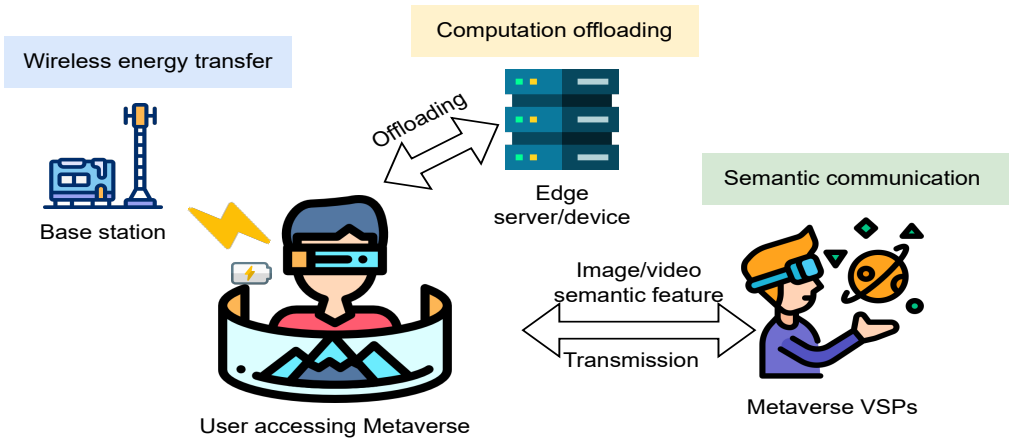


Figure 1.4: Combination of Offloading, wireless energy transfer, and semantic communication for a seamless Metaverse experience.

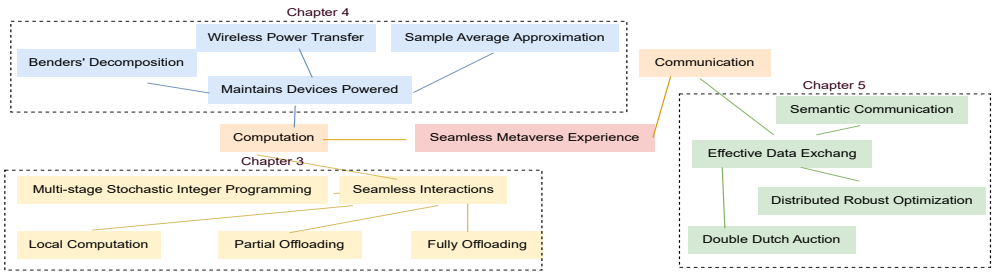


Figure 1.5: An illustration of the Chapter relationship.

List of Abbreviations

AIGC	AI-generated Content
AR	Augmented Reality
BERT	Bidirectional Encoder Representations from Transformers
BS	Base Station
CDC	Coded Distributed Computing
CV	Computer Vision
D2D	device-to-device
DID	Digital Identities
DIP	Deterministic Integer Programming
DRL	Deep Reinforcement Learning
DRO	Distributed Robust Optimization
EM	Electromagnetic
EVF	Expected-Value Formulation
HMD	Head Mounted Displays
IoT	Internet of Things
ML	Machine Learning
MMO	Massive Multiplayer Online
MMORPG	Massive Multiplayer Online Role-playing Games

MR	Mixed Reality
NLP	Natural Language Processing
PSP	Physical Service Providers
QoE	Quality of Experience
QoS	Quality of Service
RL	Reinforcement Learning
RSU	Roadside Unit
SAA	Sample-Average Approximation
SCOS	Stochastic Coded Offloading Scheme
SIP	Stochastic Integer Programming
SSP	Sensing Service Providers
SSRA	Stochastic Semantic Resource Allocation
SSQ	Simulator Sickness Questionnaire
UAV	Unmanned Aerial Vehicles
UGC	User-generated Content
VLC	Visible Light Communication
VR	Virtual reality
VSP	Virtual Service Provider
YOLO	You Only Look Once

Chapter 2

Literature Review

The Metaverse is an *embodied* version of the Internet that comprises a *seamless* integration of *interoperable, immersive, and shared virtual ecosystems* navigable by user avatars. In the following, we explain each keyword in this definition.

- *Embodied:* The Metaverse blurs the boundary between the virtual and physical worlds [30], where users can “physically" interact with the virtual worlds with a tangible experience, e.g., by using 3D visual, auditory, kinesthetic, and haptic feedback. The Metaverse can utilize AR to extend the virtual worlds into the physical world.
- *Seamless/Interoperable:* Much like the physical world, the belonging of users’ avatars in one virtual world should not lose its value when brought to another seamlessly, even if the virtual worlds are developed by separate entities. In other words, no single company can “own” the Metaverse.
- *Immersive:* The Metaverse can be “experienced" beyond 2D interactions that allow users to interact with other users similar to that in the physical world.
- *Shared:* Much like the physical world, thousands of users should be able to co-exist in a single server session, rather than having users separated into different virtual servers. With users accessing the Metaverse and immersing themselves anytime and anywhere, the life-like interaction of users is thus shared globally, i.e., an action may affect any other users just as in an open world, rather than only for users at a specific server.

- *Ecosystem:* The Metaverse will support end-to-end service provisions for users with digital identities (DIDs), e.g., content creation, social entertainment, in-world value transfer regardless of nationalities of users, and physical services that will cross the boundary between the physical and virtual worlds. The Metaverse ecosystem, empowered by the decentralized design of blockchains, is expected to be sustainable as it has a closed-loop independent economic system with transparent operating rules.

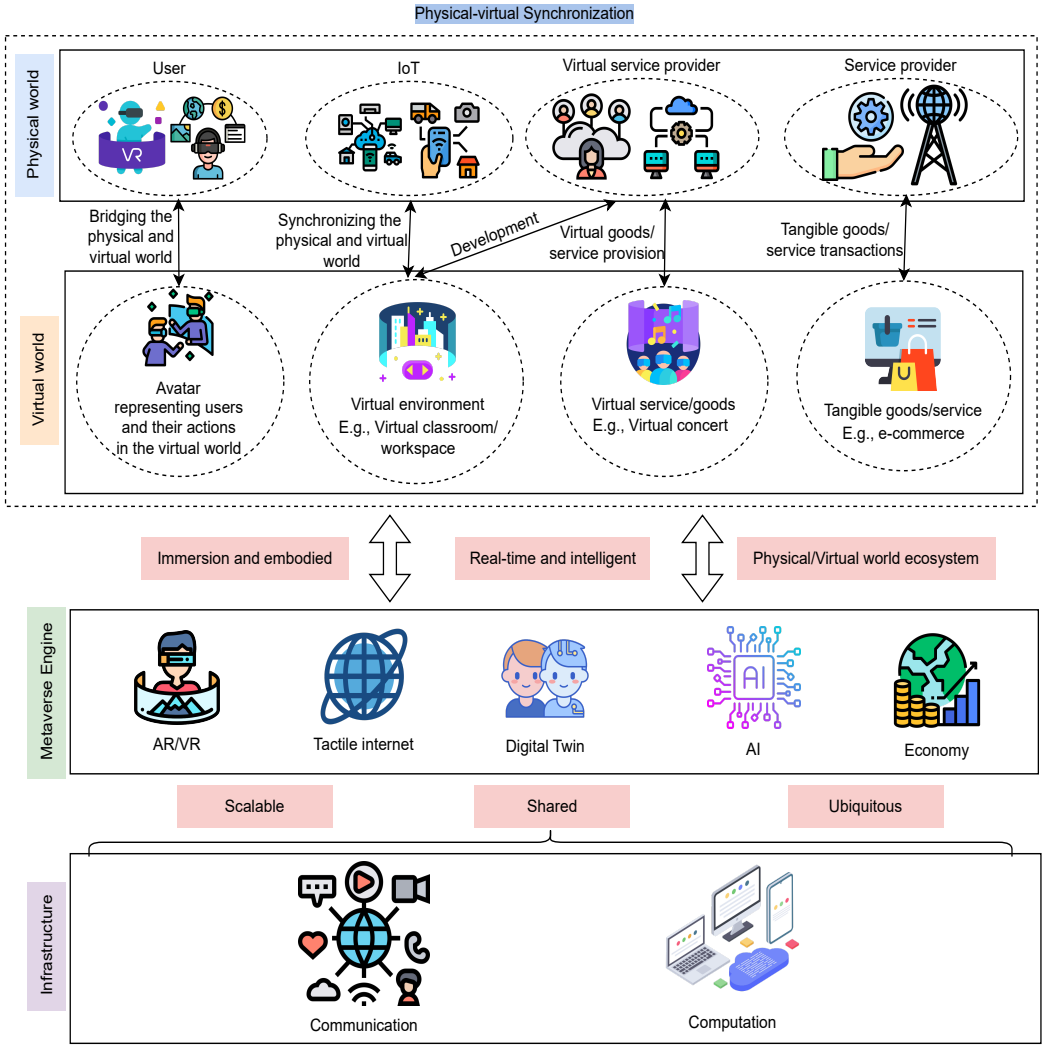


Figure 2.1: The Metaverse architecture features immersive and real-time physical-virtual synchronization supported by the communication and computation infrastructure.

2.1 The Architecture of the Metaverse

The metaverse can be considered to follow a three-layer architecture [31], i.e., **physical-virtual synchronization**, **the Metaverse engine**, and **the infrastructure** layer as shown in Fig.2.1.

2.1.1 Physical-virtual Synchronization Layer

The **physical-virtual synchronization** layer consists of physical and virtual worlds. Each non-mutually-exclusive stakeholder in the physical world controls components that influence the virtual world. The action of the stakeholder has consequences in virtual worlds that may, in turn, give feedback to the physical world. The key stakeholders are:

- *Users* can immerse the virtual worlds as avatars through various equipment, e.g., Head Mounted Displays (HMDs) or AR goggles [32]. In virtual societies, the users can, in turn, execute actions to interact with other users or virtual objects. For instance, a user wearing a VR headset can participate in a virtual conference, interacting with other participants' avatars and engaging with multimedia presentations or virtual whiteboards. A significant challenge here is the user's experience quality, heavily reliant on the latency and resolution of the VR equipment. Moreover, ensuring accessibility and mitigating VR-induced symptoms like motion sickness are ongoing concerns that developers need to address[33].

- *IoT and sensor networks* deployed in the physical world collect data from the physical world. The insights derived are used to update the virtual worlds, e.g., to maintain digital twins (DTs) for physical entities. Sensor networks may be independently owned by sensing service providers (SSPs) that contribute live data feeds (i.e., sensing as a service [34]) to virtual service providers (VSPs) to generate and maintain the virtual worlds. For example, sensors within a manufacturing plant can provide real-time data to a DT in the virtual world, simulating the factory's operations to forecast output or diagnose potential disruptions. Challenges include ensuring the security and privacy of the data transmitted and the scalability of sensor networks to handle vast amounts of data across various sources without latency issues[35].

- *Virtual service providers (VSPs)* develop and maintain the virtual worlds of the Metaverse. Similar to user-created videos today (e.g., YouTube), the Metaverse

is envisioned to be enriched with user-generated content (UGC) that includes games, art, and social applications [36]. These UGCs can be created, traded, and consumed in the Metaverse. Consider a platform similar to Roblox or Minecraft where users not only engage with the environment but also build and share their own game levels or art installations[37].

- *Physical service providers (PSPs)* operate the physical infrastructure that supports the Metaverse engine and respond to transaction requests that originate from the Metaverse. This includes the operations of communication and computation resources at edge networks or logistics services for the delivery of physical goods transacted in the Metaverse. For instance, a company might leverage edge computing to process user interactions within a city-scale augmented reality game to minimize latency[38].

2.1.2 Metaverse Engine Layer

The **Metaverse engine** layer obtains inputs such as data from stakeholder-controlled components that are generated, maintained, and enhanced by entities and their activities in the physical and virtual worlds.

- *AR/VR* enables users to experience the Metaverse visually, whereas *haptics* enables users to experience the Metaverse through the additional dimension of touch, e.g., using haptic gloves. This enhances user interactions, e.g., through transmitting a handshake across the world, and opens up the possibilities of providing physical services in the Metaverse, e.g., remote surgery. These technologies are developed by the standards that facilitate interoperability, e.g., Virtual Reality Modelling Language (VRML)[1], that govern the properties, physics, animation, and rendering of virtual assets, so that users can traverse the Metaverse smoothly and seamlessly.

- *Tactile Internet* enables users in the Metaverse to transmit/receive haptic and kinesthetic information over the network with a round-trip delay of approximately 1 ms [39]. For example, Meta's haptic glove project[2] aims to provide a viable solution to one of Metaverse's essential problems: using the tactile Internet with haptic feedback to enable users in the physical world to experience the embodied haptic and kinesthetic feeling of their avatars during interacting with virtual objects or other avatars.

[1] https://www.web3d.org/documents/specifications/14772/V2.0/part1/javascript.html

[2] https://about.fb.com/news/2021/11/reality-labs-haptic-gloves-research/

- *Digital Twin* enables some virtual worlds within the Metaverse to be modeled after the physical world in real time. This is accomplished through modeling and data fusion [40, 41]. DT adds to the realism of the Metaverse and facilitates new dimensions of services and social interaction [42]. For example, Nvidia Omniverse[3] allows BMW to blend its physical automotive factories and VR, AI, and robotics to improve its industrial precision and flexibility, which finally increases BMW's planning efficiency by around 30 percent.

- *Artificial Intelligence (AI)* can be leveraged to incorporate intelligence into the Metaverse for improved user experience, e.g., for efficient 3D object rendering, cognitive avatars, and AIGC. For example, the MetaHuman project[4] by EpicGames utilizes machine learning (ML) to generate life-like digital characters quickly. The generated characters may be deployed by VSPs as conversational virtual assistants to populate the Metaverse. A comprehensive review of AI training and inference in the Metaverse is presented in [43].

- *Economy* governs incentives and content creation, UGC trading, and service provisioning to support all aspects of the Metaverse ecosystem. For example, VSP can pay the price to SSP in exchange for data streams to synchronize the virtual and physical world [44]. Metaverse service providers can also purchase computing resources from cloud services [45] to support resource-constrained users. Moreover, the economic system is the driving engine that incentivizes the sustainable development of digital assets and DIDs for the sustainable development of the Metaverse.

2.1.3 Infrastructure Layer

The **infrastructure layer** enables the Metaverse to be accessed at the edge.

- *Computation:* Today, MMO games can host more than a hundred players in a single game session and hence require high-specification GPU requirements. VRMMO games, which are the rudiments of the Metaverse system, are still scarce in the industry. The reason is that VRMMO games may require devices such as HMDs to be connected to powerful computers to render both the immersive virtual worlds and the interactions with hundreds of other players. To enable ubiquitous access to the Metaverse, a promising solution is the cloud-edge-end computation paradigm. Specifically, local computations can be

[3] https://blogs.nvidia.com/blog/2021/04/13/nvidia-bmw-factory-future/
[4] https://www.unrealengine.com/en-US/digital-humans

(a) Meta collaborates with leading museums to provide the users with a brand new perspective when connecting with the culture [49].

(b) 3D objects can be created with the assistance of AI(Builder Bot) [50].

Figure 2.2: Screenshots of Meta AR and Meta Builder Bot.

performed on edge devices for the least resource-consuming task, e.g., computations required by the physics engine to determine the movement and positioning of an avatar. To reduce the burden on the cloud for scalability and further reduce end-to-end latency, edge servers can be leveraged to perform costly foreground rendering, which requires less graphical details but lower latency [46]. The more computation-intensive but less delay-sensitive tasks, e.g., background rendering, can, in turn, be executed on cloud servers. Moreover, AI techniques of model pruning and compression, as well as distributed learning, can reduce the burden on backbone networks. We further discuss these concepts in Section 2.3.

- *Communication:* To prevent breaks in the presence (BIP), i.e., disruptions that cause a user to be aware of the real-world setting, AR/VR and haptic traffic have stringent rate, reliability, and latency requirements [47]. Due to the expected explosive data traffic growth, ultra-dense networks deployed in edge networks can potentially alleviate the constrained system capacity. Moreover, the B5G/6G communication infrastructure [48] will play an instrumental role towards enabling post-Shannon communication to alleviate bandwidth constraints in managing the explosive growth of communication costs. We discuss these concepts in greater detail in Section 2.4.

2.2 Current Development

Companies across many sectors, such as entertainment, finance, healthcare, and education, can work closely to develop Metaverse hand in hand. For example, theme parks have become one of the ideal candidates to be situated in the Metaverse since they are expensive to construct in the physical world but can be done quickly and

safely in the Metaverse. In 2020, Disney unveiled their plans to create a theme park in the Metaverse. The company reiterated this vision at the 2021 Q4 earnings call [51]. This section looks at some existing and prospective developments that are “early forms" of the Metaverse. These developments are mainly classified into two categories as follows.

Table 2.1: Features of representative Metaverse examples.

Metaverse Example			Blockchain	AR/VR	DT	UGC	AI	
MMORPG	World of Warcraft		✗	✗	✗	✓	✓	AI enemies
	Grand Theft Auto Online		✗	✗	✓	✓	✓	AI in rockstar advanced game engine
	Final Fantasy XIV		✗	✗	✗	✓	✓	AI enemies
	Pokemon Go		✗	✓	✓	✗	✓	Plan route more strategically
	Animal Crossing: New Horizons		✗	✗	✗	✓	✓	AI NPCs
	Second Life		✗	✗	✗	✓	✓	AI NPCs
	Minecraft		✗	✓	✗	✓	✓	Platform to train AI
	DragonSB		✓	✗	✗	✓	✓	AI enemies
Applications of the Metaverse	Smart Cities	Metaverse Seoul	✓	✓	✓	✓	✓	AI NPCs
		Barbados Metaverse Embassy	✓	✓	✓	✓	✓	AI NPCs
	Entertainment	AltspaceVR	✓	✓	✓	✓	✓	AI NPCs
		Decentraland	✓	✓	✗	✓	✓	AI NPCs
	Education	Xirang	✓	✓	✓	✓	✓	AI NPCs
		CUHKSZ	✓	✗	✓	✓	✓	AI NPCs
	Work	Horizon Workrooms	✓	✓	✓	✓	✓	Tools for designing virtual content
		Microsoft Mesh	✓	✓	✓	✓	✓	Tools for avatars, holoportation, and spatial rendering
	Healthcare	Telemedicine	✓	✓	✓	✓	✓	Data analysis and collaboration

2.2.1 Massive Multiplayer Online Role-playing Games

Many have argued that the Metaverse is not new since it has long been implemented in video games. While video games of today may not completely be classified as the Metaverse according to the technical definition provided in Chapter 2, many video games have features that are rather similar to the Metaverse. For example, in massive multiplayer online role-playing games (MMORPG), players can enter the game and live in the digital world using their avatars. These avatars have roles tagged to the players, just as how one has a job in the physical world. MMORPGs can support a massive number of players to interact in a 3D-rendered virtual world. Some key examples are as follows:

- **Second Life:** This is among the first attempt to create a complete virtual world in which players could live. In second life, players control their avatars and can do almost anything they can in the physical world, from job seeking to getting married [52]. Besides, players can customize the environment they live in. The economy in Second Life is supported by the Linden Dollar[5], a virtual currency, which can be traded for real-world currency.

- **Minecraft:** The popular game allows anyone to create their own content by using 3D cubes in an infinite virtual world, and it supports user access using VR devices such as Oculus Rift. Digital copies of cities around the world, e.g., Kansas and Chicago, have been built using Minecraft and can be accessed by any user[6].

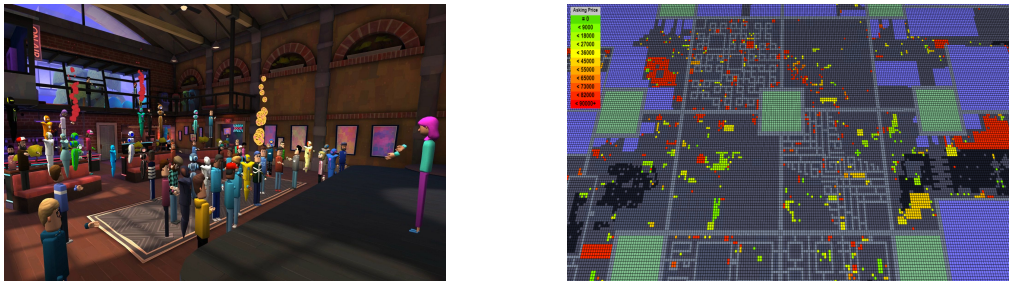
- **DragonSB:** DragonSB is the first Metaverse MMORPG built on Terra Protocol and Binance Smart Chain Platform [53]. The players will be rewarded with SB tokens, where the token will be listed on both centralized and decentralized exchanges.

Other popular MMORPGs are World of Warcraft, Grand Theft Auto Online, Final Fantasy XIV, Pokemon Go, and Animal Crossing: New Horizons. For a detailed discussion of these games, the reader may refer to [31]. Following the classification framework [31], we present a taxonomy for the above gaming Metaverse according to their features in Table 2.1.

2.2.2 Applications of the Metaverse

Beyond the gaming industry, multiple entities, from governments to tech companies, have announced their interest in establishing a presence in the Metaverse. This

[5] https://www.investopedia.com/terms/l/linden-dollar.asp
[6] https://www.minecraftmaps.com/tags/real-cities-in-minecraft



(a) Screenshot of AltspaceVR platform [54].

(b) Screenshot of the non-fungible LAND tokens constitute the Decentraland Metaverse [55].

Figure 2.3: Screenshots of (a) AltspaceVR and (b) Decentraland.

includes the smart cities [56], entertainment industry, education institutions, working environments, and healthcare (Fig. 2.3).

- **Metaverse Seoul:** The Seoul Metropolitan Government plans to release Metaverse Seoul in Seoul Vision 2030 [57]. This platform contains a virtual city hall, tourism destinations, social service centers, and many other features. The local users can meet with avatar officials for consultations and provisions of civil services. Foreign users can tour vividly replicated locations and join in celebrating festival events while wearing their VR headsets. This aims to boost the tourism industry.

- **Barbados Metaverse Embassy:** Barbados, an island nation, is prepared to declare digital real estate sovereign property by establishing a Metaverse embassy [57]. The government works with multiple Metaverse companies to aid the country in land acquisition, the design of virtual embassies and consulates, the development of infrastructure to provide services such as “e-visas", and the construction of a “teleporter" that will allow users to transport their avatars across worlds.

- **AltspaceVR:** AltspaceVR is a virtual meeting space designed for live and virtual events, and it is compatible with a variety of operating systems [58]. Basically, the users of AltspaceVR are allowed to meet and engage online through the use of avatars. It recently implemented a safety bubble feature to reduce the risk of inappropriate behavior and harassment.

- **Decentraland:** Decentraland is built on the Ethereum blockchain [55]. It is a decentralized virtual world that allows users to buy and sell virtual land. Users can make their own micro-worlds complete with virtual trees, planes, and other features. Everything that is available in the physical world can be found in Decentraland, from transportation systems to hotels.

- **Xirang:** The Chinese tech giant Baidu has released a Metaverse application named Xirang (which means land of hope), allowing users to explore the virtual worlds using portable gadgets such as smartphones and VR devices [59]. In addition, personalized avatars can be automatically generated from the user's uploaded images. Xirang hosted a three-day virtual AI developers' conference in 2021 that supports 100,000 users at once. This is an important step towards realizing the shared Metaverse.

- **The Chinese University of Hong Kong, Shenzhen (CUHKSZ):** CUHKSZ Metaverse is a virtual copy of the physical campus. It is powered by the blockchain to provide the students with an interactive Metaverse, a hybrid environment in which students' real-world behaviors can have an impact on the virtual world, and vice versa [31].

- **Telemedicine:** Telemedicine is immersion in the healthcare industry, allowing patients to connect virtually with healthcare professionals when they are physically apart [60]. Doctors and their patients can meet in the virtual world through VR to provide telemedicine consultations. While physical follow-ups, medical scans, and tests can then be booked and performed in a facility near the patient, the results can be sent to any preferred doctor.

2.3 Computation in Metaverse

Beyond reliable and low-latency communication systems, the resource-intensive services provided by the Metaverse will require costly computation resources to support [31]. Some screenshots of exemplary Metaverse applications, such as Meta AR, hand gesture AR, and AI-generated content (AIGC) applications, are illustrated in Fig. 2.4. In specific, the Metaverse requires the execution of the following computations:

- **High-dimensional Data Processing:** The physical and virtual worlds will generate vast quantities of high-dimensional data, such as spatiotemporal data [61], that allows users to experience realism in the Metaverse. For example, when a user drops a virtual object, its behavior should be governed by the laws of physics for realism. The virtual worlds with physical properties can be built upon physics game engines, such as the recently developed real-time Ray Tracing technology to provide the ultimate visual quality by calculating more bounces in each light ray in a scene [62]. During these activities, this generated

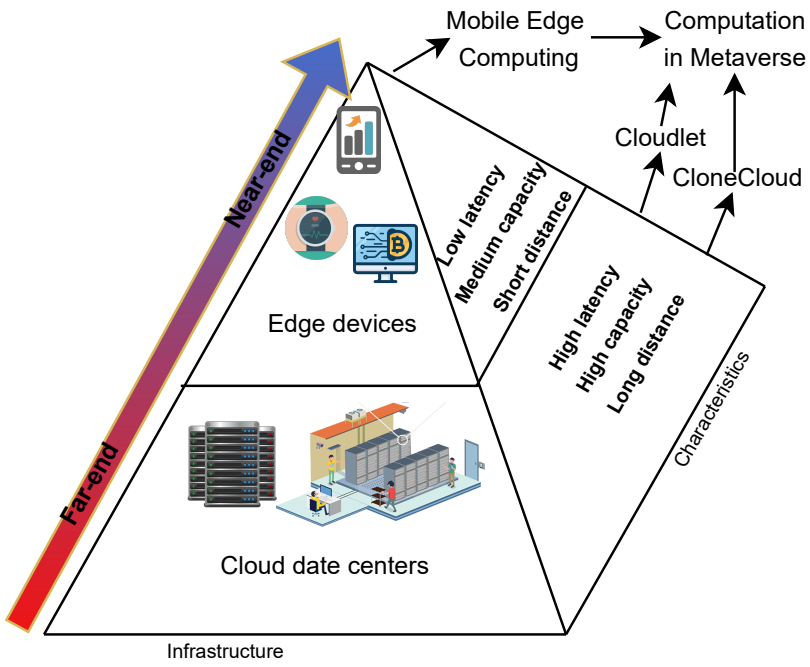


Figure 2.4: Various types of computing infrastructure to support the computation in the Metaverse and their characteristics.

high-dimensional data will be processed and stored in the databases of the Metaverse.

- **3D Virtual World Rendering:** Rendering is the process of converting raw data of virtual worlds into displayable 3D objects. When users are immersed in the Metaverse, all the images/videos/3D objects require a rendering process to display on AR/VR devices for users to watch. For example, Meta operates research facilities that set up collaborations with museums to transform 2D paintings into 3D [49] as illustrated in Fig. 2.2a. In addition, the Metaverse is expected to produce and render 3D objects in an intelligent way. For example, as shown in Fig. 2.2b, 3D objects, e.g., sky, islands, mountains, sea, and even sounds, can be generated with the assistance of AI (i.e., AIGC) through audio inputs [50].

- **Avatar Computing:** Avatars are digital representations of users in the virtual worlds that require ubiquitous computation and intelligence in avatar generation and interaction. On the one hand, avatar generation is based on the ML technologies, such as computer vision (CV) and natural language processing (NLP). On the other hand, the real-time interaction between avatars and users is computationally intensive to determine and predict the interaction results. For example, AI can synergize with AR applications to provide avatar services such

Figure 2.5: Screenshot from Microsoft Flight Simulator trailer [1].

as retrieving text information from the users' view, text amplification on paper text, and using a hand gesture to copy and paste files between computers [63].

2.3.1 The Cloud-Edge-End Computing Paradigm

In this subsection, we introduce the mobile cloud-edge-end collaborative computing paradigm, as shown in Fig. 2.4, that provides ubiquitous computing and intelligence for users and service providers in the Metaverse.

Mobile Cloud Computing: Several of the world's tech behemoths have made the full dive Metaverse as their newest macro-goal. In the next decade, billions of dollars will be invested in cloud gaming [64], with the notion that such technologies will underlie our online-offline virtual future [30]. Cloud gaming integrates cloud computing to enable high-quality video games. Mobile cloud computing provides computing services to either mobile devices or mobile embedded systems. The cloud provider installs computing servers in their private infrastructure, and services are provided to clients via virtual machines [65]. However, the most cost-effective approach for an organization is to combine cloud and in-house resources rather than choosing one over the other [66]. Fortunately, a few system architectures such as Cloudlet [67] and CloneCloud [68] are proposed to manage mobile cloud computing and provide support for the device with weak computation so that they can access the Metaverse. Microsoft Flight Simulator is one of the applications that combines both cloud and in-house resources. In Microsoft Flight Simulator, there are 2 trillion individually rendered trees, 1.5 billion buildings, and almost all the roads, infrastructure, and airports of international cities. Microsoft Flight Simulator provides the users with a realistic flying experience, as shown in Fig. 2.5. However, Microsoft Flight Simulator requires over 2.5 petabytes of data to run. It is even more challenging to run on mobile devices such as portal gaming devices or smartphones. Therefore,

most simulator data is stored and processed in the cloud. The gameplay content (e.g., bird's eye view of cities) is directly streamed to the user device (i.e., personal computers). The local device only stores a core amount of data, with computations offloaded where necessary.

Edge Computing: The Metaverse is a black hole of computational resources. It will always require as much computation capacity as possible and the mobile users accessing the Metaverse require the latency to be as low as possible to deliver real-time, advanced, computation-intense capabilities like speech recognition and AR/VR [69]. Therefore, edge computing provides low latency, high efficiency, and security to sustain the Metaverse. The edge computing observes the computation capability in the local network and computes the task that could have been carried out in the cloud, thereby reducing latency and burden on backbone networks. The additional computation capacity and storage of devices on edge are used [70].

2.3.2 Efficient Edge-enabled Computation in the Metaverse

The edge-enabled Metaverse can be accessed using various types of mobile and wired devices, including HMD and AR goggles, VR allows the users to experience a simulated virtual world while in the physical world. The VR devices generate sensory images that form the virtual worlds [31]. Real-time rendering is required in the Metaverse so that the sensory images can be produced quickly enough to form a continuous flow rather than discrete events. For example, education platforms can incorporate VR to improve lesson delivery in the virtual world. Unlike VR, AR allows the user to interact with the physical world that has been digitally modified or augmented in some way. AR applications are best suited for scenarios in which users are physically present in the physical world, such as on-the-job training and computer-assisted tasks [71]. However, users' devices have limited computation capability, memory storage, and battery. Intensive AR/VR applications that are required for the immersive Metaverse cannot run smoothly on such devices. Fortunately, based on the cloud-edge-end collaborative computing paradigm, users' devices can leverage ubiquitous computing resources and intelligence for remote rendering and task offloading [72] to overcome these challenges.

At mobile edge networks, the cloud-edge-end computing architecture allows the computations, such as high-dimensional data processing, 3D virtual world rendering, and avatar computing, to be executed where data is generated [11]. For example, when the user is interacting with avatars in the Metaverse from a vehicle to perform task computation, the vehicle can send the data to nearby vehicles or roadside units to perform the computation instead of cloud offloading, thereby dramatically reducing end-to-end latency. The proposed cloud-edge-end collaborative computing paradigm

is a promising solution to reduce latency for mobile users [73] by leveraging computing capacities at the mobile edge networks [74]. This is important as mobile users should be able to access Metaverse anytime and anywhere. Furthermore, mobile edge computing lightens the network traffic by shifting the computation from the Internet to the edge. Therefore, mobile edge computing is increasingly recognized as a viable solution to enhance computation and storage capacities, essential for various advanced vehicular applications. Significant research focuses on optimizing energy efficiency during computational offloading. [75, 76, 77, 78, 79] discuss various strategies to reduce energy consumption while maintaining performance constraints like latency.

Specifically, [75] proposes an energy-efficient mechanism that considers both the energy costs of computing tasks and data transfers. [76] examines multi-user resource allocation to minimize overall mobile energy usage with latency constraints. [77] offers a framework for joint optimization of radio and computational resources, balancing energy use and latency. [78] explores offloading strategies aimed at reducing energy use on mobile devices. Lastly, [79] focuses on minimizing energy consumption through optimal local execution or cloud offloading of mobile applications.

At mobile edge networks, edge servers are equipped with edge servers that can provide computing resources to mobile users [73]. For example, the authors in [80] partition a task into smaller sub-tasks and decide if the mobile users should perform binary offloading to edge servers or perform minimum offloading. Hence, VR content can be transmitted to nearby edge servers for real-time rendering to improve the QoS of the users. It can be observed that the most common advantage of them is that they can adapt to various environments and requirements of AR/VR rendering. However, they have different disadvantages, such as they might be myopic, unfair, unscalable, or inaccurate solutions. In the following sections, I discuss some of the solutions that can be used to support Edge-enabled Computation in the Metaverse.

2.3.3 Wireless Energy Transfer

The fusion of mobile edge computing (MEC), wireless energy transfer, and the Metaverse is paving the way for a transformative digital future. MEC brings computation and data storage closer to the location where it is needed, reducing latency and enhancing application performance in immersive metaverse environments. Meanwhile, wireless energy transfer ensures an uninterrupted power supply to edge devices, facilitating continuous, real-time interactions in the Metaverse without the limitations of battery dependency [81]. This synergy not only boosts the efficiency of virtual worlds but also supports the creation of more complex and sustainable virtual

realities, where users can engage seamlessly with digital content and services.

Wireless Power Transfer (WPT) enhances the battery longevity of smart devices, allowing them to function autonomously[82]. The incentive to use harvested power from base stations (BS) for offloading computations to a nearby MEC server is discussed[83], along with the role of multi-antenna APs that broadcast power enabling local processing or server offloading [84]. Applications of WPT-MEC offloading range from IoT and UAVs to mobile devices. Additionally, deploying WPT systems supports IoT sensors in gathering and transmitting data for processing [85]. Monitoring sensor data, considering faults and disturbances, is critical for stabilizing WPT and IoT networks [86]. A distributed WPT system with multiple BSs also powers IoT devices [87], and UAVs can serve as data aggregators and power suppliers [88].

2.3.4 Coded Distributed computing

In the Metaverse, all users are immersed simultaneously in the shared virtual worlds. The stragglers on edge networks with poor connections can affect the immersive experience of other users by causing delayed responses to social interaction. The reliance on the AR device can be reduced when parts of the task are offloaded to other edge devices. The authors in [89] propose a parallel computing approach that leverages the computation capabilities of edge servers and nearby vehicles. This approach partitions the AR task into two sub-tasks at a calibrated proportion so that the edge server and the nearby vehicle can compute the two sub-tasks simultaneously. A joint offloading proportion and resource allocation optimization algorithm is proposed to identify the offloading proportion, communication, and computation resource allocation. Whenever the computational task complexity is high, most of the computations are performed within the edge server rather than in the vehicle so that the computing resources in edge servers can be fully utilized. Nevertheless, there are many design problems, i.e., computing frameworks are vulnerable to uncertain disturbances, such as node failures, communication congestion, and straggler nodes [90]. Only in recent years, coding techniques gained great success in improving the resilience of communication, storage, and cache systems to uncertain system noises [91].

As several AI-driven AR/VR rendering tasks in the Metaverse, e.g., training DL models, are mainly performing matrix multiplications, the computation process can be sped up if we can accelerate the matrix multiplication process. The authors in [92] propose a coded distributed computing scheme known as polynomial codes to accelerate the computation process of large-scale distributed matrix-matrix multiplications. It can achieve the optimal recovery threshold, and it is proven that the recovery threshold does not increase with the number of worker nodes. The system structure of a CDC system

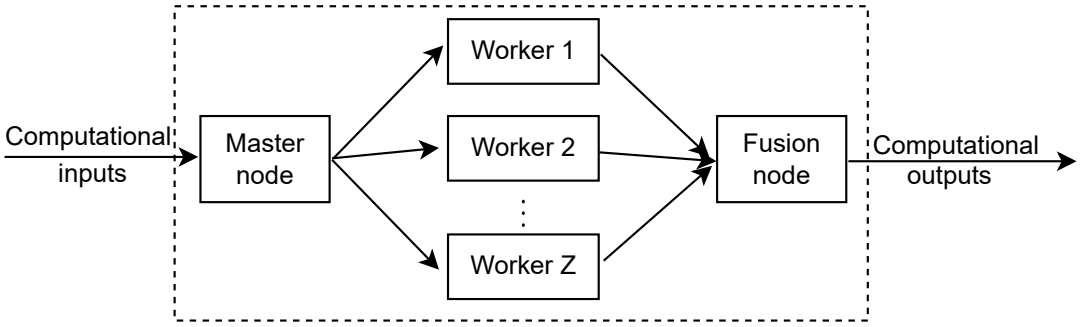


Figure 2.6: An illustration of a computational system.

is shown in Fig.2.6.

- **Master Node:** receives computation inputs, encodes and distributes them to the worker nodes.
- **Worker Node:** perform pre-determined computations on their respective inputs in parallel.
- **Fusion Node:** receives outputs from successful worker nodes and decodes them to recover the final output.

Definition 1. [Recovery threshold] The recovery threshold is the worst-case minimum number of successful workers required by the VSP to complete the computation [93].

In the vehicular Metaverse, the authors in [94] propose a collaborative computing paradigm based on CDC for executing Metaverse rendering tasks collaboratively among vehicles. To improve the reliability and sustainability of the proposed paradigm, they leverage the reputation-based coalition formulation and the Stackelberg game to select workers in the vehicular Metaverse. In addition, they propose a blockchain-based secure reputation calculation and management scheme for physical entities, i.e., vehicles, to store historical records during CDC. Their experimental results demonstrate the effectiveness and efficiency in alleviating stragglers with malicious workers. However, the large number of physical entities in the Metaverse increases the encoding and decoding cost in CDC. In response, various types of coding schemes, such as MatDot codes and PolyDot codes, are proposed in [93], which can be adopted in CDC of the Metaverse. These coding techniques can be implemented in edge computing to achieve smooth performance degradation with low-complexity decoding with a fixed deadline [95]. MatDot codes have a lower recovery threshold than the polynomial codes by only computing the relevant cross-products of the sub-tasks, but it has a higher communication cost. PolyDot codes use polynomial codes and MatDot codes as the two extreme ends and construct a trade-off between the recovery threshold and communication cost.

There have been many other works to reduce the communication load [96, 97] that are capable of improving the overall communication time. The authors in [96] introduced a Coded MapReduce framework to reduce the inter-server communication load by a multiplicative factor that grows linearly with the number of servers in the system. The authors in [97] presented a technique known as Short-Dot to reduces the cost of computation, storage, and communication. Besides reducing the communication load, Short-Dot also tackles the straggler issue. It completes the computation successfully by ignoring the stragglers.

However, the works mentioned above mainly focus on the designing of different CDC schemes. Therefore, in this study, I adopt CDC in the Metaverse network to alleviate the straggler problem and improve network reliability.

2.3.5 Stochastic Integer Programming

The Metaverse will host millions of users concurrently and try to satisfy each user who has a different demand schedule and usage requirements. When an AR/VR device is used, the AR/VR device may associate with an edge server to provide computation support for real-time services. However, the optimal number of edge computation services cannot be precisely subscribed in advance if the user's demand is uncertain. Moreover, given the stochastic fluctuations in the demand schedule, it is not practical for the users' demand to be assumed to a constant. If the Metaverse VSP subscribes to insufficient edge servers for the AR/VR devices to offload, the AR/VR applications would be unable to provide real-time services. The authors in [45] propose two-stage stochastic integer programming (SIP) to model the uncertainty in user demand. The simulation results show that despite the uncertainty regarding user demand, the optimal number of edge servers required to meet the latency requirement can be obtained while minimizing the overall network cost.

Stochastic integer programming is one of the important tools to incorporate uncertainty in optimization problems [98]. SIP can be applied to various fields to solve the optimization problem, e.g., production planning [99]. SIP assumes uncertain data as random variables with known probability distributions and uses sampled values from this distribution to build a scenario tree and optimize over the expectation [100]. SIP models can correct the decisions using the concept of recourse. In this idea, some decisions have to be made before realizing uncertain parameters and some decisions after their realization [101]. SIP models can be formulated as two-stage and multi-stage problems. The general model formulation of the two-stage SIP is expressed as follows [101]:

$$\min c^{\intercal}x + \mathbb{E}[Q(x,\omega)], \tag{2.1}$$

where

$$Q(x,\omega) = q_{\omega}^{\intercal}y(\omega), \tag{2.2}$$

subject to:

$$Ax = b, \tag{2.3}$$
$$Tx + Wy(\omega) \leq h, \tag{2.4}$$
$$x \geq 0, \tag{2.5}$$
$$y(\omega) \geq 0. \tag{2.6}$$

The first stage is represented by (2.1), and the second stage is represented by (2.2). c is the cost parameter, and $\intercal$ represents the transition matrix. A denotes the coefficients, and b is the right-hand side of the first stage constraints (2.3). $\omega = (q, T, W, h)$ is the data of the second-stage problem and $\mathbb{E}$ denote mathematical expectation. In a two-stage SIP, x is a "here-and-now" decision variable [102]. The decision must be made without realizing the uncertain data ω. The first stage is to minimize the cost $c^{\intercal}x$ and the expected cost of the second stage decision. Once the uncertain data is known, the recourse variable y can make a corresponding decision using constraints (2.4) and (2.6). Each possible realization of the uncertain data is represented by $\omega_i \in \Omega$ and is called a scenario. ω_i refers to the uncertain data in scenario i.

The two-stage SIP can be reformulated and solve using the deterministic equivalent (DE). DE assumes that the random data have a finite distribution and each scenario ω_i occurs with probability $P(\omega_i) = p_i$, $\forall i = 1, \ldots, I$ and $\sum_i p_i = 1$. Thus, $\mathbb{E}[Q(x,\omega)] = \sum_i p_i q^{\intercal}y_i$, where y_i is the optimal corrective decision variable in the second-stage for the scenario ω_i. The DE formulation is expressed as follows [103]:

$$\min c^{\intercal}x + p_1 q^{\intercal}y_1 + \ldots + p_I q^{\intercal}y_I, \tag{2.7}$$

subject to:

$$
\begin{aligned}
Ax &= b, && (2.8)\\
T_1x + W_1y(\omega_1) &\leq h_1, && (2.9)\\
T_2x + W_2y(\omega_2) &\leq h_2, && (2.10)\\
&\vdots &&\\
T_Ix + W_Iy(\omega_I) &\leq h_I, && (2.11)\\
x &\geq 0, && (2.12)\\
y(\omega_i), \ldots, y(\omega_I) &\geq 0. && (2.13)
\end{aligned}
$$

The authors in [104] applied the two-stage SIP to optimize the resource provisioning cost in cloud computing. In the courier delivery serves, the authors in [105] use the two-stage SIP to plan an optimal vehicle delivery route. A multi-stage SIP is a generalization of the two-stage SIP to the sequential realization of uncertainties. For example, the authors in [106] use a multi-stage SIP to optimize electricity generation, storage, and transmission investments over a long planning horizon. The recourse is the key concept behind SIP. With the idea of recourse, corrective action can be made after a random event has taken place. Apart from resource provisioning in cloud computing, planning a delivery route and optimizing electricity generation, there are new areas (CDC and the Metaverse) that require SIP to model the real-world uncertainties. To the best of my knowledge, the application of stochastic programming to coded distributed computing has been less studied.

2.4 Communication in Metaverse

The immersive user experience of the Metaverse requires high responsiveness and high-bandwidth connectivity [107]. Table 2.2 summarizes the bandwidth and latency requirements for various VR technologies. The mobile edge networks are expected to provide users with efficient communication and networking coverage with high-speed, low-delay, and wide-area wireless access so that users can immerse themselves in the Metaverse and have seamless and real-time experience [31]. Unlike the traditional transmission of 2D images, the need to transmit 3D virtual items and scenarios in the Metaverse to provide users with an immersive experience places a significant burden on the current 5G network infrastructure. Overall, the following characteristics of the Metaverse present significant new challenges to mobile edge networks in providing Metaverse services:

Table 2.2: Latency requirements for various VR.

	VR Resolution	FPS	Equivalent Resolution	Maximum Throughput (Mbps)	Maximum Streaming Latency	Maximum Interactive Latency
Early VR	1k×1k	30	240p	25	40	10
Entry VR	2k×2k	30	SD	100	30	10
Advance VR	4k×4k	60	HD	400	20	10
Extreme VR	8k×8k	120	4K	1000-2350	10	10

1. *Immersive 3D streaming:* To blur the boundary of the physical and virtual worlds, 3D streaming enables the immersive experience of users in the Metaverse to be delivered to millions of users at mobile edge networks.

2. *Multi-sensory communications:* The Metaverse consists of multiple 3D virtual worlds [108], where users can immerse as avatars with multi-sensory perception, including visual, auditory, haptic, and kinesthetic. Therefore, edge users need to have ubiquitous connectivity to the Metaverse to access 3D multimedia services, such as AR/VR and the tactile Internet, anytime and anywhere.

3. *Real-time interaction:* Massive types of interactions form the basis for user-user and user-avatar real-time interactions in the Metaverse. Therefore, social multimedia services in the Metaverse have to be delivered under stringent requirements in real-time interactions, such as motion-to-photon latency [109], interaction latency, and haptic perception latency.

4. *Seamless physical-virtual synchronization:* To provide seamless synchronization services among the physical and virtual worlds, physical and virtual entities need to communicate and update their real-time statuses with each other. The importance and valuation of synchronizing data may be affected by the age of information (AoI) or the value of information (VoI) [110].

To address these issues, we provide a review of cutting-edge communication and networking solutions for the edge-enabled Metaverse. The challenges can be solved from three areas: Resource Allocation for VR/AR Streaming, semantic communication, and resource allocation for physical-virtual synchronization.

2.4.1 Resource Allocation for VR Streaming

In the Metaverse, the most direct way for wireless users to access the virtual worlds at mobile edge networks is through VR streaming. Edge servers are expected to stream

VR content to users immersed in the Metaverse with high-performance guarantees in terms of rate, reliability, and latency [111]. However, the spectrum resources of edge networks and wireless devices' energy resources are limited, leading to challenging resource allocation problems for VR streaming services. Performance analysis and optimization of VR streaming services in the Metaverse require a trade-off between multiple objectives and a shift from simple averaging to fine-grained analysis to address spurs, distribution, and quality of service (QoS) issues. For example, to address the spurs of heterogeneous users, mobile edge networks provide a variety of alternative VR transmission modes, allowing users to choose the service mode that best suits their needs. Meanwhile, users can dynamically decide the quality and utility of multimedia services based on their own demand and cost of distribution.

The VR content of the Metaverse is expected to be delivered in 6G heterogeneous networks (HetNets) [112, 113]. First, VR content delivered in macrocell broadcasting can provide the largest coverage but identical QoS to all VR users within the coverage of marcocells [114]. Second, to improve the performance of macrocell broadcasting, small cell unicasting VR streaming [115] can help macrocells to extend their coverage and thus improve QoS of the VR streaming system. To efficiently utilize the available communication resources of mobile devices, VR users can also adaptively participate in device-to-device (D2D) multicast clusters and send shared reused tracking signals to service providers. In this way, the performance of the VR steaming system can be improved by increasing the total bit rate within the cell while ensuring short-term equality among users [116]. To provide accurate indoor positioning services to VR users, the authors in [117] develop a VR service provisioning model over terahertz (THz) /visible light communication (VLC) wireless networks. The THz/VLC wireless networks [118] are capable of providing larger available bandwidth for the VR content transmission requiring unprecedentedly high data rate [119]. In the proposed framework, VR users can turn on a suitable VLC access point and select the base station with which they are associated. Aiming at the quick adaptation of new users' movement patterns, they propose a meta-reinforcement learning (RL) algorithm to maximize the average number of successfully served users.

2.4.2 Resource Allocation for AR Adaptation

In addition to VR streaming, another way to access the Metaverse is to adapt to the real-world environment through AR or mobile AR (MAR) [120]. At mobile edge networks, users use edge-supported AR devices to upload and analyze their environment. After AR customization, they download the appropriate AR content from edge servers and access the Metaverse. However, resources on the edge networks are limited, while the

customization and communication resources used for AR adaptation are enormous [121, 122, 123]. Therefore, the resource allocation problem has to be solved for efficient AR adaptation. Specifically, users need to balance a trade-off between the accuracy of adaptation and the latency of interaction, i.e., the longer the adaptation time, the higher the accuracy of AR adaptation.

In general, the higher the latency of AR services (i.e., the longer the processing time of the AR service provider and the longer the upload time of the MAR user), the higher the analysis accuracy of the service. To improve performance over edge MAR systems, the authors in [124] develop a network coordinator to balance the trade-off between analysis accuracy and service latency. They rely on the block coordinate descent method [125] to develop a multi-objective analysis algorithm for MAR services that enables quick and precise target analysis of AR content. This way, the AR server's resource allocation efficiency is improved, and it can better serve MAR users with different requirements in frame size and frame rate.

2.4.3 Semantic Communication

The Metaverse is a comprehensive multimedia system consisting of many semantically oriented applications. Metaverse content dissemination services can place a massive burden on today's data-oriented communication [126], where real-time traffic requires a channel with infinite capacity. Therefore, applications in the Metaverse require service diversity and service-level optimization to reduce the load on the wireless channels at mobile edge networks.

As one of the promising approaches to overcome the Shannon capacity limit [127], semantic communication is proposed as an intelligent communication scheme that focuses on the significance of transmitted messages rather than bitstream transmission [128]. The authors in [129] proposed a semantic communication framework in the wireless network, where a BS is used to transmit the semantic contents that are extracted from textual data to each user. A semantic similarity metric is used to measure the semantic accuracy and completeness of the recovered text. With the recent development of deep learning technologies, multiple semantic communication systems were developed for text transmission. The authors in [128] designed a deep learning-enabled semantic communication system known as DeepSC. DeepSC is used for semantic text transmission. With the help of Transformer, DeepSC seeks to maximize system capacity and reduce semantic errors when restoring the meaning of sentences. To understand the meaning in a sentence, *word2vec* model is introduced in [130] to capture the relationship between words. When the words are similar, their distance is closer in the vector space. *Word2vec* model can be used to

extract the semantic information from sentences. However, *word2vec* model has to be redesigned whenever the context is different. Therefore, a general word representation model bidirectional encoder representations from transformers (BERT) is introduced in [131]. Depending on the context, BERT can generate more than one vector representation for the same word.

2.4.4 Resource Allocation for Physical-Virtual Synchronization

Efficient resource allocation in internal communication between physical and virtual entities in the mobile edge network during the synchronization of physical and virtual worlds is key to seamless real-time immersion in the Metaverse [48]. To this end, the authors in [44] design a data collection incentive mechanism for self-interested edge devices. In this mechanism, IoT devices act as PSPs to collect data in the physical world. This mechanism uses a dynamic evolutionary approach that allows different device owners to enter into various contracts to update their policies. Eventually, service providers converge to a policy that copes with time-varying resource supply and demand to maximize the dynamic rewards. To extend [44] into dynamic IoV scenarios, the authors in [132] propose a distributed incentive mechanism based on the alternating direction method of multipliers to incentivize vehicles contributing their resources to the physical-virtual synchronization in the Metaverse. This mechanism can maximize the energy efficiency of vehicular twin networks in dynamic environments.

2.5 Conclusion and Chapter Discussion

This chapter has presented a review of Metaverse and the technologies that can be used to support it. Firstly, we begin with the definitions that are used to define the Metaverse. Then, we elaborate on the architecture of the Metaverse and describe the components that each layer contains. Afterward, we listed out the current development of the Metaverse across many sectors, such as entertainment, finance, healthcare, and education. Furthermore, we provide detailed reviews and analyses of the technologies that can be used to support the Metaverse from the communication and computation aspects. Below, we summarize the research gap from our review of the literature, which paves the subsequent chapters ahead in this thesis

1. In the Metaverse, which relies fundamentally on extensive datasets for creating immersive virtual experiences, one solution to support this data-intensive environment is the deployment of Unmanned Aerial Vehicles (UAVs),

commonly known as drones. These UAVs serve as airborne base stations (BS) capable of collecting and processing data from terrestrial nodes [133, 134]. They offer several advantages, including the ability to maintain line-of-sight communication with end-users and wireless connections can be established without a fixed infrastructure to extend communication coverage. However, apart from all those benefits, UAVs are faced with energy constraints [135], and thus, they cannot complete their computation tasks if the energy utilization is not scheduled correctly. Therefore, in Chapter 3, a CDC technique is adopted to allow the UAV to partially or fully offload their computation tasks. Then, a z-stage (multi-stage) SIP offloading scheme is proposed to optimize the offloading decision by considering the UAV energy consumption and stochastic uncertainties.

2. Edge-enabled Metaverse leverages the storage, communication, and computation capabilities of end devices and edge servers to bring computing power closer to where data are produced. To improve the scalability of edge computing, next-generation communication systems are envisioned to provide energy transfer and harvesting for energy-constrained end devices, e.g., through 6G base stations (BSs) [136]. However, the wireless charging technology is relatively nascent with variable charging efficiencies [22]. Therefore, in Chapter 4, we propose a stochastic resource optimization approach. Given the stochastic nature of wireless charging efficiency and edge servers computation capacities, which can only be observed *ex-post*, a computation strategy for each user is determined using the two-stage SIP. Furthermore, we introduce the efficient computation methods of Benders' decomposition and sample average approximation to address the complexity of the SIP problem.

3. In the future, machine learning models in the Metaverse will eventually be used to generate 3D images, animations, speech, and support blockchain technologies that allow for virtual transactions. However, the current Metaverse faces with certain limitations, i) the Metaverse facilities require incredible amounts of energy, i.e., data centers rely on AI and cloud services to store information [137]. ii) The current 5G communication system is insufficient to support the Metaverse, where millions or billions of devices are connected in the intersection between the virtual and physical world [138]. Given that current technologies are approaching the Shannon limit, traditional transmission methods are increasingly inadequate for meeting the rapidly growing and diversifying data traffic demands [139, 140]. In response, in Chapter 5, we propose a semantic cropping module and a resource

allocation scheme to minimize the operation cost/energy consumption of VSPs in the presence of stochastic uncertainties.

Chapter 3

Stochastic Coded Offloading Scheme

This chapter delves into the integration of Unmanned Aerial Vehicles (UAVs) within the rapidly evolving Metaverse, where the increasing proliferation of IoT technologies has led to an exponential growth in wirelessly connected devices. These devices are essential for supporting real-time applications like augmented and virtual reality, and smart city implementations, which demand high responsiveness and minimal delays. We address the substantial strain this growth places on traditional wireless communication infrastructures, particularly in dense urban areas or highly mobile environments. To mitigate these challenges, we explore the deployment of UAVs as dynamic airborne base stations that enhance data collection and processing capabilities while navigating energy constraints and computational demands.

The focus of this chapter is on the development and evaluation of a Stochastic Coded Offloading Scheme (SCOS) that employs Coded Distributed Computing (CDC) techniques combined with strategic computation offloading. This approach is designed to minimize energy consumption and optimize resource allocation amidst uncertainties such as weather variability, fluctuating data volumes, and task completion rates. Through a series of numerical experiments and system simulations, we demonstrate how SCOS significantly reduces operational costs and energy usage in UAV networks, providing a resilient framework capable of adapting to changes in environmental conditions and network demands.

Following the introduction, the chapter is structured to first present the system model and problem formulation. Subsequent sections detail the UAV type allocation and task allocation strategies, addressing how each phase of SCOS confronts specific

uncertainties. We conclude with a comprehensive analysis of our simulation results and a discussion on the broader implications of our findings for UAV integration in complex network environments.

3.1 Introduction

Due to the rapid advancement of Internet of Things (IoT) technologies within the Metaverse, the number of wirelessly connected devices is increasing exponentially [5], generating vast amounts of data essential for creating dynamic and responsive virtual environments [6]. These environments include real-time applications such as augmented and virtual reality [27] and smart cities [28], which are particularly sensitive to delays. In the Metaverse, for instance, real-time traffic [141] and parking information [142] are crucial for simulating realistic urban environments and managing virtual traffic flow seamlessly.

The surge in wirelessly connected devices places a significant strain on wireless communication infrastructures, especially in densely built urban areas or congested regions, such as those encountered in high-speed vehicular networks [143]. Here, traditional static roadside units (RSUs) and base stations (BSs) often struggle to deliver content efficiently to end-users, which can detract from the immersive quality of the Metaverse.

One innovative solution is the deployment of Unmanned Aerial Vehicles (UAVs), or drones, which can serve as airborne base stations to collect and process data from terrestrial nodes [133, 134]. UAVs, available in various designs like fixed wings or multi-rotors, maintain line-of-sight communication with end-users, significantly enhancing the quality of service (QoS). Their flexibility allows deployment in otherwise inaccessible terrains or during disaster relief operations, and they can establish wireless connections independently of fixed infrastructure, thus extending communication coverage crucial for expansive virtual worlds.

However, despite these benefits, UAVs face limitations such as energy constraints [135], which can prevent them from completing computation-intensive tasks crucial for maintaining the Metaverse's continuity.

In this chapter, we explore a network comprising various UAVs, mobile charging stations, and edge servers attached to BSs designed to run critical applications like traffic monitoring [144, 145]. These UAVs perform essential computations such as distributed matrix multiplication, central to many modern computing applications including machine learning and scientific computing [93, 146], which are vital for tasks like post-disaster relief assistance [147] and crowd detection in virtual

scenarios [148]. To mitigate battery limitations, the matrix multiplication can be offloaded to ground-based edge servers. This distributed computing framework faces a significant challenge: latency from the slowest processors, or "stragglers" [92]. Coded distributed computing (CDC) is introduced to address this, enabling UAVs to only wait for the fastest subset of workers before recovering the output, a method known as the recovery threshold.

Beyond employing the CDC to address stragglers, this network encounters three major challenges: weather, demand, and shortfall uncertainties. Weather uncertainties may compromise UAV stability if not properly managed, particularly if the UAVs are not equipped with sufficiently powerful engines and adequate battery capacity. Demand uncertainty arises as edge servers typically require advanced subscription fees, yet the actual computational demand, such as matrix size, varies—leading to potential over-and under-subscription issues. Lastly, shortfall uncertainty refers to the risk of delays or link failures when UAVs offload computations to edge servers, potentially resulting in fewer returned copies than the recovery threshold, thereby necessitating additional costs for correcting these shortfalls. This correction also includes a hovering cost, as UAVs must remain airborne during re-computation.

To overcome the three challenges mentioned above, we introduce the Stochastic Coded Offloading Scheme (SCOS). SCOS is a two-phase optimization scheme that adopts a CDC technique to reduce the total cost of the network:

- **Phase one (UAV type allocation):** The application owner will first allocate the appropriate UAV to each mobile charging station by considering the weather condition in each time slot. This weather uncertainty is modeled by a two-stage Stochastic Integer Programming (SIP) [101].

- **Phase two (task allocation):** There are a different number of time-frames/periods within the same time slot. For example, when the morning is the first time slot, each hour is treated as one period, and task allocation occurs in each period. Demand and shortfall are the two uncertainties in task allocation. Instead of performing local computation as the correction action to correct the shortfalls, the same decision options are provided to the UAVs until the z stage. Therefore, z-stage SIP is used to model the demand and shortfall uncertainty in various stages.

Extensive simulations are performed to evaluate the effectiveness of SCOS. The results show that SCOS can minimize the total cost and the UAVs' energy consumption, especially compared with the traditional deterministic baseline scheme.

The contributions of this chapter are summarized as follows.

- The combination/integration yields fully dynamic on-demand computing solutions for emerging applications such as road traffic prediction for autonomous vehicles in which traditional approaches are ineffective due to their rigid and fixed deployment.
- Our SCOS is able to provide strategic scenario-based decision that adapts well with the weather condition in which the current solutions for UAVs are limited.
- The proposed SCOS can minimize the UAVs' overall costs by optimizing the task allocation. At the same time, it can also minimize all the UAVs' energy consumption. The optimal solution is achieved by considering both the demand and shortfall uncertainty.
- From the performance evaluation, we use the real data to validate that SCOS is the optimal scheme when the performance is compared with the Expected-Value Formulation (EVF) and random scheme.

The remainder of the chapter is organized as follows: In Section 3.2, we present the system model. In Sections 3.3 and 3.4 we formulate the problem. We discuss and analyze the simulation result in Section 3.5. Section 3.6 concludes the chapter.

3.2 System Model

The overall system model is shown in Fig. 3.1. We model the phase one (UAV type allocation) and phase two (task allocation) to complete applications defined by an application owner, e.g., road traffic monitoring [144] while considering various uncertainties. Since each edge server has limited computation capability, by deploying many edge servers at the BS, we can use constraints (3.48) and (3.49) to ensure that there will be enough computation resources to support the computation required by each UAV. The following sets are used to denote time slots, UAV types, mobile charging stations, and BSs.

- $\mathcal{T} = \{1, \ldots, \bar{t}, \ldots, \bar{T}\}$ represents the different time slot.
- $\bar{\mathcal{P}}^{\bar{t}} = \{1, \ldots, \bar{p}^{\bar{t}}, \ldots, \bar{P}^{\bar{t}}\}$ represents the period in time slot $\bar{t}$.
- The available UAVs are clustered into $|\mathcal{X}|$ types denoted by set $\mathcal{X}$, where $\mathcal{X} = \{1, \ldots, x, \ldots, X\}$. Specifically, the type refers to the battery capacity of the UAV in ascending order. For example, X is the largest type UAV that has the most battery and therefore leads to a longer flight time. The UAVs are owned by service provider $\bar{A}_1$. We use $x^{\bar{t}}$ to denote when type x UAV is used in time slot $\bar{t}$.

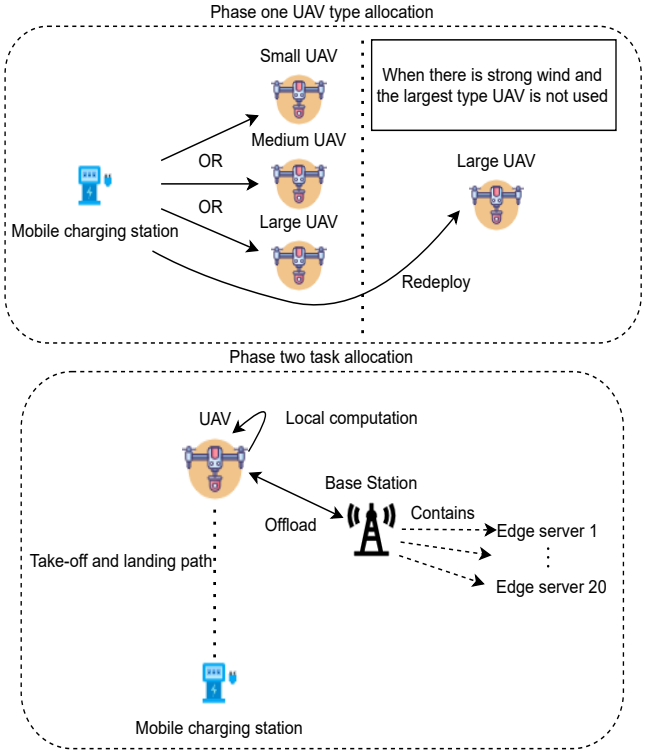


Figure 3.1: An illustrative example of the network with $\mathcal{X} = \{1 : small, 2 : medium, 3 : large\}$, 1 mobile charging station $Y = 1$, 20 edge servers $q_1 = 20$ attached to 1 BS $F = 1$.

- $\mathcal{Y} = \{1, \ldots, y, \ldots, Y\}$ represents the UAV mobile charging stations, owned by service provider $\bar{A}_1$. All the mobile charging stations are deployed at pre-specified locations defined by application owner $\bar{A}_3$.
- Each of BS f is attached with q_f number of edge servers. $\mathcal{F} = \{1, \ldots, f, \ldots, F\}$ represents BSs with the height of H_f. Edge servers are owned by service provider $\bar{A}_2$.

In phase one, the application owner first considers the weather uncertainty to pre-allocate the UAV types to each mobile charging station, also known as a UAV depot. Once the phase one optimization is done, all the UAVs will take off from their respective mobile charging stations which are located at (a_y, b_y). (a_y, b_y) are the x-y coordinates of mobile charging station y. At time slot $\bar{t}$, type x UAV will take off vertically to the height of $H_{y,x^{\bar{t}}}$ and hover in the sky for purposes such as traffic monitoring. $(a_y, b_y, H_{y,x^{\bar{t}}})$ and (a_f, b_f, H_f) are the three-dimensional coordinates of the type x UAV associated with mobile charging station y and edge servers in BS f, respectively, where $H_{y,x^{\bar{t}}} > H_f$ to maintain a line-of-sight (LoS) communication link between type x UAV and edge servers in BS f. For simplicity, we assume that UAV maintain a LoS link with the edge servers in the RSUs. Due to the hovering capability, we consider only the rotary-wing

UAVs [149].

After the UAVs reach their respective heights, they can receive and process computation tasks. In this chapter, we consider the task that the type x UAV computes is the matrix-matrix product $\mathbf{AB}$ involving the two matrices $\mathbf{A}$ and $\mathbf{B}$. However, the UAV has limited computing and storage capability [150]. Therefore, the UAV can choose to offload a portion or the whole matrix multiplication to the edge servers [150]. In phase two, it derives the offloading decision to minimize the overall operation cost by considering the demand and shortfall uncertainties. Note that the key notations used in this chapter are listed in Table 3.1. In the following, we discuss the coded distributed computing model and UAV energy consumption model.

Table 3.1: List of key notation

Symbol	Definition
$\mathcal{T}$	Set of time slots while $\bar{t} \in \mathcal{T}$ denotes the time slot index
$\bar{\mathcal{P}}^{\bar{t}}$	Set of periods in $\bar{t}$ while $\bar{p}^{\bar{t}} \in \bar{\mathcal{P}}^{\bar{t}}$ denotes the period index
$\mathcal{X}$	Set of UAV types while $x \in \mathcal{X}$ denotes the UAV type index
$\mathcal{Y}$	Set of mobile charging stations while $y \in \mathcal{Y}$ denotes the mobile charging station index
$\mathcal{F}$	Set of BSs while $f \in \mathcal{F}$ denotes the BS index
k	Recovery threshold
z	The number of stages in multi-stage SIP
$\gamma^{\bar{t}}$	Set of weather condition scenarios in $\bar{t}$ while $\mu_i^{\bar{t}} \in \gamma^{\bar{t}}$ denotes the weather condition scenario index
$\Theta^{\bar{t}}$	Set of demand scenarios in $\bar{t}$ while $\lambda_i^{\bar{t}} \in \Theta^{\bar{t}}$ denotes the demand scenario index
$\Omega^{(\bar{z},\bar{t})}$	Set of shortfall scenarios in stage $\bar{z}-1$, where $2 < \bar{z} \leq z$, and $\bar{t}$ while $\omega_i^{(\bar{z},t)} \in \Omega^{(\bar{z},\bar{t})}$ denotes the shortfall scenario index
$T_y^{x^{\bar{t}}}$	Binary variable at time slot $\bar{t}$ for mobile charging station y indicates whether type x UAV is used.
$T_y^{(\bar{t},X)}(\mu_i^{\bar{t}})$	Binary variable at time slot $\bar{t}$ for mobile charging station y indicates whether a correction on-demand type-X UAV is used in scenario $\mu_i^{\bar{t}}$, and X represents the largest UAV type.
$M_f^{(s)}$	Binary variable to indicate whether the edge servers in BS f will be used or not
$M_{y,x^{\bar{t}}}^{(L,2)}(\lambda_i^{\bar{t}})$	Decision variable represents the number of copies computed locally by the type x UAV that is associated with mobile charging station y in stage 2, time slot $\bar{t}$ and scenario $\lambda_i^{\bar{t}}$
$M_{y,x^{\bar{t}},f}^{(O,2)}(\lambda_i^{\bar{t}})$	Decision variable represents the number of copies offloaded to the edge servers in BS f by the type x UAV that is associated with mobile charging station y in stage 2, time slot $\bar{t}$ and scenario $\lambda_i^{\bar{t}}$
$\vdots$	$\vdots$
$M_{y,x^{\bar{t}}}^{(L,\hat{z})}(\lambda_i^{\bar{t}},\ldots,\omega_i^{(\bar{z},\bar{t})})$	Decision variable represents the number of copies computed locally by the type x UAV that is associated with mobile charging station y in time slot $\bar{t}$, scenario $\lambda_i^{\bar{t}}$, …, scenario $\omega_i^{(\bar{z},\bar{t})}$ and stage $\hat{z}$, where $1 < \hat{z} \leq z$
$M_{y,x^{\bar{t}},f}^{(O,\hat{z})}(\lambda_i^{\bar{t}},\ldots,\omega_i^{(\bar{z},\bar{t})})$	Decision variable represents the number of copies offloaded to the edge servers in BS f by the type x UAV that is associated with mobile charging station y in time slot $\bar{t}$, scenario $\lambda_i^{\bar{t}}$, …, scenario $\omega_i^{(\bar{z},\bar{t})}$ and stage $\hat{z}$

3.2.1 Coded Distributed Computing

Massive parallelization can speed up matrix multiplication. However, it has a computational bottleneck due to stragglers or faults. Coded computation is introduced to make matrix multiplications resilient to faults and delays, i.e., PolyDot codes [93]. In PolyDot codes, the system model typically consists of the followings [93]:

- Master node receives computation inputs, encodes and distributes them to the worker nodes.
- Worker nodes perform pre-determined computations on their respective inputs in parallel.
- Fusion node receives outputs from successful worker nodes and decodes them to recover the final output.

We consider that the type x UAV is our proposed network's master and fusion node. Each edge server in BS f is the worker and has the computation capability of $\tau_{f,q}$, where $\tau_{f,q}$ denotes the CPU computation capability of the edge server q in BS f (in CPU cycles per second).

The definitions of copy, successful workers, recovery threshold, shortfall, and demand are given as follows.

Definition 1. [Copy] a fraction of matrices **A** and **B** [93].

Definition 2. [Successful workers] Workers that finish their computation task and the task is received successfully by the UAV.

Definition 3. [Recovery threshold] The recovery threshold is the worst-case minimum number of successful workers required by the UAV to complete the computation [93].

Definition 4. [Shortfall] There exists a shortfall when the total returned copies from the local computation and from the workers are less than the recovery threshold.

Definition 5. [Demand] The demand is size of the matrix input N_y. It is always different as the input of the matrix multiplication is not always the same.

Following [93], two $N_y \times N_y$ square matrices $\mathbf{A}_y$ and $\mathbf{B}_y$ are considered. Note that our model can be applied to other matrices, e.g., non square matrices. Each of matrices $\mathbf{A}_y$ and $\mathbf{B}_y$ is sliced both horizontally and vertically. For example, $\mathbf{A}_y$ is sliced into $\frac{N_y}{t} \times \frac{N_y}{s}$ matrices and $\mathbf{B}_y$ is sliced into $\frac{N_y}{s} \times \frac{N_y}{t}$. We choose s and t such that they satisfy $st = m$ [93] and a copy is the m-th fractions of matrices **A** and **B**. Each edge server has a storage constraint that limits the edge server to store only m fractions of matrices **A** and **B** [93]. The recovery threshold k is defined [93] as:

$$k = t^2(2s - 1). \tag{3.1}$$

The processing by the workers may take a longer time when it is currently occupied with some other tasks. Therefore, the processing in the offloaded tasks is perceived to have failed if the duration exceeds the threshold time limit [151]. To recover the computed task, the sum of returned offloaded copies and locally computed copies must be greater than or equal to recovery threshold k.

The decision scenario of phase one and phase two are explained using recovery threshold k. In phase one, $|\mathcal{X}|$ scenarios may occur. Mobile charging station y chooses the UAV type to be used. In phase two, three scenarios may occur.

- The UAV can compute all copies locally, $M^{(L)}_{y,x^{\bar{t}}} \geq k$, where $M^{(L)}_{y,x^{\bar{t}}}$ indicates the number of copies that type x UAV from mobile charging station y computes locally at time slot $\bar{t}$ and k is defined in (3.1).

- The UAV can offload all copies to BS f, $M^{(O)}_{y,x^{\bar{t}},f} \geq k$, where $M^{(O)}_{y,x^{\bar{t}},f}$ denotes the number of copies that are offloaded to the edge servers in BS f at time slot t by the type x UAV from mobile charging station y.

- The UAV can compute some copies locally and offload some copies to the edge servers in BS f, $M^{(L)}_{y,x^{\bar{t}}} + M^{(O)}_{y,x^{\bar{t}},f} \geq k$

The final output can be decoded from all the return copies $M^{(L)}_{y,x^{\bar{t}}} + M^{(O)}_{y,x^{\bar{t}},f}$.

Similar to [93], the type x UAV associated with mobile charging station y uses $d_{enc} = N_y^2(M^{(L)}_{y,x^{\bar{t}}} + M^{(O)}_{y,x^{\bar{t}},f})$ symbols for encoding of matrices and $d_{dec} = N_y^2 t^2(2s-1)\log^2 t^2(2s-1)$ to decode the returned matrices. Each copy contains m-th fractions of matrices $\mathbf{A}$ and $\mathbf{B}$. UAV will transmit $d^{to}_{comm} = \frac{N_y^2}{m}$ symbols to each of the edge servers. Each copy requires $d_{cmp} = \frac{N_y^3}{mt}$ symbols for computation. After computation is completed, the edge server will send $d^{fr}_{comm} = \frac{N_y^2}{t^2}$ symbols back to the UAV.

3.2.2 UAV Hovering Energy

The propulsion energy consumption is needed to provide the UAV with sufficient thrust to support its movement. Note that we drop the time notation for ease of presentation. The propulsion power of a rotary-wing UAV with speed V can be modeled as follows [149]:

$$P_x(V) = P_{x,0}\left(1 + \frac{3V^2}{U_{tip}^2}\right) + P_{x,1}\left(\sqrt{1 + \frac{V^4}{4v_0^4}} - \frac{V^2}{2v_0^2}\right)^{\frac{1}{2}} + \frac{1}{2}d_0\rho^{\bar{t}}\mathbb{A}V^3, \tag{3.2}$$

where

$$P_{x,0} = \frac{\delta}{8}\rho \mathbf{s}\mathbb{A}\triangle_x^3 R^3, \tag{3.3}$$

$$P_{x,1} = (1+r)\frac{W_x^{3/2}}{\sqrt{2\mathbb{A}\rho}}. \tag{3.4}$$

$P_{x,0}$ and $P_{x,1}$ are two constants related to UAV's weight, rotor radius, air density, etc. U_{tip} denotes the tip speed of the rotor blade, v_0 is known as the mean rotor induced velocity in hover, d_0 and s are the fuselage drag ratio and rotor solidity, respectively. ρ and $\mathbb{A}$ are the air density and rotor disc area, respectively. r is the incremental correction factor to induced power. W_x is the type x UAV weight, δ is the profile drag coefficient, and $\triangle_x$ denotes blade angular velocity of the type x UAV. By substituting $V = 0$ into (3.2) [149], we obtain the power consumption for hovering status as follows:

$$P_{x,h} = P_{x,0} + P_{x,1}. \tag{3.5}$$

3.2.3 Local Computing Model

The local computation execution time of the type x UAV is expressed as [152]:

$$t_{y,x}^{local}(N_y) = \frac{C_x d(R)}{\tau_x}, \tag{3.6}$$

where C_x is the number CPU cycles needed to process a bit, τ_x denotes the total CPU computing capability of the type x UAV, and $d(\cdot)$ is a function to translate the number of symbols to the number of bits for computation, i.e., if the 16 Quadrature Amplitude Modulation (QAM) is used, each symbol carries 4 bits [153] and R is the number of symbols. When one copy is processed locally, $R = d_{cmp} = \frac{N_y^3}{mt}$. The type x UAV takes $t_{y,x}^{enc}$ seconds to encode one copy of the matrices N_y^2 symbols ($R = N_y^2$), and it is expressed as follows:

$$t_{y,x}^{enc}(N_y) = \frac{C_x d(N_y^2)}{\tau_x}. \tag{3.7}$$

After the type x UAV obtains at least k copies, it will take $t_{y,x}^{dec}$ seconds to decode and $R = d_{dec} = N_y^2 t^2(2s-1)\log^2 t^2(2s-1)$. $t_{y,x}^{dec}$ is defined as follows:

$$t_{y,x}^{dec}(N_y) = \frac{C_x d(N_y^2 t^2(2s-1)\log^2 t^2(2s-1))}{\tau_x}. \tag{3.8}$$

3.2.4 UAV Communication Model

We assume that each UAV is allocated with an orthogonal spectrum resource block to avoid the co-interference among the UAVs [154]. The transmission rate from the type x UAV which is associated with mobile charging station y to the edge servers in BS f can be represented as [155]:

$$r_{y,x,f} = B_x \log_2(1 + P_x^C h_{y,x,f}/N_o), \tag{3.9}$$

where the wireless transmission power of the type x UAV at time slot $\bar{t}$ is expressed as P_x^C and B_x is the bandwidth. $h_{y,x,f}$ is the channel gains, and N_0 is the variance of complex white Gaussian noise. The UAV to edge server communication is most likely to be dominated by LoS links. Therefore, the air-to-ground channel power gain from the type x UAV to the edge servers in BS f can be modeled as follows [156]:

$$h_{y,x,f} = \frac{\beta_0}{\mathbb{D}_{y,x,f}^2}, \tag{3.10}$$

where

$$\mathbb{D}_{y,x,f}^2 = (a_y - a_f)^2 + (b_y - b_f)^2 + (H_{y,x} - H_f)^2. \tag{3.11}$$

$\mathbb{D}_{y,x,f}$ denotes the distance between the type x UAV that is associated with mobile charging station y and the edge servers in BS f, and β_0 represents the reference channel gain at distance $d_0 = 1$m in an urban area [156]. We assume that for all the edge servers in the same BS f, they will have the same $\mathbb{D}_{y,x,f}^2$. The transmission time to offload one copy of matrix from the type x UAV to a edge server in BS f can be given as follows:

$$t_{y,x,f}^{to}(N_y) = \frac{d(\frac{N_y^2}{m})}{r_{y,x,f}}. \tag{3.12}$$

The energy $e_{y,x}$ required by the type x UAV to receive data from the edge server in BS f is defined as follows [157]:

$$e_{y,x}(N_y) = P_x^{re} \frac{d(\frac{N_y^2}{t^2})}{r_{f,y,x}}, \tag{3.13}$$

where P_x^{re} is the receiving power of type x UAV. $r_{f,y,x}$ is the transmission rate from edge servers in BS f to type x UAV which is associated with mobile charging station y. It is define similar to (3.9).

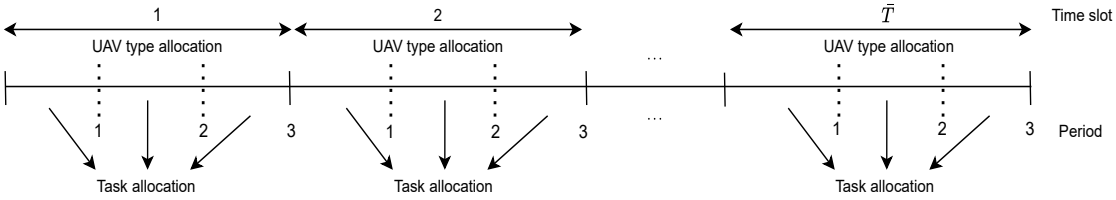


Figure 3.2: The decision process of the system across all the time slots $\bar{T}$.

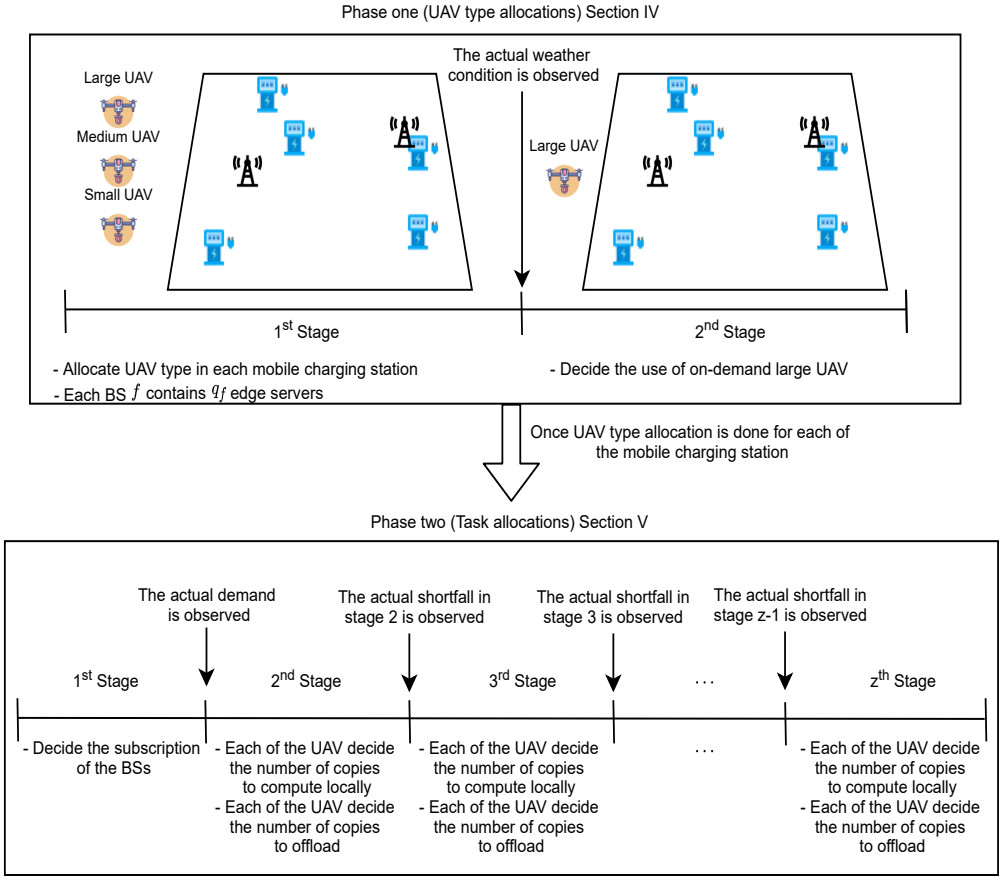


Figure 3.3: Decision making process of the system in one time slot with the using of three different types of UAV, $\mathcal{X} = \{1 : small, 2 : medium, 3 : large\}$.

3.2.5 Problem Formulation

As an illustration, Fig. 3.2 depicts the decision process of the system across all the time slots. UAV type allocation is performed in each time slot $\bar{t}$. Throughout $\bar{t}$, the mobile charging stations will use the same UAV type to perform the task allocation in each period. Fig. 3.3 shows a detailed diagram that zooms into one-period in one-time slot, and it is explained in details in both Sections 3.3 and 3.4. In Section 3.3, the application owner $\bar{A}_3$ pays a reservation cost to make an advance booking for a different time slot for the use of the UAVs. The application owner can observe the weather condition via weather forecast as it may affect the status of the UAV. If the wind is too strong and the UAV used is not large type, the UAV may crash as it has insufficient energy to hover against the wind [158]. For example, a strong wind has high kinetic energy, kinetic energy leads to a higher density of the air, and it increases the UAV hovering power consumption. Low wind speed is referred to as wind speed that is less than $11m/s$ and turbulence level < 5 [159]. As a result, $\bar{A}_3$ has to request an on-demand X type UAV to perform the job. In order for SCOS to model the weather uncertainty, we formulate the two-stage SIP to optimize the UAV type allocation.

To achieve cost minimization, phase two (task allocation) in Section 3.4 has to consider two sources of uncertainty, i.e., the demand uncertainty and shortfall uncertainty. Demand uncertainty refers to the task required by the applications, such as traffic monitoring can be of different sizes, i.e., the task's size depends on the image resolution. Shortfall uncertainty refers to if the UAV offloads the copies to the edge servers, the computed copies may not return, or the number of copies returned is less than the recovery threshold due to delays and link failure. Therefore, we use multi-stage SIP to model the demand uncertainty to optimize the number of copies to compute locally $M^{(L)}_{y,x^{\bar{t}}}$ and offload $M^{(O)}_{y,x^{\bar{t}},f}$. For example, when the recovery threshold is $k = 4$ and the UAV decide to offload two copies of the task for the edge servers in BS f to compute, i.e., $M^{(O)}_{y,x^{\bar{t}},f} = 2$. Therefore, the UAV has to compute at least two more copies locally to match the recovery threshold $M^{(L)}_{y,x^{\bar{t}}} = 2$. In time slot $\bar{t}$, type x UAV will hover in the sky for a threshold time limit $t^{thresh}_{y,x^{\bar{t}}}$ to wait for the offloaded copies to return. Without loss of generality, $t^{thresh}_{y,x^{\bar{t}}}$ is set as the worst-case scenario, i.e., the time required to compute all copies locally by the UAV, i.e., $M^{L}_{y,x^{\bar{t}}} = k$. However, there is a probability that the edge servers in BSs may fail, i.e., the computed task is not returned to the UAV before $t^{thresh}_{y,x^{\bar{t}}}$. As a result, the UAV cannot complete the full task if the total returned copies are less than 4. When the UAV fails to receive sufficient number of copies, there are shortfalls, and hence, the UAV has to re-compute the shortfalls locally or re-offload to the edge servers. Since the UAV has limited computation capabilities, it can choose to re-compute the shortfall locally or re-offload to the edge servers until z

stages, where z is the number of times of re-computations. In the meantime, the UAV has to continue hover in the sky when performing the re-computation. In order to model the shortfall uncertainty, we formulate z-stage SIP to optimize the numbers of copies to compute locally and to offload, and we can also optimize the number of stages required. Hence, this scheme will minimize the overall network cost, and the system model of this network is formulated as follows:

$$\min_{T_y^{\bar{x}^{\bar{t}}}, T_y^{x^{\bar{t}}}(\mu_i^{\bar{t}}), M_f^{(s)}, \ldots, M_{y,x^{\bar{t}},f}^{(O,\hat{z})}(\lambda_i^{\bar{t}}, \ldots, \omega_i^{(\bar{z},\bar{t})})} :$$

$$\sum_{\bar{t}\in\bar{\mathcal{T}}} O_{allocation}^{UAV}(\bar{t}) + \sum_{\bar{t}\in\bar{\mathcal{T}}} \sum_{\bar{p}^{\bar{t}}\in\bar{\mathcal{P}}^{\bar{t}}} O_{allocation}^{Task}(\bar{p}^{\bar{t}}), \tag{3.14}$$

subject to: (3.19)-(3.22), (3.39)-(3.51)
where $O_{allocation}^{UAV}(\bar{t})$ is the UAV type allocation cost in time slot $\bar{t}$ and it is defined in (3.17) in Section 3.3. $O_{allocation}^{Task}(\bar{p}^{\bar{t}})$ is the task allocation cost within period $\bar{p}$ and period $\bar{p}$ is in time slot $\bar{t}$. The task allocation is defined in (3.35) in Section 3.4.2.

3.3 Phase one: UAV type allocation

This section introduces the SIP to optimize phase one (UAV type allocation) in SCOS by minimizing the total allocation. As described in Section 3.2, the application owner $\bar{A}_3$ needs to make a reservation in advance to secure certain types of UAVs, which are own by $\bar{A}_1$. However, the weather condition is unknown and may vary at a different time slot $\bar{t}$. If the wind is too strong, the UAV is required to use more energy to hover at a fixed location [158]. As a result, the UAV will crash with insufficient energy, and the application owner has to make an on-demand request with a X type UAV. Fig. 3.3 illustrates the decision-making process of the system with the use of three UAV types, 1, 2 and 3, which represents small, medium and large, respectively.

Hence, we formulate this scheme as the two-stage SIP model.

- **First stage:** The application owner makes a reservation on the types of UAVs to be used. The decision will be made based on the available cost information and the probability distribution of the weather condition.
- **Second stage:** After knowing the exact weather condition, the application owner decides the correction action, which is the on-demand request to use the largest type X UAV.

Let $\mu_i^{\bar{t}} = \{G_1^{\bar{t}}(\mu_i^{\bar{t}}), \ldots, G_y^{\bar{t}}(\mu_i^{\bar{t}})\}$ denote weather condition scenarios i of all mobile charging stations at time slot $\bar{t}$. The set of all weather scenarios is denoted by $\gamma^{\bar{t}}$, i.e.,

Table 3.2: Weather uncertainty

Symbol	Definition
$G_y^{\bar{t}}(\mu_i^{\bar{t}}) = 0$	At time slot $\bar{t}$, the wind is weak or there is no wind at mobile charging station y. Low wind speed is referring to wind speed that is less than 11 m/s and turbulence level < 5 [159]
$G_y^{\bar{t}}(\mu_i^{\bar{t}}) = 1$	At time slot $\bar{t}$, the wind is strong at mobile charging station y. High wind speed is referring to wind speed that is greater than 11 m/s and turbulence level > 5 [159]

$\mu_i^{\bar{t}} \in \gamma^{\bar{t}}$ [160]. $G_y^{\bar{t}}(\mu_i^{\bar{t}})$ represents a binary parameter of the weather condition at time slot $\bar{t}$. For tractability, we only consider that each mobile charging station experiences only two types of weather condition. As shown in Table 3.2, $G_y^{\bar{t}}(\mu_i^{\bar{t}}) = 1$ means that at time slot $\bar{t}$, the wind is strong in mobile charging station y and the UAV has crashed, and $G_y^{\bar{t}}(\mu_i^{\bar{t}}) = 0$ means otherwise. $\mathcal{P}(\mu^{\bar{t}})$ denotes the probability if scenario $\mu^{\bar{t}} \in \gamma^{\bar{t}}$ is realized. All of the scenarios can be obtained from historical records [105] or weather forecast. The cost function is proportional to the resources used. In total, there are $|\mathcal{X}|+2$ types of payments. Note that we drop the time notation.

- C_r^x is the reservation cost for the type x UAV. It is defined as follows:

$$C_r^x = \alpha_1 B_x, \tag{3.15}$$

  where B_x is the battery capacity of the type x UAV and α_1 is the cost coefficient.

- C_o^X is the on-demand cost for the type X UAV, which represents the largest UAV type. It is defined as follows:

$$C_o^X = \alpha_2 B_X, \tag{3.16}$$

  where α_2 is the cost coefficient with a similar role to α_1 and $\alpha_2 > \alpha_1$. B_X is the battery capacity of the type X UAV.

- C_p is the penalty cost. This penalty cost is the repair cost for the crashed UAV.

We formulate the UAV type allocation as a two-stage SIP model. There are $|\mathcal{T}|(|\mathcal{X}|+1)$ decision variables in this model.

- $T_y^{x^{\bar{t}}}$ is a binary variable at time slot $\bar{t}$ for mobile charging station y indicates whether type x UAV is used. When $T_y^{x^{\bar{t}}} = 1$, at time slot $\bar{t}$, mobile charging station y uses type x UAV and $T_y^{x^{\bar{t}}} = 0$ means otherwise.

- $T_y^{(\bar{t},X)}(\mu_i^{\bar{t}})$ is a binary variable at time slot $\bar{t}$ for mobile charging station y indicates whether a correction on-demand type X UAV is used in scenario $\mu_i^{\bar{t}}$, and X represents the largest UAV type. When $T_y^{(\bar{t},X)}(\mu_i^{\bar{t}}) = 1$, at time slot $\bar{t}$, mobile charging station y performs a correction action by using the largest type-X UAV in scenario $\mu_i^{\bar{t}}$ and $T_y^{(\bar{t},X)}(\mu_i^{\bar{t}}) = 0$ means otherwise.

The objective function given in (3.17) and (3.18) is to minimize the cost of the UAV type allocation. The expressions in (3.17) and (3.18) represent the first- and second-stage SIP, respectively. The SIP formulation can be expressed as follows:

$$\min_{T_y^{x^{\bar{t}}}, T_y^{(\bar{t},X)}(\mu_i^{\bar{t}})} :$$

$$\sum_{\bar{t}\in\mathcal{T}} O^{UAV}_{allocation}(\bar{t}) = \sum_{\bar{t}\in\mathcal{T}} \sum_{y\in\mathcal{Y}} \sum_{x\in\mathcal{X}} T_y^{x^{\bar{t}}} C_r^x + \mathbb{E}\Big[Q(T_y^{x^{\bar{t}}}(\mu_i^{\bar{t}}))\Big], \tag{3.17}$$

where

$$Q(T_y^{x^{\bar{t}}}(\mu_i^{\bar{t}})) = \sum_{\bar{t}\in\mathcal{T}} \sum_{\mu_i^{\bar{t}}\in\gamma^{\bar{t}}} \mathcal{P}(\mu_i^{\bar{t}}) \sum_{y\in\mathcal{Y}} T_y^{(\bar{t},X)}(\mu_i^{\bar{t}})(C_o^X + C_p), \tag{3.18}$$

subject to:

$$\sum_{x\in\mathcal{X}} T_y^{x^{\bar{t}}} = 1, \qquad \forall \bar{t} \in \mathcal{T}, \forall y \in \mathcal{Y}, \tag{3.19}$$

$$\sum_{x\in\mathcal{X}\setminus\{X\}} T_y^{x^{\bar{t}}}(1 - G_y(\mu_i^{\bar{t}})) + T_y^{X^{\bar{t}}} + T_y^{(\bar{t},X)}(\mu_i^{\bar{t}}) = 1, \quad \forall \bar{t} \in \mathcal{T}, \forall \mu_i^{\bar{t}} \in \gamma^{\bar{t}}, \forall y \in \mathcal{Y}, \tag{3.20}$$

$$T_y^{x^{\bar{t}}} \in \{0,1\}, \qquad \forall \bar{t} \in \mathcal{T}, \forall y \in \mathcal{Y}, \forall x \in \mathcal{X}, \tag{3.21}$$

$$T_y^{(\bar{t},X)}(\mu_i^{\bar{t}}) \in \{0,1\}, \qquad \forall \bar{t} \in \mathcal{T}, \forall y \in \mathcal{Y}, \forall \mu_i^{\bar{t}} \in \gamma^{\bar{t}}. \tag{3.22}$$

The constraint in (3.19) ensures that the application owner makes a reservation on the types of UAV. On the other hand, (3.20) ensures that the UAV crashes because of strong wind if the application owner previously reserves a UAV that is not largest type X. Then, the application owner has to perform a correction action by using the largest type X on-demand UAV. (3.21) and (3.22) are boundary constraints for the decision variables.

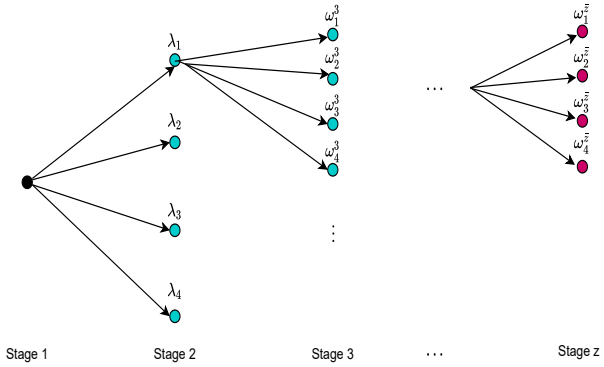


Figure 3.4: A scenario tree structure for z-stage SIP in task allocation.

3.4 Phase two: task allocation

Once the types of the UAVs are optimized from phase one in SCOS, we introduce the Deterministic Integer Programming (DIP) and SIP to optimize phase two (the number of copies to compute locally and to offload) by minimizing the UAV network cost. Note that for simplicity, we drop notation $\bar{p}^{\bar{t}}$ from phase two task allocation.

3.4.1 Deterministic Integer Programming System Model

In an ideal case, when the actual demand, which is the actual matrix size and the number of shortfalls, are precisely known ex-ante, the UAVs can choose the exact number of copies to compute locally or offload. Therefore, the correction for the shortfall is not needed, and the correction cost is zero. Similar to [161], the cost function is proportional to the UAVs offloaded data and to their demand for consuming computation resources. Choosing different sizes of UAVs will affect the payment value. In total, five types of payments are considered in DIP.

- $\bar{C}_f$ is the subscription cost for the edge servers in BS f.
- $\bar{C}_{y,x}$ denotes the UAV local computation cost and encoding cost for computing of one copy, i.e.,
$$\bar{C}_{y,x}(\mathcal{D}) = \alpha_3(t^{local}_{y,x}(\mathcal{D}) + t^{enc}_{y,x}(\mathcal{D})), \tag{3.23}$$
where α_3 is the cost coefficient associated to the energy consumption and $\mathcal{D}$ is the actual demand.
- $C_{y,x^{\bar{t}},f}$ denotes the offloading cost and it consists of three parts. The first part is related to the transmission $t^{to}_{y,x^{\bar{t}},f}$ and encoding delay $t^{enc}_{y,x}$. The second part is the type x UAV energy consumption cost and the last part C_f is the service cost for

edge servers in BS f. It is modeled as follows:

$$C_{y,x^{\bar{t}},f}(\mathcal{D}) = \alpha_3(t^{to}_{y,x^{\bar{t}},f}(\mathcal{D}) + t^{enc}_{y,x}(\mathcal{D})) + \alpha_4 e_{y,x^{\bar{t}}}(\mathcal{D}) + C_f, \tag{3.24}$$

where α_4 is the cost coefficient with a similar role to α_3.

- $\underline{C}^{thresh}_{y,x^{\bar{t}}}$ denotes the hovering cost for $t^{thresh}_{y,x^{\bar{t}}}$ seconds. They are defined as follows:

$$\underline{C}^{thresh}_{y,x^{\bar{t}}} = t^{thresh}_{y,x^{\bar{t}}} k \alpha_5 P_{x^{\bar{t}},h}, \tag{3.25}$$

where α_5 is the cost coefficient with a similar role to α_3.

- $\hat{C}_{y,x}$ denotes the type x UAV decoding cost for the returned matrices as follows:

$$\hat{C}_{y,x}(\mathcal{D}) = \alpha_3 t^{dec}_{y,x}(\mathcal{D}). \tag{3.26}$$

A DIP can be formulated and minimize the total cost of the UAVs as follows:

$$\min_{M^{(s)}_f,\ldots,M^{(O)}_{y,x^{\bar{t}},f}} :$$

$$\sum_{\bar{t}\in\mathcal{T}} O^{Task}_{allocation}(\bar{t}) = \sum_{\bar{t}\in\mathcal{T}}\sum_{f\in\mathcal{F}} M^{(s)}_f \bar{C}_f + \sum_{\bar{t}\in\mathcal{T}}\sum_{y\in\mathcal{Y}}\sum_{f\in\mathcal{F}}\Big(M^{(L)}_{y,x^{\bar{t}}}\bar{C}_{y,x}(\mathcal{D}) + M^{(TH)}_{y,x^{\bar{t}},f}\underline{C}^{thresh}_{y,x} + M^{(O)}_{y,x^{\bar{t}},f} C_{y,x,f}(\mathcal{D}) + \hat{C}_{y,x}(\mathcal{D})\Big), \tag{3.27}$$

subject to:

$$\sum_{y\in\mathcal{Y}} M^{(O)}_{y,x^{\bar{t}},f} \le \sigma M^{(s)}_f, \qquad \forall \bar{t}\in\mathcal{T}, \forall f\in\mathcal{F}, \tag{3.28}$$

$$\sum_{y\in\mathcal{Y}} M^{(O)}_{y,x^{\bar{t}},f} \le q_f, \qquad \forall \bar{t}\in\mathcal{T}, \forall f\in\mathcal{F}, \tag{3.29}$$

$$M^{(O)}_{y,x^{\bar{t}},f} \le \sigma M^{(TH)}_{y,x^{\bar{t}},f}, \qquad \forall y\in\mathcal{Y}, \forall \bar{t}\in\mathcal{T}, \forall f\in\mathcal{F}, \tag{3.30}$$

$$\sum_{f\in\mathcal{F}} M^{(O)}_{y,x^{\bar{t}},f} \ge S^{\bar{t}}_y - M^{(L)}_{y,x^{\bar{t}}}, \qquad \forall \bar{t}\in\mathcal{T}, \forall y\in\mathcal{Y}, \tag{3.31}$$

$$M^{(L)}_{y,x^{\bar{t}}} + \sum_{f\in\mathcal{F}} M^{(O)}_{y,x^{\bar{t}},f} - (S^{\bar{t}}_y - M^{(L)}_{y,x^{\bar{t}}}) \ge k, \qquad \forall y\in\mathcal{Y}, \forall \bar{t}\in\mathcal{T}, \forall f\in\mathcal{F}, \tag{3.32}$$

$$M_f^{(s)}, M_{y,x^{\bar{t}},f}^{(TH)} \in \{0,1\}, \qquad \forall y \in \mathcal{Y}, \forall f \in \mathcal{F}, \tag{3.33}$$

$$M_{y,x^{\bar{t}}}^{(L)}, M_{y,x^{\bar{t}},f}^{(O)} \in \{0,1,\ldots\}, \qquad \forall \bar{t} \in \mathcal{T}, \forall y \in \mathcal{Y}, \forall f \in \mathcal{F}. \tag{3.34}$$

$M_f^{(s)}$ is a binary variable to indicate whether the edge servers in BS f will be used or not. $M_{y,x^{\bar{t}},f}^{(TH)}$ is a binary variable to indicate in time slot $\bar{t}$ whether the type x UAV which is associated with mobile charging station y will choose to offload or not. When $M_{y,x^{\bar{t}},f}^{(TH)} = 1$, in time slot $\bar{t}$ the UAV associated with mobile charging station y choose to offload some of the copies to BS f and $M_{y,x^{\bar{t}},f}^{(TH)} = 0$ means otherwise. The objective function in (3.27) is to minimize UAVs' total cost involving the UAVs' local computation cost and the UAVs' offloading cost. The constraint in (3.28) ensures that the subscription cost of the edge servers in the BS will be paid if they are used in any of the stages, where σ is a sufficiently large number. (3.29) ensures that the total number of copies offloaded to the edge servers must not exceed the total number of edge servers in BSs. (3.30) ensures that the threshold cost will be paid if the UAV perform offloading action. (3.31) ensures that the shortfalls should only exist if the number of copies offloaded is more than or equal to the shortfalls. (3.32) ensures that the number of copies computed locally and offloaded have to be at least equal to or larger than recovery threshold k. (3.33) indicates $M_f^{(s)}$ and $M_{y,x^{\bar{t}},f}^{(TH)}$ are binary variables. (3.34) indicates that $M_{y,x^{\bar{t}}}^{(L)}$ and $M_{y,x^{\bar{t}},f}^{(O)}$ are positive decision variables.

3.4.2 Stochastic Integer Programming System Model

This section introduces the SIP to minimize the total cost of the network by optimizing the number of copies to compute locally and to offload to the edge servers in BSs. The first stage consists of all decisions that have to be selected before the demand and shortfall are realized and observed. In the second stage and onwards, decisions are allowed to adapt to this information. In each stage, decisions are limited by constraints that may depend on previous decisions and observations.

As described in Section 3.2, there is a subscription cost when the service provider $\bar{A}_3$ wants to use the edge servers in BSs for computation. Then, without knowing the demand, the type x UAV can decides the number of copies to compute locally $M_{y,x^{\bar{t}}}^{(L)}$ and the number of copies to offload $M_{y,x^{\bar{t}},f}^{(O)}$.

The computation process in the edge servers are not very reliable, as the edge servers might be processing some other task or congested. As a result, the computation time is much longer than the threshold limit $t_{y,x^{\bar{t}}}^{thresh}$. Therefore, if a copy is offloaded, there is a probability that the computation might fail, and it will require the type x UAV to re-offload again or compute it locally.

Hence, we formulate this framework as a z-stage SIP model.

- **First stage:** The application owner $\bar{A}_3$ decides to use the edge servers in BS f or not. The decision will be made based on the available cost information, the probability distribution of the demand, and the shortfall.
- **Second stage:** After knowing the exact demand, the application owner $\bar{A}_3$ decides the number of copies that are computed locally and the number of copies to be offloaded to the edge servers in BS f.
- **Third stage:** After knowing the exact shortfall in the previous stage, the $\bar{A}_3$ performs a correction action to re-decide the number of copies that is computed locally and the number of copies to be offloaded to the edge servers in BS f.

  ⋮

- z **stage:** After knowing the exact shortfall in the z-1 stage, $\bar{A}_3$ performs a correction action to re-decide the number of copies that is computed locally and the number of copies to be offloaded to the edge servers in BS f. To promote the UAV to complete the task, a huge penalty will occur if there is still a shortfall in stage z.

Let $\lambda_i^{\bar{t}} = \{D_1^{\bar{t}}, \ldots, D_y^{\bar{t}}\}$ denote the UAV demand scenario i across all mobile charging station y in time slot $\bar{t}$ and the set of demand scenarios is denoted by $\Theta^{\bar{t}}$, i.e., $\lambda_i^{\bar{t}} \in \Theta^{\bar{t}}$ [160]. $D_y^{\bar{t}}$ contains a discrete value from a finite set $\mathcal{W} = \{1, \ldots, W\}$, it represents the size of the task in UAV that is associated with mobile charging station y. Specifically, $D_y^{\bar{t}} = 1000$ means that in time slot $\bar{t}$ the matrix that UAV receives is in the size of 1000×1000. Let $\omega_i^{(\bar{z},\bar{t})} = \{\mathbb{F}_1^{(\bar{z},\bar{t})}, \ldots, \mathbb{F}_y^{(\bar{z},\bar{t})}\}$ denote the i-th shortfall scenario of the UAV in time slot $\bar{t}$ that is associated with its individual mobile charging station in stage $\bar{z}-1$, where $2 < \bar{z} \leq z$. The set of shortfall scenarios is denoted by $\Omega^{(\bar{z},\bar{t})}$, i.e., $\omega_i^{(\bar{z},\bar{t})} \in \Omega^{(\bar{z},\bar{t})}$. $\mathbb{F}_y^{(\bar{z},\bar{t})}$ represents a binary parameter of the shortfall in time slot $\bar{t}$ from the type x associated with mobile charging station y in stage $\bar{z}-1$. For example, $\mathbb{F}_y^{(\bar{z},\bar{t})} = 1$ means that, in time slot $\bar{t}$ from the copies that the UAV has offloaded, at least $A_y^{(\bar{z},\bar{t})}$ copy did not return. As a result, the total number of copies that the UAV currently has is less than k, and $\mathbb{F}_y^{(\bar{z},\bar{t})} = 0$ means otherwise. In stage z, $\mathbb{F}_y^{(\bar{z},\bar{t})} = 0$ when $\mathbb{F}_y^{(\bar{z}-1,\bar{t})} = 0$. When there is no shortfall in the previous stage then, there will not be any shortfall in the next stage. Fig. 3.4 illustrates the stages with four scenarios at each stage. All of the scenarios can be obtained from the historical records.

The cost function used in SIP is similar to DIP with an additional penalty cost $\tilde{C}$. $\tilde{C}$ occurs when the UAV still has to perform a corrective action. In total, six types of payments are considered in z-stage SIP.

We formulate the task allocation as the z-stage SIP model. There are $|\bar{\mathcal{T}}|(z(f+2)-1)$ decision variables in this model.

- $M_f^{(s)}$ is a binary variable to indicate whether the edge servers in BS f will be used or not. When $M_f^{(s)} = 1$, edge servers in BS f will be used and $M_f^{(s)} = 0$ means otherwise.

- $M_{y,x^{\bar{t}},f}^{(O,2)}(\lambda_i^{\bar{t}})$ indicates in time slot $\bar{t}$ the number of copies to be offloaded to the edge servers in BS f by type x UAV which is associated with mobile charging station y in stage 2.

- $M_{y,x^{\bar{t}}}^{(L,2)}(\lambda_i^{\bar{t}})$ indicates in time slot $\bar{t}$ the number of copies computed locally by the type x UAV which is associated with mobile charging station y in stage 2.

- $M_{y,x^{\bar{t}},f}^{(TH,3)}(\lambda_i^{\bar{t}})$ is a binary variable to indicate in time slot $\bar{t}$ whether the type x UAV which is associated with mobile charging station y choose to offload or not in stage 3. When $M_{y,x^{\bar{t}},f}^{(TH,3)}(\lambda_i^{\bar{t}}) = 1$, the type x UAV associated with mobile charging station y chooses to offload some of the copies to the edge servers in BS f and $M_{y,x^{\bar{t}},f}^{(TH,3)}(\lambda_i^{\bar{t}}) = 0$ means otherwise.

$$\vdots$$

- $M_{y,x^{\bar{t}},f}^{(O,\hat{z})}(\lambda_i^{\bar{t}}, \ldots, \omega_i^{(\bar{z},\bar{t})})$ indicates in time slot $\bar{t}$ the number of copies to be offloaded by type x UAV which is associated with mobile charging station y to the edge servers in BS f in stage $\hat{z}$, where $1 < \hat{z} \leq z$.

- $M_{y,x^{\bar{t}}}^{(L,\hat{z})}(\lambda_i^{\bar{t}}, \ldots, \omega_i^{(\bar{z},\bar{t})})$ indicates in time slot $\bar{t}$ the number of copies that the type x UAV which is associated with mobile charging station y is computing locally in stage $\hat{z}$.

- $M_{y,x^{\bar{t}},f}^{(TH,\hat{z})}(\lambda_i^{\bar{t}}, \ldots, \omega_i^{(\bar{z},\bar{t})})$ indicates is a binary variable to indicate in time slot $\bar{t}$ whether the type x UAV which is associated with mobile charging station y choose to offload or not in stage $\hat{z}$.

The objective function given in (3.35) - (3.38) is to minimize the total cost of the network. The expressions in (3.35), (3.36), (3.37) and (3.38) represent the first, second, third and up till z stage objectives, respectively. $\mathcal{P}(\lambda_i^{\bar{t}})$ and $\mathcal{P}(\omega_i^{(\bar{z},\bar{t})})$ denote the probabilities if scenarios $\lambda_i^{\bar{t}} \in \Theta^{\bar{t}}$ and $\omega_i^{(\bar{z},\bar{t})} \in \Omega^{(\bar{z},\bar{t})}$ are realized, respectively. The SIP formulation can be expressed as follows:

$$\min_{M_f^{(s)},\ldots,M_{y,x^{\bar{t}},f}^{(O,\hat{z})}(\lambda_i^{\bar{t}},\ldots,\omega_i^{(\bar{z},\bar{t})})} :$$

$$\sum_{\bar{t}\in\mathcal{T}} O_{allocation}^{Task}(\bar{t}) = \sum_{\bar{t}\in\mathcal{T}} \sum_{f\in\mathcal{F}} M_f^{(s)} \bar{C}_f + \mathbb{E}\Big[Q(M_f^{(s)}(\lambda_i^{\bar{t}}))\Big], \tag{3.35}$$

where

$$\mathcal{Q}(M_f^{(s)}(\lambda_i^{\bar{t}})) = \sum_{\bar{t}\in\mathcal{T}} \sum_{\lambda_i^{\bar{t}}\in\Theta^{\bar{t}}} \mathcal{P}(\lambda_i^{\bar{t}}) \sum_{y\in\mathcal{Y}} \sum_{f\in\mathcal{F}} \Big(M_{y,x^{\bar{t}}}^{(L,2)}(\lambda_i^{\bar{t}}) \bar{C}_{y,x}(D_y(\lambda_i^{\bar{t}})) + M_{y,x^{\bar{t}},f}^{(O,2)}(\lambda_i^{\bar{t}}) C_{y,x,f}(D_y(\lambda_i^{\bar{t}})) + \underline{C}_{y,x}^{thresh} + \hat{C}_{y,x}(D_y(\lambda_i^{\bar{t}})) + \mathbb{E}\Big[\mathcal{Q}(M_{y,x^{\bar{t}},f}^{(O,2)}(\lambda_i^{\bar{t}}, \omega_i^{(3,\bar{t})}))\Big] \Big), \tag{3.36}$$

$$\mathcal{Q}(M_{y,x^{\bar{t}},f}^{(O,2)}(\lambda_i^{\bar{t}}, \omega_i^{(3,\bar{t})})) = \sum_{\bar{t}\in\mathcal{T}} \sum_{\omega_i^{(3,\bar{t})}\in\Omega^{(3,\bar{t})}} \mathcal{P}(\omega_i^{(3,\bar{t})}) \sum_{y\in\mathcal{Y}} \sum_{f\in\mathcal{F}} \Big(M_{y,x^{\bar{t}}}^{(L,3)}(\lambda_i^{\bar{t}}, \omega_i^{(3,\bar{t})}) \bar{C}_{y,x}(D_y(\lambda_i^{\bar{t}})) + M_{y,x^{\bar{t}},f}^{(O,3)}(\lambda_i^{\bar{t}}, \omega_i^{(3,\bar{t})}) C_{y,x,f}(D_y(\lambda_i^{\bar{t}}) + M_{y,x^{\bar{t}},f}^{(TH,3)}(\lambda_i^{\bar{t}}, \omega_i^{(3,\bar{t})}) \underline{C}_{y,x}^{thresh} + \mathbb{E}\Big[\mathcal{Q}(M_{y,x^{\bar{t}},f}^{(O,3)})(\lambda_i^{\bar{t}}, \omega_i^{(3,\bar{t})}, \omega_i^{(4,\bar{t})})\Big] \Big), \tag{3.37}$$

$$\vdots$$

$$\mathcal{Q}(M_{y,x^{\bar{t}},f}^{(O,\hat{z})})(\lambda_i^{\bar{t}}, \omega_i^{(3,\bar{t})}, \ldots, \omega_i^{(\bar{z},\bar{t})}) = \sum_{\bar{t}\in\mathcal{T}} \sum_{\omega_i^{(\bar{z},\bar{t})}\in\Omega^{(\bar{z},\bar{t})}} \mathcal{P}(\omega_i^{(\bar{z},\bar{t})}) \sum_{y\in\mathcal{Y}} \sum_{f\in\mathcal{F}} \Big(M_{y,x^{\bar{t}}}^{(L,\hat{z})}(\lambda_i^{\bar{t}}, \ldots, \omega_i^{(\bar{z},\bar{t})}) \bar{C}_{y,x}(D_y(\lambda_i^{\bar{t}})) + M_{y,x^{\bar{t}},f}^{(TH,\bar{z})}(\lambda_i^{\bar{t}}) \underline{C}_{y,x}^{thresh} + M_{y,x^{\bar{t}},f}^{(O,\hat{z})}(\lambda_i^{\bar{t}}, \ldots, \omega_i^{(\bar{z},\bar{t})}) C_{y,x,f}(D_y(\lambda_i^{\bar{t}})) + \mathbb{F}_y^{\bar{z}}(\lambda_i^{\bar{t}}, \ldots, \omega_i^{(\bar{z},\bar{t})}) \tilde{C} \Big), \tag{3.38}$$

subject to:

$$\sum_{y\in\mathcal{Y}} M_{y,x^{\bar{t}},f}^{(O,2)}(\lambda_i^{\bar{t}}) \leq \sigma M_f^{(s)}, \qquad \forall \bar{t} \in \mathcal{T}, \forall f \in \mathcal{F}, \forall \lambda_i^{\bar{t}} \in \Theta^{\bar{t}}, \tag{3.39}$$

$$\vdots$$

$$\sum_{y\in\mathcal{Y}} M^{(O,\hat{z})}_{y,x^{\bar{t}},f}(\lambda^{\bar{t}}_i,\ldots,\omega^{(\bar{z},\bar{t})}_i) \leq \sigma M^{(s)}_f,$$
$$\forall \bar{t}\in\mathcal{T},\forall f\in\mathcal{F},\forall\lambda^{\bar{t}}_i\in\Theta^{\bar{t}},\forall\omega^{(3,\bar{t})}_i\in\Omega^{(3,\bar{t})},\ldots,$$
$$\forall\omega^{(\bar{z},\bar{t})}_i\in\Omega^{(\bar{z},\bar{t})}, \tag{3.40}$$

$$M^{(O,3)}_{y,x^{\bar{t}},f}(\lambda^{\bar{t}}_i,\omega^{(3,\bar{t})}_i) \leq \sigma M^{(TH,3)}_{y,x^{\bar{t}},f}(\lambda^{\bar{t}}_i,\omega^{(3,\bar{t})}_i),$$
$$\forall \bar{t}\in\mathcal{T},\forall y\in\mathcal{Y},\forall f\in\mathcal{F},\forall\lambda^{\bar{t}}_i\in\Theta^{\bar{t}},\forall\omega^{(3,\bar{t})}_i\in\Omega^{(3,\bar{t})}, \tag{3.41}$$

$$\vdots$$

$$M^{(O,\hat{z})}_{y,x^{\bar{t}},f}(\lambda^{\bar{t}}_i,\ldots,\omega^{(\bar{z},\bar{t})}_i) \leq \sigma M^{(TH,\bar{z})}_{y,x^{\bar{t}},f}(\lambda^{\bar{t}}_i,\ldots,\omega^{(\bar{z},\bar{t})}_i),$$
$$\forall \bar{t}\in\mathcal{T},\forall y\in\mathcal{Y},\forall f\in\mathcal{F},\forall\lambda^{\bar{t}}_i\in\Theta^{\bar{t}},\forall\omega^{(3,\bar{t})}_i\in\Omega^{(3,\bar{t})},\ldots,$$
$$\forall\omega^{(\bar{z},\bar{t})}_i\in\Omega^{(\bar{z},\bar{t})}, \tag{3.42}$$

$$\sum_{f\in\mathcal{F}} M^{(O,3)}_{y,x^{\bar{t}},f}(\lambda^{\bar{t}}_i,\omega^{(3,\bar{t})}_i) \geq \mathbb{F}^{(3,\bar{t})}_y(\lambda^{\bar{t}}_i,\omega^{(3,\bar{t})}_i)$$
$$\left(\bar{A}^{(3,\bar{t})}_y(\lambda^{\bar{t}}_i,\omega^{(3,\bar{t})}_i) - M^{(L,3)}_{y,x^{\bar{t}}}(\lambda^{\bar{t}}_i,\omega^{(3,\bar{t})}_i)\right),$$
$$\forall \bar{t}\in\mathcal{T},\forall y\in\mathcal{Y},\forall\lambda^{\bar{t}}_i\in\Theta^{\bar{t}},\forall\omega^{(3,\bar{t})}_i\in\Omega^{(3,\bar{t})}, \tag{3.43}$$

$$\vdots$$

$$\sum_{f\in\mathcal{F}} M^{(O,\hat{z})}_{y,x^{\bar{t}},f}(\lambda^{\bar{t}}_i,\ldots,\omega^{(\bar{z},\bar{t})}_i) \geq \Big(\bar{A}^{(\bar{z},\bar{t})}_y(\lambda^{\bar{t}}_i,\ldots,\omega^{(\bar{z},\bar{t})}_i) -$$
$$M^{(L,\hat{z})}_{y,x^{\bar{t}}}(\lambda^{\bar{t}}_i,\ldots,\omega^{(\bar{z},\bar{t})}_i)\Big)\mathbb{F}^3_y(\lambda^{\bar{t}}_i,\omega^{(3,\bar{t})}_i)\ldots\mathbb{F}^{\bar{z}}_y(\lambda^{\bar{t}}_i,\ldots,\omega^{(\bar{z},\bar{t})}_i),$$
$$\forall \bar{t}\in\mathcal{T},\forall y\in\mathcal{Y},\forall\lambda^{\bar{t}}_i\in\Theta^{\bar{t}},\forall\omega^{(3,\bar{t})}_i\in\Omega^{(3,\bar{t})},\ldots,$$
$$\forall\omega^{(\bar{z},\bar{t})}_i\in\Omega^{(\bar{z},\bar{t})}, \tag{3.44}$$

$$M^{(L,2)}_{y,x^{\bar{t}}}(\lambda^{\bar{t}}_i) + M^{(O,2)}_{y,x^{\bar{t}},f}(\lambda^{\bar{t}}_i) \geq k,$$
$$\forall \bar{t}\in\mathcal{T},\forall y\in\mathcal{Y},\forall f\in\mathcal{F},\forall\lambda^{\bar{t}}_i\in\Theta^{\bar{t}}, \tag{3.45}$$

$$
\begin{aligned}
&M^{(L,2)}_{y,x^{\bar{t}}}(\lambda^{\bar{t}}_i) + \sum_{f\in\mathcal{F}} M^{(O,2)}_{y,x^{\bar{t}},f}(\lambda^{\bar{t}}_i) + \mathbb{F}^{(3,\bar{t})}_y(\lambda^{\bar{t}}_i, \omega^{(3,\bar{t})}_i) \\
&M^{(L,3)}_{y,x^{\bar{t}}}(\lambda^{\bar{t}}_i, \omega^{(3,\bar{t})}_i) + \sum_{f\in\mathcal{F}} \mathbb{F}^{(3,\bar{t})}_y(\lambda^{\bar{t}}_i, \omega^{(3,\bar{t})}_i) M^{(O,3)}_{y,x^{\bar{t}},f}(\lambda^{\bar{t}}_i, \omega^{(3,\bar{t})}_i) \\
&\quad -\mathbb{F}^{(3,\bar{t})}_y(\lambda^{\bar{t}}_i, \omega^{(3,\bar{t})}_i)\Big(\bar{A}^{(3,\bar{t})}_y(\lambda^{\bar{t}}_i, \omega^{(3,\bar{t})}_i) - M^{(L,2)}_{y,x^{\bar{t}}}(\lambda^{\bar{t}}_i, \omega^{(3,\bar{t})}_i)\Big) \\
&\qquad\qquad \geq k, \\
&\forall \bar{t} \in \mathcal{T}, \forall y \in \mathcal{Y}, \forall f \in \mathcal{F}, \forall \lambda^{\bar{t}}_i \in \Theta^{\bar{t}}, \forall \omega^{(3,\bar{t})}_i \in \Omega^{(3,\bar{t})},
\end{aligned} \tag{3.46}
$$

$$\vdots$$

$$
\begin{aligned}
&M^{(L,2)}_{y,x^{\bar{t}}}(\lambda^{\bar{t}}) + \sum_{f\in\mathcal{F}} M^{(O,2)}_{y,x^{\bar{t}},f}(\lambda^{\bar{t}}_i) + \ldots + \sum_{f\in\mathcal{F}} \mathbb{F}^{(3,\bar{t})}_y(\lambda^{\bar{t}}_i, \omega^{(3,\bar{t})}_i) \\
&\qquad \ldots \mathbb{F}^{(\bar{z},\bar{t})}_y(\lambda^{\bar{t}}_i, \ldots, \omega^{(\bar{z},\bar{t})}_i) M^{(O,\hat{z})}_{y,x^{\bar{t}},f}(\lambda^{\bar{t}}_i, \ldots, \omega^{(\bar{z},\bar{t})}_i) + \\
&\mathbb{F}^{(3,\bar{t})}_y(\lambda^{\bar{t}}_i, \omega^{(3,\bar{t})}_i) \ldots \mathbb{F}^{\bar{z}}_y(\lambda^{\bar{t}}_i, \ldots, \omega^{(\bar{z},\bar{t})}_i) M^{(L,\hat{z})}_{y,x^{\bar{t}}}(\lambda^{\bar{t}}_i, \ldots, \omega^{(\bar{z},\bar{t})}_i) \\
&\quad -\mathbb{F}^{(3,\bar{t})}_y(\lambda^{\bar{t}}_i, \omega^{(3,\bar{t})}_i)\Big(\bar{A}^{(3,\bar{t})}_y(\lambda^{\bar{t}}_i, \omega^{(3,\bar{t})}_i) - M^{(L,2)}_{y,x^{\bar{t}}}(\lambda^{\bar{t}}_i, \omega^{(3,\bar{t})}_i)\Big) \\
&\qquad - \ldots - \mathbb{F}^{(3,\bar{t})}_y(\lambda^{\bar{t}}_i, \omega^{(3,\bar{t})}_i) \ldots \mathbb{F}^{(\bar{z},\bar{t})}_y(\lambda^{\bar{t}}_i, \ldots, \omega^{(\bar{z},\bar{t})}_i) \\
&\qquad \Big(\bar{A}^{(\bar{z},\bar{t})}_y(\lambda^{\bar{t}}_i, \ldots, \omega^{(\bar{z},\bar{t})}_i) - M^{(L,\hat{z})}_{y,x^{\bar{t}}}(\lambda^{\bar{t}}_i, \ldots, \omega^{(\bar{z},\bar{t})}_i)\Big) \geq k, \\
&\forall \bar{t} \in \mathcal{T}, \forall y \in \mathcal{Y}, \forall f \in \mathcal{F}, \forall \lambda^{\bar{t}}_i \in \Theta^{\bar{t}}, \forall \omega^{(3,\bar{t})}_i \in \Omega^{(3,\bar{t})}, \ldots, \\
&\qquad\qquad \forall \omega^{(\bar{z},\bar{t})}_i \in \Omega^{(\bar{z},\bar{t})},
\end{aligned} \tag{3.47}
$$

$$
\sum_{y\in\mathcal{Y}} M^{(O,2)}_{y,x^{\bar{t}},f}(\lambda^{\bar{t}}_i) \leq q_f, \forall \bar{t} \in \mathcal{T}, \forall f \in \mathcal{F}, \forall \lambda^{\bar{t}}_i \in \Theta^{\bar{t}}, \tag{3.48}
$$

$$\vdots$$

$$
\begin{aligned}
&\sum_{y\in\mathcal{Y}} M^{(O,\hat{z})}_{y,x^{\bar{t}},f}(\lambda^{\bar{t}}_i, \ldots, \omega^{(\bar{z},\bar{t})}_i) \leq q_f, \\
&\forall \bar{t} \in \mathcal{T}, \forall f \in \mathcal{F}, \forall \lambda^{\bar{t}}_i \in \Theta^{\bar{t}}, \forall \omega^{(3,\bar{t})}_i \in \Omega^{(3,\bar{t})}, \ldots, \\
&\qquad\qquad \forall \omega^{(\bar{z},\bar{t})}_i \in \Omega^{(\bar{z},\bar{t})},
\end{aligned} \tag{3.49}
$$

$$
\begin{aligned}
M_f^{(s)}, M_{y,x^{\bar{t}},f}^{(TH,3)}(\lambda_i^{\bar{t}}, \omega_i^{(3,\bar{t})}), \ldots, M_{y,x^{\bar{t}},f}^{(TH,\bar{z})}(\lambda_i^{\bar{t}}, \ldots, \omega_i^{(\bar{z},\bar{t})}) \\
\in \{0,1\}, \\
\forall \bar{t} \in \mathcal{T}, \forall y \in \mathcal{Y}, \forall f \in \mathcal{F}, \forall \lambda_i^{\bar{t}} \in \Theta^{\bar{t}}, \\
\forall \omega_i^{(3,\bar{t})} \in \Omega^{(3,\bar{t})}, \ldots, \forall \omega_i^{(\bar{z},\bar{t})} \in \Omega^{(\bar{z},\bar{t})},
\end{aligned} \tag{3.50}
$$

$$
\begin{aligned}
M_{y,x^{\bar{t}},f}^{(O,2)}(\lambda_i^{\bar{t}}), M_{y,x^{\bar{t}}}^{(L,2)}(\lambda_i^{\bar{t}}), \ldots, M_{y,x^{\bar{t}},f}^{(O,\hat{z})}(\lambda_i^{\bar{t}}, \ldots, \omega_i^{(\bar{z},\bar{t})}), \\
M_{y,x^{\bar{t}}}^{(L,\hat{z})}(\lambda_i^{\bar{t}}, \ldots, \omega_i^{(\bar{z},\bar{t})}) \in \{0,1,\ldots\}, \\
\forall \bar{t} \in \mathcal{T}, \forall y \in \mathcal{Y}, \forall f \in \mathcal{F}, \forall \lambda_i^{\bar{t}} \in \Theta^{\bar{t}}, \\
\forall \omega_i^{(3,\bar{t})} \in \Omega^{(3,\bar{t})}, \ldots, \forall \omega_i^{(\bar{z},\bar{t})} \in \Omega^{(\bar{z},\bar{t})}.
\end{aligned} \tag{3.51}
$$

The constraints in (3.39) and (3.40) are the same as the constraint (3.28). (3.41) and (3.42) are the same as (3.30). (3.43) and (3.44) ensure that if the previous stage does not have a shortfall, and then the shortfall for the next stage should be zero. The shortfalls should only exist if the number of copies offloaded in the previous stage is more than or equal to the shortfall. They also ensure that the number of shortfalls can be reduced when the UAV performs local computations. (3.45) ensures that the number of copies computed locally and offloaded should be at least k. (3.46) and (3.47) ensure that if the UAV has a shortfall in the previous stage, the UAV has to compute the shortfall locally or re-offload to the edge servers in the BS to match k, i.e., the recovery threshold for task completion. (3.48) and (3.49) ensure that the total number of copies offloaded to the edge servers in each stage must not exceed the total number of edge servers in BSs. (3.50) indicates $M_f^{(s)}$, $M_{y,x^{\bar{t}},f}^{(TH,3)}(\lambda_i^{\bar{t}}, \omega_i^{(3,\bar{t})})$, $\ldots, M_{y,x^{\bar{t}},f}^{(TH,\bar{z})}(\lambda_i^{\bar{t}}, \ldots, \omega_i^{(\bar{z},\bar{t})})$ are binary variables. (3.51) indicates $M_{y,x^{\bar{t}},f}^{(O,2)}(\lambda_i^{\bar{t}})$, $M_{y,x^{\bar{t}}}^{(L,2)}(\lambda_i^{\bar{t}}), \ldots, M_{y,x^{\bar{t}},f}^{(O,\hat{z})}(\lambda_i^{\bar{t}}, \ldots, \omega_i^{(\bar{z},\bar{t})})$, $M_{y,x^{\bar{t}}}^{(L,\hat{z})}(\lambda_i^{\bar{t}}, \ldots, \omega_i^{(\bar{z},\bar{t})})$ are positive decision variables.

To solve SIP, we assume that the probability distribution of all scenarios in set $\gamma^{\bar{t}}$, $\Theta^{\bar{t}}$, $\Omega^{(3,\bar{t})}, \ldots, \Omega^{(\bar{z},\bar{t})}$ are known [162], then, the complexity of the problem increases exponentially when the total number of scenarios across all the stages increases [162, 163]. For phase one in Section 3.3, the total number of decision variables can be calculated from (3.52), where t', y', and x' denote the total number of time slots, mobile charging stations, and UAV types in the set, respectively. μ' is the total number of weather scenarios. The total number of constraints can be calculated from (3.53). For example, with six time slots, six mobile charging stations, three UAV types, and ten weather scenarios, phase one will have 468 decision variables and 864 constraints.

$$decisionVariables1 = t'y'x' + \mu't'y' \tag{3.52}$$

$$constraints1 = t'y' + 2\mu't'y' + t'y'x' \tag{3.53}$$

For phase two in Section 3.4.2, the total number of decision variables can be calculated from (3.54), where f' denotes the total number of BSs in the set. λ', $\omega^{3'}$, ..., $\omega^{\bar{z}'}$ are the total number of demand, shortfall in stage 2, ..., shortfall in stage $\bar{z}-1$ scenarios. The total number of constraints can be calculated from (3.55).

$$\begin{aligned} decisionVariables2 = t'f' + t'\lambda'y'f' + t'\omega^{3'}y'f' + \ldots \\ +t'\omega^{\bar{z}'}y'f' \end{aligned} \tag{3.54}$$

$$\begin{aligned} constraints2 = 2(t'f'\lambda'\omega^{3'} + \ldots + t'f'\lambda'\omega^{3'}\ldots\omega^{\bar{z}'} \\ t'f'\lambda') + 2(t'y'f'\lambda'\omega^{3'} + \ldots + t'y'f'\lambda'\omega^{3'}\ldots\omega^{\bar{z}'}) \\ +t'y'\lambda'\omega^{3'} + \ldots + t'y'\lambda'\omega^{3'}\ldots\omega^{\bar{z}'} \\ +t'y'f'\lambda' + 2(t'y'f'\lambda'\omega^{3'}\ldots\omega^{\bar{z}'}) \end{aligned} \tag{3.55}$$

3.4.3 Stochastic Coded Offloading Scheme Flowchart

In Fig. 3.5, the flowchart of SCOS algorithm is shown. The algorithm for solving SCOS is presented in four steps (i.e., Step-1 to Step-4) as follows.

In Step-1, we obtain the weather probability for the first time slot, and it can be obtained from the historical records [105] or weather forecast. The weather uncertainty is then modeled using SIP from (3.17).

In Step-2, we save the solution from Step-1.

In Step-3, we obtain the demand probability from the road traffic data set released by Land Transport Authority Singapore [164].

In Step-4, we obtain the shortfall probability from the historical records. Using the allocated UAV type from Step-2, we modeled the demand probability from Step-3 and the shortfall probability using SIP from (3.35). After solving this problem, if the next period has task allocation, the solution is saved, and the algorithm will proceed to Step-3. Otherwise, the algorithm proceeds to the next decision box. If the timeslot is the last time slot for the next decision box, the algorithm will end and output the SIP decision variables solution. Otherwise, the algorithm will proceed to Step-1.

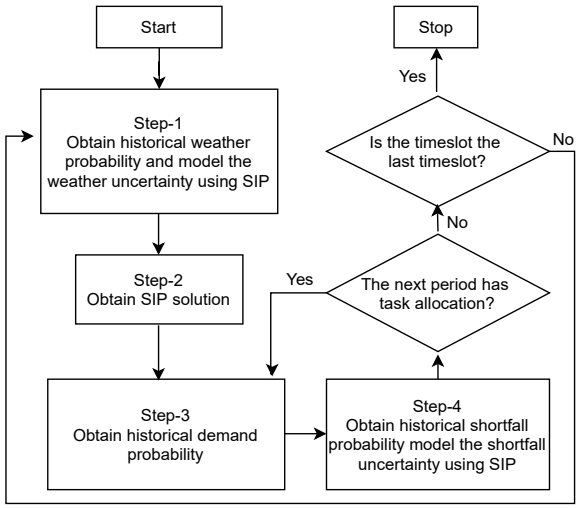


Figure 3.5: Flowchart of SCOS algorithm.

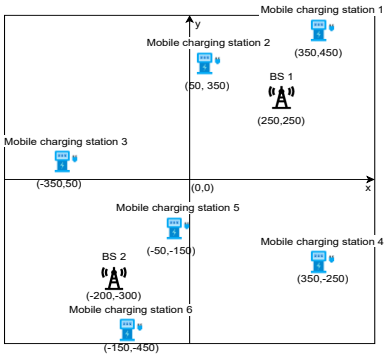


Figure 3.6: x-y coordinates of all the UAVs and BSs.

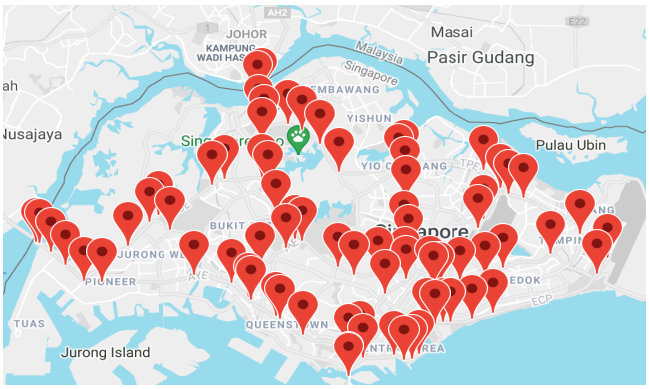


Figure 3.7: Singapore traffic camera location.

3.5 Simulation result and analysis

In this simulation, we consider the system model with one time slot $|\mathcal{T}| = 1$, six mobile charging stations, two BSs, sixty edge servers and thirty UAVs equally split into three UAV types. $\mathcal{X} = \{1 : small, 2 : medium, 3 : large\}$. The battery capacities B_1, B_2 and B_3 are $2375mAh$, $3500mAh$ and $5200mAh$, respectively [165]. A conceptual illustration of the system model is shown in Fig. 3.6. Fig. 3.6 is an x-y plane that shows the locations of the UAVs and the BSs. All the UAVs associated with their respective mobile charging stations are hovering at 100m, and the height of all edge servers in the BSs is 20m. They are randomly allocated in the area of 1000×1000 m^2. Each small grid is 25×25 m^2. We consider the case with $m = 2$ [93]. Therefore, we can substitute $s = \frac{2}{t}$ into (3.1),

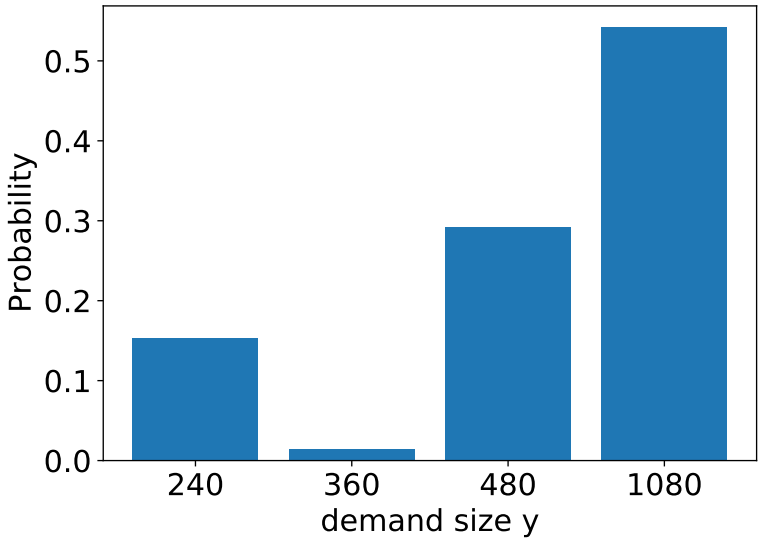


Figure 3.8: The probability distribution of the demand size.

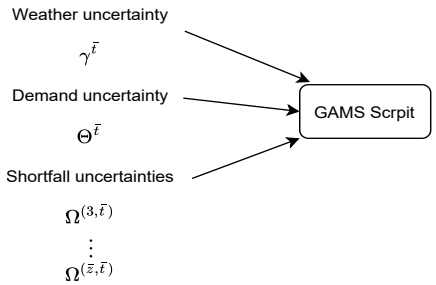


Figure 3.9: The value of uncertainties are inserted into GAMS script.

and we obtain the following:

$$k = 4t - t^2, \tag{3.56}$$

by differentiating (3.56) with respect to the variable t and equating the result to zero. Then, we are able to obtain $t = 2$, $s = 1$, and $k = 4$. The simulation parameters are summarized in Table 3.3 and their values are from [155, 149]. For the presented experiments, we implement the SIP model using GAMS script [166, 167].

We obtain the finite set $\mathcal{W}$ from the road traffic data set released by Land Transport Authority Singapore [164]. As depicted in Fig. 3.7, there are 72 cameras in Singapore to monitor the traffic flow. We generate 1 set of the data in real-time to obtain $\mathcal{Y}$. The data are in the dimension of $row \times column$. Following [93] we are only considering square matrices, therefore, $W = row = column$. The probability distribution of the demand size is shown in Fig. 3.8 and $\mathcal{W} = \{240, 260, 480, 1080\}$. We use scenarios to represent the weather, demand, and shortfall uncertainties. In each scenario, each uncertainty will have a corresponding value. For example, when $G_y^{\bar{t}}(\mu_i^{\bar{t}}) = 1$. It means that the weather uncertainty in scenario i in mobile charging station y at time slot $\bar{t}$ is represented by

Table 3.3: Experiment parameters

Parameter	**Values**
UAV weight in kg, W_x	[8-12]
Air density in kg/m^3, ρ	1.225
Rotor radius in meter, R	0.5
Rotor disc area in m^2, $\mathbb{A}$	0.79
Blade angular velocity in radians/second, $\triangle_x$	[380-420]
Tip speed of the rotor blade, U_{tip}	200
Number of blades, b	4
Blade or aerofoil chord length, c	0.0196
Rotor solidity, defined as the ratio of the total blade area bcR to the disc area $\mathbb{A}$, **s**	0.05
Fuselage drag ratio, d_0	0.3
Mean rotor induced velocity in hover, v_0	7.2
Profile drag coefficient, δ	0.012
Incremental correction factor to induced power, r	0.1
UAV hover height in meter, $H_{y,x}$	[80-100]
Height of edge servers in BS, H_f	20
UAV bandwidth in MHz, B_x	2
UAV transmit power in mW, P_x^C	32
UAV receiving power in mW, P_x^{re}	32
White Gaussian noise in dBm, N_0	-100
UAV computation power in GHz, τ_x	[0.6-1]
Number of CPU cycles needed to process a bit, C_x	20
Channel gains, β_0	-60dB
C_f in \$	0.05
α_1	0.001
α_2	0.0015
α_3	0.5
α_4	0.5
α_5	0.0001

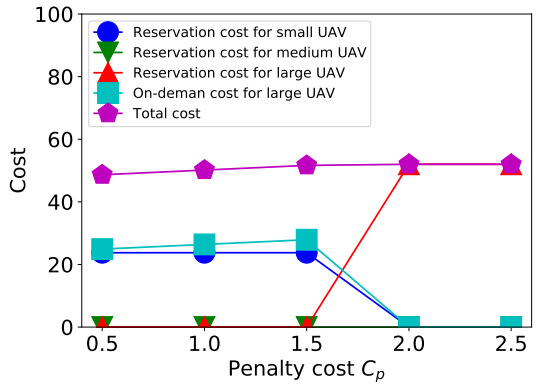


Figure 3.10: The cost of UAV type allocation when varying the penalty cost.

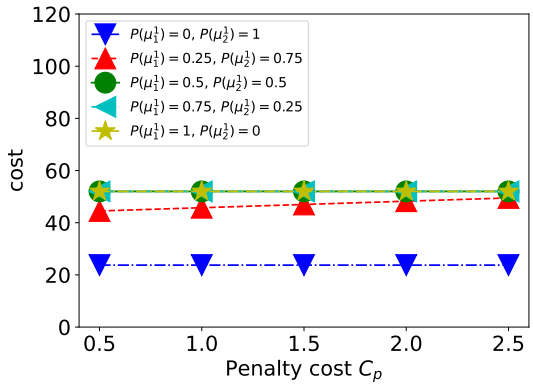


Figure 3.11: The cost of the network by varying the penalty cost and the weather probability.

the value of 1. For the road traffic data set, we obtain the dimension of the data, and we post-process the data to obtain $\mathcal{W}$ which is the demand size. We use the demand size to represent the actual demand uncertainty. Similar to weather uncertainty, $\mathcal{W}$ is represented by its corresponding values 240, 260, 480, and 1080. As shown in Fig. 3.9, weather, demand, and shortfall uncertainties can be injected into the GAMS script using their corresponding values.

3.5.1 Allocation of UAV

We first evaluate the UAV type allocation for each mobile charging station. We consider a two-stage SIP, where the first stage is (3.17) and the second stage is (3.18). In this UAV type allocation, we consider two scenarios of weather condition, i.e., strong wind and weak wind, which are $|\gamma^1| = 2$ [104]. The two scenarios are that all locations have strong wind μ_1^1 and no location has strong wind μ_2^1. We consider a stochastic system with $\mathcal{P}(\mu_1^1) = 0.3$ and $\mathcal{P}(\mu_2^1) = 0.7$. We perform the sensitivity analysis by varying the incurred cost and the weather condition probability.

Penalty cost C_p

We evaluate the UAV type allocation by varying the penalty cost C_p, and the result is presented in Fig. 3.10. When the penalty cost is low, the application owner still prefers to reserve the small UAV even though there is a crashing probability. However, the decision changed when the $C_p \geq 1.5$. The cost of the correction action, which is the on-demand cost of using a type 3 UAV, becomes much higher. $\bar{A}_3$ has to make a reservation on the type 3 UAV in the initial phase.

Probability of the weather condition

Next, we consider the same setting as that in Section 3.5.1. We investigate the impact on the network by varying both the weather probability $P(\mu_1^1)$ and C_p. The result is shown in Fig. 3.11. When the weather is good $P(\mu_1^1) = 0$, all the charging stations will choose to use a type 1 UAV in stage 1 since there is no strong wind. When the probability of bad weather increases, e.g., $P(\mu_1^2) \geq 0.5$ all the charging stations tend to choose type 3 UAVs even when the penalty cost is low due to a high probability of UAV crashes.

3.5.2 Allocation of Task

We now perform the sensitivity analysis on task allocation using the UAV type allocation derived when $C_p = 2$. We perform the sensitivity analysis by varying the number of stages z, hovering cost $C_{y,x^{\bar{i}}}^{thresh}$, the probability of shortfalls, recovery threshold k as well as the types of UAV use, and we discuss the results in this subsection.

Cost structure

We first study the cost structure of the network. As an illustration, a primitive UAV network is considered with zero local computation $M_{y,x}^{L,2} = 0$, $M_{y,x}^{L,3} = 0$, and we consider a three-stage SIP, where the first stage is (3.35), the second stage is (3.36) and the third stage is (3.38). Furthermore, we first consider six mobile charging stations, one BS, one demand scenario $|\Theta^1| = 1$ and $|\Omega^{(3,1)})| = 1$. The demand is 1080. The shortfall scenario is that there are shortfalls from the copies that the UAV has offloaded. Since $k = 4$, the number of copies that are offloaded $M_{y,x,f}^{(O,2)}$ begins with 4. In Fig. 3.12, the costs in the first, second, third stages and the total cost under the different number of the offloaded copies are presented. Since offloading is the only decision, the edge servers subscription cost in stage one should always paid. In the second stage, the cost is expected to increase as the number of copies that are offloaded increases. However, the cost in the third stage after knowing the actual shortfall decreases as the number of offloaded copies increases since the UAV needs to perform fewer re-computation to match the shortfall. Fig. 3.12 shows the optimal solution in this simple network. It can be identified that even in this simple network, the optimal solution is not trivial to obtain due to the uncertainty of shortfall. For example, the optimal cost is not the point where the cost in the second stage is the lowest of the cost in the third stage is the lowest. Therefore, SIP formulation is required to guarantee the minimum cost to the UAVs.

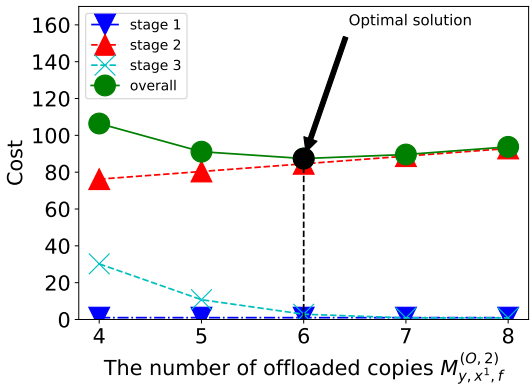


Figure 3.12: The optimal solution in a simple three-stage SIP UAV network.

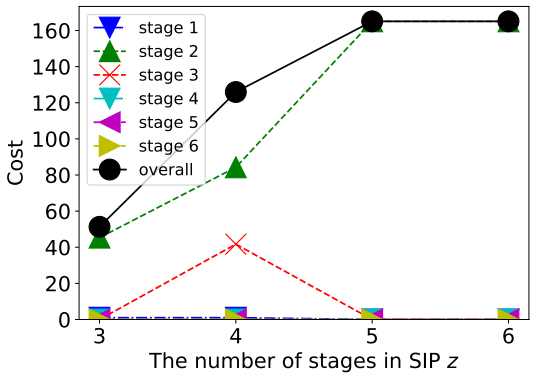


Figure 3.13: The cost of the network by varying the number of stages in SIP. Considered only UAV 1, BS 1 and BS 2.

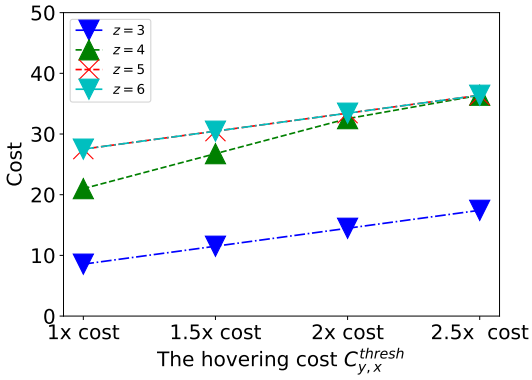


Figure 3.14: The cost of the network by varying the hovering cost $C_{y,x^{\bar{t}}}^{thresh}$.

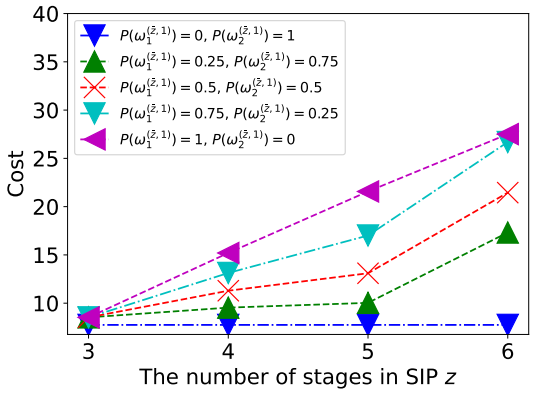


Figure 3.15: The cost of the network by varying the probability of shortfall in each stage.

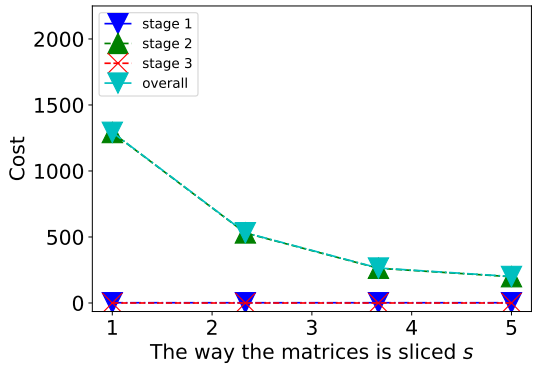


Figure 3.16: The cost of the network by varying the variable s to change the recovery threshold.

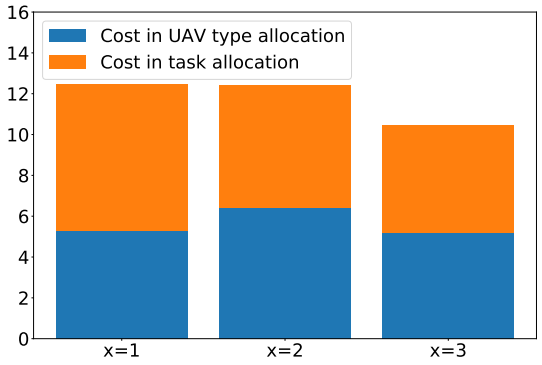


Figure 3.17: The overall network cost when the UAV type x is varied.

Number of stages z

We consider the case with four demand scenario $|\Theta^1| = 4$, one shortfall scenario $|\Omega^{(\bar{z},1)}| = 1$ for each SIP stage, and there is a shortfall in each stage. The four demand scenarios are made up of 4 demand size each, $\lambda_1^1 = 480$, $\lambda_2^1 = 240$, $\lambda_3^1 = 1080$ and $\lambda_4^1 = 360$. We vary the value of z from 3 to 6. The cost of the network is shown in Fig. 3.13 and Table 3.4 indicates the value of the variables. The number of variables in the six-stage SIP is large. Therefore, for illustration, we only indicate the value for all the variables in mobile charging station 1 when $z = 3$. Then, We use the result from $z = 3$ to explain the findings. When the data size is 240, the offloading cost is more expensive than the local computation cost. Therefore, the UAV will perform local computation in stage 2 when the data size is 240. $M_{1,3^1}^{(L,2)}(\lambda_2^1) = 4$, $M_{1,3^1,f}^{(O,2)}(\lambda_2^1) = 0$, $M_{1,3^1,f}^{(L,3)}(\lambda_2^1) = 0$ and $M_{1,3^1,f}^{(O,3)}(\lambda_2^1) = 0$. However, for the rest of the data size, the offloading action is chosen. The UAV will offload more in stage 2 to overcome the shortfall, and this offloading cost is cheaper than the local computation cost. The total number of copies offloaded to the edge servers in BSs 1 and 2 is 24 each. The UAVs will always offload to the edge servers that is closer to it as the cost is lower. To support the offloading process, 24 edge servers have to be placed in both BSs 1 and 2. Since the shortfalls exist in every stage, as the number of stages increases, the UAV will perform more local computation in stage 2. When $z \geq 5$, the UAV will only consider local computation regardless of the demand, and the overall cost will remain the same. Since there is no offloading process, edge servers' locations are no longer necessary as they are no longer needed.

Hovering cost $C_{y,x}^{thresh}$

We consider the case with mobile charging station 1 and BS 1. We vary both the hovering cost and the number of stages z to observe the impact on the network. We keep the rest of the parameters the same as Section 3.5.2. Fig. 3.14 shows the cost of the network. It can be identified that when the hovering cost increases, it will also increase the re-offloading cost, therefore increasing the network cost. Since there is a shortfall scenario in every stage, the increase of hovering costs in every stage will affect the decision of the UAV. For example, when stage 4 has 2.5 times hovering cost, the UAV will perform full local computation regardless of the data size. Full local computation is also performed when $z \geq 5$.

Probability of the shortfalls

We consider the setting similar to Section 3.5.2. There are two shortfall scenarios in each stage $|\Omega^{(\bar{z},1)}| = 2$. The two shortfall scenarios are i) all the UAVs have shortfalls $\omega_1^{(\bar{z},1)}$ in stage $\bar{z}$ and ii) all the UAVs do not have shortfall $\omega_2^{(\bar{z},1)}$. We observe the impact on the network by varying both the shortfall probabilities $P(\omega_1^{(\bar{z},1)})$, $P(\omega_2^{(\bar{z},1)})$ and the number of stages z. The result is shown in Fig. 3.15. Since there is no shortfall $P(\omega_1^{(\bar{z},1)}) = 0$, when z increases, the decision made by the UAV remains the same. The UAV can choose the cheapest decision, which is the offloading action. When $P(\omega_1^{(\bar{z},1)})$ increases, the number of copies that are computed locally and offloaded in the earlier stage also increases to reduce the penalty cost by the shortfalls. However, the number of scenarios increases exponentially with z [103], and this is the reason that it leads to a sharp increase in cost from stages 5 to 6. For example, in a 6-stage SIP with two scenarios per stage, this results in a total of $5^6 = 15,625$ 64 scenarios [103]. When $z = 6$ and $P(\omega_1^{(\bar{z},1)}) = 1$, with the high re-offloading cost, full local computation is perform in the earlier stage.

Recovery threshold k

Similar to Section 3.5.2, we consider three-stage SIP with four demand scenarios $|\Theta^1| = 4$, one shortfall scenario $|\omega^{(3,1)}| = 1$ and there is shortfalls in each offloading process. We monitor the impact of the network by varying the recovery threshold. We set $m = 4$ and we vary recovery threshold k by changing s from 1 to 5. Since k is controlled by (3.1), as s increases, k varies from 16 to 6. When $s = 1$, the recovery threshold is the highest $k = 16$. The result is shown in Fig. 3.16. The stage 1 cost remains constant as the BS is always subscribed. The cost of the network is also the highest as the UAV has to compute the most number of copies. When the data size is 240, 360, and 480, in stage 2, the UAV prefers to compute most of the copies locally as the local computation cost is much lower

Table 3.4: Decision Variable value

Variable	**z=3**	$z = 4$	$z = 5$	$z = 6$
$M_1^{(s)}$	1	1	0	0
$M_2^{(s)}$	0	0	0	0
$M_{1,3^1}^{(L,2)}(\lambda_1^1)$	**0**	4	4	4
$M_{1,3^1}^{(L,2)}(\lambda_2^1)$	**4**	4	4	4
$M_{1,3^1}^{(L,2)}(\lambda_3^1)$	**0**	0	4	4
$M_{1,3^1}^{(L,2)}(\lambda_4^1)$	**0**	4	4	4
$M_{1,3^1,1}^{(O,2)}(\lambda_1^1)$	**8**	0	0	0
$M_{1,3^1,1}^{(O,2)}(\lambda_2^1)$	**0**	0	0	0
$M_{1,3^1,1}^{(O,2)}(\lambda_3^1)$	**8**	4	0	0
$M_{1,3^1,1}^{(O,2)}(\lambda_4^1)$	**8**	0	0	0
$M_{1,3^1}^{(L,3)}(\lambda_1^1, \omega^{(3,1)})$	0	0	0	0
$M_{1,3^1}^{(L,3)}(\lambda_2^1, \omega^{(3,1)})$	0	0	0	0
$M_{1,3^1}^{(L,3)}(\lambda_3^1, \omega^{(3,1)})$	0	0	0	0
$M_{1,3^1}^{(L,3)}(\lambda_4^1, \omega^{(3,1)})$	0	0	0	0
$M_{1,3^1,1}^{(O,3)}(\lambda_1^1, \omega^{(3,1)})$	0	0	0	0
$M_{1,3^1,1}^{(O,3)}(\lambda_2^1, \omega^{(3,1)})$	0	0	0	0
$M_{1,3^1,1}^{(O,3)}(\lambda_3^1, \omega^{(3,1)})$	0	8	0	0
$M_{1,3^1,1}^{(O,3)}(\lambda_4^1, \omega^{(3,1)})$	0	0	0	0
Total number of edge server in BS 1	**24**	24	0	0
Total number of edge server in BS 2	**24**	24	0	0

than the offloading cost plus the re-offloading cost in stage 3. The reason is that when k is high, the number of shortfalls is also high, which increases the re-offloading cost. This is why the stage 3 cost is zero. As s increases, the value of k decreases, the cost in local computation increases. Therefore, instead of local computation, some of the UAVs perform offloading process in stage 2. Throughout the simulations, the cost in stage 3 is zero. With the additional penalty and hovering cost, no UAVs prefer to compute the shortfalls in stage 3.

UAV types

Similar to Section VI-A1 and Section VI-B5, we consider 1 UAV and $C_p = 2$ and investigate the differences in cost when we change the UAV type. The overall network cost is shown in Fig. 3.17. If the service provider initially uses type 1 and 2 UAVs, and when the wind is strong, the UAV cannot have sufficient energy to withstand the strong wind (weather uncertainty). As a result, the service provider has to deploy another UAV (type 3) to perform the task. However, if the service provider initially deploys a type 3 UAV, it will have sufficient energy to withstand the strong wind. There is no additional correction cost. Therefore, the UAV type allocation cost is cheaper if the service provider chooses to book type 3 UAV in advance. Furthermore, using type 3 UAV in task allocation makes the computation capability stronger than the other UAV types. As a result, it will have a shorter latency and will lead to a lower computation cost. Hence, the computation cost is also the lowest when a type 3 UAV is used. Therefore, the overall network cost is the lowest when a type 3 UAV is used.

3.5.3 Comparing between EVF, SIP and random scheme

We set $z = 3$ and compare the SIP with Expected-Value Formulation (EVF) [168] as well as the random scheme. Expected-value formulation uses the average values of shortfall as well as the demand and solves a DIP. For EVF, the total number of copies $(M^{(L)}_{y,x^{\bar{t}}} + M^{(O)}_{y,x^{\bar{t}},f})$ is fixed using the average value of shortfall and demand, an approximation scheme. In a random scheme, the values of the decision variables are randomly generated. We vary the price of the offloading action to compare the difference between EVF, SIP, and random schemes. This cost is combined with the UAV type allocation cost to observe the total cost of the network. Fig. 3.18 depicts the energy consumption comparison in percentage by using SIP as a reference and Fig. 3.19 depicts the comparison result. As shown in the result, EVF and random scheme cannot adapt to the change in cost. On the other hand, SIP can always achieve the best solution among the three to reduce the shortfall cost.

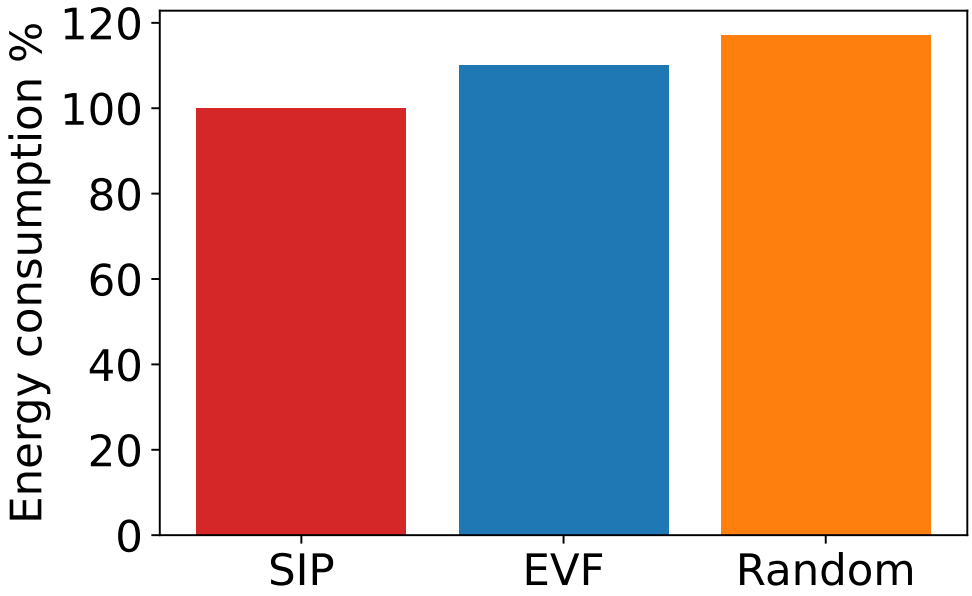


Figure 3.18: Energy consumption comparison in percentage by using SIP as reference between EVF and random scheme.

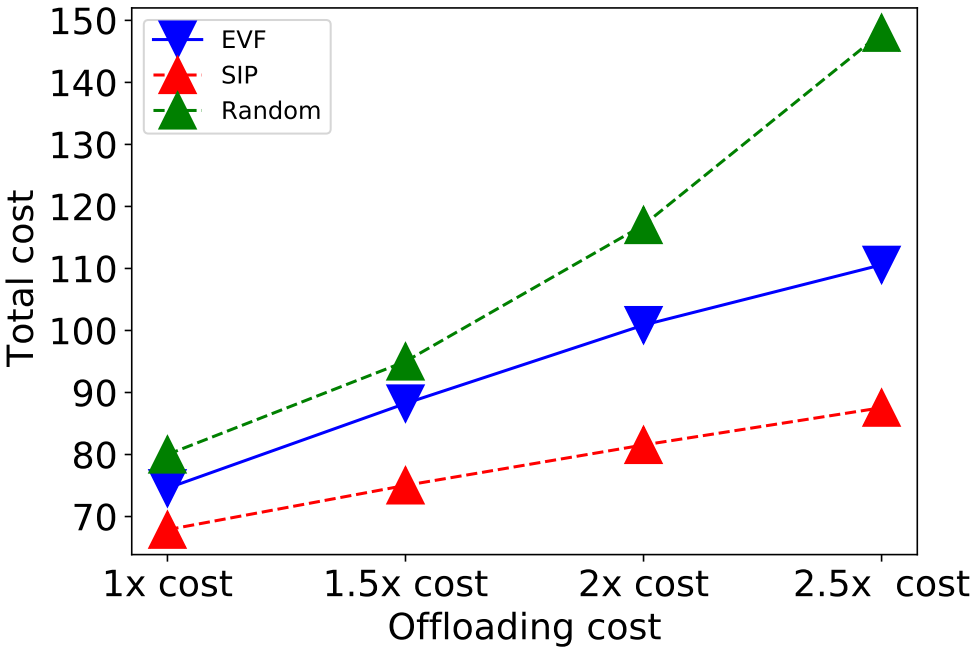


Figure 3.19: SIP comparing with EVF and random scheme.

3.6 Conclusion

In this chapter, we have proposed the Stochastic Coded Offloading Scheme (SCOS) that employs a CDC technique to combine with computation offloading to minimize the energy consumption of the UAV network. We have first formulated the SCOS as the two-stage stochastic programming to optimize the UAV type allocation to account for the weather uncertainty. Then, we have formulated the z-stage stochastic programming to account for data size and task completion uncertainty. We have also conducted numerical experiments to verify that our proposed SCOS can optimize the total cost and the UAVs' energy consumption. Compared to the benchmark, SCOS based on SIP can achieve the best solution as it can adapt to changes in data size and task failure probability. We also know that a three-stage SIP is enough to formulate the problem through the simulations as it has the lowest cost.

Chapter 4

Wireless Charging Hybrid network

This chapter introduces a novel hybrid coded edge computing network, powered by wireless technology, designed to address these challenges. Through a combination of local computation and coded computation offloading, this network aims to optimize energy usage and reduce latency in IoT operations. We explore a stochastic offloading scheme that strategically utilizes base stations for wireless energy transfer and edge servers for efficient data processing, accommodating uncertainties such as energy availability and computational demands. The chapter is structured to first present the system model, followed by a detailed discussion on stochastic resource optimization strategies, including two-stage and z-stage Stochastic Integer Programming (SIP) models. We conclude with a comprehensive performance evaluation of the proposed solutions, showcasing their effectiveness in enhancing the capabilities of IoT devices while minimizing operational costs.

4.1 Introduction

Today, the Internet-of-Things (IoT) has become an integral aspect of our lives, disrupting the way in which we track our personal health [169], maintain manufacturing equipment [170], and monitor the environment [171]. The next-generation communication systems are expected to further promote the growth of the *Internet of Everything* paradigm [136], by leveraging the enormous amounts of sensing data to drive the development of ubiquitous intelligence using Artificial Intelligence (AI) techniques. Moreover, the burgeoning Metaverse, an immersive

virtual world facilitated by advances in virtual and augmented reality, offers a new frontier for IoT integration, enhancing user interaction and data visualization in complex IoT networks [172].

However, the data-driven AI models, crucial for enriching Metaverse environments with real-time, context-aware interactions, rely heavily on massive quantities of training data. This in turn requires significant computation power for processing and model training. Moreover, due to the energy constraints, IoT devices face challenges in meeting the required quality-of-service (QoS) demands of the end-users, especially for latency-sensitive applications.

In response, edge computing is expected to be a key supporting pillar not only of next-generation communication systems but also of the expansive metaverse infrastructure. Specifically, edge computing leverages the storage, communication, and computation capabilities of end devices and edge servers to bring computing power closer to where data are produced. This is crucial for real-time Metaverse applications, where delay can disrupt user experiences. To improve the scalability of edge computing, next-generation communication systems are envisioned to provide energy transfer and harvesting for energy-constrained end devices, e.g., through 6G base stations (BSs) [136].

However, the wireless charging technology is relatively nascent with variable charging efficiencies [22]. For example, while magnetic resonant coupling has higher charging efficiency as compared to electromagnetic (EM) radiation power transfer, it is found in an earlier study that a 60W power transfer over 2 meters can only be completed with 40% efficiency [24]. As such, apart from relying solely on wireless power charging-enabled local computation, the resource-constrained IoT devices can also offload their computation tasks to the edge servers. Offloading computation-intensive tasks from the IoT devices to the edge servers helps to enhance the performance of the IoT devices while minimizing the device energy usage [173]. The IoT devices can leverage the resources of multiple edge servers to perform distributed computation tasks collaboratively. In order to mitigate the straggler effects in the distributed computing network where the computation tasks are delayed by the slower edge servers, coding techniques [174] can be used to split a computation task into multiple subtasks, which are in turn allocated to the edge servers. In particular, the IoT devices can complete the computation tasks upon receiving the computed results from a subset, instead of all edge servers. Moreover, coding redundancy alleviates the need to wait for the tail-end straggling edge servers to complete their allocated tasks, hence reducing the latency of the computation tasks.

In this Chapter, we propose a wireless-powered hybrid coded edge computing

network in which IoT devices, i.e., users, can perform their computation tasks using one of the following approaches:

1. *Full local computation:* Each cell of users is supported by a base station that powers the users through wireless power charging, thereby enabling the users to locally complete the computation tasks. However, the charging efficiency and resultant computation shortfall are only known *ex-post*.

2. *Full coded offloading:* Each user utilizes coding techniques to allocate the computation subtasks to the dedicated or non-dedicated edge servers for computation, thereby completing their task while reducing the computation latency. Note that a non-dedicated edge server, e.g., Docker [175], is cheaper to deploy as it shares its computation resources with other applications [176]. However, the computation process is delayed if it is occupied by too many users simultaneously, and this knowledge of its computation capacity is only known ex-post. In contrast, a dedicated edge server can be used exclusively by a requester [177], e.g., the Azure private multi-access edge compute [178], but is more costly. If the non-dedicated edge servers are occupied, the subtasks have to be re-offloaded to the dedicated edge servers. This is known as *corrective edge offloading*.

3. *Hybrid approach:* Each user partially offloads to dedicated/non-dedicated edge servers while simultaneously performing local computation.

To minimize the network cost while accounting for the two sources of uncertainty (i.e., charging efficiency and edge computation capacity uncertainty), we utilize a stochastic resource optimization approach. We formulate the computation decision problem into a two-stage stochastic integer programming (SIP) model. In the first stage, the computation decision is made. In the second stage, the corrective edge offloading decision can be determined if there is a shortfall, which is known ex-post. As the SIP problem may become computationally intractable since it scales with the size of the network, we introduce the Benders' decomposition and sample average approximation (SAA) to solve the SIP problem efficiently. Moreover, we consider the special case of z-stage SIP in which corrective edge offloading actions can be implemented over multiple stages for less time-sensitive tasks.

The contributions of this chapter are summarized as follows:

1. We present a novel wireless powered hybrid coded edge computing network that jointly leverages the use of base stations as wireless charging stations and edge servers for coded computation offloading. The network caters to

energy-constrained IoT devices to perform computation-intensive tasks. Moreover, the coded edge offloading mitigates the stragglers effect of distributed computation.

2. We propose a stochastic resource optimization solution for efficient resource management amid stochastic uncertainties that can only be realized ex-post. Our solution is able to derive the lowest cost of the network while accounting for the two sources of uncertainty in terms of wireless power charging inefficiency and unsuccessful edge offloading.

3. We analyze and reduce the complexity of the two-stage SIP model using Benders' decomposition and sample-average approximation. As a result, the solution is computationally tractable even as the network scales. In consideration of less time-sensitive computation tasks, we also introduce the special case using z-stage that enables corrective edge offloading over multiple stages.

The remainder of the Chapter is organized as follows. Section 4.2 presents the system model. Section 4.3 and 4.4 discuss the two-stage and $z-$ stage SIP models respectively. Then, we evaluate the performance of the proposed framework in Section 4.5, followed by the conclusion in Section 4.6.

4.2 System Model

We consider a wireless power charging enabled coded edge computing network (Fig. 4.1) with a coverage area that is divided into I cells, represented by a set $\mathcal{I} = \{1, \ldots, i, \ldots, I\}$. Each cell i contains W_i IoT devices, i.e., users, where the total number of users in each cell i may be different. Let d_{iw} denote the w^{th} user in cell i. The users aim to perform machine learning computation tasks using the data collected from their surroundings. Given the large amounts of data collected, the users may not have sufficient computation resources, e.g., energy, to complete the machine learning tasks individually.

In general, the hybrid network enables users to complete the computation tasks using the following approaches:

1. *Full local computation:* The I users are supported by H BSs, the set of which is denoted by $\mathcal{L} = \{1, \ldots, l, \ldots, L\}$ which can power the users through wireless power charging. The wireless power charging allows the users to have sufficient computation power to complete the machine learning tasks locally. While BSs are able to transfer their power to the users in a wireless manner, they may not

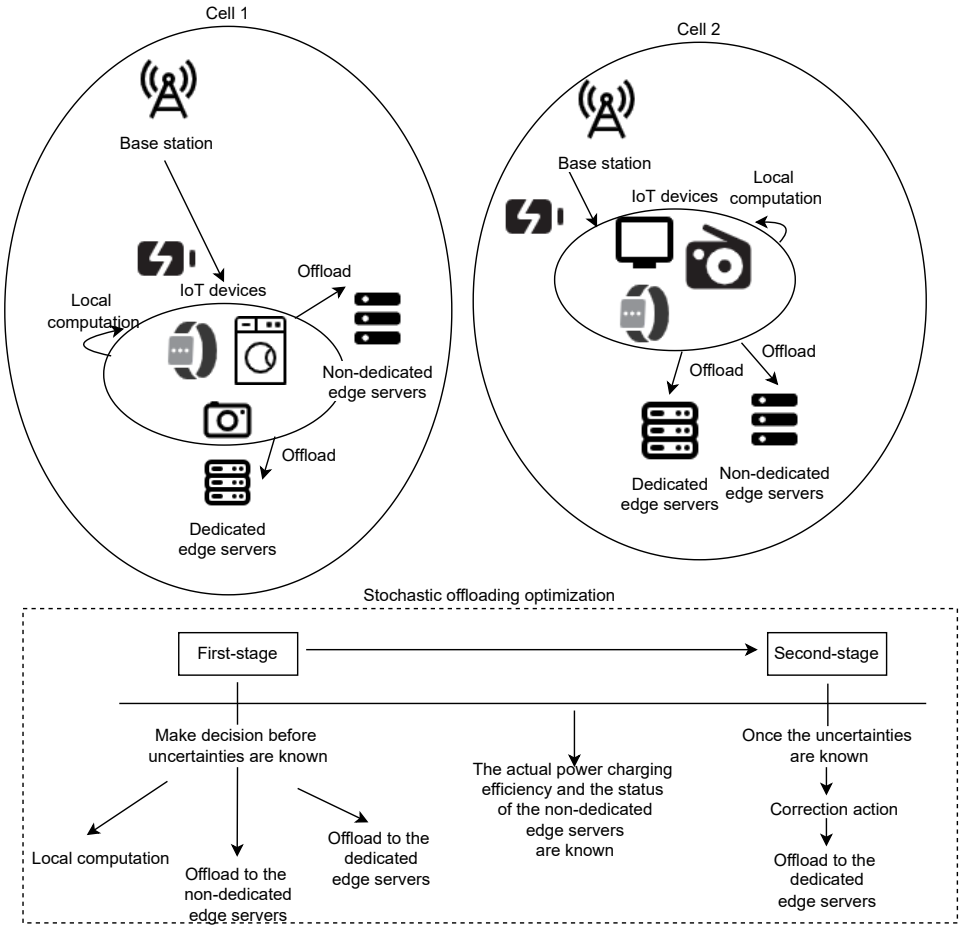


Figure 4.1: The system model of multi-tier computing.

always be reliable due to the variability in the efficiency of the magnetic coupling resonant. More specifically, the fluctuation of resonance frequency affects the efficiency of power transfer from the BSs to the users [179].

2. *Full coded offloading:* The I users are supported by K non-dedicated and X dedicated edge servers which are represented by sets $\mathcal{K} = \{1, \ldots, k, \ldots, K\}$ and $\mathcal{X} = \{1, \ldots, x, \ldots, X\}$, respectively. The edge servers provide coded computation offloading support for the users. Specifically, the users use coding techniques, e.g., Polynomial codes [92], to allocate the distributed computation subtasks to the edge servers that have sufficient resources. The use of coding techniques minimizes the computation latency by reducing the recovery threshold, which is defined as the number of edge servers that need to return their computed results for the users to reconstruct the final result. However, the non-dedicated edge server may be occupied and congested, and thus the edge offloading fails. In this case, the corrective edge offloading of the computation task to dedicated edge servers has to be implemented.

3. *Hybrid approach:* The users may adopt a combined approach in some cases, e.g., adopt wireless charging for *partial* local computation while simultaneously offloading the other subtasks to the edge servers.

In the following subsections, we discuss the BS charging model and coded distributed computing model. Then, we present the stochastic resource optimization problem formulation and optimization strategy adopted to achieve the objective of cost minimization in the hybrid network.

4.2.1 BSs Charging Model

To provide wireless power charging, BS l can provide P_{il}^{ower} power to charge users in cell i and the power is expressed as follows [2]:

$$P_{il}^{ower} = \sum_{w \in W_i} (\| e_{iwl} \|_2)^2, \tag{4.1}$$

and:

$$e_{iwl} = \frac{\bar{\beta}_l}{\bar{d}_{w,i,l}} \left[\cos(\frac{2\pi}{\kappa_l} \bar{d}_{w,i,l} + \phi_l) - j \sin(\frac{2\pi}{\kappa_l} \bar{d}_{w,i,l} + \phi_l) \right], \tag{4.2}$$

where $\bar{\beta}_l = \sqrt{\frac{Z_0 G_h P_l^o}{4\pi}}$, Z_0 is a physical constant indicating the wave-impedance of a plane wave in free space, G_l is the gain of BS l, and P_l^o is the output power of BS l. κ_l is the

wavelength of energy beam transmitted from BS l and $\bar{d}_{w,i,l}$ is the distance between BS l and d_{iw} user in cell i. (4.2) is a complex expression and the second term is the imaginary part. ϕ_l is the phase shift of the charging wave transmitted by BS l.

The power of the BSs is transferred to the users in the cell through the magnetic resonant coupling technique which allows high transmit efficiency over a relatively large distance. However, the performance of the BSs may be hindered by the uncertainty in the efficiency of the magnetic coupling resonant, which in turn affects the amount of energy that can be gained by the users. There are several factors that affect the efficiency of the magnetic coupling resonant such as the change in resonance frequency and the load value of the users [179]. In this chapter, we denote the efficiency of the magnetic coupling resonant when BS l charges cell i by β_l^i. The values of β_l^i, $\forall i \in \mathcal{I}$ and $\forall l \in \mathcal{L}$, are between 0 and 1. For example, $\beta_l^i = 0.4$ indicates that 40% of the charging power of BS l can be transferred efficiently to user i.

The efficiency of the magnetic coupling resonant cannot be known precisely before the wireless power charging is completed. In a large-scale heterogeneous network, there may exist varying combinations of charging efficiencies across the different BSs. In this chapter, we refer to each of these combinations as *scenarios*. Specifically, each scenario λ_y refers to a particular scenario indexed y, and is represented by a matrix of charging efficiencies $\beta_l^i(\lambda_y)$, $\forall i \in \mathcal{I}$ and $\forall l \in \mathcal{L}$. This matrix is expressed as follows:

$$\lambda_y = \begin{bmatrix} \beta_1^1(\lambda_y) & \beta_1^2(\lambda_y) & \cdots & \beta_1^I(\lambda_y) \\ \beta_2^1(\lambda_y) & \beta_2^2(\lambda_y) & \cdots & \beta_2^I(\lambda_y) \\ \vdots & \vdots & \ddots & \vdots \\ \beta_L^1(\lambda_y) & \beta_L^2(\lambda_y) & \cdots & \beta_L^I(\lambda_y) \end{bmatrix}. \tag{4.3}$$

The set of scenarios are represented by $\Lambda = \{\lambda_1, \ldots, \lambda_y, \ldots, \lambda_Y\}$ where Y is the total number of possible scenarios. The probability that scenario λ_y happens is denoted as $P(\lambda_y)$. For the rest of the chapter, we use $\beta_l^i(\lambda_y)$ and β_l^i interchangeably where confusion does not arise from dropping the (λ_y) notation. Accordingly, the available power to be gained by user i from BS l, which is denoted as p_{il}^v, is expressed as follows:

$$p_{il}^v = \beta_l^i P_{il}^{ower}. \tag{4.4}$$

Besides local computation, we assume that the amount of power P_{il}^{ower} required by users in cell i also includes the power for data transmission.

4.2.2 Coded Distributed Computing Model

The users can also partially or completely offload their subtasks to the edge servers, where the resources of multiple edge servers are leveraged to complete the distributed computation tasks collaboratively. One of the main challenges of a distributed computing network is the straggler effects where the overall computation time is greatly increased by the straggling edge servers. Specifically, the distributed computation tasks are only completed after the slowest edge server completes its allocated computation task. In order to mitigate the straggler effects, coding techniques are used to split the computation tasks into smaller tasks, i.e., subtasks, and distribute them to the edge servers for computation. The objective of the coding schemes is to minimize the recovery threshold α_{iw}, which is defined as the number of edge servers that are required to return their computed results to the users in order to obtain the final result.

In many AI-based applications such as scientific computing, machine learning, and graph processing, matrix multiplication has been widely used. In this chapter, we consider that each of the user d_{iw} adopts the Polynomial codes [92] to perform the distributed matrix multiplication computations[1], i.e., $\mathbf{C}^{iw} = \mathbf{A}^{iw\top}\mathbf{B}^{iw}$ where $\mathbf{A}^{iw}$ and $\mathbf{B}^{iw}$ are input matrices of user d_{iw}, where $\mathbf{A}^{iw} \in \mathbb{F}_q^{s\times r}$ and $\mathbf{B}^{iw} \in \mathbb{F}_q^{s\times t}$ for integers s, r, and t and a sufficiently large finite field $\mathbb{F}_q$. Note that the Polynomial codes used in this chapter can be extended by replacing Entangled Polynomial codes [180], PolyDot [181], and Lagrange codes [182], and it will change the recovery threshold. Entangled Polynomial codes is a generalized version of polynomial code. It allows column-wise partitioning of matrices with a parameter of $\hat{p}$, and it will achieve the recovery threshold of $\hat{p}mn + \hat{p} - 1$. When $\hat{p} = 1$, Entangled Polynomial codes equals Polynomial codes. PolyDot codes provides a trade-off between communication costs and recovery thresholds. A trade-off can be achieved by varying the value of s and t. The corresponding recovery threshold is $t^2(2s - 1)$. Lagrange codes is a batch processing approach for coded distributed matrix multiplication. Sub-matrices can be grouped into $\mathcal{K}$ blocks, where $\mathcal{K} = \frac{\hat{n}}{g}$. $\hat{n}$ is the total number of workers and each worker stores g coded sub-matrices.To perform the coded distributed computation using Polynomial codes, the four important steps are as follows [183]:

1. *Distribution of computation tasks:* In this sub-section, we use k and x interchangeably where they are both referred to as the edge server. Given that the edge servers are able to store up to $\frac{1}{m_{iw}}$ and $\frac{1}{n_{iw}}$ fractions of matrices $\mathbf{A}^{iw}$ and

[1]Our system model can be easily extended to consider other types of distributed computation problems, e.g., gradient descent, Fourier transform and convolution as well as the multiplication that involves more than two matrices.

$\mathbf{B}^{iw}$ respectively, user d_{iw} divides the input matrices into submatrices $\tilde{\mathbf{A}}_k^{iw} = f_k(\mathbf{A}^{iw})$ and $\tilde{\mathbf{B}}_k^{iw} = g_k(\mathbf{B}^{iw})$, where $\tilde{\mathbf{A}}_k^{iw} \in \mathbb{F}_q^{s\times\frac{r}{m_{iw}}}$ and $\tilde{\mathbf{B}}_k^{iw} \in \mathbb{F}_q^{s\times\frac{t}{n_{iw}}}$ respectively. Specifically, $\mathbf{f}$ and $\mathbf{g}$ represent the vectors of functions such that $\mathbf{f} = (f_1, \ldots, f_k, \ldots, f_K)$ and $\mathbf{g} = (g_1, \ldots, g_k, \ldots, g_K)$, respectively. Then, the user encodes and distributes the submatrices to the edge servers over the wireless channels for computations. Note that each edge server k may only receive one subtask, whereas the number of subtasks offloaded to the edge servers is based on the cost minimization optimization discussed in Section 4.3. Overall, the system model of the Polynomial codes consists of the followings [92]:

- The user receives computation inputs, encodes them, and distributes them to the edge servers.
- Edge servers perform pre-determined computations on their respective inputs in parallel.
- The user receives outputs from successful worker nodes and decodes them to recover the final output.

Note that data encoding and decoding actions exist in both local and offloading computation. Therefore, the energy consumption for data encoding and decoding is negligible as it exists in both.

2. *Computation by the edge servers:* Each edge server k is allocated submatrices $\tilde{\mathbf{A}}_k^{iw}$ and $\tilde{\mathbf{B}}_k^{iw}$ by the user. Based on the allocated submatrices, the edge servers perform the matrix multiplication, i.e., $\tilde{\mathbf{A}}_k^{iw\top}\tilde{\mathbf{B}}_k^{iw}$.

3. *Transmission of computed results:* Upon completion of the local computations, each edge server transmits its computed results, i.e., $\tilde{\mathbf{C}}_k^{iw} = \tilde{\mathbf{A}}_k^{iw\top}\tilde{\mathbf{B}}_k^{iw}$ to the user over the wireless communication channels.

4. *Reconstruction of final result:* By using the Polynomial codes, the user is able to decode and reconstruct the final result upon receiving α_{iw} computed results by using decoding functions. In other words, although the computation task is split and distributed to more than α_{iw} edge servers, the user does not need to wait for all the edge servers to return their computed results. The user only needs α_{iw} computed results where $\alpha_{iw} \leq K + X$. Note that although there is no constraint on the decoding functions to be used, a low-complexity decoding function such as the Reed-Solomon decoding algorithm [184] ensures the efficiency of the overall matrix multiplication computations.

Given that each edge server is able to store up to $\frac{1}{m_{iw}}$ and $\frac{1}{n_{iw}}$ fractions of matrices $\mathbf{A}^{iw}$ and $\mathbf{B}^{iw}$ respectively, the optimum recovery threshold that can be achieved by user d_{iw} using the Polynomial codes [92] is expressed as follows:

$$\alpha_{iw} = m_{iw} n_{iw}. \tag{4.5}$$

Compared with other coding techniques, such as MatDot codes [181], Polynomial codes have a lower computation cost per edge server and a higher communication cost from edge servers back to IoT devices, but they have a higher recovery threshold. However, we can carefully tune m_{iw} and n_{iw} from (5) to reduce the recovery threshold [181]. For example, if $m_{iw} = n_{iw} = 2$, the recovery threshold of Polynomial codes and MatDot codes are 4 and 3, respectively. For a square matrix of 36×36, the communication cost (symbols) of Polynomial codes and MatDot codes are 324 and 1296, respectively [92]. The computation cost (symbols) of Polynomial codes and MatDot codes are 11664 and 23328, respectively. Therefore, even though the recovery threshold of Polynomial codes is more than MatDot codes by 1, the overall number of symbols is lesser. The cost of deploying edge server k to complete the computation subtask is denoted as c_k. Hence, in order to complete its coded distributed computing task, user d_{iw} needs to pay a total amount of $\sum_{k \in \mathcal{L}_{iw}} c_k$, where $\mathcal{L}_{iw}$ represents the set of edge servers that successfully complete the computation subtasks allocated by user d_{iw}. For simplicity, we consider the cost of employing non-dedicated edge server k the same for all non-dedicated edge servers, i.e., $c_k = c, \forall k \in \mathcal{K}$. For example, the non-dedicated edge servers could belong to multiple service providers.

However, the desired response time is unable to be achieved when the non-dedicated edge server is used by multiple service providers at the same time [176]. Similarly, the cells do not know the information of whether other applications are using the non-dedicated edge servers *ex-ante*. The scenario of the non-dedicated edge server status is expressed as follows:

$$\eta_y = \{\gamma_1, \ldots, \gamma_K\}. \tag{4.6}$$

The offloaded subtasks have to be re-offloaded in this case to the dedicated edge server so that the computation process can be completed. The cost of employing the dedicated edge server x for corrective edge offloading is $\bar{c}$ and $\bar{c} > c$. The set of the current statuses of non-dedicated edge servers is denoted by $\eta_y \in \Omega$ and γ_k is a binary parameter. $\gamma_k = 0$, if the non-dedicated edge server k is fully occupied by another service provider and $\gamma_k = 1$ indicates otherwise.

Table 4.1: System Model Parameters.

Parameter	Description
$\mathbf{A}^{iw}$, $\mathbf{B}^{iw}$	Input matrices
α_{iw}	Recovery threshold of user
β_l^i	Efficiency of magnetic coupling resonant
$\underline{c}_{il}$	Cost of BS l for charging cell i
c_k	Cost of non-dedicated edge server k
$\bar{c}_x$	Cost of dedicated edge server x
c_i^p	Penalty cost
d_{iw}	w^{th} user in cell i
$d_{w,i,l}$	Distance between BS l and user d_{iw}
G_l	Gain of BS l
I	Number of cells
K	Number of non-dedicated edge servers
κ_l	Wavelength of energy beam transmitted from BS l
l	Number of BSs
λ_y	Scenario
$\frac{1}{m_{iw}}$, $\frac{1}{n_{iw}}$	Fractions of input matrices
p_l^c	BS l maximum charging energy
p_{ij}^v	Available power to be gained
P_l^o	Output power of BS l
$P(\lambda_y)$	Probability that scenario y happens
ϕ_l	Phase shift of the charging wave transmitted by BS l
X	Number of dedicated edge servers
Z_0	Wave-impedance of a plane wave

4.3 Stochastic Offloading Optimization

In this section, we first consider the case where the wireless power charging efficiency and the current status of the non-dedicated edge servers are stochastic in nature, thereby necessitating the stochastic offloading optimization approach. Then, we introduce the deterministic integer programming approach (DIP), which is a benchmark (ideal but non-realistic) case when the power transfer efficiency and the status of the non-dedicated edge servers are precisely known ex-ante.

4.3.1 Two-stage Stochastic Offloading Optimization

For easier representation, the uncertainty of charging efficiency from equation (4.3) and the uncertainty of the current status of the non-dedicated edge servers from (4.6) can be re-expressed as follows:

$$\lambda_y = \begin{bmatrix} F_1^1 & F_1^2 & \cdots & F_1^I \\ F_2^1 & F_2^2 & \cdots & F_2^I \\ \vdots & \vdots & \ddots & \vdots \\ F_L^1 & F_L^2 & \cdots & F_L^I \end{bmatrix}, \tag{4.7}$$

where $F_l^i = (\beta_l^i, \gamma_1, \ldots, \gamma_K)$. For example, $F_l^i = (\beta_l^i : 0.6, \gamma_1 : 0, \gamma_2 : 1, \gamma_3 : 1)$ means that the wireless power charging efficiency of BS l in cell i is 0.6. The non-dedicated edge server 1 is fully occupied while the non-dedicated edge servers 2 and 3 are available to be

used. Similar to (4.3), the set of scenarios are represented by $\Lambda = \{\lambda_1, \ldots, \lambda_y, \ldots, \lambda_Y\}$.

To account for the stochastic nature of wireless power charging efficiency and the status of the non-dedicated edge servers, we formulate the cost minimization problem into a two-stage SIP model in this section [105].

The two stages are as follows:

- *First stage:* The network makes the ex-ante decision between i) full local computation, ii) full coded edge offloading, and iii) hybrid computation with partial local and partial offloading.
- *Second stage:* The scenarios of wireless charging efficiency and non-dedicated edge servers availability are realized and observed. This means that each cell now has the full information on the charging efficiency of each BS and the status of the non-dedicated edge servers. If there is a shortfall, the cell makes the ex-post decision of corrective edge offloading to offload the remaining incomplete computation subtasks to the dedicated edge servers.

Using the SIP model, the network is able to alter its first stage ex-ante decision in response to the expected outcomes of the second. As an illustration, Fig. 4.2 shows a simplistic scenario tree with two scenarios in each stage. The scenarios in stage two are: i) experienced wireless power charging inefficiency and the non-dedicated edge servers are fully occupied by other applications, and ii) BSs have efficient wireless power charging and the non-dedicated edge servers are free. The scenario tree is a reduced form of an ensemble of scenarios or realizations of a process [185]. The tree clusters those realizations into a set of branches with specified probabilities of occurrence. Each scenario contains two nodes. In stage 1, all scenarios in the tree share the same node. In stage 2, the tree branches out and each branch belongs to only one scenario.

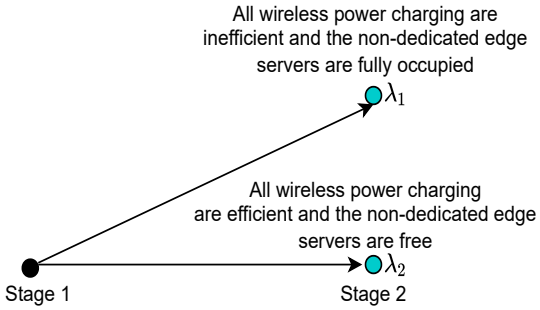


Figure 4.2: The scenario tree of a two-stage SIP with the scenarios of λ_1 and λ_2.

The two-stage SIP can be solved by generating a finite set of scenarios Λ with the likelihood of each scenario within the set occurring to be the corresponding probability

$P(\lambda_y)$. The SIP problem can be solved efficiently with a reasonable accuracy provided that the following conditions are met [186]:

1. The number of linear constraints is fixed.
2. For every linear constraint and every possible realization of scenario, the second stage problem is feasible.
3. The number of scenarios is not too large.

In our problem formulation, condition 1 is satisfied. By the definition of feasibility given in [187], condition 2 is also satisfied since there is a corrective action to be selected in stage two to correct the eventual discrepancies in stage one. However, condition 3 may not be satisfied especially since the number of scenarios scales with the size of the network. As such, we utilize the Benders' decomposition and sample-average approximation in the subsequent subsections to solve the SIP in a computationally efficient manner. The decision variables are as follows:

- $\delta_{il} \in \{0, 1\}$ indicates whether BS l is allocated to cell i. When $\delta_{il} = 1$, BS l is allocated to cell i and $\delta_{il} = 0$ indicates otherwise.
- $\bar{\delta}_{il} \in \{1, 2, \ldots, \}$ indicates the number of sub-tasks that are computed locally by the cell i when the power is provided by BS l. When $\bar{\delta}_{il} = 1$, cell i computes 1 sub-task locally.
- $\mu_{ik} \in \{0, 1\}$ indicates whether the sub-task is offloaded to the non-dedicated edge server k by cell i in stage one. Specifically, $\mu_{ik} = 1$ if cell i offload 1 sub-task to the edge server k and $\mu_{ik} = 0$ indicates otherwise.
- $\bar{\mu}_{ix} \in \{0, 1\}$ indicates whether the sub-task is offloaded to the dedicated edge server x by cell i in stage one. Unlike edge server k, edge server x does not share its computation capability with other applications.
- $\bar{\mu}^{re}_{ix}(\lambda_y) \in \{0, 1\}$ indicates whether the sub-task is re-offloaded to the dedicated edge server x by cell i in stage two under scenario λ_y.
- $\mu^p_i \in \{0, 1\}$ indicates the implementation of the penalty cost c^p_i in stage 2. When $\mu^p_i = 1$, cell i performs re-offloading action with an additional cost c^p_i and $\mu^p_i = 0$ otherwise.

In total, there are four types of the costs as follows:

- c is the cost to employ a non-dedicated edge server k, i.e., the cost for edge offloading of a subtask in stage 1.

- $\underline{c}_{il}$ is the cost that BS h received from cell i, i.e., the cost for local computation, and it is expressed as follows:

$$\underline{c}_{il} = \alpha_0 p_{il}^{v}, \tag{4.8}$$

  where α_0 is the cost coefficient.

- $\bar{c}$ is the cost to employ dedicated edge server x and $\bar{c} > c$.

- c_i^p is the penalty cost for cell i that occurs in stage 2. It is the penalty cost for the time that is wasted when the wireless power charging is not efficient and a shortfall occurs that has to be corrected.

The objective function given in (4.9) and (4.10) is to minimize the total cost of the network. The expressions in (4.9) and (4.10) represent the first and second stage objectives, respectively. $\mathbb{E}\Big[Q(\bar{\mu}_{ix}^{re}(\lambda_y))\Big]$ is the expectation over scenario $\lambda_y \in \Lambda$. The optimization variables, are the elements of the set $\Xi^{ts} = \{\delta_{il}, \bar{\delta}_{il}, \mu_{ik}, \bar{\mu}_{ix}, \mu_i^p(\lambda_y), \bar{\mu}_{ix}^{re}(\lambda_y)\}$. The SIP formulation can be expressed as follows:

$$\min_{\Xi^{ts}}:$$

$$\sum_{i\in\mathcal{I}}\sum_{l\in\mathcal{L}}\Big(\delta_{il} + \bar{\delta}_{il}\underline{c}_{il}\Big) + \sum_{i\in\mathcal{I}}\Big(\sum_{k\in\mathcal{K}}\mu_{ik}c + \sum_{x\in\mathcal{X}}\bar{\mu}_{ix}\bar{c}\Big) + \mathbb{E}\Big[Q(\bar{\mu}_{ix}^{re}(\lambda_y))\Big], \tag{4.9}$$

where:

$$Q(\bar{\mu}_{ix}^{re}(\lambda_y)) = \sum_{\lambda_y\in\Lambda} P(\lambda_y)\sum_{i\in\mathcal{I}}\Big(\mu_i^p(\lambda_y)c_i^p + \sum_{x\in\mathcal{X}}\bar{\mu}_{ix}^{re}(\lambda_y)\bar{c}\Big), \tag{4.10}$$

subject to:

$$\sum_{l\in\mathcal{L}}\bar{\delta}_{il}\sum_{w\in W_i}\alpha_{iw} + \sum_{k\in\mathcal{K}}\mu_{ik} + \sum_{x\in\mathcal{X}}\bar{\mu}_{ix} \geq \sum_{w\in W_i}\alpha_{iw}, \quad \forall i \in \mathcal{I}, \tag{4.11}$$

$$\sum_{l\in\mathcal{L}}\bar{\delta}_{il}\beta_l^i(\lambda_y)\sum_{w\in W_i}\alpha_{iw} + \sum_{k\in\mathcal{K}}\mu_{ik}\gamma_k(\lambda_y) + \sum_{x\in\mathcal{X}}\bar{\mu}_{ix} + \sum_{x\in\mathcal{X}}\bar{\mu}_{ix}^{re}(\lambda_y) \geq \sum_{w\in W_i}\alpha_{iw}, \quad \forall i \in \mathcal{I}, \forall\lambda_y \in \Lambda, \tag{4.12}$$

$$\bar{\delta}_{il} \leq \bar{A}\delta_{il}, \qquad \forall i \in \mathcal{I}, \forall l \in \mathcal{L}, \tag{4.13}$$

$$\sum_{x \in \mathcal{X}} \bar{\mu}_{ix}^{re}(\lambda_y) \leq \bar{A}\mu_i^p(\lambda_y), \qquad \forall i \in \mathcal{I}, \forall \lambda_y \in \Lambda, \tag{4.14}$$

$$\sum_{i \in \mathcal{I}} \delta_{il} \leq 1, \qquad \forall l \in \mathcal{L}, \tag{4.15}$$

$$\sum_{l \in \mathcal{L}} \delta_{il} \leq 1, \qquad \forall i \in \mathcal{I}, \tag{4.16}$$

$$\sum_{i \in \mathcal{I}} \mu_{ik} \leq 1, \qquad \forall k \in \mathcal{K}, \tag{4.17}$$

$$\sum_{i \in \mathcal{I}} \bar{\mu}_{ix} \leq 1, \qquad \forall x \in \mathcal{X}, \tag{4.18}$$

$$\sum_{i \in \mathcal{I}} \bar{\mu}_{ix}^{re}(\lambda_y) \leq 1, \qquad \forall x \in \mathcal{X}, \forall \lambda_y \in \Lambda, \tag{4.19}$$

$$\bar{\delta}_{il} p^s \sum_{w \in W_i} \alpha_{iw} \leq p_l^c, \qquad \forall i \in \mathcal{I}, \forall l \in \mathcal{L}, \tag{4.20}$$

$$\delta_{il}, \mu_{ik}, \mu_{ix}, \mu_i^p(\lambda_y), \bar{\mu}_{ix}^{re}(\lambda_y) \in \{0, 1\}, \quad \forall i \in \mathcal{I}, \forall l \in \mathcal{L}, \forall k \in \mathcal{K}, \forall x \in \mathcal{X}, \forall \lambda_y \in \Lambda, \tag{4.21}$$

$$\bar{\delta}_{il} \in \{0, \ldots, 1\}, \qquad \forall i \in \mathcal{I}, \forall l \in \mathcal{L}. \tag{4.22}$$

(4.11) and (4.12) ensure that the total number of sub-tasks that are computed should be greater than the total recovery threshold. If the non-dedicated edge servers are occupied, the sub-tasks are re-offloaded again in the next stage. (4.13) ensures that if cell i performs local computation, a BS should be allocated to cell i, where $\bar{A}$ is any large number. (4.14) ensures that the penalty cost is paid when the cell has to perform re-offloading. (4.15) and (4.16) ensure a one-to-one matching between the cell and the BS, i.e., a maximum of only one BS can be allocated to a cell, and a cell cannot choose another BS that has already been allocated. (4.17) - (4.19) ensures that the cell i cannot choose an occupied edger server when performing offloading and re-offloading. (4.20) ensures that the amount of energy that cell i uses should not exceed the maximum energy p_l^c that BS l provides and p^s is the energy used per sub-task. (4.21) indicates that the variables are binary variables. (4.22) indicates that $\bar{\delta}_{il}$ is a positive variable with a value between 0 and 1.

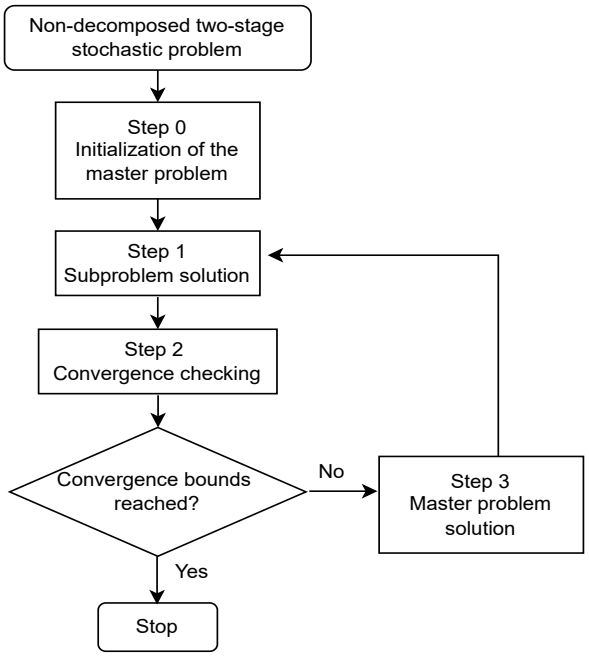


Figure 4.3: Flow chart of Benders' decomposition algorithm.

4.3.2 Benders' Decomposition

The problems formulated in (4.9) and (4.10) become complicated when the number of scenarios is large. The complexity of the problem depends on the number of variables, parameters, and constraints, which increases with the size of the network. SIP will potentially become computationally intractable if there are many scenarios [188]. Therefore, we propose the use of Benders' decomposition to decompose the original problem into multiple smaller problems [104]. Each of these can be solved iteratively to arrive at a final solution to the overall problem [104]. The benefit of this technique is that the decomposed problems are easier to solve than the original problem, so even if they have to be solved iteratively to obtain the solution, it will still be faster than trying to solve the original problem. However, (4.9) and (4.10) contain complicating variables that prevent the problem from being decomposed, i.e., first-stage variables [188]. Therefore, we can fix the complicating variables with a fixed value [104] so that the two-stage SIP can be decomposed into the independent optimization of master (4.23) and sub-problems (4.33).

The Benders' decomposition algorithm is performed iteratively to solve the decision variables and calculate the lower and upper bound. The algorithm terminates when the optimal solution converges, i.e., the difference between the lower bound and upper bounds are less than ϵ, i.e., a very small tolerance value. Figure 4.3 shows the flowchart of Benders' decomposition algorithm. The algorithm mainly consists of four steps as

follows.

Step 0: Initialization of the master problem. We initialize the parameters in the master problem once at the beginning of the algorithm. Let τ denote the iteration counter, v denote the iteration before τ and initially set $\tau = 1$. The optimization variables, are the elements of the set $\Xi^{ms} = \{\delta_{il}^{\tau}, \bar{\delta}_{il}^{\tau}, \mu_{ik}^{\tau}, \bar{\mu}_{ix}^{\tau}\}$. The master problem is expressed as follows.

$\min_{\Xi^{ms}}$:

$$\sum_{i\in\mathcal{I}}\sum_{l\in\mathcal{L}}\left(\delta_{il}^{\tau} + \bar{\delta}_{il}^{\tau}\mathfrak{c}_{il}\right) + \sum_{i\in\mathcal{I}}\left(\sum_{k\in\mathcal{K}}\mu_{ik}^{\tau}c + \sum_{x\in\mathcal{X}}\bar{\mu}_{ix}^{\tau}\bar{c}\right) + \alpha_{\tau}, \tag{4.23}$$

subject to:

$$\sum_{l\in\mathcal{L}}\bar{\delta}_{il}^{\tau}\sum_{w\in W_i}\alpha_{iw} + \sum_{k\in\mathcal{K}}\mu_{ik}^{\tau} + \sum_{x\in\mathcal{X}}\bar{\mu}_{ix}^{\tau} \geq \sum_{w\in W_i}\alpha_{iw}, \quad \forall i \in \mathcal{I}, \tag{4.24}$$

$$\bar{\delta}_{il}^{\tau} \leq \bar{A}\delta_{il}^{\tau}, \qquad \forall i \in \mathcal{I}, \forall l \in \mathcal{L}, \tag{4.25}$$

$$\sum_{i\in\mathcal{I}}\delta_{il}^{\tau} \leq 1, \qquad \forall l \in \mathcal{L}, \tag{4.26}$$

$$\sum_{l\in\mathcal{L}}\delta_{il}^{\tau} \leq 1, \qquad \forall i \in \mathcal{I}, \tag{4.27}$$

$$\sum_{i\in\mathcal{I}}\mu_{ik}^{\tau} \leq 1, \qquad \forall k \in \mathcal{K}, \tag{4.28}$$

$$\sum_{i\in\mathcal{I}}\mu_{ix}^{\tau} \leq 1, \qquad \forall x \in \mathcal{X}, \tag{4.29}$$

$$\bar{\delta}_{il}^{\tau}p^{s}\sum_{w\in W_i}\alpha_{iw} \leq p_l^{c}, \qquad \forall i \in \mathcal{I}, \forall l \in \mathcal{L}, \tag{4.30}$$

$$\alpha_{\tau} \geq \alpha^{(lb)}, \tag{4.31}$$

$$\alpha_\theta \geq \Big[\sum_{\lambda_y\in\Lambda} P(\lambda_y)\sum_{i\in\mathcal{I}}\Big(\mu_i^{(p,v)}(\lambda_y)c_i^p + \sum_{x\in\mathcal{X}}\bar{\mu}_{ix}^{(re,v)}(\lambda_y)\bar{c}\Big)\Big]+$$
$$\sum_{i\in\mathcal{I}}\sum_{\lambda_y\in\Lambda}\Big(\sum_{l\in\mathcal{L}}\rho_{il}^v(\lambda_y)(\delta_{il}^\tau-\delta_{il}^v)+$$
$$\sum_{l\in\mathcal{L}}\bar{\rho}_{il}^v(\lambda_y)(\bar{\delta}_{il}^\tau-\bar{\delta}_{il}^v)+$$
$$\sum_{k\in\mathcal{K}}\rho_{ik}^v(\lambda_y)(\mu_{ik}^\tau-\mu_{ik}^v)+$$
$$\sum_{x\in\mathcal{X}}\rho_{ix}^v(\lambda_y)(\mu_{ix}^\tau-\mu_{ix}^v)\Big). \tag{4.32}$$

The objective function (4.23) and the constraints (4.24) - (4.30) are modified from (4.9), (4.11), (4.13), (4.15) - (4.18), and (4.20) by adding an iteration counter τ. α_τ is an auxiliary variable, representing the total objective function of the second-stage problem in iteration τ and it will be improved in subsequent iterations. $\alpha^{(lb)}$ is the lower bound of α_τ and it can be estimated from historical data of prior solutions [104]. (4.32) is the Benders' optimality cut and it does not exist when $\theta = 1$ as the variables and parameters are initialized with zero. Once the first-stage problem is solved, the algorithm proceeds to step 1. $\rho_{il}^v(\lambda_y)$, $\bar{\rho}_{il}^v(\lambda_y)$, $\rho_{ik}^v(\lambda_y)$, and $\rho_{ik=x}^v(\lambda_y)$ are the dual value of constraints (4.37)-(4.40), respectively.

Step 1: Sub-problem solution. Step 1 formulate and solve multiple sub-problems. For each scenario λ_y, the sub-problem is formulated as follows:

$$\min_{\mu_i^{(p,\tau)}(\lambda_y),\bar{\mu}_{ix}^{(re,\tau)}(\lambda_y)} :$$

$$P(\lambda_y)\sum_{i\in\mathcal{I}}\Big(\mu_i^{(p,\tau)}(\lambda_y)c_i^p + \sum_{x\in\mathcal{X}}\bar{\mu}_{ix}^{(re,\tau)}(\lambda_y)\bar{c}\Big), \tag{4.33}$$

subject to: (4.25),

$$\sum_{l\in\mathcal{L}}\bar{\delta}_{il}^\tau\beta_l^i(\lambda_y)\sum_{w\in W_i}\alpha_{iw} + \sum_{k\in\mathcal{K}}\mu_{ik}^\tau\gamma_k(\lambda_y) + \sum_{x\in\mathcal{X}}\bar{\mu}_{ix}^\tau + \sum_{x\in\mathcal{X}}\bar{\mu}_{ix}^{(re,\tau)}(\lambda_y) \geq \sum_{w\in W_i}\alpha_{iw},$$
$$\forall i\in\mathcal{I},\forall\lambda_y\in\Lambda, \tag{4.34}$$

$$\sum_{x\in\mathcal{X}}\bar{\mu}_{ix}^{(re,\tau)}(\lambda_y) \leq \bar{A}\mu_i^{(p,\tau)}(\lambda_y), \qquad \forall i\in\mathcal{I},\forall\lambda_y\in\Lambda, \tag{4.35}$$
$$\sum_{i\in\mathcal{I}}\bar{\mu}_{ix}^{(re,\tau)}(\lambda_y) \leq 1, \qquad \forall x\in\mathcal{X},\forall\lambda_y\in\Lambda, \tag{4.36}$$

$$\delta_{il}^{\tau} = \delta_{il}^{(\tau,fixed)}, \qquad \forall i \in \mathcal{I}, \forall l \in \mathcal{L}, \tag{4.37}$$

$$\bar{\delta}_{il}^{\tau} = \bar{\delta}_{il}^{(\tau,fixed)}, \qquad \forall i \in \mathcal{I}, \forall l \in \mathcal{L}, \tag{4.38}$$

$$\mu_{ik}^{\tau} = \mu_{ik}^{\tau,fixed}, \qquad \forall i \in \mathcal{I}, \forall k \in \mathcal{K}, \tag{4.39}$$

$$\bar{\mu}_{ix}^{\tau} = \bar{\mu}_{ix}^{\tau,fixed}, \qquad \forall i \in \mathcal{I}, \forall x \in \mathcal{X}. \tag{4.40}$$

The objective function (4.33) and the constraints (4.34) - (4.36) are modified from (4.10), (4.12), (4.14), and (4.19) by adding an iteration counter τ. (4.37)-(4.40) is to fix the complicating variables with fixed variables.

Step 2: Convergence checking. The master problem is the lower bound and the sub-problem is the upper bound of the solution. Both of the bounds are checked at the end of each iteration. Let $z_v^{(lb)}$ and $z_v^{(ub)}$ denotes the lower and upper bounds in iteration v from (4.23) and (4.33), respectively. The Benders' decomposition algorithm stops when $|z_v^{(lb)} - z_v^{(ub)}| < \epsilon$, which ϵ is a small tolerance value. Otherwise, the algorithm proceeds to the next iteration.

Step 3: First-stage problem solution. In this step, the current iteration counter will be increased by 1. Then the master problem in (4.23) can be further relaxed by constraint (4.32). The solution to the master problem will also adjust α_τ, which is the on-demand cost. Once the master problem is solved, the algorithm proceeds to step 2 with the same iteration counter.

4.3.3 Sample-Average Approximation

The computation time of the master problem in (4.23) increases when more Benders' optimality cuts are added to the problem [189]. In this section, we use a different approach known as SAA to optimize stochastic problems when a large number of scenarios are used. We then compare the two methods in the performance evaluation section.

The SAA approach is about approximating the expected objective value utilizing sample scenarios Γ_N. $\Gamma_N = \{\lambda_1, \ldots, \lambda_n, \ldots, \lambda_N\}$ is the generated scenarios when the sample size is N. Instead of solving the problem using the true value, SAA approximates the true value by generating a sample of the stochastic parameters. This approach chooses a set of scenarios, e.g., N scenarios, where N is smaller than the total number of scenarios $|\Omega|$. The set of sample indices is represented by $\mathcal{N} = \{1, \ldots, n, \ldots, N\}$. This approach can obtain an optimal solution if N is large enough and thereby represents the scenarios adequately, where N can be verified numerically. We revisit the numerical validation of N in Section 4.5.

The optimization variables, are the elements of the set

$\Xi^{SAA} = \{\delta_{il}, \bar{\delta}_{il}, \mu_{ik}, \bar{\mu}_{ix}, \mu_i^p(\lambda_n), \bar{\mu}_{ix}^{re}(\lambda_n)\}$. The corresponding SAA objective function is expressed as follows:

$\min_{\Xi^{SAA}}$:

$$\sum_{i\in\mathcal{I}}\sum_{l\in\mathcal{L}}\Big(\delta_{il} + \bar{\delta}_{il}\underline{c}_{il}\Big) + \sum_{i\in\mathcal{I}}\Big(\sum_{k\in\mathcal{K}}\mu_{ik}c + \sum_{x\in\mathcal{X}}\bar{\mu}_{ix}\bar{c}\Big) + \frac{1}{N}\sum_{n\in\mathcal{N}}\sum_{i\in\mathcal{I}}\Big(\mu_i^p(\lambda_n)c_i^p + \sum_{x\in\mathcal{X}}\bar{\mu}_{ix}^{re}(\lambda_n)\bar{c}\Big), \tag{4.41}$$

subject to: (4.11)-(4.22)

The SAA computation will first generate an N sample scenario and then the stochastic problem is solved by M times to produce a set of M possible solutions. The value of N and M are chosen in a way that the number of scenarios is large enough while yet maintaining the computability of the problems. When the same solution is found in these problems, we can choose the solution as the desired solution.

4.3.4 The Benchmark Case: Deterministic Offloading Optimization

In the ideal case, when the actual charging efficiency and the status of the non-dedicated edge servers are precisely known ex-ante, all cells can choose the correct action, whether to charge by the BSs or offload the tasks to the non-dedicated or dedicated edge server. The correction action is not required to perform and therefore, the total correction cost will be zero. The optimization variables, are the elements of the set $\Xi^{DIP} = \{\delta_{il}, \bar{\delta}_{il}, \mu_{ik}, \bar{\mu}_{ix}\}$. The optimization problem can be formulated by the DIP model [189], which is expressed as follows:

$\min_{\Xi^{DIP}}$:

$$\sum_{i\in\mathcal{I}}\sum_{l\in\mathcal{L}}\delta_{il} + \sum_{i\in\mathcal{I}}\sum_{l\in\mathcal{L}}\bar{\delta}_{il}\underline{c}_{il} + \sum_{i\in\mathcal{I}}\sum_{k\in\mathcal{K}}\mu_{ik}c + \sum_{i\in\mathcal{I}}\sum_{x\in\mathcal{X}}\bar{\mu}_{ix}\bar{c} \tag{4.42}$$

subject to: (4.13), (4.15)-(4.18), (4.20), (4.22),

$$\sum_{l\in\mathcal{L}}\bar{\delta}_{il}\bar{\beta}_l^i\sum_{w\in W_i}\alpha_{iw} + \sum_{k\in\mathcal{K}}\bar{\gamma}_k\mu_{ik} + \sum_{x\in\mathcal{X}}\bar{\mu}_{ix} \geq \sum_{w\in W_i}\alpha_{iw}, \quad \forall i \in \mathcal{I}, \tag{4.43}$$

$$\delta_{il}, \mu_{ik}, \bar{\mu}_{ix}, \in \{0,1\}, \quad \forall i \in \mathcal{I}, \forall l \in \mathcal{L}, \forall k \in \mathcal{K}, \tag{4.44}$$

where (4.43) ensures that the cell chooses a computation method, i.e., compute locally, offload to the non-dedicated edge servers, offload to the dedicated edge servers or perform both actions when the actual charging efficiency $\bar{\beta}_l^i$ and actual non-dedicated edge servers computation status $\bar{\gamma}_k$ are known. (4.44) indicates that the variables are binary.

4.4 Case Study: Corrective Edge Offloading Decision Making Over Multiple Stages

The two-stage SIP is applicable when the cell has to complete its computation by the second stage. This means that the cell has to offload to a dedicated edge server at the second stage as it is unable to tolerate any uncertainty.

In contrast, we will study a special case whereby the cells are not urgent to implement their correction actions. For each stage after the first, the corrective action of re-offloading to the non-dedicated or dedicated edge servers can thus be performed. Specifically, the cheaper option of re-offloading to non-dedicated edge servers is made available due to the fact that the computation does not have to be completed by the second stage, and additional uncertainties can be tolerated.

We use a z-stage (multi-stage) SIP to formulate the decision of the cell in the z time steps. The uncertainty from (4.7) is added with a z symbol to indicate the scenario in each stage. The modified uncertainty in stage z is expressed as follows:

$$\lambda_y^z = \begin{bmatrix} F_1^{1,z} & F_1^{2,z} & \cdots & F_1^{I,z} \\ F_2^{1,z} & F_2^{2,z} & \cdots & F_2^{I,z} \\ \vdots & \vdots & \ddots & \vdots \\ F_L^{1,z} & F_L^{2,z} & \cdots & F_L^{I,z} \end{bmatrix}, \tag{4.45}$$

where $F_l^{i,z} = (\beta_l^{i,z}, \gamma_1^z, \ldots, \gamma_K^z)$. The set of scenarios are represented by $\Lambda^z = \{\lambda_1^z, \ldots, \lambda_y^z, \ldots, \lambda_Y^z\}$. Since the uncertainty of charging efficiency is only used in stage 2, we can drop the $\beta_l^{i,z}$ from $F_l^{i,z}$ from stage 3 onward to reduce the number of parameters. As shown in Fig. (4.4), the z-stages are as follows:

- *First stage:* In the first stage, the network makes the ex-ante decision between i) full local computation, and ii) partial local computation. This decision is made before the scenarios are realized and observed. Note that the scenarios refer to the combinations as described in Section 4.2.1.

- *Second stage:* In the second stage, the scenarios are realized and observed. In

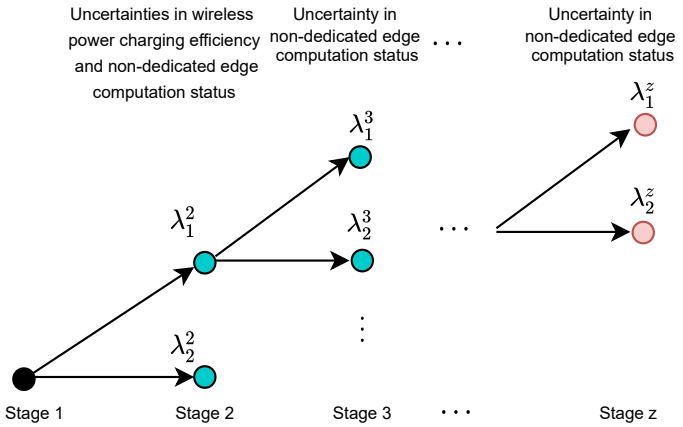


Figure 4.4: The scenario tree of a z-stage SIP.

other words, each user now knows the shortfall in computation. To account for this shortfall, the user makes the ex-post decision to offload the remaining incomplete computation subtasks to the dedicated edge servers or non-dedicated edge servers. Similarly, this decision is made before the second stage computation status of the non-dedicated edge servers can be realized.

- *Third stage:* In the third stage, the scenarios of the computation status of the non-dedicated edge servers from the second stage are realized and observed. If the computation is not complete (e.g., occupied non-dedicated edge servers in the second stage are used), the users are given an additional opportunity to offload to other or dedicated edge servers until the final stage z.

  ⋮

- *z-stage:* In the z-stage, the scenarios of the computation status of the non-dedicated edge servers from $(z-1)$-stage are realized and observed. If the computation is not complete, the users are given a final opportunity to offload to other dedicated or non-dedicated edge servers, and the users are penalized by a large penalty cost $\tilde{c}$.

In our problem formulation, the decision variables are as follows:

- $\mu_{ik}^{(2)}(\lambda_y^2) \in \{0,1\}$ indicates whether the sub-task is offloaded to the non-dedicated edge server k by cell i in stage two and scenario λ_y^2. Specifically, $\mu_{ik}^{(2)}(\lambda_y^2) = 1$ if cell i offload 1 sub-task to the edge server k.

- $\bar{\mu}_{ix}^{(2)}(\lambda_y^2) \in \{0,1\}$ indicates whether the sub-task is offloaded to the dedicated edge server x by cell i in stage two and scenario λ_y^2. Specifically, $\bar{\mu}_{ix}^{(2)}(\lambda_y^2) = 1$ if cell i offload 1 sub-task to the edge server x.

- $\mu_i^{(2,p)}(\lambda_y^2) \in \{0,1\}$ indicates the implementation of the penalty cost c_i^p in stage 2. When $\mu_i^{(2,p)}(\lambda_y^2) = 1$, cell i performs re-offloading action in stage 2 with an additional cost of c_i^p and $\mu_i^{(2,p)}(\lambda_y^2) = 0$ means otherwise.

$$\vdots$$

- $\bar{\mu}_{ix}^{(z)}(\lambda_y^2, \ldots, \lambda_y^z) \in \{0,1\}$ indicates whether the sub-task is re-offloaded to the dedicated edge server x by cell i in stage z.

- $\mu_{ik}^{(z)}(\lambda_y^2, \ldots, \lambda_y^z) \in \{0,1\}$ indicates whether the sub-task is re-offloaded to the non-dedicated edge server k by cell i in stage z.

- $\mu_i^{(z,p)}(\lambda_y^2, \ldots, \lambda_y^z) \in \{0,1\}$ indicates the implementation of the penalty cost c_i^p in stage z.

The objective function given in (4.46) - (4.49) is to minimize the total cost of the network. The expressions in (4.46), (4.47), (4.48) and (4.49) represent the first, second, third, and z stage objectives, respectively. $\mathbb{E}\Big[Q(\mu_{ik}^{(2)}(\lambda_y^2))\Big]$, $\mathbb{E}\Big[Q(\mu_{ik}^{(3)}(\lambda_y^2, \lambda_y^3))\Big]$, $\ldots$, $\mathbb{E}\Big[Q(\mu_{ik}^{(z)}(\lambda_y^2, \ldots, \lambda_y^z))\Big]$ are the expectations over scenario $\lambda_y^2 \in \Lambda^2, \ldots, \lambda_y^z \in \Lambda^z$. $\tilde{c}$ is an additional large penalty cost that occurs in the final stage z when the network still has to perform a corrective action. The optimization variables, are the elements of the set $\Xi^z = \{\delta_{il}, \bar{\delta}_{il}, \mu_{ik}, \bar{\mu}_{ix}, \mu_{ik}^{(2)}(\lambda_y^2), \bar{\mu}_{ix}^{(2)}(\lambda_y^2), \ldots, \mu_i^{(z,p)}(\lambda_y^2, \ldots, \lambda_y^z)\}$. The SIP formulation can be expressed as follows:

$$\min_{\Xi^z}:$$

$$\sum_{i\in\mathcal{I}}\sum_{l\in\mathcal{L}}\Big(\delta_{il} + \bar{\delta}_{il}\underline{c}_{il}\Big) + \sum_{i\in\mathcal{I}}\Big(\sum_{k\in\mathcal{K}}\mu_{ik}c + \sum_{x\in\mathcal{X}}\bar{\mu}_{ix}\bar{c}\Big) + \mathbb{E}\Big[Q(\mu_{ik}^{(2)}(\lambda_y^2))\Big], \tag{4.46}$$

where:

$$Q(\mu_{ik}^{(2)}(\lambda_y^2)) = \sum_{\lambda_y^2\in\Lambda^2} P(\lambda_y^2)\sum_{i\in\mathcal{I}}\Big(\sum_{k\in\mathcal{K}}\mu_{ik}^{(2)}(\lambda_y^2)c + \sum_{x\in\mathcal{X}}\bar{\mu}_{ix}^{(2)}(\lambda_y^2)\bar{c} + \mu_i^{(2,p)}(\lambda_y^2)c_i^p + \mathbb{E}\Big[Q(\mu_{ik}^{(3)}(\lambda_y^2, \lambda_y^3))\Big]\Big), \tag{4.47}$$

$$Q(\mu_{ik}^{(3)}(\lambda_y^2,\lambda_y^3)) = \sum_{\lambda_y^3\in\Lambda^3} P(\lambda_y^3)\sum_{i\in\mathcal{I}}\Big(\sum_{k\in\mathcal{K}} c\mu_{ik}^{(3)}(\lambda_y^2,\lambda_y^3) + \sum_{x\in\mathcal{X}}\bar{\mu}_{ix}^{(3)}(\lambda_y^2,\lambda_y^3)\bar{c} + \mu_i^{(3,p)}(\lambda_y^2,\lambda_y^3)c_i^p + \mathbb{E}\Big[Q(\mu_{ik}^{(4)}(\lambda_y^2,\lambda_y^3,\lambda_y^4))\Big]\Big), \tag{4.48}$$

$$\vdots$$

$$Q(\mu_{ik}^{(z)}(\lambda_y^2,\ldots,\lambda_y^z)) = \sum_{\lambda_y^z\in\Lambda^z} P(\lambda_y^z)\sum_{i\in\mathcal{I}}\Big(\sum_{k\in\mathcal{K}} c\mu_{ik}^{(z)}(\lambda_y^2,\ldots,\lambda_y^z) + \sum_{x\in\mathcal{X}}\bar{\mu}_{ix}^{(z)}(\lambda_y^2,\ldots,\lambda_y^z)\bar{c} + \mu_i^{(z,p)}(\lambda_y^2,\ldots,\lambda_y^z)\tilde{c}\Big), \tag{4.49}$$

subject to: (4.11), (4.13), (4.15) - (4.18), (4.20), (4.22),

$$\sum_{k\in\mathcal{K}}\mu_{ik}^{(2)}(\lambda_y^2)\gamma_k^2(\lambda_y^2)\sum_{x\in\mathcal{X}}\bar{\mu}_{ix}^{(2)}(\lambda_y^2) \geq \sum_{w\in W_i}\alpha_{iw} - \sum_{l\in\mathcal{L}}\bar{\delta}_{il}\beta_l^i(\lambda_y^2)\sum_{w\in W_i}\alpha_{iw} \quad \forall i\in\mathcal{I},\forall\lambda_y^2\in\Lambda^2, \tag{4.50}$$

$$\vdots$$

$$\sum_{k\in\mathcal{K}}\Big(\mu_{ik}^{(2)}(\lambda_y^2)\gamma_k^2(\lambda_y^2) + \mu_{ik}^{(3)}(\lambda_y^2,\lambda_y^3)\gamma_k^3(\lambda_y^2,\lambda_y^3) + \ldots + \mu_{ik}^{(z)}(\lambda_y^2,\ldots,\lambda_y^z)\Big) + \sum_{x\in\mathcal{X}}\Big(\bar{\mu}_{ix}^{(2)}(\lambda_y^2) + \bar{\mu}_{ix}^{(3)}(\lambda_y^2,\lambda_y^3) + \ldots + \bar{\mu}_{ix}^{(z)}(\lambda_y^2,\ldots,\lambda_y^z)\Big) \geq \sum_{w\in W_i}\alpha_{iw} - \sum_{l\in\mathcal{L}}\bar{\delta}_{il}\beta_l^i(\lambda_y^2)\sum_{w\in W_i}\alpha_{iw} \quad \forall i\in\mathcal{I},\forall\lambda_y^2\in\Lambda^2,\ldots,\forall\lambda_y^z\in\Lambda^z, \tag{4.51}$$

$$\sum_{k\in\mathcal{K}}\mu_{ik}^{(2)}(\lambda_y^2) + \sum_{x\in\mathcal{X}}\bar{\mu}_{ix}^{(2)}(\lambda_y^2) \leq \bar{A}\mu_i^{(2,p)}(\lambda_y^2), \quad \forall i\in\mathcal{I},\forall\lambda_y^2\in\Lambda^2, \tag{4.52}$$

$$\vdots$$

$$\sum_{k\in\mathcal{K}}\mu_{ik}^{(z)}(\lambda_y^2,\ldots,\lambda_y^z)+\sum_{x\in\mathcal{X}}\bar{\mu}_{ix}^{(z)}(\lambda_y^2,\ldots,\lambda_y^z)\leq \bar{A}\mu_i^{(z,p)}(\lambda_y^2,\ldots,\lambda_y^z), \quad \forall i\in\mathcal{I},\forall\lambda_y^2\in\Lambda^2,\ldots,\forall\lambda_y^z\in\Lambda^z, \tag{4.53}$$

$$\sum_{i\in\mathcal{I}}\mu_{ik}^{(2)}(\lambda_y^2)\leq 1, \qquad \forall k\in\mathcal{K},\forall\lambda_y^2\in\Lambda^2, \tag{4.54}$$

$$\vdots$$

$$\sum_{i\in\mathcal{I}}\mu_{ik}^{(z)}(\lambda_y^2,\ldots,\lambda_y^z)\leq 1, \quad \forall k\in\mathcal{K},\forall\lambda_y^2\in\Lambda^2,\ldots,\forall\lambda_y^z\in\Lambda^z, \tag{4.55}$$

$$\sum_{i\in\mathcal{I}}\bar{\mu}_{ix}^{(2)}(\lambda_y^2)\leq 1, \qquad \forall x\in\mathcal{X},\forall\lambda_y^2\in\Lambda^2, \tag{4.56}$$

$$\vdots$$

$$\sum_{i\in\mathcal{I}}\bar{\mu}_{ix}^{(z)}(\lambda_y^2,\ldots,\lambda_y^z)\leq 1, \quad \forall x\in\mathcal{X},\forall\lambda_y^2\in\Lambda^2,\ldots,\forall\lambda_y^z\in\Lambda^z, \tag{4.57}$$

$$\delta_{il},\bar{\mu}_{ix}^{(2)}(\lambda_y^2),\ldots,\bar{\mu}_{ix}^{(z)}(\lambda_y^2,\ldots,\lambda_y^z)\in\{0,1\}, \quad \forall i\in\mathcal{I},\forall l\in\mathcal{L},\forall k\in\mathcal{K},\forall\lambda_y^2\in\Lambda^2,\ldots,\forall\lambda_y^z\in\Lambda^z. \tag{4.58}$$

(4.50) and (4.51) perform the same function as that of (4.11) and (4.12). (4.52) and (4.53) are the same as (4.14) to ensure that the penalty cost is paid when the cell requires to perform re-offloading. (4.54) - (4.57) are the same as (4.17) - (4.19). (4.58) indicates that the variables are binary variables. Similarly, when the number of scenarios are large, Benders' decomposition and SAA are applicable in the multi-stage SIP.

To solve the SIP, we assume that the probability distribution of all scenarios in set Λ^2,..., Λ^z are known [162], then, the complexity of the problem increases exponentially when the total number of scenarios across all the stages increases [162, 163]. The optimization algorithm is processed on the user terminal with their respective edge devices such as laptops and the algorithm is repeated for every computing tasks. By comparing the power usage for running optimization algorithm with the amount of power used in computation, computation consumes way more power. For example, a single V100 GPU can consume between 250 and 300 watt [190], while the optimization algorithm consumes around 0.8 watt (measured using open hardware monitor). Therefore, the energy consumption for optimization algorithm is negligible.

4.5 Performance Evaluation

Table 4.2: System Simulation Parameter Values [2]

Parameter	Values
Phase shift of BS l, ϕ_l	$[0, 2\pi)$
Gain of BS l, G_l	12dB
Output power of BS l, P_l^o	10W
Cost of non-dedicated edge server c	\$1000
Cost of dedicated edge server $\bar{c}$	\$2000
Penalty cost c_i^p	\$500

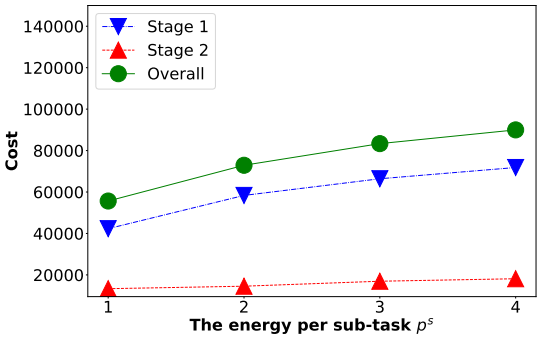


Figure 4.5: The cost of the network by varying p^s.

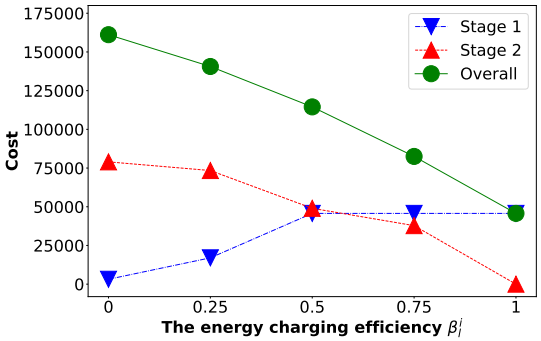


Figure 4.6: The cost of the network by varying β_l^i.

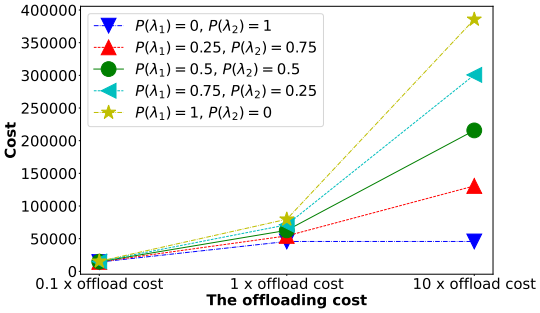


Figure 4.7: The cost of the network by varying the offloading cost.

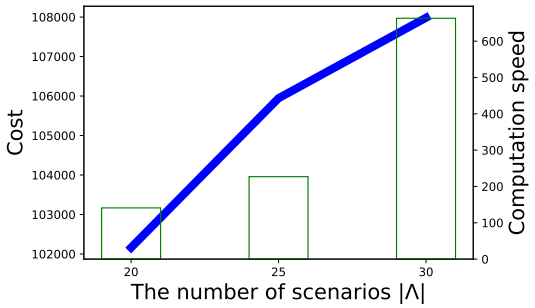


Figure 4.8: The number of scenarios required.

The extensive experiments are conducted with two hundred edge servers, three cells, and three BSs. For the presented experiments, we implement the SIP model using GAMS script [166, 167]. The simulation parameters are presented in Table 4.2. We first present the illustrative example of two scenarios. This enables us to understand the results when the parameters are varied. Then, we present the stochastic resource optimization when there are several (500) scenarios in more realistic network settings. Finally, we present the resource optimization over z-stages.

4.5.1 Illustrative Example of Two Scenarios

We vary α_{iw} randomly from 1 to 5 and set $p^s = 1$. Then, we consider two scenarios $|\Lambda| = 2$. The two scenarios are i) the wireless power charging efficiency from all the BSs is not efficient and the non-dedicated edge servers are fully occupied by other service providers λ_1, i.e., $\beta_l^i < 1$, and ii) all the wireless power charging efficiency are efficient and the non-dedicated edge servers are free λ_2, i.e., $\beta_l^i = 1$. The probabilities of each scenarios occurring are $P(\lambda_1) = 0.3$ and $P(\lambda_2) = 0.7$. We first use only scenarios λ_1 and λ_2 to perform the sensitivity test to evaluate the different parameters in a two-stage SIP.

Energy per sub-task p^s

We vary the energy per sub-task p^s and Fig. 4.5 shows the cost breakdown of the network. Among the three computation cost, $\underline{c}_{il}$, c and $\bar{c}$, the local computation is the cheapest. When $p^s = 1$, cells 1, 2, and 3 have sufficient energy to process most of the sub-tasks using three BSs. However, as the energy per sub-task increases, all the cells have to offload some sub-tasks to the non-dedicated edge servers. Therefore, the cost in stage 1 increases. The cost in stage 2 increases because when more sub-tasks are offloaded to the non-dedicated edge servers, more sub-tasks have to be re-offloaded whenever λ_1 occurs (i.e., the corrective edge offloading has to occur). Therefore, it is

clear that all cells will use local computation where possible, when it is not limited by the energy constraint in (4.20).

Wireless power charging efficiency β_l^i

We set $p^s = 0.5$, $P(\lambda_1) = 0.8$, $P(\lambda_2) = 0.2$, vary the wireless power charging efficiency β_l^i and keep the rest of the parameters same as Section 4.5.1. The simulation result is shown in Fig. 4.6. When the wireless power charging efficiency is low ($\beta_l^i = 0$) all the cells except 3 will offload the sub-tasks to the edge servers, which means that cells 1 and 2 perform zero local computation. The cost is cheaper for cell 3 to perform the local computation first using BS 1 and then perform the re-offloading action to the dedicated edge servers when efficiency is low. As the efficiency increases, more cells are willing to perform local computation in stage 1 and re-offload in stage 2 if the efficiency is low. Therefore, the cost in stage 1 increases as efficiency increases, and the cost in stage 2 decreases as efficiency increases.

Probability sensitivity

Similar to the setup in Section 4.5.1, we observe the impact on the solution when varying both the wireless power charging efficiency probability $P(\lambda_1)$ and offloading cost. The result is shown in Fig. 4.7. We observe that the probability does not affect the decisions of all the cells when the offloading cost is low. All the cells will offload to the edge servers due to the cheap cost. As the offloading cost increases, the cost differences between each probability become larger. However, all the cells will still choose local computation even if the probability of power charging efficiency is low as the offloading cost is too high. The cells will perform local computation in the first stage and then offload the rest of the sub-tasks to the edge server when the efficiency is low.

4.5.2 Stochastic Resource Optimization Under Several Scenarios

In the following, we consider when there are 500 scenarios.

Estimation of Number of Scenarios Required for SAA

We perform the simulation using the SAA approach to find the minimum number of scenarios required to best approximate the cost of the network. The simulations are performed $M = 2$ times. We first start the simulation by using 20 scenarios. The result is shown in Fig. 4.8. The bar chart that is shown below the line graph represents the

Table 4.3: Computation speed of SAA and Benders' decomposition in term of seconds.

	SAA	Benders' decomposition (500 scenarios)
20 Scenarios	141s	173s
25 Scenarios	227s	
30 Scenarios	663s	

computation speed in terms of seconds. The cost approximation depends on the sampled scenarios. When the number of scenarios are large enough, the approximated cost will be closer to the actual cost of the network [104]. However, when the number of scenarios increases, the complexity of the SAA solution increases. Therefore the computation time increases. In order to best approximate the network cost while balancing the computation cost/time, we choose to use 30 scenarios for the SAA approach unless otherwise stated for the rest of this section.

Convergence of Benders' Decomposition

Fig. 4.9 shows the bound convergence obtained by solving the Benders' decomposition algorithm. In initialization (iteration 0), the upper bound and the lower bounds are set at a very large positive and negative value, i.e., +10e10 and -10e-10, respectively. However, for better visualization, we limit the upper bound and the lower bound of the figure to 100000 and 40000 respectively, and we extend the simulation by a few more iterations even if the algorithm converges. In each iteration, the upper bound and the lower bound are adjusted and they converge at the second iteration. The Benders' decomposition breaks down the two-stage SIP into the master and sub-problem. We observe that the algorithm can handle a large number of scenarios, i.e., 500, as the sub-problems can be solved efficiently due to their smaller number of variables and parallelization.

Extraction of Scenarios

We extract 30 scenario samples from three different percentiles of the distribution, head (10^{th}), middle (50^{th}) and tail (90^{th}). The cost of the SAA is shown in Fig. 4.10. We can observe that the costs are different when the scenarios are extracted from different locations. This means that the cost from SAA is dependent on the sampled scenarios. Using the scenario generated from the middle portion of the distribution can lead to a high cost. However, the differences between them are not significant. The differences can be reduced when the scenarios are well sampled, as shown in the last (Mixed) column.

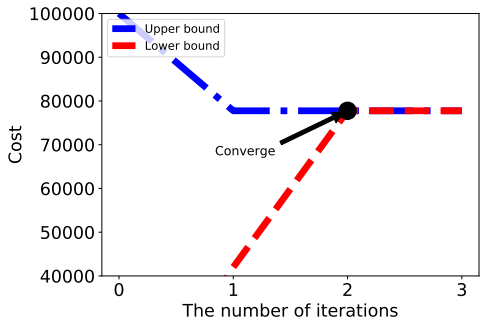


Figure 4.9: The convergence of Benders' decomposition.

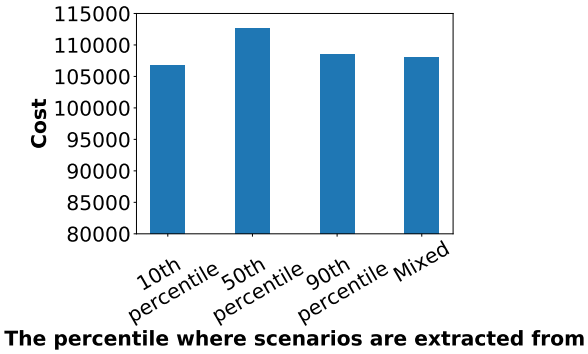


Figure 4.10: Comparing the cost of SAA by using scenarios from different percentiles.

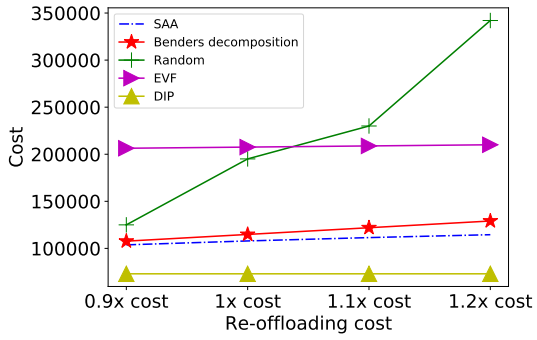


Figure 4.11: Comparing between SIP scheme vs random scheme vs EVF scheme vs DIP scheme.

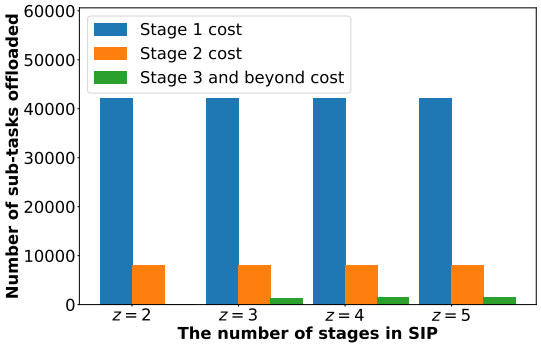


Figure 4.12: The network cost when varying the number of stages z.

Computation time comparison between SAA and Benders' Decomposition

We compare the computation time difference between SAA and Benders' decomposition. The result is shown in Table 4.3. The computation time of the SAA tends to increase as the number of scenarios increases. Compared to the SAA, Benders' decomposition can maintain a much lower computation speed of 173s while using a high number of scenarios (500).

Comparing different schemes

We compare the Benders' decomposition, SAA, DIP, expected-value formulation (EVF) [168] scheme, as well as the random scheme. The EVF uses the average value of the computation decision in stage 1 and solves an SAA with 30 scenarios. We use EVF as a fixed scheme and compare the results by varying the offloading price. For EVF, $\bar{\delta}_{il}$, μ_{ik} and $\bar{\mu}_{ix}$ are fixed using the average values of the computation decision from the historical records. The DIP is the ideal but unrealistic case in which we assume that the exact uncertainty of the wireless power charging efficiency and computation uncertainty for non-dedicated edge servers is exactly known. In a random scheme, the values of the decision variables are randomly generated. As shown in Fig. 4.11, both EVF and the random scheme derive the highest costs as they cannot adapt to the change in the corrective edge offloading cost. EVF is the highest when the costs are 0.9× and 1×. The random scheme is the highest when the costs are 1.1× and 1.2×. SAA which is an approximating solution, manage to achieve a relatively close cost as compared with the Benders' decomposition.

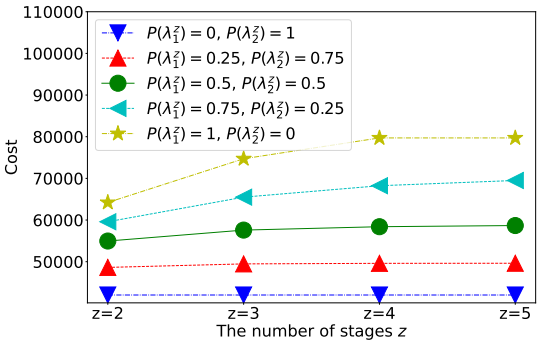


Figure 4.13: The network cost when varying the probability of uncertainty in each stage.

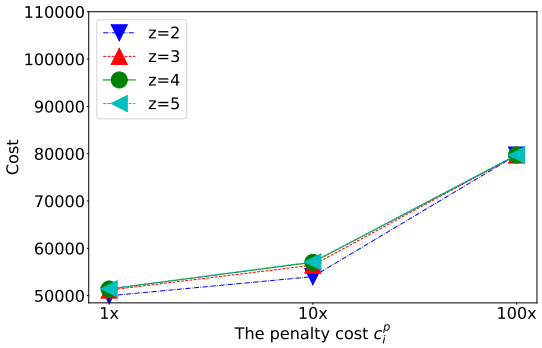


Figure 4.14: The network cost when varying the penalty cost c_i^p.

Table 4.4: Decision variable value

Variable	$z = 2$	$\mathbf{z = 3}$	$z = 4$	$z = 5$
δ_{11}	0	0	0	0
δ_{12}	0	0	0	0
δ_{13}	0	0	0	0
δ_{21}	1	**1**	1	1
δ_{22}	0	0	0	0
δ_{23}	0	0	0	0
δ_{31}	0	0	0	0
δ_{32}	0	0	0	0
δ_{33}	1	**1**	1	1
$\bar{\delta}_{11}$	0	0	0	0
$\bar{\delta}_{12}$	0	0	0	0
$\bar{\delta}_{13}$	0	0	0	0
$\bar{\delta}_{21}$	1	**1**	1	1
$\bar{\delta}_{22}$	0	0	0	0
$\bar{\delta}_{23}$	0	0	0	0
$\bar{\delta}_{31}$	0	0	0	0
$\bar{\delta}_{32}$	0	0	0	0
$\bar{\delta}_{33}$	1	**1**	1	1
$\sum_{k\in\mathcal{K}} \bar{\mu}^{(2)}_{1k}(\lambda^2_1)$	13	**13**	13	13
$\sum_{k\in\mathcal{K}} \bar{\mu}^{(2)}_{2k}(\lambda^2_1)$	6	**6**	6	6
$\sum_{k\in\mathcal{K}} \bar{\mu}^{(2)}_{3k}(\lambda^2_1)$	6	**6**	6	6
$\sum_{k\in\mathcal{K}} \bar{\mu}^{(3)}_{1k}(\lambda^2_1, \lambda^3_1)$	-	**6**	6	6
$\sum_{k\in\mathcal{K}} \bar{\mu}^{(3)}_{2k}(\lambda^2_1, \lambda^3_1)$	-	**3**	3	3
$\sum_{k\in\mathcal{K}} \bar{\mu}^{(3)}_{3k}(\lambda^2_1, \lambda^3_1)$	-	**3**	3	3
$\sum_{k\in\mathcal{K}} \bar{\mu}^{(4)}_{1k}(\lambda^2_1, \lambda^3_1, \lambda^4_1)$	-	-	3	6
$\sum_{k\in\mathcal{K}} \bar{\mu}^{(4)}_{2k}(\lambda^2_1, \lambda^3_1, \lambda^4_1)$	-	-	2	3
$\sum_{k\in\mathcal{K}} \bar{\mu}^{(4)}_{3k}(\lambda^2_1, \lambda^3_1, \lambda^4_1)$	-	-	2	3
$\sum_{k\in\mathcal{K}} \bar{\mu}^{(4)}_{1k}(\lambda^2_1, \ldots, \lambda^5_1)$	-	-	-	2
$\sum_{k\in\mathcal{K}} \bar{\mu}^{(4)}_{2k}(\lambda^2_1, \ldots, \lambda^5_1)$	-	-	-	1
$\sum_{k\in\mathcal{K}} \bar{\mu}^{(4)}_{3k}(\lambda^2_1, \ldots, \lambda^5_1)$	-	-	-	1

4.5.3 Resource optimization over z-stages

Next, we will perform the simulation on z-stage SIP.

Number of stages z

We first consider the case with two computation scenarios $|\Lambda^z| = 2$ for each of the SIP stages. The two scenarios are i) the wireless power charging from all the BSs is not efficient and some of the non-dedicated edge servers are occupied by other service providers λ_1^z, e.g., $\beta_l^i \neq 1$, and ii) all the wireless power charging are efficient and the non-dedicated edge servers are free λ_2^z, e.g, $\beta_l^i = 1$. The probabilities of uncertainty are $P(\lambda_1^z) = 0.3$ and $P(\lambda_2^z) = 0.7$. We vary the value of z from 2 to 5. Fig. 4.12 shows the cost breakdown, and the value of the decision variables are indicated in Table 4.4. The number of decision variables are large. Therefore, for illustration, Table 4.4 does not contain any decision variables that have a value of zero. Then, we use the result from $z = 3$ to explain the findings. In stage 1, Cells 2 and 3 choose BS 1 and 3, respectively, to perform the local computation as the local computation is cheaper than the offloading. While cell 2 chooses to offload to the non-dedicated edge servers. When the efficiency of local computation and the uncertainty in computation status of the non-dedicated edge servers is low, cells 1, 2, and 3 will then re-offload the remaining sub-tasks to the non-dedicated edge servers. Since $P(\lambda_1^z)$ is low at each SIP stage, it is cheaper for cells to re-offload the sub-tasks to non-dedicated edge servers than to dedicated edge servers. The cells can perform the correction action only when it is needed by offloading the sub-tasks to the non-dedicated edge servers.

Probability Sensitivity in z-stage

We observe the impact on the network by varying both the probabilities $P(\lambda_1^z)$ and $P(\lambda_2^z)$ and the number of stages z. The cost of the network is shown in Fig. 4.13. If there is no uncertainty $P(\lambda_1^z) = 0$, when z increases, the decision made by the cells remains the same. When $P(\lambda_1^z)$ increases, it means that the chances that the BS is not efficient and some of the non-dedicated edge servers are occupied by other service providers is high. Therefore, the cells will compute all the tasks in the earlier stage, i.e., stage one to avoid the high re-offloading cost.

Penalty Sensitivity

We consider a setup similar to Section 4.5.3. We observe the cost of the network by varying the penalty cost of the re-offloading. The cost of the network is shown in Fig. 4.14. When the penalty cost is 1× or 10×, all the cells will still perform

re-offloading as the correction actions. However, all the cells change their decision when the penalty cost is 100× as the correction action that happens at all the stages is higher than using the dedicated edge server in stage one.

4.6 Conclusion

In this chapter, we have introduced a stochastic resource optimization framework for the wireless powered hybrid coded edge computing network. Given the uncertainties of wireless charging efficiency and edge server availability, we have proposed a stochastic integer programming approach for resource optimization. As a result, we can obtain an optimal cost-minimizing computation strategy despite the fact that the uncertainties are only observed ex-post of making the computation decisions. For our future work, we will further incorporate other sources of uncertainties, such as malicious edge servers, into the problem formulation. This will further introduce additional properties of secure coded offloading into the problem formulation.

Chapter 5

Semantic Transmission selection

This chapter explores the integration of advanced technologies within the Metaverse, focusing particularly on the pivotal role of Artificial Intelligence (AI) in scaling virtual environments. The burgeoning demands of AI-driven Metaverse applications require substantial data to train machine learning models for interpreting and responding to diverse user inputs, from text and images to complex video interactions. Addressing the critical challenges of excessive energy consumption by data centers and the inadequacies of current 5G communications, this chapter proposes a shift towards 6G-enabled semantic communication systems. These systems aim to enhance the sustainable development of the Metaverse by minimizing data transmission sizes, thereby reducing energy usage and supporting more efficient virtual transactions. We will present a case study on developing a virtual transportation network within the Metaverse using real-world data from Singapore. The study examines the strategies of a virtual service provider (VSP) employing semantic data to create immersive user experiences while managing the dual challenges of energy efficiency and robust data handling through a two-phase Stochastic Semantic Resource Allocation (SSRA). This introductory approach, employing Distributed Robust Optimization (DRO) and Deep Reinforcement Learning (DRL)-driven auction mechanisms, ensures cost and energy minimization across uncertain network conditions, paving the way for a resilient and sustainable Metaverse.

5.1 Introduction

The Metaverse is expected to be built around seven key technologies: 6G communication systems, extended reality, brain-computer interfaces, cloud/edge computing, blockchain, digital twins, and artificial intelligence (AI) [191]. Out of those emerging technologies, AI is the most important one in the Metaverse puzzle because of its ability to enable the Metaverse to scale. The AI-enabled Metaverse applications require a tremendous amount of data to train machine learning models so that the Metaverse applications can understand user inputs ranging from text to images and even videos and respond appropriately regardless of the user's input languages. In the future, machine learning models in the Metaverse will eventually be used to generate 3D images, animations, speech, and support blockchain technologies that allow for virtual transactions. However, the current Metaverse face with certain limitations, i) the Metaverse facilities require incredible amounts of energy, i.e., data centers rely on AI and cloud services to store information [137]. ii) The current 5G communication system is insufficient to support the Metaverse, where millions or billions of devices are connected in the intersection between the virtual and physical world [138]. Given that current technologies are approaching the Shannon limit, traditional transmission methods are increasingly inadequate for meeting the rapidly growing and diversifying data traffic demands [139, 140].

In response, semantic communication enabled 6G future communication systems may contribute to the development of the sustainable Metaverse. In contrast to existing communication technologies, transmission in semantic communication is deemed effective if the received information maintains the same meaning as the transmitted information [192]. For instance, when a user requests an image, semantic communication systems reduce the transmitted data and transmit only the region in which the user is interested. As a result, the Metaverse energy consumption is reduced by reducing the transmitted data size.

In this chapter, we propose a case study on the development of a virtual transportation network in the Metaverse that utilizes self-collected real-world data from selected streets in Singapore [193]. We consider a virtual service provider (VSP) that is an entity that offers a virtual service within the Metaverse. Using data from the physical domain, such as weather conditions or images of geographical landmarks and vehicles, the VSP can develop the Metaverse with the objective to provide users with immersive experiences, such as for realistic test driving of vehicles or the safe training of autonomous vehicles subject to practical constraints. Typically, data captured from physical domains can be traded on data markets or retrieved via crowdsensing [194].

Specifically, devices at the network's periphery may sell semantic data after it is collected and processed from the geographic regions in which they are deployed. The VSPs can purchase the semantic data transmissions from the edge sensing units by using two semantic data subscription plans, i.e., the reservation and on-demand plans [195]. The reservation plan is the long-term plan to purchase the semantic data that has been previously collected, processed, and stored by edge sensing units. On the other hand, the on-demand plan is an ad-hoc plan triggered whenever the VSPs require additional semantic data from the edge sensing units, e.g., when the VSP receives user feedbacks for low quality of experience (QoE) due to outdated physical-virtual world synchronization. As a result, the on-demand plan depends on parameters such as the uncertainty of QoE feedback from users, the interest of VSPs in the type of data required, and the number of data points required. To further reduce the energy consumption of the Metaverse, the cost function of the edge sensing units is formulated based on their energy consumption. Therefore, when the cost of the network is minimized, the energy consumption of the network is also minimized.

To derive the cost-minimizing semantic data subscription strategies of the VSP, we introduce a two-phase stochastic semantic resource allocation (SSRA). A two-phase SSRA is needed to allocate the semantic resource before and after observing the demand. In phase one, we first derive the expected on-demand cost for each corresponding set of uncertainty parameters. With our derived on-demand cost structure over the various uncertain scenarios, we formulate and solve for the subscription strategies of the VSPs as a distributed robust optimization (DRO) problem. The two phases are as follows.

- **Phase One (On-demand cost derivation):** Utilizing a single parameter set for each uncertainty scenario, we introduce a DRL-driven double dutch auction for one-to-one matching and price determination between VSPs and edge sensing units, thereby deriving the on-demand cost.
- **Phase Two (DRO for Subscription Strategy):** Addressing demand uncertainties in the Metaverse's cost minimization problem, we initially employ Stochastic Integer Programming (SIP) with historically-based random variables. Recognizing the limitations of SIP due to unpredictable distributions, we pivot to a more conservative, data-driven Distributed Robust Optimization (DRO) strategy. DRO constructs a modifiable confidence set from historical data to ensure network robustness, even when the true underlying distribution is unclear.

The contributions of this chapter are summarized as follows.

- For the phase one of the SSRA scheme, our proposed DRL-driven-double dutch auction mechanism is tractable, dynamic, and adaptive to the stochastic network

conditions. This enables us to devise the cost/energy consumption minimization strategy at scale. In addition, together with semantic communication enabled, a sustainable Metaverse can be realized.

- In the phase two of the SSRA scheme, instead of assuming the historical observation data is the true distribution, we construct a confidence set using the reference distribution derived from historical observation data and distance metrics. Our proposed DRO cost minimization formulation is able to mitigate the negative impacts of inaccurate uncertainty modeling of user demands in complex networks.
- Using our self-collected real-world data, we compare SSRA with other baselines, such as deterministic, random, and stochastic integer programming. From the performance evaluation, we demonstrate that SSRA can reduce significant amount of energy and achieve better outcomes relative to other baselines for extreme conditions, highlighting the practicality of our proposed scheme when applied to realistic scenarios.

The remainder of the chapter is organized as follows: In Section 5.2, we present the system model. In Sections 5.3 and 5.4 we formulate the problem. We discuss and analyze the simulation result in Section 5.5. Section 5.6 concludes the chapter.

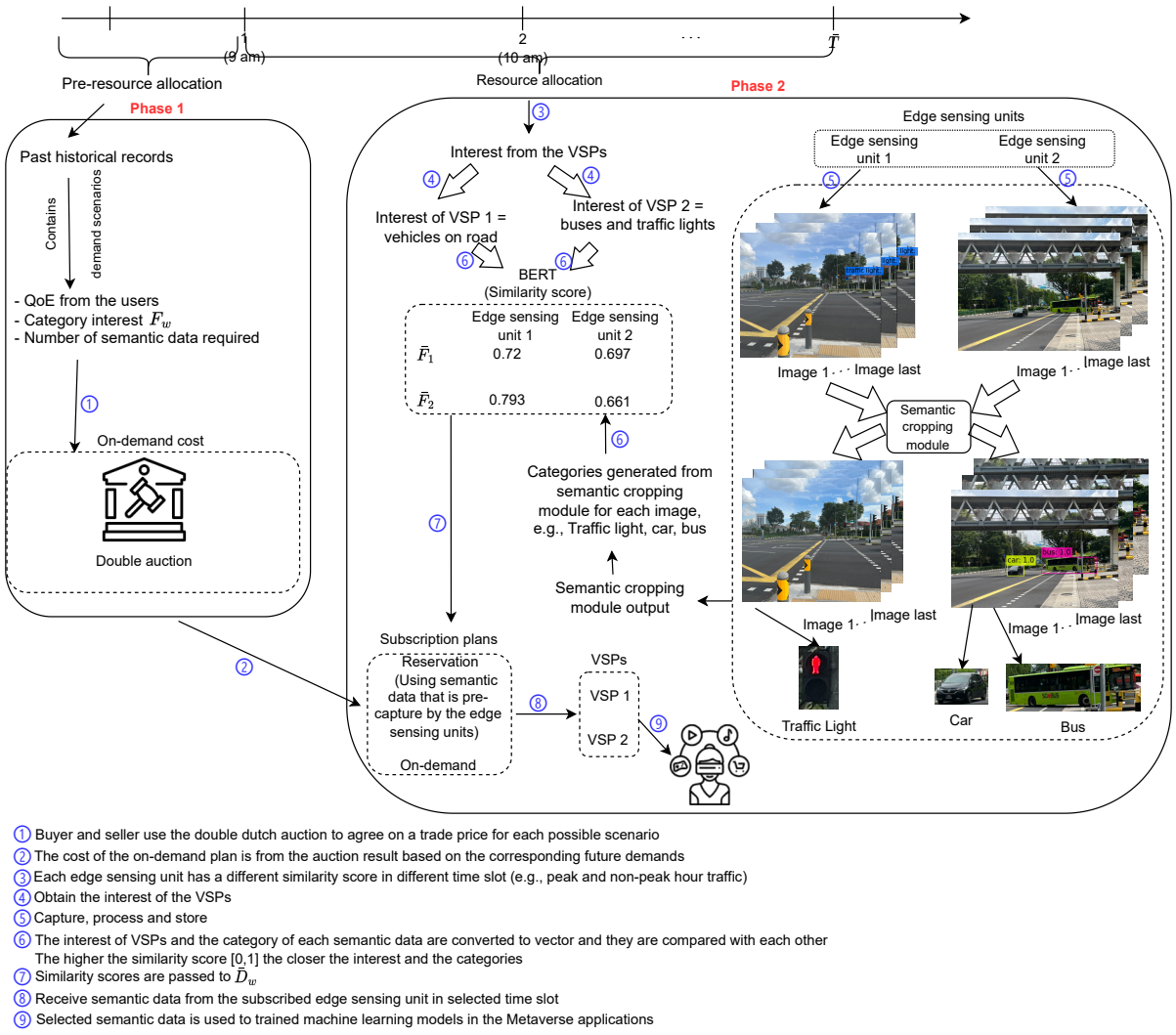
5.2 System Model

We consider a Metaverse network shown in Fig. 5.1 and Fig. 5.2 shows the network topology. To develop the Metaverse, a set $\mathcal{W} = \{1, \ldots, w, \ldots, W\}$ of W VSPs subscribe to the semantic data provision plans offered by edge sensing units denoted by the set $\mathcal{E} = \{1, \ldots, e, \ldots, E\}$.

5.2.1 Semantic Market Data Provision Plans

The VSPs subscribe to the edge sensing units using two types of plans as follows:

- **Reservation plan:** This is a long-term plan in which the VSP w pays a *fixed* cost $c^r_{w,e}$ for n_e semantic data that has been previously collected and stored by edge sensing unit e. The fixed cost $c^r_{w,e}$ is dependent on the data collection, processing, and communication costs defined in Section 5.2.2. As each edge sensing unit e has a different field of view at time slot $\bar{t}$, marked by coordinates $(x_e(\bar{t}), y_e(\bar{t}))$, the images that have been captured differ. The VSP chooses the reservation plan provided by an edge sensing unit that has the highest category similarity to its



① Buyer and seller use the double dutch auction to agree on a trade price for each possible scenario
② The cost of the on-demand plan is from the auction result based on the corresponding future demands
③ Each edge sensing unit has a different similarity score in different time slot (e.g., peak and non-peak hour traffic)
④ Obtain the interest of the VSPs
⑤ Capture, process and store
⑥ The interest of VSPs and the category of each semantic data are converted to vector and they are compared with each other
The higher the similarity score [0,1] the closer the interest and the categories
⑦ Similarity scores are passed to $\bar{D}_w$
⑧ Receive semantic data from the subscribed edge sensing unit in selected time slot
⑨ Selected semantic data is used to trained machine learning models in the Metaverse applications

Figure 5.1: An illustrative example of the network with two VSPs and two edge sensing units. A detailed illustration of the double dutch auction mechanism can be found in Fig.5.4

category interests, i.e., the VSP subscribes to the edge sensing unit that is able to transmit the semantic data that best satisfies its interest. Note that this is possibly a *many-to-one* matching between VSPs and the edge sensing unit. In other words, the edge sensing unit is able to sell its previously collected semantic data to as many interested VSPs as possible. The category similarity is further defined in Section 5.2.3.

- **On-demand plan:** This is an ad-hoc plan triggered when the VSP requests for additional semantic data to be newly collected by the edge sensing unit e at time slot $\bar{t}$ for the on-demand *variable* cost $c^{o}_{w,e}(\lambda_i)$. λ_i denotes the demand scenario defined in (5.1). The ad-hoc plan is triggered when the semantic data from the reservation plan is insufficient to achieve a defined level of user QoE. We further

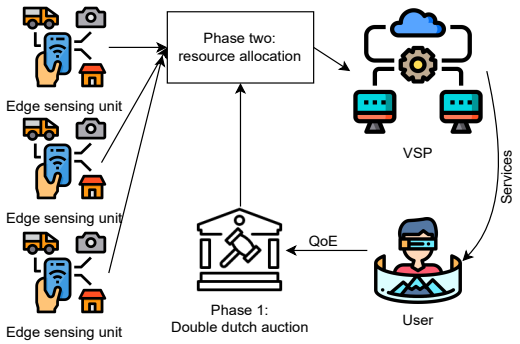


Figure 5.2: An illustrative example of the network topology.

define the QoE score in Section 5.2.4. Unlike the reservation plan which is a fixed cost, the cost of the on-demand plan depends on the demand scenario i of all VSPs. Due to the unpredictability nature of such demands and *one-to-one* matching between VSP and edge sensing unit, more than one VSPs may compete for the on-demand sensing service of the edge sensing unit. As such, we devise a DRL-based double dutch auction as presented in Section 5.3 to efficiently match the ad-hoc demand of the VSP to the limited supply of available edge sensing units.

In general, the reservation plan is cheaper than the on-demand plan in data markets [168], e.g., since additional data has to be collected on-demand within a short notice period. However, the on-demand plan has to be triggered due to scenario uncertainty. Let λ_i denote the i^{th} scenario of all the VSPs:

$$\lambda_i = (\bar{D}_1, \ldots, \bar{D}_W), \tag{5.1}$$

where $\bar{D}_w = (\gamma_{w,e}^{ave}, F_w, \bar{F}_w)$, $\gamma_{w,e}^{ave}$ is the average QoE score of the users who are accessing the Metaverse application developed by VSP w using semantic data from edge sensing unit e, $\bar{F}_W$ represents the set of category interest of VSP W, and F_W represents the number of semantic data transmissions that VSP W requires. Each scenario λ_i occurs with a probability of $P(\lambda_i)$.

5.2.2 Sensing Model

At its respective location, each edge sensing unit can sense (capture) images and process them to generate semantic data. The energy consumption for data collection can be expressed as follows [196]:

$$\bar{E}_e = (P_e^{idle}\hat{T}_e^{idle} + P_e^{active}\hat{T}_e^{active}), \tag{5.2}$$

where P_{idle} is the power consumption in idle state, P_{active} is the power consumption in active state, $\hat{T}_{idle}$ is the time in idle state, and $\hat{T}_{active}$ is the time in active state. $\hat{T}_{active} = \frac{d}{\tau_e}$, where d is the number of pixels in an image and τ_e denotes the total CPU computing capability of edge sensing unit e. When the captured data is processed locally, the local computation execution time of edge sensing unit e is expressed as [152]:

$$\hat{t}_e = \frac{C_e d}{\tau_e}, \tag{5.3}$$

where C_e is the number of CPU cycles needed to process a bit.

When the semantic data is requested by the VSP, it is processed and transmitted through base station b which is located at (x_b, y_b, z_b). In an urban area, many obstacles surround the edge sensing units and prevent the units from maintaining a line of sight (LoS) transmission link with the BS. In order to identify whether a transmission link is LoS or Non-LoS (NLoS), the height of the obstacles has to be known, which is difficult in practice. Therefore, we adopt a probabilistic LoS channel model [197]. The LoS transmission probability from edge sensing unit e to BS b is defined as follows [4]:

$$P_{e,b}^{L} = B_3 + \frac{B_4}{1 + e^{-(B_1 + B_2\theta_{e,b})}}, \tag{5.4}$$

where $B_1 < 0$, $B_2 > 0$, $B_4 > 0$, and B_3 are the constants with $B_3 + B_4 = 1$. The probability of NLoS $P_{e,b}^{N}$ can be obtained as $1 - P_{e,b}^{L}$. $\theta_{e,b}$ is the elevation angle from e to b and is defined as:

$$\theta_{e,b} = \arctan(\frac{z_b}{d_{e,b}}), \tag{5.5}$$

where z_b is the height of BS b and $d_{e,b}$ is the horizontal distance between the edge sensing unit e and BS b. Then, the channel power gain between the edge sensing unit e and BS b can be modeled as:

$$h_{e,b} = P_{e,b}^{L}\beta_0 d_{e,b}^{-\alpha_L} + (1 - P_{e,b}^{L})\mu\beta_0 d_{e,b}^{-\alpha_N}, \tag{5.6}$$

where β_0 is the average channel power gain at a reference distance of 1m, μ is the signal attenuation factor due to the NLoS propagation, α_L and α_N denote the average path loss exponents for the LoS and NLoS, respectively and $d_{e,b}$ is the distance between edge sensing unit e and BS b. To lighten the burden of the wireless network, we follow [198] to consider an orthogonal frequency-division multiple access (OFDMA) system to enable multiple concurrent uplinks and downlinks

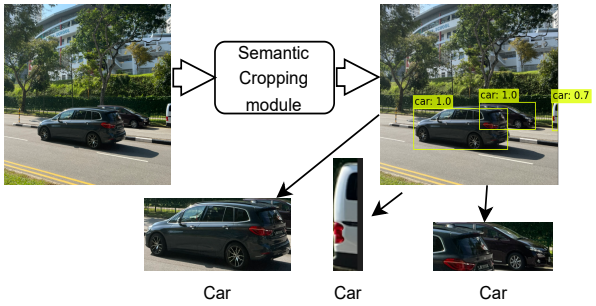


Figure 5.3: An illustrative example of semantic cropping module (YOLO) output.

communication. Each orthogonal resource block can be occupied only by at most one transmission so that the interference caused by other edge sensing units can be avoided. The uplink bitrate for each edge sensing unit is given as follows:

$$r_e = B_e\Big(\log_2(1 + \frac{\hat{P}_e h_{e,b}}{\sigma^2 \Gamma})\Big), \tag{5.7}$$

where B_e denotes the allocated bandwidth of edge sensing unit e, $\hat{P}_e$ refers to the transmit power of edge sensing unit e, σ^2 denotes the receiver noise power and Γ denotes the signal-to-noise ratio (SNR) gap.

Accordingly, the reservation cost $c^r_{w,e}$ is defined as follows:

$$c^r_{w,e} = n_e \delta_1 (\frac{\hat{P}_e \bar{q}}{r_e} + \bar{E}_e + \hat{t}_e) + \hat{F}_{w,e}, \tag{5.8}$$

where $\bar{q}$ is the average transmitted data size, δ_1 is the cost coefficient to convert energy consumption into cost, and $\hat{F}$ is the profit margin of the edge sensing unit, e.g., a fixed rental cost for its services. Similarly, the minimum on-demand cost (i.e., bid floor) is defined as follows to preserve the individual rationality of the auction:

$$\min(c^o_{w,e}) = \delta_2 (\frac{\hat{P}_e \bar{q}}{r_e} + \bar{E}_e + \hat{t}_e), \tag{5.9}$$

where δ_2 is the cost coefficient and it should be chosen in such a way that satisfies $\frac{c^r_{w,e}}{n} \leq \min(c^o_{w,e})$ as the cost of reservation plan $\leq$ the cost of the on-demand plan. The cost of the on-demand plan takes reference from the auction result by using the corresponding future demands. The actual on-demand cost is obtained from the double dutch auction output in Section 5.3.

5.2.3 Category Generation and Similarity Matching

Using the plans, the VSPs can obtain semantic data captured by the edge sensing units. However, it is difficult to identify which edge sensing unit produces images that are important or relevant to the interests of the VSPs. This chapter adopts a pre-trained machine learning model, you only look once (YOLO) from [199]. YOLO provides real-time detection with relatively high accuracy. With the help of YOLO, objects (semantic data) and their corresponding categories can be extracted from the respective images. Figure 5.3 shows an example of semantic cropping module (YOLO) output. When the category (label of the object) is within the interest of the VSPs, the semantic data, which is the snapshot of the object, can be transmitted.

However, once the categories are generated from the images, the VSPs cannot identify to which edge sensing unit to subscribe to as the VSPs do not know which semantic data is relevant to their interest. In additional, each image may output multiple semantic data and corresponding categories. It is not practical for the VSPs to search manually through the categories. Therefore, we propose to use category similarity for the VSPs to subscribe to the edge sensing unit that produces the best semantic data that meets the interest of the VSPs. Category similarity convert the interest of VSPs and the category of the semantic data into vectors so that they can be compared. However, in different contexts, the same word might have multiple definitions. For example, "wind" can mean the current of air or the action turn. The traditional method, such as *word2vec* cannot recognize polysemy. The issue arises when the same word cannot be represented by the same numerical vector in different contexts. One of the solutions is to use BERT [128]. BERT is a powerful pre-trained machine learning model that has been trained by billions of sentences for extracting semantic information. It is used to convert words into vectors according to different contexts. In this chapter, we use cosine similarity [128] to measure the similarity between the sentences, i.e.,

$$\text{match}(\mathbf{A},\mathbf{B}) = \frac{\mathbf{A}\cdot\mathbf{B}}{||\mathbf{A}||||\mathbf{B}||}, \tag{5.10}$$

where $\mathbf{A}$ is the vectorized of $\bar{F}_w$ which is the interest of VSP w and $\mathbf{B}$ is the vectorized categories output generated from BERT. The category similarity defined in (5.10) is a number between 0 and 1, which indicates how similar $\mathbf{A}$ is to $\mathbf{B}$, with 1 representing the highest similarity and 0 representing no similarity. The average similarity of VSP w's demands and edge sensing unit e's data types is represented by $S_{w,e}$ and can be calculated from $S_{w,e} = \frac{\sum_{i=1}^{y_2}\sum_{j=1}^{y_1}\text{match}(\mathbf{A},\mathbf{B}_{ij})}{y_2}$, where y_1 is the total number of semantic data in an image and y_2 is the total number of images in edge sensing unit e. As a result, the VSPs can

subscribe to receive the semantic data from the edge sensing unit that has the highest similarity score. $S_{w,e}$ prevents the VSP from over- and under-semantic data transmission. For example, the reservation plan from edge sensing unit 1 supports VSP 1 with 150 semantic data transmissions. After the interest and the demand (requires 300 semantic data, e.g., image segments) of VSP 1 is realized, the average similarity score $S_{1,1} = 0.8$. VSP 1 faces a shortfall of 180 as $300 - 0.8(150) = 180$. An additional on-demand plan is used to accommodate the shortfall of 225 semantic data, $\frac{180}{0.8}$. Therefore, a resource allocation scheme is required to prevent re-transmission and reduce the network's energy consumption.

5.2.4 User Experience Metrics

VSPs provide users with immersive virtual environment experiences by using AR/VR devices. However, in the current virtual environment technology, some users exhibit cybersickness symptoms. There are numerous reasons that can contribute to cyber sicknesses, such as gender, age, illness, display, and technology issues [200]. Unfortunately, due to human complexity, it is difficult to tackle the cybersickness problem that is caused by individual factors. However, we can reduce the cybersickness effects if it is caused by display and technology, i.e., improve the rendering of the virtual environment. For example, when the lag occurs due to processing delays (insufficient data to render the 3D models), there is a delay between an individual's action and the system's reaction, and this can contribute to cybersickness symptoms [201]. By synthesizing images with neural networks, real-time rendering of 3D virtual environments can be achieved with low processing delays [202]. After the QoE from the users is known, more semantic data that are within the interest of VSPs can be requested from the edge sensing units to improve the QoE. In this chapter, we use cybersickness to evaluate QoE [3]. Specifically, QoE can be quantified using parameters from the Simulator Sickness Questionnaire (SSQ) [203].

The SSQ provides a comprehensive evaluation of a user's experience by capturing various symptoms of simulator sickness, such as nausea, oculomotor issues, and disorientation. This allows for a more nuanced understanding of the user's experience than other simpler metrics. Comparing with other metrics such as Weber-Fechner Law which represented by a logarithm function. SSQ directly captures the user's subjective experience. This is crucial for Metaverse applications where the user's comfort and well-being are primary concerns. As shown in Table 5.1, SSQ consists of 16 parameters categorized into nausea (N), oculomotor (O), and disorientation (D). Each symptom is rated using a 4-point scale (0; no sickness, 1; mild sickness, 2;

considerable sickness, 3; severe sickness). The score of each category is calculated by adding all the symptom ratings that fall under the same category. The total score is then measured by combining every single score for each category with the weight. The value of each SSQ symptom is between 0 and 4. The average SSQ score received by VSP w from its users for the nausea category can be calculated as [3]:

$$\gamma_w^{Nau} = \frac{\sum_{u=1}^{U_w}(s_u^{gd} + s_u^{sw} + s_u^{is} + s_u^{dc} + s_u^{na} + s_u^{bu} + s_u^{sa})}{U_w}, \tag{5.11}$$

where s_u^{gd}, s_u^{sw}, s_u^{is}, s_u^{dc}, s_u^{na}, s_u^{bu}, and s_u^{sa} are the ratings for general discomfort, sweating, increased salivation, difficulty concentrating, nausea, burping, and stomach awareness, respectively. U_w represents the total number of users that use the Metaverse application provided by VSP w. The average oculomotor score received by VSP w from its users can be calculated as:

$$\gamma_w^{Ocu} = \frac{\sum_{u=1}^{U_w}(s_u^{gd} + s_u^{he} + s_u^{fa} + s_u^{es} + s_u^{dc} + s_u^{df} + s_u^{bv})}{U_w}, \tag{5.12}$$

where s_u^{he}, s_u^{fa}, s_u^{es}, s_u^{df}, and s_u^{bv} are the rating for headache, fatigue, eyestrain, difficulty focusing, and blurred vision, respectively. The average disorientation score received by VSP w from its users can be calculated as:

$$\gamma_w^{Dis} = \frac{\sum_{u=1}^{U_w}(s_u^{df} + s_u^{na} + s_u^{fh} + s_u^{bv} + s_u^{dzc} + s_u^{ve} + s_u^{dzo})}{U_w}, \tag{5.13}$$

where s_u^{fh}, s_u^{dzc}, s_u^{ve}, and s_u^{dzo} are the rating for the fullness of head, dizzy (eye close), vertigo, and dizzy (eye open), respectively.

To account for the QoE, the total average SSQ score is integrated with the semantic data that a VSP has:

$$\gamma_{w,e}^{ave} = \frac{1}{v_{w,e}}(v_1\gamma_w^{Nau} + v_2\gamma_w^{Ocu} + v_3\gamma_w^{Dis}), \tag{5.14}$$

$$\text{where } v_{w,e} = \log(1 + S_{w,e}F_w), \tag{5.15}$$

v_1, v_2, and v_3 are the weights for γ_w^{Nau} γ_w^{Ocu}, and γ_w^{Dis} respectively. The variable $v_{w,e}$ is introduced in the formula to modulate the weight of the average SSQ scores based on the semantic data available to a Virtual Service Provider (VSP). Essentially, it adjusts the impact of user Quality of Experience (QoE) scores (nausea, oculomotor, disorientation) by considering the volume and significance of semantic data a VSP utilizes to improve user experience. The use of a logarithmic function to define $v_{w,e}$ helps in capturing

Table 5.1: 16 symptoms of cybersickness in SSQ [3]

SSQ Symptoms	N	O	D
General discomfort	1	1	
Fatigue		1	
Headache		1	
Eye strain		1	
Difficulty focusing		1	1
Increased salivation	1		
Sweating	1		
Nausea	1		1
Difficulty concentrating	1	1	
Fullness of head			1
Blurred vision		1	1
Dizzy (Eye open)			1
Dizzy (Eye closed)			1
Vertigo			1
Stomach awareness	1		
Burping	1		

the diminishing returns effect — as more semantic data is added, its marginal utility decreases, reflecting a more realistic impact on improving QoE. This mechanism ensures that the VSPs' scores are adjusted based on both the discomfort scores and the richness of the data they employ to mitigate these discomforts. The relationship is constructed by using a logarithmic function following [204, 205].

5.2.5 Problem Formulation

To minimize the cost of the network, we introduce a two-phase optimization approach.

- **Phase one - On demand cost derivation:** For each scenario λ_i, the similarity score, SSQ score, and average transmitted semantic data size differ across the VSPs. Accordingly, the on-demand price fluctuates based on these factors, since the VSPs are facing different extents of shortfalls that affect their bidding decisions for the semantic data. We use the double dutch auction to obtain the distribution of on-demand cost paid by the VSPs to the owners of edge sensing units across the different scenarios.

- **Phase two - Distributed robust resource allocation:** In phase two, the VSPs will decide on the subscription plans to use, i.e., reservation or on-demand. The decision is made depending on the uncertainty of the demands and on-demand cost. Whenever there is insufficient data, the VSPs are allowed to purchase more

semantic data with the use of an on-demand plan. We first formulate and optimize the problem using two-stage SIP. SIP uses historical data as the true distribution to model the uncertainties. However, the use of historical data to estimate real-world uncertainties is difficult and inaccurate. DRO is used to characterize the uncertainty of probability distribution of demands by constructing the confidence set, which is actually an alias of the uncertainty set and consists of a family of ambiguous probability distributions. With the help of observed historical data, which contains i) semantic interest from the VSPs, ii) the number of semantic data required, iii) QoE from the users who are accessing the Metaverse applications, iv) semantic data category from the edge sensing units, and v) the average transmitted semantic data size, the confidence set can be adjusted with a guaranteed confidence level. Then, resource allocation decisions can be made by using the constructed ambiguous probability distribution.

5.3 Phase One: On-demand Cost Derivation

In the pre-resource allocation stage, by using historical data, the VSPs and the edge sensing units can open a call market to place orders to buy or sell semantic data at certain bid prices and match at predetermined time intervals. The VSPs will buy semantic data transmission from the owners of the edge sensing units in this call market. Therefore, the VSPs are the buyer, owners of the edge sensing units are the sellers, and they can be supervised by an auctioneer such as the network operator or administrator. We design a DRL-based double dutch auction mechanism to determine the buying price from the VSPs and the selling prices from the owners of the edge sensing units. The price that both the seller and buyer agree with each other is the on-demand cost. The detailed explanation of the different roles in the call market can be found below:

- **Buyer:** At the start of the auction, the buyer will submit their buy-bid to the Metaverse administrator. The buy-bid of VSP w at each time step t is denoted as $k^t_{w,e}$. This is the maximum price that VSP w is willing to pay to the edge sensing unit, and it is defined as follows [206]:

$$k^t_{w,e} = \delta_0 \gamma^{ave}_{w,e}, \tag{5.16}$$

where δ_0 is the cost coefficient. When the $\gamma^{ave}_{w,e}$ score is high, it means that the users are faced with severe simulator sickness and have low QoE. Therefore, VSP w is willing to pay more to improve its application.

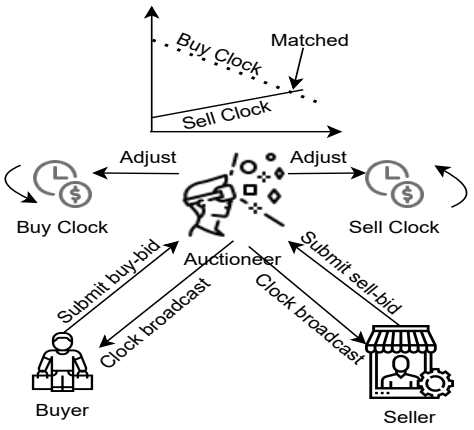


Figure 5.4: An illustrative example of the double dutch auction mechanism.

- **Seller:** Since the edge sensing unit is providing the communication resource to the buyer, the selling bid of the edge sensing unit should depend on the transmission energy consumption. This is the minimum price that the seller is willing to sell to prevent a loss in profit. At the start of the auction, the seller submits their sell-bid o_e^t to the Metaverse administrator, and it is defined as follows [206]:

$$o_e^t = \frac{\hat{P}_e \bar{q} \delta_1}{r_e}, \tag{5.17}$$

  where $\bar{q}$ is the average transmitted data size and δ_1 is the cost coefficient with a similar role to δ_0.

- **Auctioneer:** The auctioneer manages clocks, denoted as $C_{B,w,e}^t$ and $C_{S,e}^t$ for the buyer side and the seller side, respectively. $C_{B,w,e}^t$ is the buying clock that displays the buying price for buyer w to seller e. It descends over time and displays the buying price at each step. $C_{S,e}^t$ is the selling clock, and it ascends and displays the selling price for the seller e at each step. At the beginning of the auction, $C_{B,w,e}^0$ opens at the highest price p^{max}, and $C_{S,e}^0$ opens at the lowest price p^{min}.

5.3.1 Double Dutch Auction Mechanism Actions

An illustrative example of the double dutch auction mechanism is shown in Fig. 5.4. Generally, the buyers in a double dutch auction begin to count at a very high price and continue to count downward. In contrast, the sellers begin to ascend from a very low price and continue to ascend until a seller stops it (who offers the product at that price). When the buyer and seller do not match at the trade price, the buyer clock resumes its downward movement, followed by the seller clock's upward movement. Finally, the

trading is over when the two prices cross (purchase is made at the crossover point) [207, 208]. In the double dutch auction mechanism, there are mainly three types of actions, i.e., clock broadcast, clock acceptance, and clock adjustment.

- **Clock broadcast**: The double dutch mechanism begins with a clock broadcast. The auctioneer will broadcast the current buying and selling clocks to the buyers or the sellers at the start of each iteration.

- **Clock acceptance**: After receiving the auction clock from the Metaverse administrator, the buyers or the sellers will compare their bids with the current clocks. The buyers or the sellers will bid if the current clock meets the buy-bid or sell-bid conditions. For the buyers, the w^{th} buyer who is not in $\mathcal{M}_B^t$ can submit its buy-bid $b_{w,e}^t$, which represents the maximum price that a buyer is willing to pay and $b_w^t = C_{B,w,e}^t$. $\mathcal{M}_B^t$ is a set that contains all the buyers who are matched with a seller, and $\mathcal{M}_S^t$ is a set that contains all the sellers who are matched with a buyer. Similar to the buyers, the sellers who have not participated in the trade will decide after receiving the broadcasted selling clock from the Metaverse administrator. The sellers can submit their sell-bid a_e^t, which represents the minimum price that a seller is willing to sell, and $a_e^t = C_{S,e}^t$. The buyer and seller are matched with each other if $k_{w,e}^t \geq C_{B,w,e}^t$, $o_e^t \leq C_{S,e}^t$, and the buy clock crosses the sell clock at $t+1$, i.e., $C_{S,e}^{t+1} \geq C_{B,w,e}^{t+1}$. Then, $\mathcal{M}_B^t = \mathcal{M}_B^t \cup \{w\}$, $\mathcal{M}_S^t = \mathcal{M}_S^t \cup \{e\}$, and $\mathcal{M}_{price}^t = \mathcal{M}_B^t \cup \{C_{w,e}\}$. $\mathcal{M}_{price}^t$ is a set that contains the final matched price and $C_{w,e}$ is defined as follows:

$$C_{w,e} = \frac{C_{B,w,e}^t + C_{S,e}^t}{2}. \tag{5.18}$$

  Thus, the difference between the expected buy-bid and the real buy-bid is the regret $g_{w,e}^t$ of buyer w, which can be calculated as:

$$g_{w,e}^t = k_{w,e}^t - b_{w,e}^t. \tag{5.19}$$

  On the other hand, the difference between the expected sell-bid and the real sell-bid is the regret g_e^{t+1} of seller e, which can be calculated as:

$$g_e^t = a_w^t - o_e^t. \tag{5.20}$$

- **Clock adjustment**: Clock acceptance only occurs when the sell-bid or buy-bid conditions are met. Most of the time, it will be clock adjustment, which means that none of the sellers or the buyers is participating in the auction, so the auction

clocks must be adjusted accordingly. For example, when the buy-bid condition is not met, the Metaverse administrator chooses a step size of μ^t to adjust the buyer clock as follows:

$$C_{B,w,e}^{t+1} = C_{B,w,e}^{t} - \mu^t. \tag{5.21}$$

On the other hand, if the sell-bid condition is not met, the Metaverse administrator chooses a step size of μ^t to adjust the seller clock:

$$C_{S,e}^{t+1} = C_{S,e}^{t} - \mu^t. \tag{5.22}$$

The Metaverse administrator can dynamically adjust the step size to improve the auction's efficiency via proper step size. The auction ends when all sellers' and buyers' clocks cross with each other, i.e., $C_{B,w,e}^{T+1} < C_{S,e}^{T+1}$.

5.3.2 Markov Decision Process of the Double Dutch Auction

In order to determine the proper step size and minimizes the regrets of the buyers and sellers. The double dutch auction mechanism can be formulated as a Markov Decision Process and is defined as $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} >$, where $\mathcal{S}, \mathcal{A}, \mathcal{P}$, and $\mathcal{R}$ are the state space, action space, state transition probability function, and reward, respectively.

- **State space**: The state space at the current time step t contains the buyer clock $C_{B,w,e}^t$, the seller clock $C_{S,e}^t$, the set of winning buyers $\mathcal{M}_B^t$, the set of winning sellers $\mathcal{M}_S^t$, the final matched price $\mathcal{M}_{price}^t$, and the total state space is defined as follows:

$$\mathcal{S}^t = \{C_{B,w,e}^t, C_{S,e}^t, \mathcal{M}_B^t, \mathcal{M}_S^t\} \tag{5.23}$$

- **Action space**: The action space contains the step μ^t that the Metaverse administrator used to adjust the buyer and the seller clock at time step t, i.e., $a^t = \{\mu^t\}$.

- **State transition**: The state transition of the Metaverse administrator from the current state $\mathcal{S}^t$ to the next state $\mathcal{S}^{t+1}$ depends on the action space a^t.

- **Reward**: Following [206], the reward consists of two parts, the trade regret of buyers and sellers and the auction information exchange cost to broadcast the new auction clock to buyers or sellers. The reward is shown in Algorithm 1.

Algorithm 1: Algorithm for calculating the reward in DRL

```
if k^t_{w,e} ≥ C^t_{B,w,e} then
    if o^t_e ≤ C^t_{S,e} then
        if Buy clock crosses the sell clock then
            r(S^t, a^t) = −g^t_{w,e} − g^t_e
else
    r(S^t, a^t) = −p_a(W) − p_a(E)
```

Note that $p_a(X)$ is the auction information exchange cost function for X participants and $p_a(X) = cX$, where c is denoted as the cost coefficient.

To maximize the long-term accumulated reward, the Metaverse administrator needs to determine its optimal action $a \in \mathcal{A}$ given state $s \in \mathcal{S}$. The optimal policy determined by the Metaverse administrator is defined as $\pi^* : \mathcal{S} \rightarrow \mathcal{A}$. The Q-learning can be adopted to find the optimal policy by constructing a look-up table with Q-values of state-action pairs, i.e., $Q(s, a)$. The Q-values are updated based on the experiences of the Metaverse administrator as follows:

$$Q^{new}(s, a) = (1 - \Lambda)Q(s, a) + \Lambda\Big(r(s, a) + \gamma \max_{a' \in \mathcal{A}} Q(s', a')\Big), \tag{5.24}$$

where Λ is the learning rate, γ is the discount factor, and $r(s, a)$ is the reward received. However, Q-learning has the issue of large state and action spaces. Thus, in this chapter, we use deep Q-learning (DQL) to find the optimal policy to obtain the on-demand cost which is the selling price of the VSP.

5.3.3 Deep Q-learning

DQL uses a neural network to approximate the action-value function $Q(s^t, a^t)$, so the Q-value at time step t can be rewritten as $Q(s^t, a^t|\theta)$ and θ is the parameter of the neural network. After the approximation, the optimal policy $\pi^*(s)$ will be given by [209]:

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} Q^*(s^t, a^t|\theta), \tag{5.25}$$

where $Q^*(s_t, a_t|\theta)$ is the optimal Q-value via DNN approximation. DQN will choose the approximated action $a^{t+1} = \pi^*(s^{t+1})$. Then, the approximated $\tilde{Q}(s^t, a^t|\theta)$ can be formulated as:

$$\tilde{Q}(s^t, a^t|\theta) = r(\mathcal{S}^t = s^t, a^t = a^t) + \gamma Q(s^{t+1}, \pi^*(s^{t+1}|\theta)). \tag{5.26}$$

The parameter θ is updated by minimizing the loss, defined as follows:

$$L = \frac{1}{T}\sum_{t=0}^{T}\Big(\tilde{Q}(s^t, a^t|\theta) - Q(s^{t+1}, \pi^*(s^{t+1}|\theta))\Big)^2. \tag{5.27}$$

At time instance t, the Metaverse administrator will choose action a^t according to (5.25), get a reward $r(\mathcal{S}^t, a^t)$ and proceed to the next state s^{t+1}. The vector (s^t, a^t, r^t, s^{t+1}) will be stored in experience memory. Then the parameter θ and policy π will be updated according to the data fetched from the experience memory [209]. By using the DRL-double dutch auction mechanism, a trade list Γ_i is generated for each demand scenario λ_i. The total number of trade lists generated is equal to the number of demand scenarios. The trade list contains the matched price and the matched buyer and seller. The on-demand cost from phase two uses the matched price to perform the resource allocation whenever the scenario occurs.

We analyze the complexity of the proposed DRL-enabled double dutch auction. Following [210], the complexity of the proposed DRL is $O(\hat{e}|\theta|\hat{C})$. $\hat{e}$ is the number of epochs, $|\theta|$ is the number of parameters in the neural network. $\hat{C}$ is the time complexity of calculating the gradient of each θ.

5.3.4 Economic Properties

This section discusses the desirable properties of the double dutch auction, i.e., individual rationality, truthfulness, and budget balance.

Theorem 1. *The proposed double dutch auction satisfies individual rationality.*

Proof. Individual rationality is ensured when none of the participating parties loses any money when the auction is closed. This means that the utilities for both the sellers and buyers are non-negative. A buyer may face two conditions, i) the buyer wins the auction, and ii) the buyer loses the auction. When a buyer wins the auction, the utility of a buyer w can be expressed as $\bar{u}_w = b^t_{w,e} - C_{w,e}$. Since $b^t_w = C^t_{B,w,e}$ and $C^t_{B,w,e} > C^t_{S,e}$, it leads to $b^t_{w,e} > C_{w,e}$. Therefore, in this case, the utility of a buyer w is non-negative. However, in the latter situation, a user's utility is 0 if the user loses an auction. We may generally state that buyers never experience any losses.

On the other hand, the seller may also experience the same two situations. When a seller wins the auction, the utility of a seller e can be expressed as $\bar{u}_e = C_{w,e} - a^t_e$. Since $a^t_e = C^t_{S,e}$ and $C^t_{B,w,e} > C^t_{S,e}$, it leads to $C_{w,e} > b^t_{w,e}$. Therefore, in this case, the utility of a seller e is non-negative. In the second situation, the seller who loses an auction neither earns nor loses anything. Therefore, the proposed auction satisfies individual rationality.

Theorem 2. *Proposed double dutch auction satisfies truthfulness.*

Proof. Truthful auctions stop malevolent users from placing a fraudulent bids. The payment of buyers' and sellers' revenue is determined by the common crossing price $C_{w,e}$. Therefore, buyers and sellers do not have any other incentive to submit their buy-bids or sell-bids except their true valuations to the semantic transmissions.

Theorem 3. *Proposed double dutch auction is budget balance.*

Proof. When the auctioneer makes no losses, an auction is said to be budget balanced [208]. Since the buyers and sellers trade with the common crossing price $C_{w,e}$, their utility is non-negative. The auctioneer gains a non-negative utility as the auctioneer received $p_a(W) + p_a(E)$ whenever the auction information is exchanged.

5.4 Phase Two: Distributed Robust Resource Allocation

Once the trade list Γ_i is obtained for each scenario λ_i. The cost of the on-demand plan will be equal to the matched price in Γ_i. In the following section, we first provide a detailed stochastic integer programming formulation, and then we extend the formulation to distributional robust optimization to allocate the semantic resources in the Metaverse.

5.4.1 Stochastic Integer Programming Formulation

In this section, we develop a two-stage Stochastic Integer Programming (SIP) to minimize the total cost of the network by optimizing the subscription plans. The two-stage SIP can be solved by assuming that the probability distribution of the demand is known. The first stage defines which edge sensing unit is subscribed to the reservation plan while the second stage defines the number of semantic data by utilizing the on-demand plan. We define two variables to indicate the subscription plan that is used. $m^{\mathrm{r}}_{w,e}$ is a binary variable indicates whether VSP w want to rent edge sensing unit e using the reservation plan or not. For example, $m^{\mathrm{r}}_{w,e} = 1$ means that the VSP rents edge sensing unit e and $m^{\mathrm{r}}_{w,e} = 0$ means otherwise. $m^{\mathrm{o}}_{\bar{t},w,e}(\lambda_i)$ is a positive variable to indicate the number of semantic data to purchase by VSP w from edge sensing unit e in time slot $\bar{t}$. Let a convex sample space Ω contain I possible scenarios. The two-stage SIP formulation can be expressed as follows:

$$\min_{m^{\mathrm{r}}_{w,e}, m^{\mathrm{o}}_{\bar{t},w,e}(\lambda_i)} :$$

$$\sum_{w\in\mathcal{W}}\sum_{e\in\mathcal{E}} m^{\mathrm{r}}_{w,e} c^{\mathrm{r}}_{w,e} + \sum_{\lambda_i\in\Omega} P(\lambda_i) \sum_{\bar{t}\in\bar{\mathcal{T}}}\sum_{w\in\mathcal{W}}\sum_{e\in\mathcal{E}} m^{\mathrm{o}}_{\bar{t},w,e}(\lambda_i) c^{\mathrm{o}}_{w,e}(\lambda_i)\delta_2, \tag{5.28}$$

subject to:

$$m^{\mathrm{r}}_{w,e} \geq \bar{O} m^{\mathrm{r}}_{\bar{t},w,e}, \qquad \forall \bar{t}\in\bar{\mathcal{T}}, \forall w\in\mathcal{W}, \forall e\in\mathcal{E}, \tag{5.29}$$

$$\sum_{\bar{t}\in\bar{\mathcal{T}}}\sum_{e\in\mathcal{E}} m^{\mathrm{r}}_{\bar{t},w,e} n \bar{F}_{\bar{t},w,e}(\lambda_i) + \sum_{\bar{t}\in\bar{\mathcal{T}}}\sum_{e\in\mathcal{E}} m^{\mathrm{o}}_{\bar{t},w,e}(\lambda_i)\bar{F}_{\bar{t},w,e}(\lambda_i) \geq F_w, \quad \forall w\in\mathcal{W}, \forall\lambda_i\in\Omega, \tag{5.30}$$

$$\sum_{w\in\mathcal{W}} m^{\mathrm{r}}_{w,e} \leq 1, \qquad \forall e\in\mathcal{E}, \tag{5.31}$$

$$m^{\mathrm{r}}_{w,e} \in \{0,1\}, \forall w\in\mathcal{W}, \forall e\in\mathcal{E}, \tag{5.32}$$

$$m^{\mathrm{o}}_{\bar{t},w,e}(\lambda_i) \in \mathbb{Z}^+, \forall\bar{t}\in\bar{\mathcal{T}}, \forall w\in\mathcal{W}, \forall e\in\mathcal{E}, \forall\lambda_i\in\Omega. \tag{5.33}$$

$\bar{O}$ indicates a small number and $m^{\mathrm{r}}_{\bar{t},w,e}$ is a decision variable. δ_2 is the weight to ensure the on-demand cost is within an acceptable range. Constraint (5.29) ensures that the semantic transmission is available for VSPs for all the time slots whenever an edge sensing unit is rented. At each time slot, a single edge sensing unit can support n semantic data transmission and $\bar{F}_{\bar{t},w,e}(\lambda_i)$ is the similarity score between the semantic data and the VSP interest. (5.30) ensures that the demand of the VSP has to be met using the on-demand plan whenever there is insufficient semantic data from the reservation plan. (5.31) ensures that each edge sensing unit e can only be used once in the reservation plan. We utilize the GAMS script to output the respective decision variables and use the number of constraints to analyze the complexity of the two-stage SIP [195, 211]. Let $\bar{t}'$, w', e', and λ' denote the total number of time slots, VSPs, edge sensing units, and demand scenarios. Then, the total number of constraints are $w'\lambda' + w'e'(1+\bar{t}+\lambda')$.

5.4.2 DRO Cost Minimization

In the real-world Metaverse scenario, the uncertainties generated by the users are highly dynamic and the probability distribution of demand is unknown and hard to

predict accurately. Inherent uncertainty and inaccurate estimation of the probability distribution seriously affect the network cost. Therefore, this section introduces DRO to minimize the expected total cost under the worst-case distribution realization in D, where D represents the confidence set for the ambiguous distribution, constructed in Section 5.4.2. Different from SIP, which assumes that the probability distribution of demand is known. DRO construct a confidence set and presumed that the confidence set contains the true distribution because it is data-driven.

Confidence Set Construction

In this chapter, there is no clear probability distribution for the demand and overall average SSQ score that the VSPs received. As an alternative, a confidence set for ambiguous distributions is created by explicitly associating the model's robustness when applied to the accumulated historical data.

The reference distribution is expressed as $\mathcal{P}_0 = \{p_1^0, p_2^0, \ldots, p_{I'}^0\}$ for a series of obtained historical data with a total of I' scenarios. Note that p_1^0 and $P(\lambda_i)$ are interchangeable. We use the empirical distribution as the reference distribution [212]. Let $\mathcal{P} = \{p_1, \ldots, p_I\}$ denote the ambiguity/true probability distribution. Taking into account that the reference distribution $\mathcal{P}^0$ is not always identical to the true distribution $\mathcal{P}$, we calculate the distance between them and we create a confidence set for the ambiguous distribution, denoted as D. However, a different metric used will impact the effectiveness of the system differently. Therefore, we adopt L_1 and L_∞ norm because of their excellent numerical tractability. L_∞ norm is lesser conservative than L_1 norm. This means that L_∞ norm is willing to take on more risk, is less focused on worst-case scenarios, or has a tighter confidence set than the L_1 norm. As a result, L_1 norm is more reliable at the expense of higher cost [212]. The distance measurement for the L_1 and L_∞ norm are denoted as $d_1(\mathcal{P}, \mathcal{P}_0)$ and $d_\infty(\mathcal{P}, \mathcal{P}_0)$ and the confidence sets D_1 and D_∞, which are corresponding to L_1 and L_∞ norm, respectively, can be constructed as follows [212]:

$$d_1(\mathcal{P}, \mathcal{P}_0) = \sum_{i=1}^{I} |p_i - p_i^0|, \tag{5.34}$$

$$d_\infty(\mathcal{P}, \mathcal{P}_0) = \max_{1 \geq i \geq I} |p_i - p_i^0|. \tag{5.35}$$

$$D_1 = \left\{ \mathcal{P} \in \mathcal{R}_+^I | \sum_{i=1}^{I} |p_i - p_i^0| \leq \theta_1 \right\}, \tag{5.36}$$

$$D_\infty = \left\{ \mathcal{P} \in \mathcal{R}_+^I | \max_{1 \geq i \geq I} |p_i - p_i^0| \leq \theta_\infty \right\}, \tag{5.37}$$

where θ_1 and θ_∞ are the tolerance values, and they are closely related to both the confidence level β_1, β_∞, and the size of available historical data I'. Their exact relationship can be determined from the convergence rates between the ambiguous distribution $\mathcal{P}$ and the reference distribution $\mathcal{P}^0$ under the L_1 and L_∞ norm [212]. The convergence rate are defined as follows [212]:

$$Pr\{||\mathcal{P} - \mathcal{P}^0||_1 \leq \theta_1\} \geq 1 - 2I \exp(\frac{-2I'\theta_1}{I}), \tag{5.38}$$

$$Pr\{||\mathcal{P} - \mathcal{P}^0||_\infty \leq \theta_\infty\} \geq 1 - 2I \exp(-2I'\theta_\infty). \tag{5.39}$$

Suppose $\beta_1 = 1 - 2I \exp(\frac{-2I'\theta_1}{I})$ and $\beta_\infty = 1 - 2I \exp(-2I'\theta_\infty)$, $\theta_1 = -\frac{I}{2I'} \ln \frac{1-\beta_1}{2I}$ for L_1 norm metric and $\theta_\infty = -\frac{1}{2I'} \ln \frac{1-\beta_\infty}{2I}$ for L_∞ norm metric [212]. This means that the probability of the ambiguous distribution within the confidence sets that are constructed by L_1 and L_∞ norms are at least equal to β_1 and β_∞ respectively.

DRO Problem Formulation

Based on the distributed robust optimization model and using the two-stage SIP from (5.28), we formulate our optimization problem, the objective of which is to minimize the expected network cost under the worst-case distribution realization in D. Here, D can be constructed according to the L_1 and L_∞ norms, which are corresponding to D_1 and D_∞. Then, the formulated problem is written as:

$$\min_{m^{\mathrm{r}}_{w,e}, m^{\mathrm{o}}_{\bar{t},w,e}(\lambda_i)} \max_{\mathcal{P} \in D}:$$

$$\sum_{w \in \mathcal{W}} \sum_{e \in \mathcal{E}} m^{\mathrm{r}}_{w,e} c^{\mathrm{r}}_{w,e} + \sum_{\lambda_i \in \Omega} P(\lambda_i) \sum_{\bar{t} \in \bar{\mathcal{T}}} \sum_{w \in \mathcal{W}} \sum_{e \in \mathcal{E}} m^{\mathrm{o}}_{\bar{t},w,e}(\lambda_i) c^{\mathrm{o}}_{w,e}(\lambda_i) \delta_2, \tag{5.40}$$

subject to: (5.29)-(5.33).

We solve the optimization by first interchanging the minimization and maximization operations in the objective function (5.28). Then, the optimization problem can be rewritten as follows:

$$\max_{\mathcal{P} \in D} \min_{m^{\mathrm{r}}_{w,e}, m^{\mathrm{o}}_{\bar{t},w,e}(\lambda_i)}:$$

$$\sum_{w\in\mathcal{W}}\sum_{e\in\mathcal{E}} m^{\mathrm{r}}_{w,e}c^{\mathrm{r}}_{w,e} + \sum_{\lambda_i\in\Omega} P(\lambda_i)\sum_{\bar{t}\in\bar{\mathcal{T}}}\sum_{w\in\mathcal{W}}\sum_{e\in\mathcal{E}} m^{\mathrm{o}}_{\bar{t},w,e}(\lambda_i)c^{\mathrm{o}}_{w,e}(\lambda_i)\delta_2, \tag{5.41}$$

subject to: (5.29)-(5.33).

To address the problem in (5.41), we can treat the probability distribution P as fixed and then optimize the internal minimization problem. However, there is still a maximization in the outer layer in (5.41). Thus, we turn the minimization problem into its dual problem, which is illustrated as follows:

$$\max_{y^{(1)}_{\bar{t},w,e},y^{(2)}_{\bar{t},w,e}(\lambda_i),y^{(3)}_{w,e}}:$$

$$\sum_{\bar{t}\in\bar{\mathcal{T}}}\sum_{w\in\mathcal{W}}\sum_{e\in\mathcal{E}}\sum_{\lambda_i\in\Omega} y^{(2)}_{\bar{t},w,e}(\lambda_i)F_w(\lambda_i) - \sum_{w\in\mathcal{W}}\sum_{e\in\mathcal{E}} y^{(3)}_{w,e}, \tag{5.42}$$

subject to:

$$-y^{(1)}_{\bar{t},w,e} - y^{(3)}_{w,e} \le 0, \qquad \forall \bar{t}\in\bar{\mathcal{T}}, \forall w\in\mathcal{W}, \forall e\in\mathcal{E}, \tag{5.43}$$

$$\sum_{\bar{t}\in\bar{\mathcal{T}}}\sum_{e\in\mathcal{E}}\left(\bar{O}y^{(1)}_{\bar{t},w,e} + n\bar{F}_{\bar{t},w,e}(\lambda_i)y^{(2)}_{\bar{t},w,e}(\lambda_i)\right) \le \sum_{e\in\mathcal{E}} C^{o}_{w,e}(\lambda_i)P(\lambda_i)\delta_2, \quad \forall w\in\mathcal{W}, \forall\lambda_i\in\Omega, \tag{5.44}$$

$$\sum_{w\in\mathcal{W}} y^{(3)}_{w,e} \le c^{r}_{w,e}, \qquad \forall e\in\mathcal{E}, \tag{5.45}$$

where $y^{(1)}_{\bar{t},w,e}$,$y^{(2)}_{\bar{t},w,e}(\lambda_i)$, and $y^{(3)}_{w,e}$ are dual variables corresponding to constraints (5.29)-(5.31).

As a result, the dual problem and the outer maximization operation can be combined and the final reformulated form of the original optimization problem is illustrated as follows:

$$\max_{P(\lambda_i),y^{1}_{\bar{t},w,e},y^{(2)}_{\bar{t},w,e}(\lambda_i),y^{(3)}_{w,e}}:$$

$$\sum_{\bar{t}\in\bar{\mathcal{T}}}\sum_{w\in\mathcal{W}}\sum_{e\in\mathcal{E}}\sum_{\lambda_i\in\Omega} y^{(2)}_{\bar{t},w,e}(\lambda_i)F_w(\lambda_i) - \sum_{w\in\mathcal{W}}\sum_{e\in\mathcal{E}} y^{(3)}_{w,e}, \tag{5.46}$$

subject to: (5.43)-(5.45)

$$\sum_{\lambda_i\in\Omega} P(\lambda_i) = 1, \tag{5.47}$$

$$\mathcal{P} \in D, \tag{5.48}$$

where (5.47) is the basic condition of probability distribution, and (5.48) indicates that the ambiguous distribution $\mathcal{P}$ belongs to the confidence set D. The DRO algorithm can be solved in two steps. In step 1, given historical data and using it as a reference distribution $\mathcal{P}_0$, the problem (5.46) can be solved using the GAMS script to construct the ambiguous distribution $\mathcal{P}$. In step 2, by substituting $\mathcal{P}$ into the objective function of inner minimization problem (5.41), it is a linear programming problem, and we also utilize the GAMS script to output the respective decision variables. Similar to Section 5.4.1, we also analyze the complexity of the DRO by using the number of constraints. The total number of constraints in both step 1 and step 2 of DRO are $2w'\lambda' + e' + 1 + N_{metric} + w'e'(1 + 2\bar{t} + \lambda')$. N_{metric} is the number of linear inequality and equality constraints introduced by the adopted metric [212].

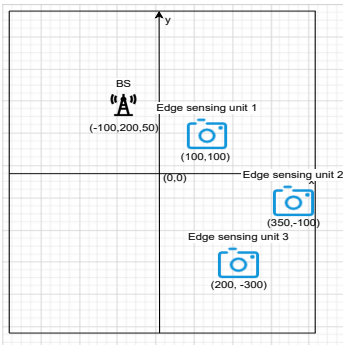


Figure 5.5: x-y coordinates of edge sensing units and base station (BS).

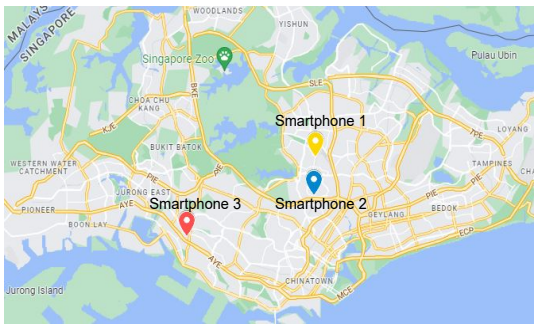


Figure 5.6: The locations of the edge sensing units, such as smartphones that are used to collect data around Singapore.

Figure 5.7: Examples of data that are collected from smartphone 1.

Figure 5.8: Examples of semantic data that are processed by smartphone 1.

5.5 Performance Evaluation

We consider three virtual service providers, and three edge sensing units i.e., smartphones (model: iPhone 13 Pro Max), e_1, e_2, and e_3. The interest of the VSPs are “vehicles on road", “buses and traffic lights" and “motorcyclist and motorcycle". Edge sensing devices are deployed in a rectangular area 800x1000m with three points as vertices where the locations are randomly generated as shown in Fig. 5.5, (100,100)m, (350,-100)m, and (200, -300)m. The location of the BS is at (-100, 200, 50)m. Figure 5.6 shows the locations where the data are collected by the edge sensing units deployed around Singapore. Figure 5.7 shows the examples of data that are collected from smartphone 1 and Fig. 5.8 shows the examples of semantic data that are processed. Each smartphone’s daily rental cost $\hat{F}$ is \$1.89 [213]. The other system parameters are shown in Table 5.2. In the simulations, we have generated the SSQ behaviors randomly to determine $\gamma_{w,e}^{ave}$. At the same time, F_W is also generated randomly in each demand scenario.

Energy efficiency

We first compare the energy efficiency between semantic data transmission and non-semantic data transmission. In semantic transmission, the energy consumption components consist of data collection, semantic data processing (deep learning model inference), and semantic data transmission. On the other hand, the energy consumption components of non-semantic transmission consist of data collection and

Table 5.2: Experiment parameters [4]

Parameter	Values
α_L	2.5
α_N	3.5
B_e	[0.1,0.2] Mhz
B_1	-0.4568
B_2	0.0470
B_3	-0.63
B_4	1.63
μ	-20dB
β_0	-60 dB
Γ	8.2 dB
σ^2	-109 dBm
Uplink transmission power,$\hat{P}_e$	20 dBm

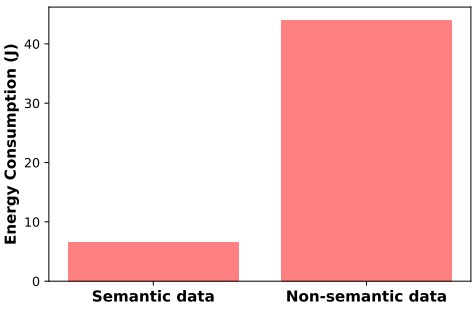


Figure 5.9: The energy consumption of semantic data and non-semantic data.

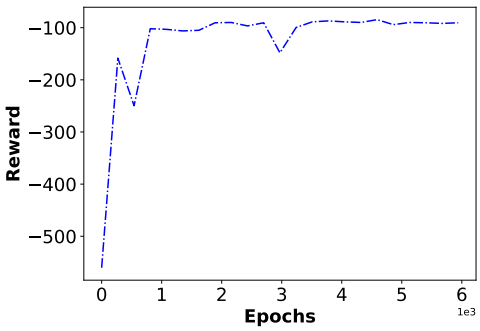


Figure 5.10: The convergence of DRL in Section 5.3.

data transmission. The energy consumption for data collection is negligible as it exists in both. The most intensive operation in semantic data processing is YOLO object detection. According to [214], the energy usage of YOLO is around 4.29 $\times 10^{-4}$ J, which is 4.7879 $\times 10^{-4}$ times less than semantic data transmission. If the energy consumption of YOLO is added to the energy consumption of semantic data transmission, the final result is not affected after we round off to two decimal places. Therefore, the energy consumption for the data process is negligible. An image's average transmitted data size is 2.2Mb, while the average transmitted semantic data is 45Kb. The reason is that the transmitted semantic data is precisely the region in which the VSP is interested (demand). Hence, the energy of semantic data transmission is 0.9J, and the energy of non-semantic data transmission is 44J. The energy usage for semantic data processing is 2.232J/image [215]. The total energy consumption is shown in Fig 5.9. With the help of semantic data communication, edge sensing units

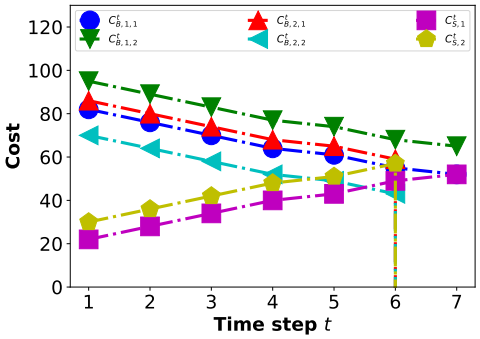


Figure 5.11: An illustration of DRL double dutch auction result.

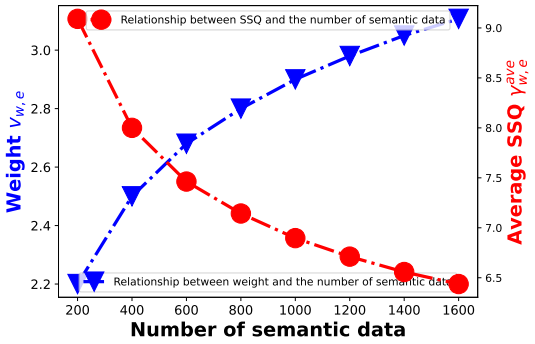


Figure 5.12: Relationship between the weight $v_{w,e}$ and the number of semantic data required.

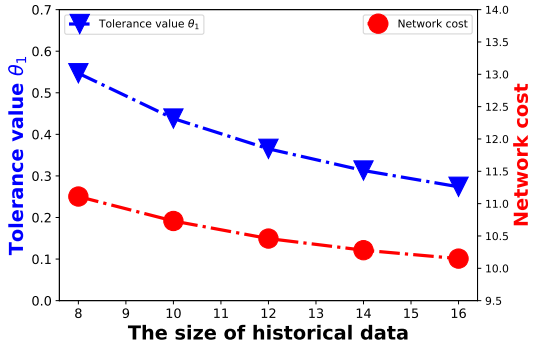


Figure 5.13: Tolerance value θ_1 plot against the historical data size I'.

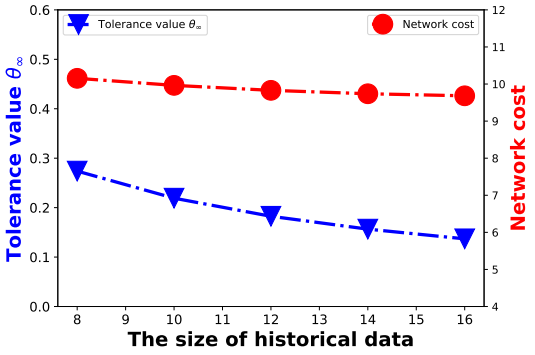


Figure 5.14: Tolerance value θ_∞ plot against the historical data size I'.

can reduce their power consumption during transmission as well as storage costs, which means that they will charge the VSPs less as the transmission cost depends on the transmission energy. In addition, it improves the sustainability of developments in the Metaverse.

5.5.1 Convergence of Double Dutch Auction

As shown in Fig. 5.10, the reward of the DQL in Section 5.3.1 can be maximized and converged at around 3500 epochs. In DQL, we use the epsilon-greedy policy to select an action to balance exploration and exploitation. Exploration allows an agent to increase its existing understanding of each action, resulting in a long-term benefit. On the other hand, exploitation is taking random actions to collect unknown information. In an optimal long-term approach, it may involve some short-term sacrifices. For

example, one exploratory attempt will lead to the lowest reward, but it is necessary to serve as an experience to avoid doing so in the future. At around 700 and 3000 epochs, the DQL is making random actions that lead to poor results and reduce the overall reward.

5.5.2 Varying Time Steps

A simulation setup is used to illustrate the DRL-double dutch auction result and it is shown in Fig. 5.11. This simple network consists of two VSPs and two edge sensing units. In this simulation, the buyer clock reduces from $k_{w,e}^{t}$, and the seller clock increases from o_{e}^{t}. The auction terminates when all the seller clocks cross the buyer clocks. Using the match condition from Section 5.3.1, seller 2 is matched with buyer 2 at $t = 6$, and therefore the matched price is determined by using the clock value from $t - 1$. Once buyer 2 and seller 2 are matched, they are removed from the auction. Similarly, for buyer 1 and seller 1, they are matched at $t = 7$ and the final trade price is determined from $t = 6$.

5.5.3 SSQ Relationship

We consider the set-up similar to Section 5.5.1 to explain the relationship between SSQ, similarity score, and the number of semantic data required. We set $S_{1,1} = 0.8$ and vary the number of semantic data required. The relationship plot is shown in Fig. 5.12. When the number of semantic data increases, weight $v_{1,1}$ increases. An increase in $v_{1,1}$ refers to a decrease in SSQ score as the weight $v_{1,1}$ is inversely proportional to the SSQ score. However, when the number of semantic data required increases, the weight increases at a decreasing rate. This is due to the fact that insufficient data cause only a portion of the discomfort from the Metaverse application. A further increase in semantic data does not significantly decrease SSQ scores.

5.5.4 Monitoring the Tolerance Value θ_1 and θ_∞

In the next set-up, we monitor the relationship between the tolerance value θ_1, θ_∞, and the size of historical data. The confidence level is set at 0.95, $I = 2$, and the relationship curves are shown in Figs. 5.13 and 5.14. From the result, L_∞ norm metrics has a smaller tolerance value than L_1 norm and it is less conservative. This is because the confidence set will shrink as θ_∞ decreases which reduces the conservativeness. As historical data size grows, θ_1 and θ_∞ decrease, and more information is available, resulting in a tighter confidence set and a less conservative output solution and, therefore, a decrease in network cost. When I' increases to infinity, θ_1 and θ_∞ decrease

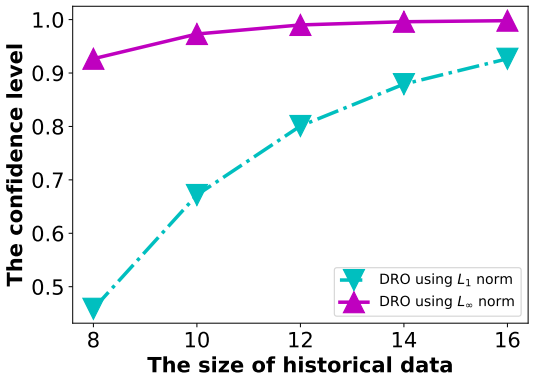


Figure 5.15: Monitor the confidence level β_1 and β_∞.

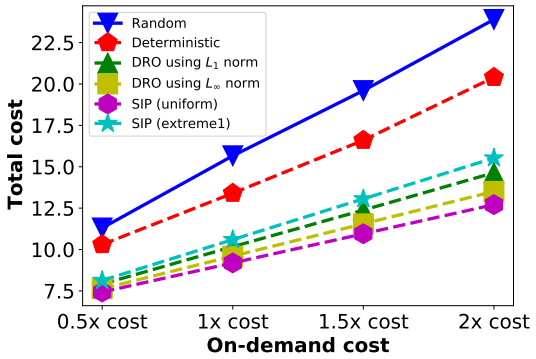


Figure 5.16: Compare DRO with deterministic, random, SIP (uniform), SIP (extreme1) schemes.

Table 5.3: Decisions of VSPs [4]

Decision Variables	Values
$m^r_{1,2}$	1
$m^r_{2,3}$	1
$m^r_{3,1}$	1
$m^o_{1,1,3}(\lambda_1)$	97
$m^o_{1,2,1}(\lambda_1)$	234
$m^o_{3,1,2}(\lambda_2)$	316
$m^o_{1,2,3}(\lambda_2)$	4
$m^o_{1,3,1}(\lambda_2)$	27

to same value that is close to zero. The reference distribution can then be utilized as the actual true distribution after the confidence set becomes a singleton. Then, the VSPs can purchase lesser semantic data transmission with the reservation plan and purchase more with the on-demand plan only when the scenario occurs. It can be readily found that L_∞ can achieve a lower network cost.

5.5.5 Monitor the Confidence Level β_1 and β_∞

We monitor the confidence levels β_1 and β_∞ by fixing the tolerance values θ_1 and θ_∞ and varies the size of historical data I'. The confidence level is shown in Fig. 5.15. As the size of the historical data used increases, it is more confident that the constructed demand probability distribution contains the true distribution. DRO that uses L_∞ norm metrics can achieve a higher confidence level than DRO that uses β_1 norm metrics. Eventually, the confidence level β_1 and β_∞ will reach 1, but the L_∞ norm will converge faster.

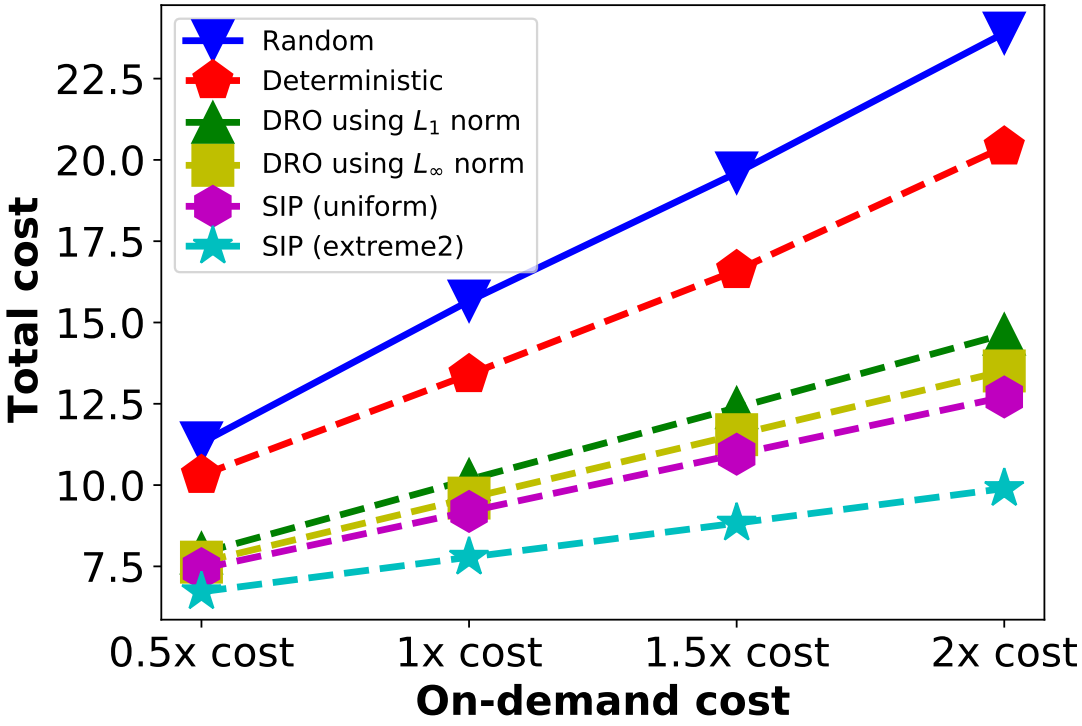


Figure 5.17: Compare DRO with deterministic, random, SIP (uniform), SIP (extreme2) schemes.

5.5.6 Decision of VSPs

We consider a simple case using L_∞ norm with three VSPs, three edge sensing units, and two scenarios $|\lambda_i| = 2$ to show the decisions of the VSPs. There are three time slots representing 1: morning, 2: noon, and 3: afternoon. In λ_1, the VSPs require semantic data that are relevant to the vehicles driving on the road. While in λ_2, the VSPs require semantic data relevant to buses and traffic lights. The decisions of VSPs are shown in Table 5.3. Each VSP subscribes to an edge sensing unit by using the reservation plan due to the low cost. Each edge sensing unit charges VSP with a different reservation cost, and the VSP subscribes to the edge sensing unit with the lowest cost. When the VSP has a demand and additional semantic data is required by using the on-demand plan, the VSP subscribes to the edge sensing unit according to the auction result. For example, in λ_1, VSP 1 is matched with edge sensing unit 1, VSP 2 is matched with edge sensing unit 3, and VSP 3 is matched with edge sensing unit 2. Both VSPs 1 and 2 request to transmit semantic data from time slot 1 as time slot 1 is morning, at the peak hour, and there are more vehicles on the road compared to other time slots.

Table 5.4: Comparing SIP with DRO.

Modeling Approach	Difference in Accuracy	Computational Efficiency
SIP (uniform)	-	0.341sec
DRO with L_∞	4.46%	36.44sec
DRO with L_1	10.66%	45.167sec

5.5.7 Comparison with Other Schemes

We use a similar setting as in Section 5.5.4 and set $n_e = 150$ and compare the DRO schemes (using L_1 and L_∞ norms) with deterministic optimization [212], random, and SIP [195] schemes. The deterministic optimization scheme uses the average SSQ scores, the number of semantic data required, and the similarity score $S_{w,e}$ to optimize the resource allocation. In the random scheme, the values of the decision variables are randomly generated. SIP uses historical data to perform the resource allocation, which can be solved by using (5.28)-(5.33). We use two types of scenarios $|\Omega|| = 2$. The VSPs have demand in both scenarios λ_1 and λ_2. However, the cost of the on-demand plan in λ_2 is higher than λ_1 as the users in λ_2 are faced with a lower QoE, i.e., the users in λ_1 have lower user demand than the users in λ_2. We compare the schemes with three types of SIP, i) uniform case, i.e., $P(\lambda_1) = 0.5$ and $P(\lambda_2) = 0.5$, ii) extreme case 1 (high probability of higher user demand), i.e., $P(\lambda_1) = 0.3$ and $P(\lambda_2) = 0.7$, and iii) extreme case 2 (low probability of higher user demand), i.e., $P(\lambda_1) = 0.7$ and $P(\lambda_2) = 0.3$.

Figure 5.16 compares the aforementioned schemes for extreme case 1 and Figure 5.17 considers extreme case 2. As shown in the results, the deterministic and random schemes cannot adapt to the changes, so their costs are high. The SIP with a uniform case has a lower cost as compared to both DRO schemes. This is due to the fact that DRO uses the uniform distribution as a reference distribution to obtain the true distribution but the DRO is more conservative than SIP (uniform). This means that DRO offers higher system reliability (user satisfaction) at the expense of higher overall network costs. Table 5.4 shows the comparison result between SIP (uniform) and DRO solutions. DRO with L_∞ has a tighter confidence set than that of L_1. Therefore, L_∞ has a lower difference in accuracy and computation efficiency as compared to L_1 norm. As an illustration, we also plot the SIP results (with extreme cases 1 and 2 known as the true distribution). For extreme case 1, the DRO with L_1 norm constructs a relatively close probability distribution of demand compared to DRO with L_∞ norm and SSP (uniform). The reason is that it is more conservative, therefore catering to extreme cases of higher user demand. However, for SIP (with extreme case 2 as the true distribution), we observe that the DRO results are further off

due to conservatives.

5.6 Conclusion

In conclusion, we have proposed a two-phase stochastic semantic resource allocation (SSRA) scheme to allocate the semantic data captured by the edge sensing units. We have shown that energy consumption can be reduced by transmitting semantic data as compared to non-semantic data. Besides semantic communications, we further reduce the network's energy consumption by implementing the SSRA scheme. We have first formulated the SSRA scheme as a double dutch auction to match the VSP and the edge sensing unit. The matching is dynamic and depends on the quality of experience (QoE) from the Metaverse users and the semantic data transmission cost from the edge sensing unit. Then, we formulated DRO to minimize the expected operation cost (energy consumption) of the VSPs as not only is the demand of the VSPs uncertain, but the demand probability distribution is also unknown. Therefore, compared to the random and deterministic the SSRA scheme can achieve the best solution as it can adapt to changes in the probability distribution of VSPs' demands. If extreme case 1 is the actual distribution, then DRO with L_1 norm is better than L_∞ norm when constructing the probability distribution of demand. On the other hand, it is the other way around if extreme case 2 is the actual distribution.

Chapter 6

Conclusions and Future Works

This thesis investigates edge computing within the Metaverse, focusing on computation offloading, wireless energy transfer, and semantic communication to enhance virtual world interactions. We initially explore the Metaverse's technological framework and the role of mobile cloud-edge-end computing in providing ubiquitous computing solutions. Subsequent Chapters present innovative approaches like stochastic coded offloading, wireless energy transfer, and hybrid coded edge computing networks to address energy constraints and computation efficiency in IoTs. The final chapters discuss advancements in semantic communication to optimize data transmission energy efficiency, concluding with future research directions in Metaverse technology.

In this thesis, we explored various methods and applications of edge computing in the context of the Metaverse, a virtual world that is immersive and interconnected. We focused on three key aspects of edge computing: computation offloading, wireless energy transfer, and semantic communication.

Chapter 2 delves into the concept of the Metaverse, its current development, and its significance in the realm of technology. Furthermore, the chapter discusses the mobile cloud-edge-end collaborative computing paradigm, which offers ubiquitous computing and intelligence for users and service providers in the Metaverse. The importance of resource allocation for synchronization between the physical and virtual worlds is also underscored. However, despite the potential of the Metaverse, challenges such as energy constraints in IoTs and the need for efficient resource allocation are highlighted. These challenges need to be addressed to ensure seamless integration and functionality of the

Metaverse in real-world applications.

Given the challenges of the current computation latency and the energy constraints in IoTs. Chapter 3 presented a stochastic coded offloading scheme (SCOS) to minimize the cost of the network energy consumption of the UAVs and prevent over- and under-subscription of the resources. Different from existing works in the literature, we adopt a CDC technique to reduce the latency caused by stragglers. Then, we utilize the z-stage stochastic integer programming model to leverage the trade-off between local and offloading computation while taking into account the computation shortfall and demand uncertainty.

To further improve the energy constraints of the IoTs, Chapter 4 extends chapter 3 by considering the energy constraint of IoT devices that can be recharged with wireless power transfer derived from base stations. The chapter proposes a resource optimization scheme for a hybrid coded edge computing network that consists of IoT devices and base stations. The chapter also introduces efficient computation methods such as Benders' decomposition and sample-average approximation to solve the stochastic integer programming problem.

Besides the energy constraint that the VSPs face in the Metaverse computation, the VSPs can also improve the energy efficiency from the communication aspect. Chapter 5 presents a communication framework whereby a semantic cropping module is introduced to extract semantic information from the captured data. The semantic cropping module reduces the size and energy consumption of data transmission while maintaining its meaning. After recognizing the limitations of SIP due to unpredictable distributions, we pivot to a more conservative, data-driven Distributed Robust Optimization strategy.

As a conclusion to the thesis, we discuss the challenges and research directions in Metaverse as follows:

1. **Unlabeled Data:** The Metaverse generates a large amount of data that can be used to optimize Metaverse service delivery and physical/virtual synchronization. However, this data is often large-scale and unlabeled. Thus, one of the potential solutions is that AI models can use self-supervised learning [216, 217, 218] to mine the supervised information from such data using pretexts. AI models can be trained with the mined supervised information to learn valuable representations for downstream tasks. In this way, the vast amount of data generated in the Metaverse can be mined for knowledge and value, which can then be fed back into the mobile edge network through physical/virtual synchronization to enable improved realizations of value in the physical world.

2. **Avatars:** The Metaverse avatars are digital representations of the users in the virtual world. It could be as straightforward as an icon or as realistic and highly customized as an avatar that engages with the surroundings. However, the realistic/human-like avatar may require a large number of resources to create and operate. Therefore, the VSPs have to establish a solution to generate these realistic avatars with limited resources on edge devices. Furthermore, there is also a need to establish a privacy protocol as each avatar has to collect and store a large amount of private biological data of the user.

3. **Sustainable Resource Allocation:** The Metaverse will always be resource-hungry. Cloud/fog/edge computing services are always needed so that mobile devices or portable gadgets with low storage and computing capability can provide mobile users with excellent and immersive QoE. Therefore, the amount of energy used to provide the Metaverse with communication and computation power will continue to rise, exacerbating energy consumption and increasing environmental concerns such as greenhouse gas emissions. Green cloud/fog/edge networking and computing are therefore needed to achieve sustainability.

4. **Multimodal Semantic Communication:** By automatically interpreting AI models, semantic communication allows mobile edge networks to evolve from data-oriented to semantic-oriented to support a wide variety of context-aware and goal-driven services in the Metaverse. However, existing semantic communication models tend to focus on semantic extraction, encoding, and decoding for a single task, such as speech or images. Therefore, semantic communication has to evolve to multimodal semantic communication to design a communication framework that is not limited to just one mode (e.g., speech or text) but often involves a combination of various modes, such as gestures, facial expressions, images, and sound.

5. **Metaverse Privacy Issues:** The Metaverse relies on a multitude of interconnected technologies. Given this complexity, it is unsurprising that the security vulnerabilities associated with the Metaverse's development are more susceptible to attacks. Moreover, VSPs continuously monitor users, resulting in an unparalleled scale of data collection, encompassing biometrics, locations, interactions, and even emotional reactions. To address these concerns, several solutions can be considered: i) VSPs could employ robust encryption for data both in transit and at rest, safeguarding user information from unauthorized access; ii) leveraging blockchain-based systems could empower users with

greater control over their data, simultaneously diminishing the risks associated with centralized data breaches.

List of Publications

Journal Articles

1. **Ng, W. C.**, Lim, W. Y. B., Xiong, Z., Niyato, D., Poor, H. V., Shen, X. S., and Miao, C., “Resource Optimization for UAV-assisted Wireless Power Charging Enabled Hybrid Coded Edge Computing Network", *IEEE Transactions on Mobile Computing (2023),* doi: 10.1109/TMC.2023.3246994.

2. **Ng, W. C.**, Lim, W. Y. B., Xiong, Z., Niyato, D., Shen, X. S., and Miao, C., “Distributionally Robust Cost Minimized Edge Semantic Intelligence in the Sustainable Metaverse", Accepted in *IEEE Transactions on Mobile Computing (2023).*

3. **Ng, W. C.**, Lim, W. Y. B., Xiong, Z., Niyato, D., Miao, C., Han, Z., and Kim, D. I., “Stochastic Coded Offloading Scheme for Unmanned-Aerial-Vehicle-Assisted Edge Computing", *IEEE Internet of Things Journal (2022)*, 10(7), 5626-5643.

4. Xu, M., **Ng, W. C.**, Lim, W. Y. B., Kang, J., Xiong, Z., Niyato, D., and Miao, C, “A full dive into realizing the edge-enabled metaverse: Visions, enabling technologies, and challenges", *IEEE Communications Surveys & Tutorials (2022),* vol. 25, no. 1, pp. 656-700.

5. **Ng, W. C.** Du, H., Lim, W. Y. B., Xiong, Z., Niyato, D., and Miao, C, “Optimizing Stochastic Resource Allocation for Multi-modal Content Generation in Metaverse", Submitted in *IEEE Transactions on Services Computing (2023).*

6. Zhang, S., Lim, W. Y. B., **Ng, W. C.**, Xiong, Z., Niyato, D., Shen, X. S., and Miao, C., “Towards Green Metaverse Networking: Technologies, Advancements and Future Directions," *IEEE Network (2023)*, doi: 10.1109/MNET.130.2200510.

7. Tan, X., **Ng, W. C.**, Lim, W. Y. B., Xiong, Z., Niyato, D., and Han, Y., “Reputation-Aware Federated Learning Client Selection based on Stochastic Integer Programming", *IEEE Transactions on Big Data (2022)*, doi: 10.1109/TBDATA.2022.3191332.

Conference Proceedings

1. Xu, M., **Ng, W. C.**, Niyato, Han, Y., Miao, C. and Kim, D. I., “Stochastic resource allocation in quantum key distribution for secure federated learning," in *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, Rio de Janeiro, Brazil, December 2022.

2. **Ng, W. C.**, Du, H., Lim, W. Y. B., Xiong, Z., Niyato, D., and Miao, C, “Stochastic resource allocation for semantic communication-aided virtual transportation networks in the metaverse", Submitted in *IEEE Wireless Communications and Networking Conference*, Dubai, United Arab Emirates, April 2024.

3. Ng, J. S., **Ng, W. C.**, Lim, W. Y. B., Xiong, Z., Niyato, D., Leung, C., and Miao, C, “UAV-assisted Wireless Power Charging for Efficient Hybrid Coded Edge Computing Network", in *Proceedings of IEEE International Conference on Communications (ICC)*, Seoul, Korea, 2022.

4. **Ng, W. C.**, Lim, W. Y. B., Ng, J. S., Xiong, Z., Niyato, D., and Miao, C, “Unified resource allocation framework for the edge intelligence-enabled metaverse", in *Proceedings of IEEE International Conference on Communications (ICC)*, Seoul, Korea, 2022.

5. **Ng, W. C.**, Lim, W. Y. B., Ng, J. S., Sawadsitang, S., Xiong, Z., and Niyato, D., “Optimal stochastic coded computation offloading in unmanned aerial vehicles network", in *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, Madrid, Spain, 2021.

Bibliography

[1] Microsoft, "Microsoft flight simulator - pre-order launch trailer," Accessed Dec. 7 2021. [Online]. Available: https://www.youtube.com/watch?v=TYqJALPVn0Y&t=53s&ab_channel=Xbox

[2] Q. Sha, X. Liu, N. Ansari, and Y. Jia, "Efficient multiple charging base stations assignment for far-field wireless-charging in green iot," in *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2021, pp. 1–6.

[3] M. Shahid Anwar, J. Wang, S. Ahmad, A. Ullah, W. Khan, and Z. Fei, "Evaluating the factors affecting qoe of 360-degree videos and cybersickness levels predictions in virtual reality," *Electronics*, vol. 9, no. 9, p. 1530, 2020.

[4] C. You and R. Zhang, "Hybrid offline-online design for uav-enabled data harvesting in probabilistic los channels," *IEEE Transactions on Wireless Communications*, vol. 19, no. 6, pp. 3753–3768, 2020.

[5] J. Du, E. Gelenbe, C. Jiang, H. Zhang, and Y. Ren, "Contract design for traffic offloading and resource allocation in heterogeneous ultra-dense networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2457–2467, 2017.

[6] K. Peng, J. Nie, N. Kumar, C. Cai, J. Kang, Z. Xiong, and Y. Zhang, "Joint optimization of service chain caching and task offloading in mobile edge computing," *Applied Soft Computing*, vol. 103, p. 107142, 2021.

[7] L. Li, X. Qiao, Q. Lu, P. Ren, and R. Lin, "Rendering optimization for mobile web 3d based on animation data separation and on-demand loading," *IEEE Access*, vol. 8, pp. 88 474–88 486, 2020.

[8] U. Moghe, P. Lakkadwala, and D. K. Mishra, "Cloud computing: Survey of different utilization techniques," in *2012 CSI Sixth International Conference on Software Engineering (CONSEG)*. IEEE, 2012, pp. 1–4.

[9] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: architecture, key technologies, applications and open issues," *Journal of network and computer applications*, vol. 98, pp. 27–42, 2017.

[10] J. Santos, J. van der Hooft, M. T. Vega, T. Wauters, B. Volckaert, and F. De Turck, "Srfog: A flexible architecture for virtual reality content delivery through fog computing and segment routing," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2021, pp. 1038–1043.

[11] A. Sunyaev, "Fog and edge computing," in *Internet computing*. Springer, 2020, pp. 237–264.

[12] N. Hassan, K.-L. A. Yau, and C. Wu, "Edge computing in 5g: A review," *IEEE Access*, vol. 7, pp. 127 276–127 289, 2019.

[13] R. Mason, "Is the next generation of the internet ready to be tested?" Accessed Oct. 18, 2022. [Online]. Available: https://www.forbes.com/sites/forbestechcouncil/2022/04/29/is-the-next-generation-of-the-internet-ready-to-be-tested/?sh=2db6cee5129b

[14] S. Carlini, "How edge computing will power the metaverse," Accessed Aug. 18, 2023. [Online]. Available: https://www.forbes.com/sites/forbestechcouncil/2022/05/18/how-edge-computing-will-power-the-metaverse/

[15] M. Slater-Robins, "Meta halts plans for mega new data center after local backlash," Accessed Aug. 18, 2023. [Online]. Available: https://www.techradar.com/news/meta-halts-plans-for-mega-new-data-center-after-local-backlash

[16] K. Smolorz, "How edge computing can unlock the metaverse," Accessed Aug. 18, 2023. [Online]. Available: https://stlpartners.com/articles/edge-computing/edge-computing-metaverse/

[17] L. Lin, X. Liao, H. Jin, and P. Li, "Computation offloading toward edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1584–1607, 2019.

[18] C. Feng, P. Han, X. Zhang, B. Yang, Y. Liu, and L. Guo, "Computation offloading in mobile edge computing networks: A survey," *Journal of Network and Computer Applications*, vol. 202, p. 103366, 2022.

[19] M. Liu and Y. Liu, "Price-based distributed offloading for mobile-edge computing with computation capacity constraints," *IEEE Wireless Communications Letters*, vol. 7, no. 3, pp. 420–423, 2017.

[20] T. Zhang and W. Chen, "Computation offloading in heterogeneous mobile edge computing with energy harvesting," *IEEE Transactions on Green Communications and Networking*, vol. 5, no. 1, pp. 552–565, 2021.

[21] U. Saleem, Y. Liu, S. Jangsher, Y. Li, and T. Jiang, "Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 360–374, 2020.

[22] L. Xie, Y. Shi, Y. T. Hou, W. Lou, H. D. Sherali, H. Zhou, and S. F. Midkiff, "A mobile platform for wireless charging and data collection in sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 8, pp. 1521–1533, 2015.

[23] L. Xie, Y. Shi, Y. T. Hou, and A. Lou, "Wireless power transfer and applications to sensor networks," *IEEE Wireless Communications*, vol. 20, no. 4, pp. 140–145, 2013.

[24] A. Kurs, A. Karalis, R. Moffatt, J. D. Joannopoulos, P. Fisher, and M. Soljacic, "Wireless power transfer via strongly coupled magnetic resonances," *science*, vol. 317, no. 5834, pp. 83–86, 2007.

[25] X. Luo, H.-H. Chen, and Q. Guo, "Semantic communications: Overview, open issues, and future research directions," *IEEE Wireless Communications*, vol. 29, no. 1, pp. 210–219, 2022.

[26] J. Zhang, J. Nie, J. Jin, T. Han, D. I. Kim, and P. Popovski, "Guest editorial: Semantic communications for the metaverse," *IEEE Wireless Communications*, vol. 30, no. 4, pp. 16–17, 2023.

[27] T. von Sawitzky, P. Wintersberger, A. Riener, and J. L. Gabbard, "Increasing trust in fully automated driving: route indication on an augmented reality head-up display," in *Proceedings of the 8th ACM International Symposium on Pervasive Displays*, Palermo Italy, June 2019.

[28] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things journal*, vol. 1, no. 1, pp. 22–32, February 2014.

[29] G. Lawton, "How seoul is creating a metaverse for a smarter city," Accessed Aug. 18, 2023. [Online]. Available: https://venturebeat.com/ai/how-seoul-is-creating-a-metaverse-for-a-smarter-city/

[30] B. Matthew, "The metaverse: What it is, where to find it, and who will build it," Accessed Dec. 7 2021. [Online]. Available: https://www.matthewball.vc/all/themetaverse

[31] H. Duan, J. Li, S. Fan, Z. Lin, X. Wu, and W. Cai, "Metaverse for social good: A university campus prototype," *arXiv preprint arXiv:2108.08985*, 2021.

[32] N.-N. Zhou and Y.-L. Deng, "Virtual reality: A state-of-the-art survey," *International Journal of Automation and Computing*, vol. 6, no. 4, pp. 319–325, Nov. 2009.

[33] CNBC, "Ceo: People will be able to wear vr headsets and have meetings on the street 'in the coming year," Accessed May. 2, 2024. [Online]. Available: https://www.cnbc.com/video/2023/10/18/people-will-be-able-to-wear-vr-headsets-and-have-meetings-on-the-street-in-the-coming-year-says-t
html

[34] X. Sheng, J. Tang, X. Xiao, and G. Xue, "Sensing as a service: Challenges, solutions and future directions," *IEEE Sensors journal*, vol. 13, no. 10, pp. 3733–3741, May 2013.

[35] C. McMahon, "Iot and digital twin: Explaining the connection," Accessed May. 2, 2024. [Online]. Available: https://www.ptc.com/en/blogs/corporate/iot-digital-twin

[36] J. Kim, "The institutionalization of youtube: From user-generated content to professionally generated content," *Media, culture & society*, vol. 34, no. 1, pp. 53–67, Jan. 2012.

[37] M. CALORE, "Roblox builds out its metaverse vision with video chat," Accessed May. 2, 2024. [Online]. Available: https://www.wired.com/story/gadget-lab-podcast-611/

[38] M. Chen, W. Liu, T. Wang, A. Liu, and Z. Zeng, "Edge intelligence computing for mobile augmented reality with deep reinforcement learning approach," *Computer Networks*, vol. 195, p. 108186, 2021.

[39] G. P. Fettweis, "The tactile internet: Applications and challenges," *IEEE Vehicular Technology Magazine*, vol. 9, no. 1, pp. 64–70, Mar. 2014.

[40] Z. Liu, N. Meyendorf, and N. Mrad, "The role of data fusion in predictive maintenance using digital twin," in *Proc. of AIP conference*, vol. 1949, no. 1, Apr. 2018, p. 020023.

[41] L. U. Khan, E. Mustafa, J. Shuja, F. Rehman, K. Bilal, Z. Han, and C. S. Hong, "Federated learning for digital twin-based vehicular networks: Architecture and challenges," 2022. [Online]. Available: https://arxiv.org/abs/2208.05558

[42] P. Bellavista, C. Giannelli, M. Mamei, M. Mendula, and M. Picone, "Digital twin oriented architecture for secure and qos aware intelligent communications in industrial environments," *Pervasive and Mobile Computing*, vol. 85, p. 101646, Sep. 2022.

[43] T. Huynh-The, Q.-V. Pham, X.-Q. Pham, T. T. Nguyen, Z. Han, and D.-S. Kim, "Artificial intelligence for the metaverse: A survey," *arXiv preprint arXiv:2202.10336*, Feb. 2022, [Online]. Available: https://arxiv.org/abs/2202.10336.

[44] Y. Han, D. Niyato, C. Leung, C. Miao, and D. I. Kim, "A dynamic resource allocation framework for synchronizing metaverse with iot service and data," in *Proc. of IEEE International Conference on Communications (ICC)*, Seoul, Korea, May 2022, pp. 1196–1201.

[45] W. C. Ng, W. Y. B. Lim, J. S. Ng, Z. Xiong, D. Niyato, and C. Miao, "Unified resource allocation framework for the edge intelligence-enabled metaverse," in *Prof. of IEEE International Conference on Communications (ICC)*, Seoul, Korea, May 2022, pp. 5214–5219.

[46] F. Guo, F. R. Yu, H. Zhang, H. Ji, V. C. Leung, and X. Li, "An adaptive wireless virtual reality framework in future wireless networks: A distributed learning approach," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 8514–8528, May 2020.

[47] J. Park and M. Bennis, "Urllc-embb slicing to support vr multimodal perceptions over wireless cellular systems," in *Proc. of IEEE Global Communications Conference (GLOBECOM)*, Abu Dhabi, UAE, Dec. 2018, pp. 1–7.

[48] X. Shen, J. Gao, W. Wu, M. Li, C. Zhou, and W. Zhuang, "Holistic network virtualization and pervasive network intelligence for 6g," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 1–30, Dec. 2021.

[49] Meta, "Reality labs," Accessed Jan. 3 2022. [Online]. Available: https://tech.fb.com/ar-vr/

[50] ——, "Inside the lab: Building for the metaverse with ai," Accessed Feb. 16 2022. [Online]. Available: https://about.fb.com/news/2022/02/inside-the-lab-building-for-the-metaverse-with-ai/

[51] R. ANWESHA, "Media and entertainment in the metaverse: The future is already here," Accessed Feb. 16 2022, https://www.xrtoday.com/virtual-reality/media-and-entertainment-in-the-metaverse-the-future-is-already-here/.

[52] V. Diami, "What comparisons between second life and the metaverse miss," Accessed Feb. 16 2022. [Online]. Available: https://slate.com/technology/2022/02/second-life-metaverse-facebook-comparisons.html

[53] DragonSB, "Dragonsb," Accessed Feb. 16 2022. [Online]. Available: https://dragonsb.finance/

[54] L. Ben, "Microsoft's social vr platform 'altspacevr' revamped for improved event hosting," Accessed Feb. 16 2022. [Online]. Available: https://www.roadtovr.com/microsoft-altspacevr-update-event-hosting/

[55] Decentraland, "Decentraland land - what drives long term value," Accessed Feb. 16 2022. [Online]. Available: https://decentral.games/blog/decentraland-land-what-drives-long-term-value

[56] K. Zhang, J. Ni, K. Yang, X. Liang, J. Ren, and X. S. Shen, "Security and privacy in smart city applications: Challenges and solutions," *IEEE Communications Magazine*, vol. 55, no. 1, pp. 122–129, 2017.

[57] W. Y. B. Lim, Z. Xiong, D. Niyato, X. Cao, C. Miao, S. Sun, and Q. Yang, "Realizing the metaverse with edge intelligence: A match made in heaven," *IEEE Wireless Communications*, pp. 1–9, Jul. 2022.

[58] AltspaceVR, "Altspacevr be together, anywhere." Accessed Feb. 16 2022. [Online]. Available: https://altvr.com/

[59] L. Jiaxing, "Baidu launches xirang metaverse environment," Accessed Dec. 16 2021. [Online]. Available: https://kr-asia.com/baidu-launches-xirang-metaverse-environment

[60] M. Bernard, "The amazing possibilities of healthcare in the metaverse," Accessed Feb. 16 2022. [Online]. Available: https://www.forbes.com/sites/bernardmarr/2022/02/23/the-amazing-possibilities-of-healthcare-in-the-metaverse/

[61] Y. Wang, Z. Su, N. Zhang, D. Liu, R. Xing, T. H. Luan, and X. Shen, "A survey on metaverse: Fundamentals, security, and privacy," *arXiv preprint arXiv:2203.02662*, Mar. 2022. [Online]. Available: https://arxiv.org/abs/2203.02662

[62] Meta, "Ray tracing – what does it mean to you?" Accessed Feb. 16 2022. [Online]. Available: https://blog.unity.com/manufacturing/ray-tracing-what-does-it-mean-to-you

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

[64] Y. Han, D. Guo, W. Cai, X. Wang, and V. Leung, "Virtual machine placement optimization in mobile cloud gaming through qoe-oriented resource competition," *IEEE transactions on cloud computing*, Jun. 2020.

[65] M. Li, Z. Sun, Z. Jiang, Z. Tan, and J. Chen, "A virtual reality platform for safety training in coal mines with ai and cloud computing," *Discrete Dynamics in Nature and Society*, vol. 2020, Oct. 2020.

[66] B.-R. Huang, C. H. Lin, and C.-H. Lee, "Mobile augmented reality based on cloud computing," in *Anti-counterfeiting, Security, and Identification*, Taipei, Taiwan, Aug. 2012, pp. 1–5.

[67] T. M. Fernández-Caramés, P. Fraga-Lamas, M. Suárez-Albela, and M. Vilar-Montesinos, "A fog computing and cloudlet based augmented reality system for the industry 4.0 shipyard," *Sensors*, vol. 18, no. 6, p. 1798, Jun. 2018.

[68] B.-G. Chun and P. Maniatis, "Augmented smartphone applications through clone cloud execution." in *HotOS*, vol. 9, 2009, pp. 8–11.

[69] Y. Zhang, L. Jiao, J. Yan, and X. Lin, "Dynamic service placement for virtual reality group gaming on mobile edge cloudlets," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 8, pp. 1881–1897, Jul. 2019.

[70] K. Bierzynski, A. Escobar, and M. Eberl, "Cloud, fog and edge: Cooperation for the future?" in *Proc. of the Second International Conference on Fog and Mobile Edge Computing (FMEC)*, Valencia, Spain, May 2017, pp. 62–67.

[71] Honeywell, "How augmented reality is revolutionizing job training," Accessed Jan. 3 2022. [Online]. Available: https://www.honeywell.com/us/en/news/2018/02/how-ar-and-vr-are-revolutionizing-job-training?utm_source=TW

[72] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. Shen, "Energy efficient dynamic offloading in mobile edge computing for internet of things," *IEEE Transactions on Cloud Computing*, vol. 9, no. 3, pp. 1050–1060, Feb. 2019.

[73] M. T. Beck, M. Werner, S. Feld, and S. Schimper, "Mobile edge computing: A taxonomy," in *Proc. of the Sixth International Conference on Advances in Future Internet*, Lisbon, Portugal, Nov. 2014, pp. 48–55.

[74] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, Sep. 2017.

[75] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks," *IEEE access*, vol. 4, pp. 5896–5907, 2016.

[76] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2016.

[77] O. Munoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 10, pp. 4738–4755, 2014.

[78] Y. Yu, J. Zhang, and K. B. Letaief, "Joint subcarrier and cpu time allocation for mobile edge computing," in *2016 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2016, pp. 1–6.

[79] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Transactions on Wireless Communications*, vol. 12, no. 9, pp. 4569–4581, 2013.

[80] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, Dec. 2017.

[81] E. Mustafa, J. Shuja, S. K. uz Zaman, A. I. Jehangiri, S. Din, F. Rehman, S. Mustafa, T. Maqsood, and A. N. Khan, "Joint wireless power transfer and task

offloading in mobile edge computing: a survey," *Cluster Computing*, vol. 25, no. 4, pp. 2429–2448, 2022.

[82] S. Bi, Y. Zeng, and R. Zhang, "Wireless powered communication networks: An overview," *IEEE Wireless Communications*, vol. 23, no. 2, pp. 10–18, 2016.

[83] J. Feng, Q. Pei, F. R. Yu, X. Chu, and B. Shang, "Computation offloading and resource allocation for wireless powered mobile edge computing with latency constraint," *IEEE Wireless Communications Letters*, vol. 8, no. 5, pp. 1320–1323, 2019.

[84] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE transactions on wireless communications*, vol. 17, no. 3, pp. 1784–1797, 2017.

[85] M. M. Rana, W. Xiang, E. Wang, X. Li, and B. J. Choi, "Internet of things infrastructure for wireless power transfer systems," *IEEE Access*, vol. 6, pp. 19 295–19 303, 2018.

[86] M. M. Rana and W. Xiang, "Iot communications network for wireless power transfer system state estimation and stabilization," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 4142–4150, 2018.

[87] K. W. Choi, A. A. Aziz, D. Setiawan, N. M. Tran, L. Ginting, and D. I. Kim, "Distributed wireless power transfer system for internet of things devices," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2657–2671, 2018.

[88] S. Lhazmir, O. A. Oualhaj, A. Kobbane, and L. Mokdad, "A decision-making analysis in uav-enabled wireless power transfer for iot networks," *Simulation Modelling Practice and Theory*, vol. 103, p. 102102, 2020.

[89] J. Zhou, F. Wu, K. Zhang, Y. Mao, and S. Leng, "Joint optimization of offloading and resource allocation in vehicular networks with mobile edge computing," in *Proc. of tje 10th International Conference on Wireless Communications and Signal Processing (WCSP)*, Hangzhou, China, Oct. 2018, pp. 1–6.

[90] B. Wang, J. Xie, K. Lu, Y. Wan, and S. Fu, "On batch-processing based coded computing for heterogeneous distributed computing systems," *arXiv preprint arXiv:1912.12559*, December 2019.

[91] B. Wang, J. Xie, K. Lu, Y. Wan, and S. Fu, "Coding for heterogeneous uav-based networked airborne computing," in *IEEE Globecom Workshops (GC Wkshps)*, Waikoloa, HI, March 2019.

[92] Q. Yu, M. Maddah-Ali, and S. Avestimehr, "Polynomial codes: an optimal design for high-dimensional coded matrix multiplication," *IEEE Transactions on Network Science and Engineering*, pp. 4403–4413, May 2017.

[93] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," *IEEE Transactions on Information Theory*, vol. 66, no. 1, pp. 278–301, July 2019.

[94] Y. Jiang, J. Kang, D. Niyato, X. Ge, Z. Xiong, and C. Miao, "Reliable coded distributed computing for metaverse services: Coalition formation and incentive mechanism design," *CoRR*, vol. abs/2111.10548, Nov. 2021. [Online]. Available: https://arxiv.org/abs/2111.10548

[95] K. T. Kim, C. Joe-Wong, and M. Chiang, "Coded edge computing," in *Proc. of IEEE INFOCOM*, Toronto, ON, Jul. 2020, pp. 237–246.

[96] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coded mapreduce," in *53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Monticello, IL, April 2015, pp. 964–971.

[97] S. Dutta, V. Cadambe, and P. Grover, ""short-dot": Computing large linear transforms distributedly using coded short dot products," *IEEE Transactions on Information Theory*, vol. 65, no. 10, pp. 6171–6193, July 2019.

[98] C. M. Wang, "Stochastic integer programming: Decomposition methods and industrial applications," June 2014.

[99] W. H. Fleming, S. Sethi, and H. M. Soner, "An optimal stochastic production planning problem with randomly fluctuating demand," *SIAM Journal on Control and optimization*, vol. 25, no. 6, pp. 1494–1502, February 1987.

[100] C. L. Lara, J. D. Siirola, and I. E. Grossmann, "Electric power infrastructure planning under uncertainty: stochastic dual dynamic integer programming (sddip) and parallelization scheme," *Optimization and Engineering*, vol. 21, no. 4, pp. 1243–1281, October 2020.

[101] J. R. Birge and F. Louveaux, "Introduction to stochastic programming," 2011.

[102] C. Li and I. E. Grossmann, "A review of stochastic programming methods for optimization of process systems under uncertainty," *Frontiers in Chemical Engineering*, vol. 2, p. 34, January 2021.

[103] *GAMS Documentation*, GAMS Development Corporation, Washington, DC, USA, 2013. [Online]. Available: https://www.gams.com/latest/docs/

[104] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimization of resource provisioning cost in cloud computing," *IEEE transactions on services Computing*, vol. 5, no. 2, pp. 164–177, February 2011.

[105] S. Sawadsitang, R. Kaewpuang, S. Jiang, D. Niyato, and P. Wang, "Optimal stochastic delivery planning in full-truckload and less-than-truckload delivery," in *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*, Sydney, NSW, Australia, June 2017.

[106] Y. Liu, R. Sioshansi, and A. J. Conejo, "Multistage stochastic investment planning with multiscale representation of uncertainties and decisions," *IEEE Transactions on Power Systems*, vol. 33, no. 1, pp. 781–791, April 2017.

[107] T. Rajamanickam, "The metaverse and communication service providers," Accessed Sep. 3 2023. [Online]. Available: https://www.nokia.com/blog/the-metaverse-and-communication-service-providers/

[108] B. CAULFIELD, "What is the metaverse?" [Online]. Available: https://www.bloomberg.com/news/articles/2022-01-15/gamefi-is-a-new-crypto-craze-what-s-it-all-about-quicktake.

[109] J. Zhao, R. S. Allison, M. Vinnikov, and S. Jennings, "Estimating the motion-to-photon latency in head mounted displays," in *Proc. of IEEE Virtual Reality (VR)*, Los Angeles, CA, Mar. 2017, pp. 313–314.

[110] P. Popovski, F. Chiariotti, V. Croisfelt, A. E. Kalør, I. Leyva-Mayorga, L. Marchegiani, S. R. Pandey, and B. Soret, "Internet of things (iot) connectivity in 6g: An interplay of time, space, intelligence, and value," *arXiv preprint arXiv:2111.05811*, Nov. 2021. [Online]. Available: https://arxiv.org/abs/2111.05811

[111] E. Bastug, M. Bennis, M. Médard, and M. Debbah, "Toward interconnected virtual reality: Opportunities, challenges, and enablers," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 110–117, Jun. 2017.

[112] G. I. Tsiropoulos, A. Yadav, M. Zeng, and O. A. Dobre, "Cooperation in 5g hetnets: Advanced spectrum access and d2d assisted communications," *IEEE Wireless Communications*, vol. 24, no. 5, pp. 110–117, Oct. 2017.

[113] L. Feng, Z. Yang, Y. Yang, X. Que, and K. Zhang, "Smart mode selection using online reinforcement learning for vr broadband broadcasting in d2d assisted 5g hetnets," *IEEE Transactions on Broadcasting*, vol. 66, no. 2, pp. 600–611, Mar. 2020.

[114] J. J. Gimenez, J. L. Carcel, M. Fuentes, E. Garro, S. Elliott, D. Vargas, C. Menzel, and D. Gomez-Barquero, "5g new radio for terrestrial broadcast: A forward-looking approach for nr-mbms," *IEEE Transactions on Broadcasting*, vol. 65, no. 2, pp. 356–368, Mar. 2019.

[115] M. S. Elbamby, C. Perfecto, M. Bennis, and K. Doppler, "Edge computing meets millimeter-wave enabled vr: Paving the way to cutting the cord," in *Proc. of IEEE Wireless Communications and Networking Conference (WCNC)*, Barcelona, Spain, Apr. 2018, pp. 1–6.

[116] L. Militano, M. Condoluci, G. Araniti, A. Molinaro, A. Iera, and G.-M. Muntean, "Single frequency-based device-to-device-enhanced video delivery for evolved multimedia broadcast and multicast services," *IEEE Transactions on Broadcasting*, vol. 61, no. 2, pp. 263–278, Mar. 2015.

[117] Y. Wang, M. Chen, Z. Yang, W. Saad, T. Luo, S. Cui, and H. V. Poor, "Meta-reinforcement learning for reliable communication in thz/vlc wireless vr networks," *IEEE Transactions on Wireless Communications*, pp. 1–1, Mar. 2022.

[118] C. Chaccour, M. N. Soorki, W. Saad, M. Bennis, and P. Popovski, "Can terahertz provide high-rate reliable low latency communications for wireless vr?" *IEEE Internet of Things Journal*, Jan. 2022.

[119] A. E. Abbas, "Constructing multiattribute utility functions for decision analysis," *Risk and Optimization in an Uncertain World*, pp. 62–98, Sep. 2010.

[120] Y. Siriwardhana, P. Porambage, M. Liyanage, and M. Ylianttila, "A survey on mobile augmented reality with 5g mobile edge computing: Architectures, applications, and technical aspects," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1160–1192, Feb. 2021.

[121] H. Qiu, F. Ahmad, F. Bai, M. Gruteser, and R. Govindan, "Avr: Augmented vehicular reality," in *Proc. of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, Munich, Germany, Jun. 2018, pp. 81–95.

[122] L. Liu, H. Li, and M. Gruteser, "Edge assisted real-time object detection for mobile augmented reality," in *Proc. of the 25th Annual International Conference on Mobile Computing and Networking*, Los Cabos, Mexico, Aug. 2019, pp. 1–16.

[123] X. Huang, W. Zhong, J. Nie, Q. Hu, Z. Xiong, J. Kang, and T. Q. S. Quek, "Joint user association and resource pricing for metaverse: Distributed and centralized approaches," Aug. 2022. [Online]. Available: https://arxiv.org/abs/2208.06770

[124] Q. Liu, S. Huang, J. Opadere, and T. Han, "An edge network orchestrator for mobile augmented reality," in *Proc. of IEEE INFOCOM*, Honolulu, HI, Apr. 2018, pp. 756–764.

[125] L. Grippo and M. Sciandrone, "On the convergence of the block nonlinear gauss–seidel method under convex constraints," *Operations research letters*, vol. 26, no. 3, pp. 127–136, Apr. 2000.

[126] E. C. Strinati and S. Barbarossa, "6g networks: Beyond shannon towards semantic and goal-oriented communications," *Computer Networks*, vol. 190, p. 107930, May 2021.

[127] Z. Qin, X. Tao, J. Lu, and G. Y. Li, "Semantic communications: Principles and challenges," *arXiv preprint arXiv:2201.01389*, Dec. 2021. [Online]. Available: https://arxiv.org/abs/2201.01389

[128] H. Xie, Z. Qin, G. Y. Li, and B.-H. Juang, "Deep learning enabled semantic communication systems," *IEEE Transactions on Signal Processing*, vol. 69, pp. 2663–2675, Apr. 2021.

[129] Y. Wang, M. Chen, W. Saad, T. Luo, S. Cui, and H. V. Poor, "Performance optimization for semantic communications: An attention-based learning approach," in *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2021, pp. 1–6.

[130] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[131] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[132] W. Sun, P. Wang, N. Xu, G. Wang, and Y. Zhang, "Dynamic digital twin and distributed incentives for resource allocation in aerial-assisted internet of

vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 8, pp. 5839–5852, Feb. 2021.

[133] H. Wang, J. Wang, J. Chen, Y. Gong, and G. Ding, "Network-connected uav communications: Potentials and challenges," *China Communications*, vol. 15, no. 12, pp. 111–121, December 2018.

[134] N. Cheng, W. Xu, W. Shi, Y. Zhou, N. Lu, H. Zhou, and X. Shen, "Air-ground integrated mobile edge networks: Architecture, challenges, and opportunities," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 26–32, August 2018.

[135] S. Koulali, E. Sabir, T. Taleb, and M. Azizi, "A green strategic activity scheduling for uav networks: A sub-modular game perspective," *IEEE Communications Magazine*, vol. 54, no. 5, pp. 58–64, May 2016.

[136] W. Saad, M. Bennis, and M. Chen, "A vision of 6g wireless systems: Applications, trends, technologies, and open research problems," *IEEE network*, vol. 34, no. 3, pp. 134–142, 2019.

[137] C. Rachel, "The metaverse: Environmental costs of virtual reality," Accessed Nov. 14, 2022. [Online]. Available: https://impakter.com/the-metaverse-environmental-costs-of-virtual-reality/

[138] P. Davies, "5g won't be enough: How are telecom companies building the metaverse with big tech?" Accessed Oct. 18, 2022. [Online]. Available: https://www.euronews.com/next/2022/03/03/5g-won-t-be-enough-how-are-telecom-companies-building-the-metaverse-with-big-tech

[139] Y. Lin, C. Wu, X. Chen, Y. Ji *et al.*, "Meta-networking: Beyond the shannon limit with multi-faceted information, semantic communication, iov, multi-faceted inforamtion," 2023.

[140] M. Xu, W. C. Ng, W. Y. B. Lim, J. Kang, Z. Xiong, D. Niyato, Q. Yang, X. S. Shen, and C. Miao, "A full dive into realizing the edge-enabled metaverse: Visions, enabling technologies, and challenges," *IEEE Communications Surveys & Tutorials*, 2022.

[141] S. Djahel, M. Salehie, I. Tal, and P. Jamshidi, "Adaptive traffic management for secure and efficient emergency services in smart cities," in *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*. IEEE, 2013, pp. 340–343.

[142] E. I. Vlahogianni, K. Kepaptsoglou, V. Tsetsos, and M. G. Karlaftis, "A real-time parking prediction system for smart cities," *Journal of Intelligent Transportation Systems*, vol. 20, no. 2, pp. 192–204, 2016.

[143] M. Chen, M. Mozaffari, W. Saad, C. Yin, M. Debbah, and C. S. Hong, "Caching in the sky: Proactive deployment of cache-enabled unmanned aerial vehicles for optimized quality-of-experience," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 5, pp. 1046–1061, March 2017.

[144] K. Ro, J.-S. Oh, and L. Dong, "Lessons learned: Application of small uav for urban highway traffic monitoring," in *45th AIAA aerospace sciences meeting and exhibit*, 2007, p. 596.

[145] J. Du, C. Jiang, H. Zhang, Y. Ren, and M. Guizani, "Auction design and analysis for sdn-based traffic offloading in hybrid satellite-terrestrial networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2202–2217, 2018.

[146] Y. Liu, J. Nie, X. Li, S. H. Ahmed, W. Y. B. Lim, and C. Miao, "Federated learning in the sky: Aerial-ground air quality sensing framework with uav swarms," *IEEE Internet of Things Journal*, 2020.

[147] C. Tang, C. Zhu, X. Wei, H. Peng, and Y. Wang, "Integration of uav and fog-enabled vehicle: application in post-disaster relief," in *IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, Tianjin, China, January 2019, pp. 548–555.

[148] N. H. Motlagh, M. Bagaa, and T. Taleb, "Uav-based iot platform: A crowd surveillance use case," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 128–134, February 2017.

[149] Y. Zeng, J. Xu, and R. Zhang, "Energy minimization for wireless communication with rotary-wing uav," *IEEE Transactions on Wireless Communications*, vol. 18, no. 4, pp. 2329–2345, 2019.

[150] Q. Hu, Y. Cai, G. Yu, Z. Qin, M. Zhao, and G. Y. Li, "Joint offloading and trajectory design for uav-enabled mobile edge computing systems," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1879–1892, 2018.

[151] S. Dutta, V. Cadambe, and P. Grover, "Coded convolution for parallel and distributed computing within a deadline," in *IEEE International Symposium on Information Theory (ISIT)*, Aachen, Germany, August 2017, pp. 2403–2407.

[152] Q.-V. Pham, M. Zeng, R. Ruby, T. Huynh-The, and W.-J. Hwang, "Uav communications for sustainable federated learning," *arXiv preprint arXiv:2103.11073*, March 2021.

[153] E. Notes, "Qam formats: 8-qam, 16-qam, 32-qam, 64-qam, 128-qam, 256-qam." [Online]. Available: https://www.electronics-notes.com/articles/radio/modulation/quadrature-amplitude-modulation-types-8qam-16qam-32qam-64qam-128qam-256qam.php

[154] Z. Zhou, J. Feng, Z. Chang, and X. Shen, "Energy-efficient edge computing service provisioning for vehicular networks: A consensus admm approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5087–5099, March 2019.

[155] J. Chen, S. Chen, S. Luo, Q. Wang, B. Cao, and X. Li, "An intelligent task offloading algorithm (itoa) for uav edge computing network," *Digital Communications and Networks*, vol. 6, no. 4, pp. 433–443, November 2020.

[156] M. Hua, Y. Wang, Q. Wu, H. Dai, Y. Huang, and L. Yang, "Energy-efficient cooperative secure transmission in multi-uav-enabled wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7761–7775, June 2019.

[157] J. S. Ng, W. Y. B. Lim, H.-N. Dai, Z. Xiong, J. Huang, D. Niyato, X.-S. Hua, C. Leung, and C. Miao, "Joint auction-coalition formation framework for communication-efficient federated learning in uav-enabled internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 4, December 2020.

[158] N. Bezzo, K. Mohta, C. Nowzari, I. Lee, V. Kumar, and G. Pappas, "Online planning for energy-efficient and disturbance-aware uav operations," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Korea (South), December 2016, pp. 5027–5033.

[159] T. Chu, M. J. Starek, J. Berryhill, C. Quiroga, and M. Pashaei, "Simulation and characterization of wind impacts on suas flight performance for crash scene reconstruction," *Drones*, vol. 5, no. 3, p. 67, 2021.

[160] S. Sawadsitang, D. Niyato, P. Tan, and P. Wang, "Joint ground and aerial package delivery services: A stochastic optimization approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 6, pp. 2241–2254, October 2019.

[161] G. Mitsis, E. E. Tsiropoulou, and S. Papavassiliou, "Data offloading in uav-assisted multi-access edge computing systems: A resource-based pricing and user risk-awareness approach," *Sensors*, vol. 20, no. 8, p. 2434, April 2020.

[162] M. Dyer and L. Stougie, "Computational complexity of stochastic programming problems," *Mathematical Programming*, vol. 106, no. 3, pp. 423–432, December 2006.

[163] S. Rajendran and A. R. Ravindran, "Platelet ordering policies at hospitals using stochastic integer programming model and heuristic approaches to reduce wastage," *Computers & Industrial Engineering*, vol. 110, pp. 151–164, 2017.

[164] L. T. Authority, "Singapore traffic data," Accessed Aug. 1, 2021. [Online]. Available: https://data.gov.sg/dataset/traffic-images

[165] DJI, "Dji," Accessed Aug. 1, 2021. [Online]. Available: https://www.dji.com/sg

[166] D. Chattopadhyay, "Application of general algebraic modeling system to power system optimization," *IEEE Transactions on Power Systems*, vol. 14, no. 1, pp. 15–22, 1999.

[167] G. D. Corporation, "General Algebraic Modeling System (GAMS) Release 24.2.1," Washington, DC, USA, 2013. [Online]. Available: http://www.gams.com/

[168] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimal virtual machine placement across multiple cloud providers," in *IEEE Asia-Pacific Services Computing Conference (APSCC)*, Singapore, January 2009, pp. 103–110.

[169] S. S. Mishra and A. Rasool, "IoT Health care Monitoring and Tracking: A Survey," in *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, Tirunelveli, India, 2019, pp. 1052–1057.

[170] C. Yang, T. Peng, S. Lan, W. Shen, and L. Wang, "Towards IoT-enabled Dynamic Service Optimal Selection in Multiple Manufacturing Clouds," *Journal of Manufacturing Systems*, vol. 56, pp. 213–226, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278612520300947

[171] P. Ferrer-Cid, J. M. Barcelo-Ordinas, and J. Garcia-Vidal, "Graph Learning Techniques Using Structured Data for IoT Air Pollution Monitoring Platforms," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[172] L.-H. Lee, T. Braud, P. Zhou, L. Wang, D. Xu, Z. Lin, A. Kumar, C. Bermejo, and P. Hui, "All one needs to know about metaverse: A complete survey on technological singularity, virtual ecosystem, and research agenda," *arXiv preprint arXiv:2110.05352*, 2021.

[173] H. Guo, J. Liu, J. Zhang, W. Sun, and N. Kato, "Mobile-Edge Computation Offloading for Ultradense IoT Networks," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4977–4988, 2018.

[174] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding Up Distributed Machine Learning Using Codes," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1514–1529, 2018.

[175] Docker, "Docker," Accessed April. 14, 2022. [Online]. Available: https://www.docker.com/pricing/?utm_source=docker&utm_medium=webreferral&utm_campaign=traffictopricing

[176] M. Desertot, C. Escoffier, P. Lalanda, and D. Donsez, "Autonomic management of edge servers," in *Self-Organizing Systems*. Springer, 2006, pp. 216–229.

[177] Azure, "Azure private multi-access edge compute (mec)," Accessed April. 14, 2022. [Online]. Available: https://azure.microsoft.com/en-us/solutions/private-multi-access-edge-compute-mec/#overview

[178] ——, "Azure," Accessed April. 14, 2022. [Online]. Available: https://azure.microsoft.com/en-us/solutions/private-multi-access-edge-compute-mec/#overview

[179] G. Zhang, C. Li, H. Zhang, and X. Jiang, "Parameters Optimization for Magnetic Resonance Coupling Wireless Power Transmission," *The Scientific World Journal*, vol. 2014, pp. 1–8, 2014. [Online]. Available: https://doi.org/10.1155/2014/321203

[180] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," *IEEE Transactions on Information Theory*, vol. 66, no. 3, pp. 1920–1933, 2020.

[181] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, "On the Optimal Recovery Threshold of Coded Matrix Multiplication," *IEEE Transactions on Information Theory*, vol. 66, no. 1, pp. 278–301, 2020.

[182] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and S. A. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security, and privacy," in *the 22nd International Conference on Artificial Intelligence and Statistics*, Naha, Okinawa, Japan, April 2019, pp. 1215–1225.

[183] J. S. Ng, W. Y. B. Lim, S. Garg, Z. Xiong, D. Niyato, M. Guizani, and C. Leung, "Collaborative coded computation offloading: An all-pay auction approach," in *ICC 2021-IEEE International Conference on Communications*. IEEE, 2021, pp. 1–6.

[184] F. Didier, "Efficient Erasure Decoding of Reed-Solomon Codes," *arXiv preprint arXiv:0901.1886*, 2009.

[185] L. S. INC., "Lingo 17 online users manual," 2021. [Online]. Available: https://www.lindo.com/doc/online_help/lingo17_0/index.html?scenario_tree.htm

[186] A. Shapiro and A. Nemirovski, "On complexity of stochastic programming problems," in *Continuous optimization*. Springer, 2005, pp. 111–146.

[187] R. M. Van Slyke and R. Wets, "L-shaped linear programs with applications to optimal control and stochastic programming," *SIAM journal on applied mathematics*, vol. 17, no. 4, pp. 638–663, 1969.

[188] A. Schwele, J. Kazempour, and P. Pinson, "Do unit commitment constraints affect generation expansion planning? a scalable stochastic model," *Energy Systems*, vol. 11, no. 2, pp. 247–282, 2020.

[189] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimal virtual machine placement across multiple cloud providers," in *2009 IEEE Asia-Pacific Services Computing Conference (APSCC)*. IEEE, 2009, pp. 103–110.

[190] M. Labbe, "Energy consumption of ai poses environmental problems," Accessed Dec. 14, 2022. [Online]. Available: https://www.techtarget.com/searchenterpriseai/feature/Energy-consumption-of-AI-poses-environmental-problems

[191] I. News, "Ai: The driving force behind the metaverse?" Accessed Oct. 18, 2022. [Online]. Available: https://www.itu.int/hub/2022/06/ai-driving-force-metaverse/

[192] X. Luo, H.-H. Chen, and Q. Guo, "Semantic communications: Overview, open issues, and future research directions," *IEEE Wireless Communications*, 2022.

[193] S. Dmytro Spilka, "How big data could form the cornerstone of the metaverse," Accessed Aug. 14, 2022. [Online]. Available: https://venturebeat.com/datadecisionmakers/how-big-data-could-form-the-cornerstone-of-the-metaverse/

[194] D. Niyato, M. A. Alsheikh, P. Wang, D. I. Kim, and Z. Han, "Market model and optimal pricing scheme of big data and internet of things (iot)," in *2016 IEEE international conference on communications (ICC)*. IEEE, 2016, pp. 1–6.

[195] W. C. Ng, H. Du, W. Y. B. Lim, Z. Xiong, D. Niyato, and C. Miao, "Stochastic resource allocation for semantic communication-aided virtual transportation networks in the metaverse," *arXiv preprint arXiv:2208.14661*, 2022.

[196] R. LiKamWa, B. Priyantha, M. Philipose, L. Zhong, and P. Bahl, "Energy characterization and optimization of image sensing toward continuous mobile vision," in *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, 2013, pp. 69–82.

[197] J. Won, D.-Y. Kim, Y.-I. Park, and J.-W. Lee, "A survey on uav placement and trajectory optimization in communication networks: From the perspective of air-to-ground channel models," *ICT Express*, 2022.

[198] Y. Wu, Y. Wang, F. Zhou, and R. Q. Hu, "Computation efficiency maximization in ofdma-based mobile edge computing networks," *IEEE Communications Letters*, vol. 24, no. 1, pp. 159–163, 2019.

[199] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[200] J. J. LaViola Jr, "A discussion of cybersickness in virtual environments," *ACM Sigchi Bulletin*, vol. 32, no. 1, pp. 47–56, 2000.

[201] S. Davis, K. Nesbitt, and E. Nalivaiko, "A systematic review of cybersickness," in *Proceedings of the 2014 conference on interactive entertainment*, 2014, pp. 1–9.

[202] E. Andrew and VitalBriefing, "Nvidia researchers make real-time rendering a virtual reality," Accessed Nov. 14, 2022. [Online]. Available: https://www.spiceworks.com/tech/innovation/articles/nvidia-researchers-make-real-time-rendering-a-virtual-reality/

[203] R. S. Kennedy, N. E. Lane, K. S. Berbaum, and M. G. Lilienthal, "Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness,"

The international journal of aviation psychology, vol. 3, no. 3, pp. 203–220, 1993.

[204] Y. Zhan, P. Li, Z. Qu, D. Zeng, and S. Guo, "A learning-based incentive mechanism for federated learning," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6360–6368, 2020.

[205] W. Y. B. Lim, J. S. Ng, Z. Xiong, J. Jin, Y. Zhang, D. Niyato, C. Leung, and C. Miao, "Decentralized edge intelligence: A dynamic resource allocation framework for hierarchical federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 3, pp. 536–550, 2021.

[206] M. Xu, D. Niyato, J. Kang, Z. Xiong, C. Miao, and D. I. Kim, "Wireless edge-empowered metaverse: A learning-based incentive mechanism for virtual reality," *arXiv preprint arXiv:2111.03776*, 2021.

[207] M. Gawinecki, P. Kobzdej, M. Ganzha, and M. Paprzycki, "Introducing interaction-based auctions into a model agent-based e-commerce system: preliminary considerations," in *Proceedings of the 2007 Euro American conference on Telematics and information systems*, 2007, pp. 1–8.

[208] V. Jain and B. Kumar, "Auction based cost-efficient resource allocation by utilizing blockchain in fog computing," *Transactions on Emerging Telecommunications Technologies*, p. e4469, 2022.

[209] H. Li, H. Gao, T. Lv, and Y. Lu, "Deep q-learning based dynamic resource allocation for self-powered ultra-dense networks," in *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2018, pp. 1–6.

[210] Y. Hu, M. Chen, W. Saad, H. V. Poor, and S. Cui, "Distributed multi-agent meta learning for trajectory design in wireless drone networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 10, pp. 3177–3192, 2021.

[211] W. C. Ng, W. Y. B. Lim, Z. Xiong, D. Niyato, C. Miao, Z. Han, and D. I. Kim, "Stochastic coded offloading scheme for unmanned aerial vehicle-assisted edge computing," *IEEE Internet of Things Journal*, pp. 1–1, 2022.

[212] Y. Chen, B. Ai, Y. Niu, H. Zhang, and Z. Han, "Energy-constrained computation offloading in space-air-ground integrated networks using distributionally robust optimization," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 11, pp. 12 113–12 125, 2021.

[213] Grover, "Smartphone rental," Accessed Jul. 10, 2022. [Online]. Available: https://www.grover.com/us-en/phones-and-tablets/smartphones

[214] S. Kim, S. Park, B. Na, and S. Yoon, "Spiking-yolo: spiking neural network for energy-efficient object detection," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 07, 2020, pp. 11 270–11 277.

[215] Y. MAO, "A memory-efficient yolo object detection convolutional neural network inference engine on the kilocore 2 manycore platform," Ph.D. dissertation, 2021.

[216] I. Misra and L. v. d. Maaten, "Self-supervised learning of pretext-invariant representations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6707–6717.

[217] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 11, pp. 4037–4058, 2020.

[218] X. Liu, F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang, and J. Tang, "Self-supervised learning: Generative or contrastive," *IEEE Transactions on Knowledge and Data Engineering*, 2021.