Data-Driven Network Management for Next-Generation Wireless Networks

by

Conghao Zhou

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2022

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Amiya Nayak Professor School of Electrical Engineering and Computer Science University of Ottawa
Supervisor(s):	Xuemin (Sherman) Shen University Professor Department of Electrical and Computer Engineering University of Waterloo
Internal Member:	Zhou Wang Professor Department of Electrical and Computer Engineering University of Waterloo
Internal Member:	Xiaodong Lin Professor Department of Electrical and Computer Engineering University of Waterloo
Internal-External Member:	Xinzhi Liu Professor Department of Applied Mathematics University of Waterloo

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

With the commercialization and maturity of the fifth-generation (5G) wireless networks, the next-generation wireless network (NGWN) is envisioned to provide seamless connectivity for mobile user terminals (MUTs) and to support a wide range of new applications with stringent quality of service (QoS) requirements. In the NGWN, the network architecture will be highly heterogeneous due to the integration of terrestrial networks, satellite networks, and aerial networks formed by unmanned aerial vehicles (UAVs), and the network environment becomes highly dynamic because of the mobility of MUTs and the spatiotemporal variation of service demands. In order to provide high-quality services in such dynamic and heterogeneous networks, flexible, fine-grained, and adaptive network management will be essential. Recent advancements in deep learning (DL) and digital twins (DTs) have made it possible to enable data-driven solutions to support network management in the NGWN. DL methods can solve network management problems by leveraging data instead of explicit mathematical models, and DTs can facilitate DL methods by providing extensive data based on the full digital representations created for individual MUTs. Data-driven solutions that integrates DL and DT can address complicated network management problems and explore implicit network characteristics to adapt to dynamic network environments in the NGWN. However, the design of data-driven network management solutions in the NGWN meets several technical challenges: 1) how the NGWN can be configured to support multiple services with different spatiotemporal service demands while simultaneously satisfying their different QoS requirements; 2) how the multi-dimensional network resources are proactively reserved to support MUTs with different mobility patterns in a resource-efficient manner; and 3) how the heterogeneous NGWN components, including base stations (BSs), satellites, and UAVs, jointly coordinate their network resources to support dynamic service demands, etc. In this thesis, we develop efficient data-driven network management strategies in two stages, i.e., long-term network planning and real-time network operation, to address the above challenges in the NGWN.

Firstly, we investigate planning-stage network configuration to satisfy different service requirements for communication services. We consider a two-tier network with one macro BS and multiple small BSs, which supports communication services with different spatiotemporal data traffic distributions. The objective is to maximize the energy efficiency of BSs by jointly configuring downlink transmission power and communication coverage for each BS. To achieve this objective, we first design a network planning scheme with flexible binary slice zooming, dual time-scale planning, and grid-based network planning. The scheme allows flexibility to differentiate the communication coverage and downlink transmission power of the same BS for different services while improving the temporal and spatial granularity of network planning. We formulate a combinatorial optimization

problem in which communication coverage management and power control are mutually dependent. To solve the problem, we propose a data-driven method with two steps: 1) we propose an unsupervised-learning-assisted approach to determine the communication coverage of BSs; and 2) we derive a closed-form solution for power control. Secondly, we investigate planning-stage resource reservation for a compute-intensive service to support MUTs with different mobility patterns. The MUTs can offload their computing tasks to the computing servers deployed at the core networks, gateways, and BSs. Each computing server requires both computing and storage resources to execute computing tasks. The objective is to optimize long-term resource reservation by jointly minimizing the usage of computing, storage, and communication resources and the cost from re-configuring resource reservation. To this end, we develop a data-driven network planning scheme with two elements, i.e., multi-resource reservation and resource reservation re-configuration. First, DTs are designed for collecting MUT status data, based on which MUTs are grouped according to their mobility patterns. Then, an optimization algorithm is proposed to customize resource reservation for different groups to satisfy their different resource demands. Last, a meta-learning-based approach is proposed to re-configure resource reservation for balancing the network resource usage and the re-configuration cost. Thirdly, we investigate operation-stage computing resource allocation in a space-air-ground integrated network (SAGIN). A UAV is deployed to fly around MUTs and collect their computing tasks, while scheduling the collected computing tasks to be processed at the UAV locally or offloaded to the nearby BSs or the remote satellite. The energy budget of the UAV, intermittent connectivity between the UAV and BSs, and dynamic computing task arrival pose challenges in computing task scheduling. The objective is to design a real-time computing task scheduling policy for minimizing the delay of computing task offloading and processing in the SAGIN. To achieve the objective, we first formulate the on-line computing scheduling in the dynamic network environment as a constrained Markov decision process. Then, we develop a risk-sensitive reinforcement learning approach in which a risk value is used to represent energy consumption that exceeds the budget. By balancing the risk value and the reward from delay minimization, the UAV can explore the task scheduling policy to minimize task offloading and processing delay while satisfying the UAV energy constraint. Extensive simulation have been conducted to demonstrate that the proposed data-driven network management approach for the NGWN can achieve flexible BS configuration for multiple communication services, fine-grained multi-dimensional resource reservation for a compute-intensive service, and adaptive computing resource allocation in the dynamic SAGIN. The schemes developed in the thesis are valuable to the data-driven network planning and operation in the NGWN.

Acknowledgments

First, I would like to express my sincerest appreciation to my supervisor, Prof. Xuemin (Sherman) Shen, for providing me an invaluable opportunity to start my Ph.D. study at the University of Waterloo. I will never forget his continuous support and helpful guidance throughout the doctoral program. He always encouraged me to have a professional research attitude and to become an independent researcher, and gave me enough time to improve my research skills. Without his patience and insightful enlightenment, the achievements in my research would have never been possible. Moreover, Prof. Shen conveyed his philosophy of life to me both in words and deeds. I learned from him the importance of balancing work, life, and family. Prof. Shen plays a significant role in my life as a positive role model.

I would like to express my deepest gratitude to Prof. Weihua Zhuang for her valuable guidance and help during my Ph.D. study, especially in AI Project. It is my honor and fortune to have a weekly meeting with Prof. Zhuang during the project. Her insightful thoughts and comprehensive understanding inspired me to do in-depth research. The experience gained from the project provides a solid foundation for my research.

I would also like to thank Prof. Xinzhi Liu, Prof. Xiaodong Lin, Prof. Zhou Wang, and Prof. Amiya Nayak for serving my thesis examination committee. Their insightful comments and valuable questions have significantly improved the quality of my thesis.

In the past four years, all the experience and memories I have gained in the BBCR group will be my greatest treasure. I would like to thank Prof. Nan Cheng, Prof. Peng Yang, Prof. Feng Lyu, Dr. Hongli He, and Dr. Wen Wu for opening my door to the world of research. I would like to thank Prof. Jie Gao, Dr. Mushu Li, Prof. Huaqing Wu, and Dr. Weisen Shi for leading my way to being a researcher. I would like to thank Prof. Haibo Zhou, Prof. Ning Zhang, Dr. Jiayin Chen, Prof Qiang Ye, Dr. Kaige Qu, Prof. Haixia Peng, Mingcheng He, Yingying Pei, Shisheng Hu, and Xinyu Huang for their significant support and help in both my research and life. I am also grateful for the wonderful time spent in Canada with Dr. Dongxiao Liu, Dr. Cheng Huang, Han Yin, Moru Yao, Liang Xue, Ruoxu Wang, Yannan Wei, and all current and former BBCR members.

Finally, I would like to greatly thank my parents Hongqin Hu and Dr. Yanlai Zhou for their endless love, encouragement, and understanding. I would like to extend my special thanks to Xiaoyue Ma for accompanying and supporting me in the long term. Their enormous efforts motivate me to achieve my dream step by step.

Conghao Zhou
July. 14, 2022
Waterloo, Ontario, Canada

Dedication

The Ph.D. thesis is dedicated to my beloved parents, *Yanlai Zhou* and *Hongqin Hu*.

The dedication is also in memory of my grandparents.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

3GPP	3rd Generation Partnership Project
5G	Fifth-generation
AI	Artificial Intelligence
AR	Augmented Reality
BS	Base Station
CAPEX	Capital Expenditure
CDF	Cumulative Distribution Function
CMDP	Constrained Markov Decision Process
CN	Core Network
DL	Deep Learning
DNN	Deep Neural Network
DOTS	Delay-Oriented Task Scheduling
DQN	Deep Q Learning
DRL	Deep Reinforcement Learning
DT	Digital Twin
DTD	Data Traffic Distributions
IoT	Internet of Things
LEO	Low Earth Orbit
LOSCS	Local Optimum Service Coverage Search
MBS	Macro Base Station
MEC	Mobile Edge Computing
ML	Machine Learning
MUT	Mobile User Terminal
NAP	Network Aggregation Point
NFV	Network Function Virtualization

NGWN	Next-generation Wireless Networks
OPEX	Operation Expenditure
PSO	Particle Swarm Optimization
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Networks
RB	Resource Block
RL	Reinforcement Learning
RPC	Random Probabilistic Configuration
RR	Resource Reservation
SAGIN	Space-Air-Ground Integrated Network
SBS	Small Base Station
SC	Service Coverage
SDN	Software-Defined Networking
SINR	Signal-To-Interference-and-Noise Ratio
SPC	Sampling-based Probabilistic Configuration
S-BS	Server at a Base Station
S-CN	Server at the Core Network
S-NAP	Server at a Network Aggregation Point
UAV	Unmanned Aerial Vehicle
UDT	Mobile User Terminal Digital Twin
ULSCS	Unsupervised Learning-assisted Service Coverage Search
VM	Virtual Machine
VR	Virtual Reality

List of Symbols

Symbols in Chapter 3

$a_{m,n}$	The binary indicator to indicate the SC size of SBS m for serving slice n
$b^t_{i,n,i',n'}$	The indicator to represent whether grid i for slice n is interfered with grid i for slice n during time interval t
C_m	The total number of resource blocks that BS m can allocate within each time interval
$d_{m,i}$	The distance between BS m and grid i
$d_{m,m'}$	The distance between BS m and BS m'
$E^t_{m,n}$	The energy consumption of slice n at BS m during time interval t
$h^t_{i,n}$	The average channel gain between grid i and BS $m_{i,n}$ over the duration of time interval t
$\mathcal{I}_{m,n}$	The set of grid indexes which are within the SC of BS m for serving slice n
L^{f}_m	The radius of full SC of SBS m
L^{r}_m	The radius of reduced SC of SBS m
$L_{m,n}$	The SC of SBS m for serving slice n
$L_{\max}$	The maximum radius of geographical coverage of all SBSs
$m_{i,n}$	The index of BS which serves grid i for slice n

$\mathbf{O}$	The key features of DTD
$p_{i,n}^t$	The downlink transmission power at grid i for slice n over time interval t
p_{MBS}	The maximum transmission power of the MBS
p_{SBS}	The maximum transmission power of a SBS
$P_{m,n}^t$	The total transmission power of BS m for slice n during time interval t
r	The diameter of hexagon grids
$\mathcal{R}_{m,n}$	The set of grids in the ring-shaped area circling BS m for slice n
$w_{i,n}^t$	The data traffic volume of slice n generated at grid i during time interval t
$\gamma_{i,n}^t$	The SINR of grid i for slice n during time interval t
$\gamma_n^{\min}$	The SINR requirement of slice n
η_n	The required resource blocks of slice n per unit data traffic
$\theta_{i,n}^t$	The resource block occupation ratio of gird i for slice n during time interval t
λ_n	The weight to reflect the value of the energy efficiency of slice n
τ	The duration of a time interval

Symbols in Chapter 4

a_k	The indicator on whether the resource reservation in time interval k should be reconfigured
$c_{e,k}^n$	The amount of computing resource reserved at server e for group n in time interval k
$f_{b,e,k}^n$	The load of computing tasks from group n covered by BS b and assigned to server e in time interval k

$f_{b,e,k}^{n,i}$	The load of computing tasks requiring chunk i from group n covered by BS b and assigned to server e in time interval k
$g_{e,k}$	The amount of storage resource reserved at server e in time interval k
$\mathcal{I}_{e,k}$	The set of chunks stored at server e in time interval k
L	The data volume of each chunk of context data
$L^{\mathbf{re}}$	The data volume of remotely accessing context data for each computing task
$m_{e,k}^{n}$	The load of computing tasks from group n assigned to server e in time interval k
$o_k^{\mathbf{v}}$	The cost from reconfiguring resource reservation in time interval k
$p_{b,k}^{i}$	The request ratio of chunk i in the coverage of BS b in time interval k
$v_{b,e,k}^{\mathbf{up}}$, $v_{b,e,k}^{\mathbf{down}}$	The communication resource usage of uploading and downloading, respectively, between server e and BS b in time interval k
$v_{n,e,k}^{\mathbf{re}}$	The communication resource usage of remote access for executing computing tasks at server e in time interval k
$w^{\mathbf{c}}$, $w^{\mathbf{s}}$, $w^{\mathbf{o}}$	The weights of communication, storage, and communication resource usage, respectively
$x_{b,k}^{n}$, $\tilde{x}_{b,k}^{n}$	The actual and predicted load of computing tasks, respectively, from group n in the coverage of BS b in time interval k
$\tilde{\boldsymbol{x}}_k^n$	The spatial task distribution of group n in time interval k
α, β, γ	The input data size, computing workload, size of computing result of each computing task, respectively
Δ_k	The network resource usage during time interval k

$\epsilon^{\mathbf{c}}_{e,k}, \epsilon^{\mathbf{s}}_{e,k}$	The computing and storage resource usage of server e, respectively, during time interval k
$\eta_{b,e}$	The communication resource usage of transmitting a bit data between BS b and server e
$\xi^{e'}_{e}$	The communication resource usage of accessing context data remotely from server e' to server e

Chapter 1

Introduction

While the fifth-generation (5G) wireless network has been able to support lots of applications, as a wide range of new applications continues to increase in number and popularity, their stringent and various service requirements may surpass what the 5G networks can provide. With the success of 5G, the communication and networking communities now anticipate the next-generation wireless network (NGWN), with higher requirements for flexibility, fine-granularity, and adaptability in support of high-quality services in an increasingly dynamic and heterogeneous network environment. However, prevalent model-driven network management paradigms do not fulfill the requirements in the NGWN due to the limitations in addressing complicated network management problems and handling highly dynamic network environments. To address the limitations, a new data-driven paradigm is envisioned to enable intelligent network management in the NGWN. In this chapter, we first provide an overview of the NGWN and network management, then elaborate on the requirements of network management in the NGWN and introduce the potential of data-driven network management. Finally, we present three key research issues investigated in this thesis.

1.1 Overview

1.1.1 Overview of NGWNs

In the past several decades, wireless communications and networking have undergone a dramatic evolution, with a new generation emerging every ten years or so. With the

commercialization and maturity of the fifth-generation (5G) wireless networks, the development of the NGWN, i.e., 6G, has attracted a great deal of attention. Both academia and industry have started discussing the vision, requirements, and driving technologies for the implementation of the NGWN in 2030. The consensus among the discussions is that the NGWN should provide anywhere and anytime connectivity and services to mobile user terminals (MUTs) for supporting multifarious applications across every aspect of human society [1–3]. The NGWN has the following three distinct features in contrast to 5G networks, in terms of applications, network architecture, and network environment.

- **Disruptive New Applications** – The increasing number of bandwidth-demanding applications drives up the demand for the capacity of 5G networks. As the successor to 5G, the NGWN is expected to support a wide range of new applications, such as virtual reality (VR) / augmented reality (AR), industry 5.0, holographic communication, and advanced autonomous driving [3]. Such new applications have more stringent service requirements, e.g., latency and data rate, than 5G applications, and the quality of service (QoS) requirements could be diversified. The new applications, e.g., compute-intensive applications, require additional network resources including computing and storage resources, in addition to the conventional communication resource, [4]. Moreover, the number of MUTs is increasing exponentially. For example, the report in [5] predicts that over 10% of companies will use VR/AR technology for business operations, and the number of head-mounted VR/AR devices will reach a minimum of 337 million by 2025. As a result, the explosion in the number of MUTs, together with strict service requirements and multi-dimensional resource needs, will pose significant challenges to network management for supporting these disruptive new applications in the NGWN.

- **Heterogeneous Network Architecture** – The NGWN will integrate multiple network components such as space, air, and ground networks, demonstrating a high heterogeneity level [6]. In the 5G era, lots of remote, rural, and suburban areas still lack a high-speed Internet connection. Around 3 billion people worldwide cannot access the Internet or enjoy high-speed services [7]. To overcome this barrier and provide global-coverage and ubiquitous services, non-terrestrial networks are expected to complement conventional terrestrial networks. In particular, leveraging plenty of satellites to provide global connectivity in a cost-effective way will become a reality due to the decreasing cost of constructing and launching satellites [8, 9]. Meanwhile, the flexible deployment of aerial networks based on unmanned aerial vehicles (UAVs) can provide services in response to dynamic service demands and emergency situations, such as disaster relief or service congestion [10, 11]. Consequently, the NGWN is envisioned to integrate non-terrestrial networks, e.g., satellite networks and aerial

networks, with conventional terrestrial networks to provide seamless and flexible services. However, due to the scalability, existing network management methods may not handle such high heterogeneity [12].

- **Highly Dynamic Network Environment** – The network environment of NGWN will be highly dynamic due to multiple factors. First, in the NGWN, the velocity of some MUTs, e.g., aircrafts or high-speed trains, can be up to 500 km/hour [2]. Second, due to the random channel fading and unpredictable interference, wireless channel conditions can be time-varying and location-dependent. Third, the data traffic and service demands from MUTs in the NGWN have spatiotemporal variations. For example, the computation demand from MUTs of a compute-intensive application can change over time. As a result, providing reliable connections and high-quality services for MUTs in highly dynamic network environment is difficult.

1.1.2 Overview of Network Management

In light of the three aforementioned features of the NGWN requiring advanced network management, we provide an overview of network management and its different categories in this subsection.

Network management refers to the process of configuring, monitoring, and maintaining a network to ensure that the network runs smoothly and efficiently for providing reliable and high-quality services to MUTs. Existing research has a relatively broad definition of network management, yet there is a consensus that network management can be categorized into two stages, i.e., planning stage and operation stage, according to their timescales [3, 13, 14]. Network management in the planning stage and operation stage are referred to as *network planning* and *network operation*, respectively. In particular, the planning period, also called planning window, may last from minutes to hours, while the operation period, called operation window, is measured in milliseconds or seconds [1]. Due to the dynamic nature of network environments, the decisions of both the network planning and operation need to be adjusted from time to time. The relation between network planning and network operation can be summarized as "planning before operation" and illustrated as Fig. 1.1. The different focuses of network planning and network operation are introduced below.

Network planning, i.e., planning-stage network management, aims to meet network performance targets, e.g., network energy efficiency and resource utilization. To achieve this goal, network planning can be further categorized into *network configuration*, i.e., coordinate and configure the network infrastructure, and *resource reservation*, i.e., reserve network resources proactively on network infrastructure [15]. We present the focus of network planning from the following three perspectives in terms of the type of the supported

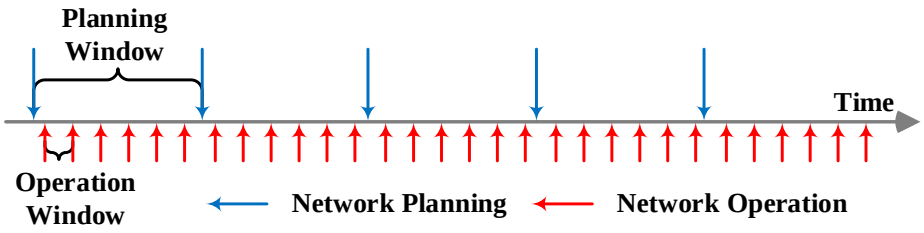


Figure 1.1: The illustration of network planning and network operation.

services. 1) To support conventional communication services, existing works on network planning mainly investigate radio access networks (RANs) configuration, including base station (BS) deployment, BS communication coverage management, and downlink reference power control, as well as communication resource (e.g., frequency bandwidth) reservation on BSs in RANs [16]. With the development of ultra-dense networks, a wide range of BSs with different transmission power and communication coverage, such as macro BSs and small BSs, are deployed in RAN, which complicates co-channel interference among BSs. Configuring BSs and reserving communication resources in multi-tier RANs become important in network planning. 2) With the rapid development of mobile edge computing (MEC), multiple computing servers can be deployed at different network nodes such as the core network, gateways, BSs, and other locations in the network, to support compute-intensive services for executing computing tasks from MUTs [17, 18]. In the NGWN, configuring computing servers or deploying service functions as well as proactive computing and storage resource reservation are crucial in supporting compute-intensive services. 3) To support multiple services with different service requirements, *network slicing* technique is used in 5G networks. Multiple isolated virtual networks, i.e., network slices, can be created on top of the physical network, and each virtual network is used to support at least one service [19]. Each virtual network can be planned in the planning stage via placing virtual network functions and reserving network resources to satisfy its QoS requirements while supporting service differentiation with other virtual networks [20].

In contrast to network planning, network operation, i.e., operation-stage network management, targets delivering services to individual MUTs based on the well-planned network for improving MUT satisfaction. We introduce the focus of network operation from the following two perspectives in terms of the type of the supported services. 1) To support communication services, existing works on network operation concentrate on associating each individual MUT to a BS and determining the quantity of communication resource al-

located to the MUT in real time [21]. 2) Network operation for compute-intensive services is more complicated than communication services due to the need for multi-dimensional network resources. Offloading computing tasks from each MUT to one of the computing servers deployed within networks should take into account not only communication between the computing server and the MUT but also computing task execution at the computing server [22]. In order to support compute-intensive services, offloading and scheduling computing tasks, as well as computing resource allocation, in real time are mainly investigated in the operation stage.

Evidently, both network planning and network operation play critical roles in improving network performance and user satisfaction. However, the existing literature for conventional networks pays much more attention to the operation stage [23,24], while only limited works target the planning stage [14].

1.2 Network Management in NGWNs

Envisioned to support unprecedentedly diverse and new applications in increasingly heterogeneous and dynamic networks, the NGWN will require innovative network management paradigms. In this section, we first introduce the network management requirements in the NGWN and why conventional network management techniques cannot satisfy them. Then, we summarize the limitations of the current network management paradigms.

1.2.1 Network Management Requirements in NGWNs

The network management requirements in the NGWN are put forward from the following three aspects:

- **Flexibility** – As the NGWN should support a wide range of services with different QoS requirements, it will require the advanced customization of network management to support multifarious applications. In the 5G era, network slicing provides the potential to create customized virtual networks on top of the physical networks for different service types with various QoS requirements, which can achieve service differentiation and satisfy service level agreement for each service [1]. Most of the existing network slicing techniques concentrate on core networks. While slicing in the core network is essential, achieving QoS isolation and improving network resource utilization in RAN are also necessary [13]. However, few investigations have been conducted on slicing-based network management for RAN, especially in the planning

stage [25]. This is because mutually-dependent network configurations (e.g., BS coverage and power) significantly complicate RAN slicing. Consequently, in the NGWN, the advanced customization of network management for RANs is required to support multifarious applications flexibly.

- **Fine-granularity** – Achieving fine-grained network planning to realize high utilization of network resources in the NGWN is the second requirement. The QoS requirements of new applications in the NGWN are increasingly stringent, leading to the growing demand of network resources in achieving QoS guarantee. To support a large number of applications with stringent QoS requirements, improving network resource utilization is imperative. Existing network planning approaches are coarse-grained, which are mostly based on aggregated information of MUTs, such as the number of MUTs covered by a BS instead of the features of individual MUTs, e.g., mobility patterns [26]. The resulting estimation of service demands are inaccurate, which may degrade network resource utilization. However, fine-grained network management can yield high-dimensional network status, such as fine-grained spatiotemporal service demands, in the optimization problem, which poses a challenge to existing network management methods [3].

- **Adaptivity** – Network management in the NGWN should adapt to a highly dynamic network environment. A few research works have paid efforts to address spatiotemporal network dynamics in 5G, but they may not suffice for the NGWN due to the following two reasons. First, the main focus of 5G is to manage terrestrial networks. With the integration of non-terrestrial networks and terrestrial networks, the high mobility of satellites and UAVs results in a higher network dynamics in the NGWN than that in 5G [6]. Second, the network architecture and the network resource in the NGWN are multi-tier (e.g., space-air-ground integrated networks and multi-tier computing servers) and multi-dimensional (e.g., computing, storage, and communication), respectively, which further challenge an adaptive network management approach [18, 27]. Evidently, network management for heterogeneous and dynamic networks requires a radical shift in adaptivity.

1.2.2 Limitations of Model-driven Network Management

Recognizing the network management requirements in the NGWN, we further summarize the limitations in the state-of-the-art paradigms to comprehend the necessity of new network management paradigms. Until 5G, network management was primarily based on *model-driven* or heuristic methods [3, 28]. Optimization methods and game theory have

been widely adopted as mathematical tools for addressing network management problems. While such model-driven methods have promoted the development of effective network management in the past several decades, they may be insufficient for the NGWN for two reasons. Firstly, the NGWN will become complex, leading to a large number of variables and complicated correlations among them. Identifying the correlation among lots of variables while achieving low computation complexity poses a tremendous challenge for existing model-driven methods [29]. Secondly, modeling the network dynamics accurately and appropriately is challenging or even impossible for model-driven methods in an uncertain and highly dynamic network environment. Even if an appropriate model is given, convergence or equilibrium may not be guaranteed in a highly dynamic environment [1]. Consequently, we are reaching the point where network dynamics and heterogeneity, as well as the QoS requirements that we expect from the communication system, will exceed the capabilities of current model-driven network management paradigms.

1.3 Data-driven Network Management

The need for flexible, fine-grained, and adaptive network management motivates the use of artificial intelligence (AI) technologies to empower NGWN [3, 28]. The past decade has witnessed rapid advancement in the research of machine learning (ML), one of the most powerful AI tools. ML broadly refers to an algorithmic technology that endows a system with the ability of leveraging extensive data instead of explicit mathematical models, which supports a wide range of applications, such as computer vision and natural language processing [30]. Traditional ML methods have been employed in the research field of communication and networking. Lots of traditional ML-based methods, e.g., Bayesian learning, decision-tree learning, and rule-based learning, have been developed for managing conventional networks. Nevertheless, such traditional ML methods still rely on mathematical models (e.g., Bayesian theory) and problem-specific assumptions, which limits their ability to address complicated problems while making them problem-specific and inapplicable to other problems [28].

Recent advancements in deep learning (DL) and digital twins (DTs) have made it possible to shift from model-driven to *data-driven* network management. DL, as an inherently data-driven ML approach, implements the learning process by elaborating the data through deep neural networks (DNNs) [31]. In contrast to traditional ML methods, DL methods do not require prior knowledge of mathematical models for designing DNNs, and DNNs can be generalized in different scenarios. Lots of research works have shown that DL methods outperform transitional ML methods in different scenarios, e.g., object detection and speech recognition, especially when a large amount of data is available [32]. Recently,

DT techniques offer the potential to address the limitation of data-driven methods in their high reliance on datasets. A DT can be characterized by a full digital representation of a physical object or a process, which contains a large volume of data from the physical object for advanced data analytics [33, 34]. Integrating DL and DT paves the way to implement data-driven methods in practice. Such data-driven methods have started attracting the attention of the communication and networking community to employ data-driven network management paradigms in the NGWN [1]. Although the investigation of data-driven network management is still in its infancy, its potential to improve the flexibility, fine-granularity, and adaptivity of network management is tremendous due to the following reasons.

1) Data-driven methods can provide close-to-optimal solutions for the network management problems which are infeasible or too complicated for model-driven methods. For example, as we mentioned in Section 1.2, the problem of network management, e.g., the RAN slicing problem, could become complicated due to highly coupled variables when we aim to improve flexibility and fine-granularity. Data-driven methods offer the potential to address this challenge from two aspects. First, deep supervised learning is capable of approximating complicated functions [35]. A dataset consists of historical network statuses and optimal network management decisions that are obtained offline using brute force or a complex model-driven method. As long as the labeled data are adequate, deep supervised learning can offline learn the mapping function from network status to the optimal network management decision and, then, online infer the management decision with low computational complexity [36]. Furthermore, even if extensive labeled data are not available for training, deep unsupervised learning, e.g., auto-encoder, can be used for feature extraction to explore implicit network characteristics and reduce the network status dimensions [37]. The flexibility and fine granularity of network management can therefore be improved as a result of using data-driven methods.

2) By collecting and analyzing real-world data, data-driven methods are able to explore network dynamics, which can help develop or fine-tune the policies for highly dynamic network environment. When the network environment changes too rapidly, and the network dynamics are unknown *a priori*, choosing an appropriate model for network management is infeasible. Deep reinforcement learning (DRL) can be used to learn the network dynamics from historical data, including network status, decision making, and network performance, and adjust the decision making adaptively according to the learned network dynamics online [38]. DRL algorithms can be guaranteed to converge to the optimal policy when the network environment is stationary [39]. Even if network environment are not stationary, DRL can still be

used to adapt to the network environment in an online manner when the collected historical data is adequate [40]. Thus, data-driven DRL methods are expected to improve the adaptivity of network management in the NGWN.

3) DTs can be used for networking to provide a large volume data. To be specific, DTs can be introduced to represent individual MUTs in the NGWN. Each DT consists of data that describes the corresponding MUT, including the MUT's mobility, service demand, and QoS satisfaction, and functions for data processing and analysis [3]. Data contained in DTs can be used to predict MUT status, which can, in turn, facilitate customized network management for highly diversified MUTs and enable fine-grained network management. Moreover, DTs can acquire the necessary amount of data to facilitate effective data-driven methods to enhance network performance. Therefore, DTs for networking are envisioned to support fine-grained network management while empowering above DL methods in the NGWN.

1.4 Research Motivations and Contributions

Although data-driven methods provide possible solutions to improve flexibility, fine-granularity, and adaptivity, the design of data-driven network management solutions in the NGWN still meets several challenges. In this section, we first introduce the challenges of designing data-driven network management solutions. Then, we outline three research efforts to address the challenges.

1.4.1 Challenges of Data-driven Network Management in NGWN

Designing data-driven network management solutions in the NGWN still faces the following challenges:

- **QoS Guarantee** – In the NGWN, network management should satisfy the QoS requirements of various services while achieving service isolation, which brings a two-fold challenge. First, different services have different QoS requirements, e.g., signal-to-interference-and-noise ratio (SINR) requirements. Enabling customized network configurations for multiple services while satisfying their different QoS requirements is challenging since the network configuration supporting any service affects other network configurations that are used to support other services. For example, the communication for one service can interfere with the communication of other services.

Second, MUTs may have different mobility patterns, resulting in different network resource usage. Customizing the resource management for MUTs based on their different mobility patterns while satisfying their diverse QoS requirements remains challenging. However, the design of data-driven network management solutions to ensure QoS is not straightforward because the commonly used data-driven methods are not able to provide closed-form solutions. It is necessary to explore the use of data-driven network management to overcome the aforementioned challenges in the NGWN.

- **Complex Decision Making** – The decision variables of network management are highly coupled with each other. From the perspective of resource management, the NGWN should provide computing, storage, and communication resources for the supported applications, yet managing multi-dimensional resources is coupled. For example, a BS's computing capacity determines how many computing tasks it can handle and how much communication resource is required, and the storage capacity of each BS affects the amount of communication resource used for downloading context data from other servers [41, 42]. From the perspective of network configuration, the decision variables of BS configuration are mutually dependent due to the co-channel interference between BSs. The number of the dimension of decision variables can be further increased when network slicing is utilized for each BS for multiple applications. Data-driven methods are more suitable for solving complex problems than model-driven methods, but we still need to investigate how to design specific data-driven methods based on the specific coupling relations in the network management problems.

- **Resource Constraints** – While BSs will become resourceful in terms of not only communication but also computing and storage in the NGWN, network resources are still limited when there are a large number of connected MUTs with stringent QoS requirements. First, leveraging limited network resources to satisfy multiple strict QoS requirements simultaneously is challenging. Thus, designing a dedicated mechanism to enable resource multiplexing or sharing for improving network efficiency, e.g., resource utilization, is non-trivial. Second, leveraging UAVs or satellites to provide services in the NGWN is energy-consuming, and UAVs and satellites are mostly energy-constrained. Another challenge is improving service performance, e.g., latency, without violating the energy budgets of UAVs or satellites. In most cases, data-driven methods are used to address single-objective problems without constraints, but they are less often used to address multi-objective problems or to address single-objective problems with constraints. [43]. Therefore, satisfying recurse constraints in the design of data-driven network management solutions should

be further studied.

- **Dynamic Service Demands** – Highly dynamic service demands, e.g., computing task requests from MUTs, bring a two-fold challenge. First, re-configuring networks (e.g., adjusting resource reservation strategy) is required to adapt to the dynamic service demands, but it could yield additional cost of network resource usage. Achieving adaptive network management while balancing the network performance and the additional cost from network re-configuration is challenging. Second, to handle dynamic service demands, making network management decisions should take the upcoming service demands into account, which leads to a long-term sequential decision-making problem. However, solving the problem with uncertain and dynamic service demands is complicated. Multidimensionality of network resources and network heterogeneity exacerbate the problem of handling dynamic service demands in the NGWN. Even though data-driven methods are advantageous for accommodating dynamic environments, when networks become complex, they still face challenges of high computational complexity and slow convergence [29].

1.4.2 Research Problems and Contributions

While data-driven methods have advantages over conventional model-driven methods, this does not mean conventional model-driven methods should be dismissed in the NGWN. The data-driven methods should be used in synergy with conventional model-driven methods and well-designed for network management. In this thesis, we focus on three research problems in the NGWN and contribute to designing data-driven network management solutions that can improve the network resource utilization and service performance.

1) We first investigate planning-stage RAN configuration to achieve QoS guarantee for multiple communication services, as shown in Problem 1 in Fig 1.2. In the considered scenario, all BSs in the RAN share the same radio spectrum to support communication services with different SINR requirements. With the network slicing technique, a network slice is created for a communication service. Different network slices have different spatiotemporal distributions of data traffic. Due to the co-channel interference among network slices, handling the dynamic spatiotemporal distributions of data traffic for all network slices while fulfilling their SINR requirements is challenging. Our research objective is to maximize the energy efficiency of terrestrial BSs by jointly configuring downlink transmission power and communication coverage for each BS. To achieve this objective, we first design a network planning scheme with three novel ideas, including flexible binary slice zooming, dual time-scale planning,

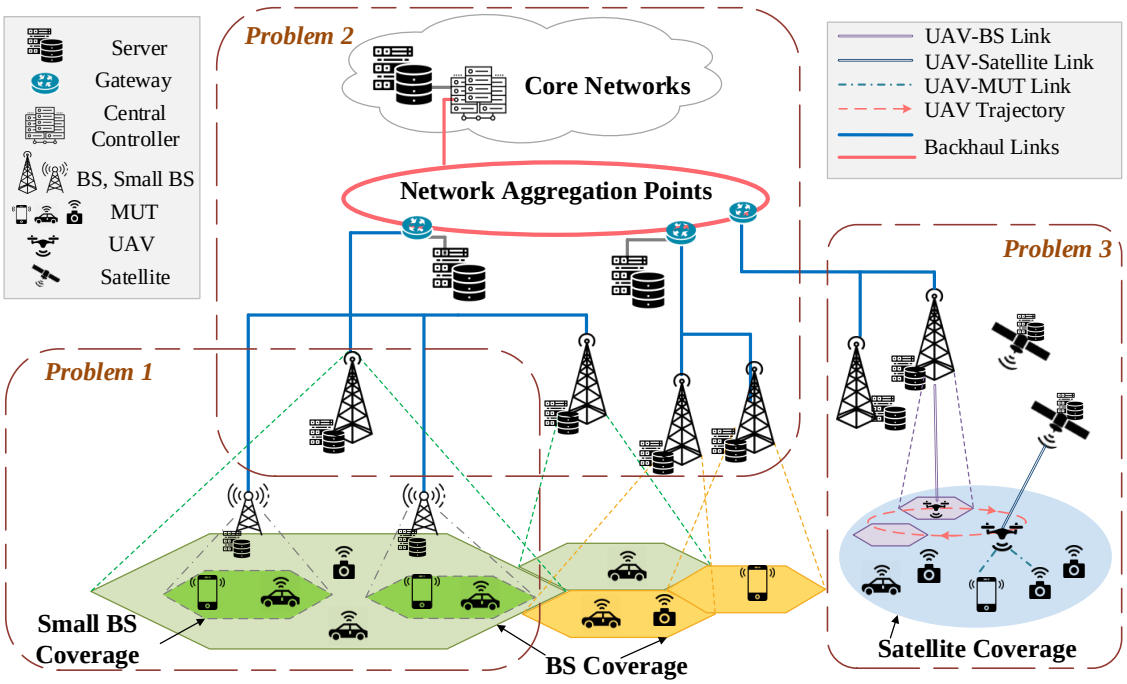


Figure 1.2: The illustration of network management for the NGWN.

and grid-based network planning. The scheme introduces flexibility to differentiate the communication coverage and downlink transmission power of the same BS for different network slices while improving the temporal and spatial granularity of network planning. With the proposed scheme, we formulate a combinatorial optimization problem in which communication coverage management and power control are mutually dependent. To solve the problem, we propose a hybrid data-model-driven method with two steps. In the first step, we propose an unsupervised-learning-assisted approach to determine the communication coverage of BSs; In the second step, we derive a closed-form solution for power control. Simulation results demonstrate that the proposed network planning scheme outperforms conventional planning schemes in energy efficiency and approaches the optimal solution in performance.

2) We then investigate planning-stage resource reservation to support a compute-intensive service with taking into account the mobility of MUTs, as shown in Problem 2 in Fig 1.2. In the considered scenario, mobile MUTs can offload their computing tasks

to the computing servers deployed at the core networks, gateways, and BSs. Each computing server requires both computing and storage resources to execute computing tasks. The mutual dependency among different resources and the impact of MUT mobility create challenges in proactive resource reservation. Through adaptively reserving adequate network resources for dynamic service demands, the research objective is to minimize the usage of computing, storage, and communication resources as well as the cost of re-configuring resource reservation. To this end, we develop a hybrid data-model-driven network planning scheme with two elements, i.e., multi-resource reservation and resource reservation re-configuration. First, DTs are designed for collecting MUT status data, based on which MUTs are grouped according to their mobility patterns. Second, an optimization algorithm is proposed to customize resource reservation for different groups to satisfy their different resource demands. Last, a meta-learning-based approach is developed to re-configure resource reservation for balancing the network resource usage and the re-configuration cost. Simulation results demonstrate that the proposed network planning scheme outperforms benchmark schemes by using less network resources and incurring lower reconfiguration costs.

3) Lastly, we investigate operation-stage computing resource allocation in a dynamic space-air-ground integrated network (SAGIN), as shown in Problem 3 in Fig 1.2. In the considered scenario, MUTs in remote areas need to offload their computing tasks to computing servers, but terrestrial BSs provide limited communication coverage for the MUTs. Since satellites can provide global coverage, and UAVs can provide flexible service delivery, satellites and UAV can be used to complement the terrestrial networks in the remote areas. A UAV is deployed to fly around MUTs and collect their computing tasks, while scheduling the collected computing tasks to be processed at the UAV locally or offloaded to the nearby terrestrial BSs or the remote satellite. The energy budget of the UAV, intermittent connectivity between the UAV and terrestrial BSs, and dynamic computing task arrival pose challenges in computing task scheduling. Our objective is to design a computing task scheduling policy for minimizing the delay of computing task offloading and processing in the SAGIN. To achieve the objective, we first formulate the online computing scheduling in the dynamic network environment as a constrained Markov decision process. Then, we develop a risk-sensitive DRL approach in which a risk value is used to represent energy consumption that exceeds the budget. By balancing the risk value and the reward from delay minimization, the UAV can explore the task scheduling policy to minimize task offloading and processing delay while satisfying the UAV energy constraint. Simulation results show the proposed data-driven resource allocation

approach reduces the delay of task offloading and processing while not exceeding the UAV's energy budget compared with conventional schemes.

1.5 Thesis Outline

The remainder of the thesis is organized as follows: In Chapter 2, we provide a comprehensive background and review of network management and the state-of-the-art network planning and network operation technologies. In chapter 3, based on unsupervised learning, we design a flexible network configuration scheme in the planning stage to support the customized communication coverage management and power control for different communication services. In chapter 4, based on meta learning, we propose a fine-grained and adaptive resource reservation scheme in the planning stage to support a computing-intensive service with considering MUT mobility. In chapter 5, based on reinforcement learning, we design an adaptive resource allocation scheme for a compute-intensive service in highly dynamic space-air-ground integrated networks. Finally, we conclude the thesis and discuss future works in Chapter 6.

Chapter 2

Literature Review

This chapter aims to provide a comprehensive background and review of network management and the state-of-the-art network management methods in the planning stage and the operation stage.

2.1 Planning-stage Network Management

Academia and industry aim to design networks and adjust network configuration adaptively for reducing both capital expenditure (CAPEX) and operation expenditure (OPEX) while satisfying dynamic service demands [14]. In the early stage, deploying network infrastructures and configuring communication networks are referred to as network management in the planning stage, i.e., network planning. Network planning decisions are made once when networks are constructed and no longer altered. The network planning decisions are mostly offline and cannot adapt to the dynamic network environment. To deal with the unpredicted network dynamics, the concept of network management in the planning stage has been extended to online but "slow-reaction" network planning as communication networks become complex and need to support the increasing number of network functionalities. Therefore, in addition to offline network planning, online long-term network configuration/adjustment (several tens of minutes to several hours) can also be considered as network management in the planning stage nowadays [13]. The existing works concentrating on network planning mainly consist of three aspects: cellular planning, virtual network (i.e., slice) configuration, and resource reservation [1].

2.1.1 Cellular Planning

Cellular planning problem is a classical problem in network management. Most works on cellular planning aim to determine the number of BSs and BS locations and provide the essential *planning parameters*, e.g., BS coverage, users clustering, and BS radiation patterns [16]. However, cellular planning problem is complex and non-scalable since lots of variables affect the optimization of cellular planning, and the variables are coupled. Just determining the location of BSs has been shown to be NP-hard [44]. To address the complex cellular planning problem, existing works decompose the planning problem into multiple subproblems and then focus on addressing a subproblem while considering a certain number of planning parameters and network constraints. The existing works are classified according to their investigated sub-problem in cellular network planning as follows.

A. BS Deployment

Determining the number of BSs is investigated in several existing works [45–48]. The minimum number of conventional BSs and the locations of these numbers are jointly determined to satisfy the given data traffic demands in [45]. Authors in [46] find the near-optimal locations of the minimum number of BSs to be deployed in LTE while serving all users and satisfying the constraints of BS resource capacity based on particle swarm optimization. Guo *et al.* leverage Poisson point process to model the real deployment of BS and adopt the outage probability as the metric to determine the number of BSs and their locations [47]. To minimize the overall power consumption of BSs, authors first optimize the locations of BSs considering a fixed number of BSs and then identify the redundant BSs and remove them while satisfying the uplink and downlink SINR requirements [48]. Moreover, lots of works pay attention to optimize the number of drone-based BSs or UAVs for NGWN [49–51]. A low-complexity algorithm is proposed to minimize the number of drone-based BSs while covering all given users [49]. The minimum number of drone-based BSs and their 3-D locations are optimized to provide communication coverage to a set of users with different SINR requirements [50]. In [51], the authors first divide the 3-D space to be covered into octahedron shapes of equal dimensions. Then, the authors adopt a statistical learning approach to build a statistical model on the user distribution, and use the model is used to associate users to BSs to minimize the latency averaged on all users.

Given the fixed number of BSs or drone-based BSs, optimizing their locations is investigated in lots of existing works [52–54]. The overall communication coverage area of all BSs are maximized by determining the latitude and longitude of all drone-based BSs based on different algorithms in [44, 52]. Considering not only the latitude and longitude of each

drone-based BS but also the height of each drone-based BS, a 3-D drone-based BS deployment problem is investigated to maximize the number of users within the service coverage areas given the fixed locations of users [53]. Considering two BSs with the fixed height, the impact of minimum distance between two drone-based BSs on the overall communication coverages areas of the two drone-based BSs is investigated in [54].

B. Planning Parameter Configuration

For relatively simple network scenario, the planning parameters of each BS, e.g., communication coverage area, uplink/downlink radiation patterns, and long-term power control, can be set once and have no need to be altered again [55]. However, when communication networks become complex and heterogeneous while the network environment is highly dynamic, fixed network configuration cannot adapt to dynamic network environment. In contrast to that changing BS locations is mostly infeasible, planning parameters can be adjusted adaptively for network dynamics. As a result, planning parameter configuration can be considered as configuring/reconfiguring the planning parameters of BSs over a long-term period (e.g., several minutes to several hours). Some existing works investigate BS coverage management in conventional wireless networks [56, 57]. Generally, long-term user association can be considered in BS coverage management in wireless networks. For example, the default user association scheme in cellular networks maximizes signal-to-interference-and-noise ratio (i.e., max-SINR scheme), which maximizes the probability of coverage and minimizes the probability of outage. In [56], a user association scheme is proposed to balance the data traffic load among BSs with fixed user locations. From the energy efficiency perspective, authors in [13] present the necessity and importance of changing the communication coverage of BSs by switching on/off BSs dynamically according to service demands. In [57], a scheme of BS coverage management, named cell zooming, is proposed to improve the energy efficiency of BSs. The coverage of any BS can be either zero or its maximum geographical coverage, corresponding to turning the BS on or off.

Some existing works study long-term transmission power control in conventional one-tier wireless networks [58, 59]. Giovanni *et al.* optimize long-term transmission power control to maximize the sum data rate of all users [58]. Specifically, the coverage area of each BS is divided into three regions, and the transmission power of each region over a long-term period is chosen from three power levels. The values of power levels are determined based on the average number of users in the corresponding region over the long-term period. In [59], considering frequency reuse and co-channel interference, the authors investigate power control to adjust the radius of BS coverage for the purpose of load balancing. The power control is determined based on a stochastic model of user location

distribution over a long-term period. The planning parameter configuration problem is also investigated in two-tier networks [60]. Given a certain number of pre-defined locations within the networks, long-term transmission power control and frequency bandwidth are jointly optimized for macro and small BSs to improve the number of locations where the data rate is satisfied while satisfying the communication resource usage budget [60].

In addition, Wu *et al.* propose a cross-layer approach to jointly optimize the timesharing in the medium access layer and the sum of max of flows assignment in the network layer for minimizing aggregate congestion while minimizing power consumption [61].

C. Load Balancing

Load balancing is an important research objective in cellular planning, which aims to associate the service demands to multiple BSs as even as possible for improving network performance, e.g., latency [14]. Planning parameter configuration is a possible way to achieve load balancing in the planning stage, but the technologies for load balancing in the planning stage are different from those in the listed works. Lots of works focus on configure planning parameters to balance the load among multiple BSs in homogeneous communication networks [62]. Based on the gathered information on the communication resource usage at each BS, the handover parameter of each BS can be customized to reduce the number of overloaded cells [62]. To be specific, a customized threshold of resource usage can be determined for each BS based on resource usage of all BSs, and users can handover among BSs based on the determined thresholds. Furthermore, some works investigate load balancing in heterogeneous networks [63, 64]. The optimal cell re-selection offset for each BS is investigated to determine the user association for the BS, which can support data traffic load balancing between BSs with LTE heterogeneous networks [63]. To mitigate co-channel interference among macro BSs and small BSs in LTE heterogeneous networks, authors in [64] propose a distributed algorithm to mute certain subframes at macro BSs while balancing the data traffic load among macro BSs and small BSs for improving users' data rates.

2.1.2 Virtual Network Configuration

Network slicing technology has been proposed and leveraged for 5G networks to support massive services with diversified service requirements [1]. Traditional communication networks, e.g., LTE, employ the one-size-fits-all approach for all services, regardless of their different service requirements, while network slicing is to create multiple virtual networks,

i.e., slice, on top of the physical (or substrate) networks [19]. Each virtual network is a combination of virtual nodes and virtual links, and virtual nodes are interconnected through virtual links, forming a virtual network topology [20]. Each virtual network can be managed and configured in a highly flexible and customized way to support a service based on its service requirement. As a result, many works focus on configuring virtual networks in the planning stage. Similar to the aforementioned planning parameter configuration, virtual networks can be configured/reconfigured dynamically. In this subsection, we first introduce the background of network slicing in detailed and then survey the existing works on virtual network configuration, i.e., slice configuration.

A. Network Slicing

Network slicing relies on the concept of network virtualization, which can be traced back to the 1960s when the first operating system was developed by IBM [65–67]. This system providing time-sharing and virtual memory to fifteen users simultaneously introduces a breakthrough in computing. With the development of hardware and software, the idea of virtualization was widely adopted for data centers by the early 80s [68]. This idea aimed to form the vision of virtual systems spanning across computing platforms, network resources, and storage devices by creating a virtual form of a physical entity through software methods and processes [69]. This technology was gradually applied to networking for connecting remote sites securely with controlled performance through the Internet. The introduction of overlay networks in the late 80s that consist of nodes connected over logical links forming a virtual network over a network composed of physical infrastructures (i.e., underlay networks) can be seen as an early form of network slicing, combining heterogeneous resources over various administrative domains. Overlay networks provide QoS guarantees in a service-oriented fashion. They are flexible in nature but not automated nor programmable. By 2000, by allowing users to obtain isolated application-specific slices, the first-generation platforms for verifying and evaluating new network protocols were established based on overlay networks. A slice was defined as a unit component with allocated resources such as computation capability on servers. However, such overlay platforms had limitations in underlay network controls [69].

By 2010, with the advancement of Internet technologies, the virtualization of both overlay networks and underlay networks are improved for core networks [70]. On the one hand, cloud computing is proposed to provide advanced virtualization of resources. Not only physical network infrastructure resources can be virtualized, i.e., infrastructure as a service, but also online platforms and commonly-used computing services, i.e., platform as a service and software as a service, can be virtualized to use and manage easily. On the

other hand, software-defined networking (SDN) and network function virtualization (NFV) technologies are proposed to enable flexible and programmable network management [71]. Specifically, SDN provides a separation between the network control and data planes, improving the flexibility of network function management and efficiency of data transfer [72]. NFV allows various network functions to be virtualized, i.e., in virtual machines, and moves the functions to different locations [73]. Based on such virtualization technologies, network slicing can be enabled to improve service provisioning and resource utilization for core networks via flexible resource sharing, dynamic network function instantiation, and agile network management.

Due to the need for more flexibility and elasticity of networks for supporting diverse services, the 3GPP has reshaped the core network in the 5G era completely and built several specifications, e.g., network slice selection and network function sharing, to enable network slicing through the creation of core network instances for different types of services [74]. To be specific, based on SDN and NFV technologies, network functions in conventional networks such as LTE have been virtualized as different virtual network functions. Such virtual network functions can be deployed, configured, migrated, and removed in a flexible way to support a slice. The virtual network functions in 5G can be divided into two groups. The first group comprises the virtual network functions handling basic services of the communication networks, such as user reregistration and mobility management. The second group includes the virtual network functions designed specifically for a slice, which cannot be shared among multiple slices. As a result, each slice can be customized and configured flexibly to support a service based on its service requirements, which brings more flexibility, elasticity, and QoS assurance to the communication networks [75].

B. Virtual Network Configuration

Generally, communication networks consists of core networks and RAN. We introduce existing works on virtual network configuration for core networks and RAN, respectively.

- *Core Networks* – In the virtual network configuration for core networks, most existing works focus on mapping virtual resources onto physical network infrastructure, referred to as the *virtual network embedding* problem. According to the type of virtual resources, the virtual network embedding problem can be further classified as two sub-problems *virtual node mapping*, where virtual resources at virtual nodes have to be mapped onto physical network nodes, and *virtual link mapping*, where virtual links connecting these virtual nodes have to be mapped to virtual paths connecting the corresponding nodes on top of the physical network. Solving the virtual network embedding problem is NP-hard [20]. Even if the virtual node mapping is given,

mapping the virtual links to a physical network is still NP-hard. Therefore, virtual network configuration is still challenging and needs to be further investigated.

Most existing works on virtual network configuration for core networks are summarized as follows. Some works pay attention to virtual node mapping, such as virtual network function placement. Authors in [76] design a demand-supply model to quantify the data rate performance degradation due to the consolidation of virtual network functions, and then place the functions on 5G core networks to maximize the overall data rate of two different slices. Two different algorithms of virtual network function placement are proposed for two objectives, i.e., path minimization and session continuity in [77]. A virtual network function placement algorithm is developed for unicast and multicast services to maximize amortized throughput subject to resource constraints on physical links and nodes [78]. Considering the end-to-end link, including core networks and the RAN, a network function placement is developed for radio units, distributed units, centralized units, and 5G core to minimize the power consumption while satisfying users' QoS requirements [79]. Furthermore, a few works focus on virtual link mapping, e.g., service or network function chain placement. Considering user mobility, authors joint optimize user association and network function chain placement to minimize end-to-end delay of users in 5G networks [80]. Qu *et al.* propose propose a delay-aware flow migration approach to guarantee average delay isolation among multiple network slices within maximal tolerable service downtime [81]. Moreover, some existing works pay efforts on service function chain placement [82]. Jang *et al.* investigate joint service function placement and flow distribution to maximize the flow rate while minimizing the energy consumption for multiple service function chains [83].

In addition to virtual network embedding problem, specific virtualization architecture or detailed procedures to support network slicing are investigated. A holistic network virtualization architecture based on AI is proposed to support user-centric networking [3]. Several technical requirements regarding performance and manageability to enable network slicing in the core networks are discussed in [75]. Authors in [84] reshape the network functions into more granular functions to break down the monolithic core network functions into more modular functions that constitute a control plane service. In [85], the authors show how the modular functions could be composed to build a control plane service tailored to a network slice. Some works address the problem of network slice selection to associate a user with the appropriate slice instance. In [86], the authors have defined the procedures to enable network slice selection and discovery for individual users.

- *Radio Access Networks* – While the advantage of network slicing is common, and

3GPP has completed specifications regarding network slicing for 5G core networks, the realization of network slicing in RAN still poses multiple open issues. A few works aim to design and realize RAN slicing in NGWN. In [87], lots of requirements for RAN slicing are discussed regarding data traffic differentiation for different users in the same slice, resource multiplexing among slices, and protection mechanisms to reduce inter-slice effects. To fulfill such requirements, architecture, protocol, and management for RAN slicing should be re-designed. For protocol design and slice configuration, physical, link, and network protocol layers are designed to manage radio resource allocation to users within the coverage of a BS while guaranteeing slice-specific isolation and support customized resource management configurations for users in different slices [88]. Different options of slice granularity and physical layer and MAC layer configurations in RAN are proposed for different service types in [89]. As for network architecture, an AI-based network architecture is proposed to support RAN slicing [1]. For the Cloud-RAN architecture, a novel network function placement approach is proposed to support RAN slicing, which can enable service differentiation in network function placement based on different service requirements [90]. To enable network management for RAN slicing, a joint BS deployment and frequency spectrum planning approach is proposed in [91] to satisfy data traffic demands for multiple slices with different spatial traffic distributions. A K-means based approach is to cluster users as a slice based on their mobility patterns, and configure radio resources on different BSs to serve different slices [92].

A few existing works focus on designing resource management frameworks for RAN slicing in different types of networks [25, 93, 94]. To support Internet of vehicles services in vehicular networks, a dynamic RAN slicing framework is presented to dynamically allocate radio spectrum and computing resource, and distribute computation workloads for multiple slices [94]. Ye *et al.* investigate a resource management framework for a two-tier cellular network, which enables user association and bandwidth reservation among BSs in the planning stage to satisfy differentiated service requirements of slices [25]. In a practical network environment, an service provisioning approach framework is designed for slice association and resource allocation to maximize bandwidth utilization while satisfying QoS requirements of users [93].

2.1.3 Resource reservation

Resource management, as a part of network management, plays an important role in improving network performance in both conventional communication networks and NGWN [1]. In the early stage, resource management in the planning stage for conventional communi-

cation networks, e.g., 3G and LTE, is referred to as determining the quantity of communication resource reserved for each BS proactively, e.g., determining the radio frequency bandwidth of a BS. Similar to the extended concept of network planning, the concept of resource management in the planning stage is extended to both offline and online long-term resource reservation [13]. Note that the terminologies "resource provisioning" and "capacity planning" adopted in some works are sometimes equivalent to "resource reservation". Generally, long-term resource reservation in the planning stage is based on aggregated information of users, such as the number of users covered by a BS, and focuses on network performance, such as resource utilization over a relatively long time period, ranging from several minutes to hours [95]. However, compared with resource management in the operation stage, only few works pay attention to resource reservation in the planning stage. Existing works on long-term resource reservation consists of three directions: centralized, decentralized, and hybrid.

- *Centralized* – Most works on resource reservation use a centralized controller to reserve network resources for different users or slices based on their service demands. To support communication services in RAN, lots of resource reservation approaches are proposed for different layers and wireless technologies. Resource reservation schemes in both network and MAC layers are designed for IEEE 802.111 wireless technologies to enable QoS assurance [96]. A semi-random back-off mechanism is proposed for resource reservation for channel access in contention-based wireless local area networks [97]. Chang *et al.* group users based on their hand-off probability and enable group-based bandwidth resource reservation to guarantee QoS for each group [98]. For network slicing, Bega *et al.* leverage a data-driven solution to predict the amount of resources needed to accommodate future demands at each network slice and reserve resources for the slices based on prediction results [99]. A joint long-term spectrum reservation and real-time radio resource allocation are investigated to improve the network resource utilization while stratifying the service requirement of each slice [100]. Reyhanian *et al.* jointly optimize communication resource reservation in RAN and core networks to satisfy uncertain user demands [101].

  A few works pay attention to joint computing and communication resource reservation due to the increasing number of diversified computing services needed to be supported in communication, especially NGWN. Based on the aggregated computing demands from all access points, a proactive computing resource reservation approach in MEC is designed to minimize the delay of executing computing tasks [102] and maximizing resource utilization in computing task execution [103], respectively. Yin *et al.* study edge server placement to minimize the network resource usage based on statistical computing demands [104]. There are limited works on long-term resource

reservation in multi-tier computing [105, 106]. Considering edge and cloud computing, Zhou *et al.* propose a computing resource reservation approach to minimize network resource usage while satisfying different delay requirements of two applications [105]. For servers located at different tiers of space-ground integrated networks, joint communication and computing resource reservation is studied to minimize the long-term cost of delay requirement violation and network reconfiguration [106].

- *Decentralized* – Some works investigate decentralized resource reservation for user privacy and computation complexity [107, 108]. In particular, each user can determine and request the required network resources based on its individual objective and requirements, and the network controller only determines whether the request succeeds or not. Since the network controller does not require user information, e.g., users' objectives and requirements to enable resource reservation, the decision-making would be computationally inexpensive and not expose private information.

- *Hybrid* – One work investigates hybrid resource reservation [109]. Specifically, users can provide partial information to the network controller for centralized resource reservation. Then, users can perform their own resource allocation algorithm to select the optimal amount of network resources without considering other users' objectives and requirements.

2.2 Operation-stage Network Management

While many works have investigated network management in the planning stage, far more studies have put effort into the network management in the operation stage [22]. Generally, network management in the operation stage, i.e., *network operation*, is referred to as providing consistent network services to individual users, and real-time resource allocation is a key point in communication networks [110]. Specifically, resource allocation in the operation stage relies on real-time information on individual users, such as user locations, and targets real-time user satisfaction. Since compute-intensive applications are expected to be supported in the NGWN as mentioned in Section 2.1, computing task offloading and scheduling should be investigated for resource allocation in NGWN. Therefore, we summarize existing works on radio resource allocation and computing task offloading and scheduling in this section.

2.2.1 Radio Resource Allocation

Most works studied the radio resource (e.g., frequency spectrum, transmission power and resource blocks) allocation among different slices for cellular networks in the operation stage. We summarize the existing works based on different network scenarios, i.e., homogeneous and heterogeneous networks.

- *Homogeneous Networks* – Resource allocation for homogeneous communication networks can be further classified into the cases with single BSs or multiple BSs. In a homogeneous network, the same radio access technology, e.g., cellular or WiFi, are applied for all BSs, and all BSs have the similar configuration, e.g., radio radiation patterns. For the case with single BS, a radio resource allocation scheme is developed to allocate RBs, which keeps track of the service contracts with the service providers and also the fairness requirements between cell-center users and cell-edge users [111]. To enable efficient spectrum slicing in the RAN domain, the unique physical layer configurations associated with each dedicated slice is investigated [112]. The authors systematically derive the relationships among most of the key physical layer parameters, including sub-carrier spacing, symbol duration, sampling rate, discrete Fourier transform size and waveforms for different scenarios. For the case of one cell comprising one slice for human type communications and one slice for machine type communications, random access based procedure and radio resource allocation are investigated to guarantee respective delay and throughput requirements of users [113]. To achieve the overall revenue maximization of slices, Han *et al.* propose a new stochastic model for network slicing that leverages on the multi-queuing system to optimally design an admission control of on-demand network slices as well as to orchestrate them once are accepted [114].

  Moreover, some resource allocation algorithms focus on the case with multiple BSs. To guarantee that the same (or similar in time/frequency) RBs are assigned to the same slice owners when multiple BSs are close enough to interfere among themselves, a NP-hard RAN slicing enforcement problem is formulated [115]. In this work, RBs are orthogonally allocated to slices to reduce inter-slice interference and enable advanced communication techniques (e.g., Coordinated multi-point transmission) and coordinated beamforming by maximizing the number of simultaneous transmissions on different BSs. Considering slices' loads may be spatially inhomogeneous and time varying rather than statistically partitioning radio resources at each BS, Caballero *et al.* provide an analysis of a distributed model for the radio resource sharing, the share-constrained proportional allocation mechanism, to realize network slicing in NGWNs [116]. Considering a criterion for resource allocation amongst slices, i.e.,

weighted proportional fairness, a dynamic radio resource slicing scheme is proposed to achieve desirable fairness across the network slices of the different tenants and their associated users [117]. Meanwhile, the Pareto-optimality of user association to BSs, the fair allocation of base stations' resources, and the gains resulting from this scheme are established in this work.

- *Heterogeneous Networks* – In contrast to homogeneous networks, heterogeneous networks are more complex since multiple access points with different radio access technologies, e.g., LTE heterogeneous networks, can provide network services to the same user [118]. Tseliou *et al.* propose a resources negotiation for network virtualization algorithm, which is suitable for application in LTE heterogeneous networks consisting of a Macro BS overlaid with multiple Small BSs [119]. The challenge of traffic variations in geographical dimension is addressed by appropriately slicing radio resources in terms of resource blocks and transferring them among multiple BSs. With the objective that is how to efficiently allocate resources over both licensed and unlicensed bands according to different QoS requirements of different services, two or more slices are allowed to access each other's licensed spectrum [120]. Specifically, multiple slices can then aggregate their distributed licensed bands to support traffic associated with the same type of service. Different slices can also negotiate and trade their rights to access unlicensed bands according to the estimated value.

  Compared with conventional communication networks, the NGWN are expected to provide seamless anytime and anywhere connectivity through integrating space, air, and ground network components. In particular, there are various initiatives to construct satellites and launch thousands of low Earth orbit satellites. Benefiting from global availability, satellite networks have the potential to supplement worldwide seamless high-bandwidth Internet connectivity in a cost-effective way. Meanwhile, the agility of aerial networks based on UAVs capacitates the flexible deployment of UAV networks in response to dynamic service demands and emergency situations, such as disaster relief or service congestion. Thus, the SAGIN is envisioned as a promising architecture to complement the terrestrial network for the NGWN. Lots of works have paid attention to resource allocation in the SAGIN [12, 121]. A data-driven network control architecture is designed to enable resource allocation for complex SAGIN environment [122]. Zhou *et al* propose adaptive access mode selection for space-ground integrated networks based on the different communication features of satellites and BSs [123]. For more complicated space-air-ground integrated vehicular networks, network slicing and SDN technologies are leveraged to support different QoS requirements of typical vehicular services via jointly proactive chancing and communication resource allocation [12, 121].

2.2.2 Computing Task Offloading and Scheduling

With the development of MEC and caching techniques, computing and storage resources are paid more attention. Several research works enable multi-dimensional resource allocation in the planning stage for supporting compute-intensive services. To develop a dynamic network resource allocation approach based on semi-Markov decision process framework which allows the network provider to jointly allocate computing, storage, and radio resources to different slice requests in a real-time manner and maximize the long-term reward under a number of available resources [124]. In this work, a deep reinforcement learning-based solution is proposed to deal with the dynamics of slicing requests, e.g., uncertain service time and resource demands. By forming a resource (including computation, memory and bandwidth) allocation auction between the slices and the data centers in 5G virtualized networks, Halabian *et al.* propose a distributed resource allocation approach for a heterogeneous cloud infrastructure [125]. Liang *et al.* formulate a virtual resource allocation and in-network caching strategy as an optimization problem, which maximizes the utility of slices [126].

Most works on real-time resource allocation focus on computing task offloading and service placement, among which many consider one-tier computing such as cloud computing or MEC [127–130]. Based on the real-time computing task arrival of each user, decentralized and centralized communication and computing resource allocation approaches are proposed to minimize the delay of computing task offloading and execution in cloud computing and MEC, respectively [127, 128]. Service placement is studied in MEC based on the real-time user location and the type of service required by each user to maximize the number of users that can be severed under each edge server's resource capacity [129, 130]. To solve the joint problem of partial offloading scheduling and resource allocation for mobile edge computing systems with multiple independent tasks, a two-level alternation method is proposed based on the Lagrangian dual decomposition [131]. To address the multi-user computation offloading problem for mobile-edge cloud computing in a multi-channel wireless interference environment, a distributed computation offloading algorithm is proposed based on a Nash equilibrium [132]. However, it is difficult for an optimization-based algorithm to adapt to the dynamic task arrival scenario since a fixed task number is required. Considering the stochastic task generation, Lyapunov optimization is leveraged in task scheduling schemes. Besides, an asymptotically optimal scheduling scheme is also proposed with partial knowledge in mobile edge computing scenarios by leveraging the Lyapunov drift [133]. In order to minimize the delay due to both radio access and computation, a user-centric energy-aware mobility management scheme is proposed based on Lyapunov functions, and multi-armed bandit theories [134]. The Lyapunov-drift-based techniques can schedule tasks to keep the task queue stable based on the current queue backlog. However, the optimality cannot be

guaranteed since the information of future status (e.g., future task arrival) is lacking.

Some works focus on resource allocation for multi-tier computing [135–139]. Computing resources on fog nodes and the cloud server are allocated to users at different locations to satisfy the delay requirements of their computing tasks [135]. Li *et al.* investigate a service placement approach for cloud and edge computing to satisfy each user's computing demands [136]. Given that the same type of computing tasks can share computing results, Yu *et al.* study joint computing task offloading and service placement in multi-tier computing to reduce the delay of executing computing tasks [137].

Only a few existing works focus on computing task offloading and task scheduling in the SAGIN. A cost-effective scheme for joint service placement and routing is proposed in [140]. To accommodate diverse services, resources of the satellite, aerial, and terrestrial components have been sliced, and a hierarchical resource management scheme is proposed to put available resources into a common and dynamic resource pool [12]. To meet the emerging computation-intensive Internet of things (IoT) applications with diverse QoS requirements, an air-ground integrated mobile edge network is presented to realize mobile edge computing [141]. In [138], to address uncertain channel conditions in remote areas, a DRL-based scheduling scheme is proposed for the virtual machine assignment and task offloading in the SAGIN. However, accommodating IoT computing task scheduling in the SAGIN still faces significant challenges since the computing task arrival from IoT devices is highly dynamic and random, and the management for both communication and computing resources is complicated.

Chapter 3

Planning-stage Service Coverage and Power Control for Network Slicing

In this chapter, we investigate planning-stage RAN configuration to achieve QoS guarantee for multiple communication services. Based on network slicing technique, a network slice can be created for a communication service on top of a physical network. In light of the dynamic spatiotemporal distribution of data traffic for all network slices, as well as the co-channel interference between them, it is difficult to meet the QoS requirements for all network slices at the same time. Our research objective is to maximize the energy efficiency of terrestrial BSs by jointly configuring downlink transmission power and communication coverage for each BS while fulfilling different SINR requirements for communication services. To achieve this objective, we first design a network planning scheme with three novel ideas, including flexible binary slice zooming, dual time-scale planning, and grid-based network planning. The scheme introduces flexibility to differentiate the communication coverage and downlink transmission power of the same BS for different network slices while improving the temporal and spatial granularity of network planning. With the proposed scheme, we formulate a combinatorial optimization problem in which communication coverage management and power control are mutually dependent. To solve the problem, we propose a hybrid data-model-driven method with two steps. In the first step, we propose an unsupervised-learning-assisted approach to determine the communication coverage of BSs; In the second step, we derive a closed-form solution for power control. Simulation results demonstrate that the proposed network planning scheme outperforms conventional planning schemes in energy efficiency and approaches the optimal solution in performance.

3.1 Background and Motivations

A major challenge for modern wireless networks is the accommodation of diverse services with various performance requirements. In networks beyond 5G, i.e., the NGWN, the diversity of services is expected to escalate, and the performance requirements will become more stringent [142]. To address this challenge, network slicing is proposed to support multiple coexisting virtual networks on the same physical network infrastructure [1]. Each virtual network is referred to a *slice*, corresponding to a specific service and possibly a unique set of performance requirements. Network slicing allows for highly flexible network management, as the creation, customization, adjustment, and annulment of slices are driven by network service demands and carried out in software [143].

Slicing-based network management consists of two stages: planning stage on a large time scale (e.g., several hours) and operation stage on a small time scale (e.g., several milliseconds) [1, 94, 144]. A planning stage reserves network resources (such as radio, computing, and storage resources) for slices based on a prediction of their service demands, and determines the network configuration such as the service coverage (SC) and transmission power of base stations (BSs) for the next planning period. The subsequent operation stage schedules the reserved resources of a slice to individual MUTs according to the real-time MUT locations and service demands. Therefore, the network management in the planning stage requires long-term and network-level information and optimization, while the network management in the operation stage requires real-time information of MUTs. Evidently, the network management in both stages affects the performance of networks [94]. However, the existing literature pays much more attention to the operation stage [23, 145], while only limited works target the planning stage [14]. This is partially due to the unique challenges in the planning stage, including but not limited to the difficulties in obtaining network-wide MUT information (e.g., MUT location distribution), handling spatiotemporal distribution of data traffic, and determining mutually-dependent network configurations (e.g., coverage and power). The above challenges must be addressed for all coexisting slices, and the result should demonstrate network-level optimality.

As the NGWN become more hierarchical, heterogeneous, and dense, effective network planning can play a major role in improving network resource utilization and service requirement satisfaction [1]. Motivated by the importance of network planning, we investigate the planning stage of slicing-based radio network management in this work. We consider a two-tier network with one macro BS (MBS) and multiple small BSs (SBSs), which supports multiple slices with different spatiotemporal data traffic distributions (DTDs) and service requirements. The objective is to maximize the energy efficiency of the network, by determining the downlink transmit power and SC of all BSs in the planning stage.

To achieve this objective, we first propose schemes to facilitate network planning, which provide an approach to control the temporal and spatial granularity of network planning. The schemes allow us to flexibly adjust the downlink transmission power and SC of BSs based on the spatiotemporal DTD of coexisting slices. With the proposed schemes, we formulate an energy efficiency optimization problem with a large number of SC and transmission power variables which are mutually dependent. To solve this problem, we adopt a two-step approach: In the first step, we propose an unsupervised learning-assisted SC search (ULSCS) algorithm to determine the SC of SBSs; In the second step, we derive a closed-form solution for power control. The contributions of this chapter are two-fold:

- We propose several new schemes for network planning, including dual time-scale planning, grid-based network planning, and flexible binary slice zooming. The first two schemes enable fine-grained and adjustable network planning by exploiting the temporal and spatial data traffic dynamics in the network, respectively. The third scheme replaces conventional cell-based coverage management with slice-based coverage and thus empowers planning with additional flexibility;
- To solve the planning problem for energy efficiency maximization, we devise an unsupervised learning-assisted approach, which searches for the best solution by leveraging the solutions of similar historical DTD instances. The performance of the proposed approach is close to the global optimum given sufficient historical information.

The remainder of this chapter is organized as follows. Section 3.2 describes the system model and the three proposed schemes. Section 3.3 presents the problem formulation. Section 3.4 introduces the ULSCS algorithm. Section 3.5 presents the simulation results, followed by the summary in Section 3.6.

3.2 System Model and Proposed Planning Schemes

In this section, we first introduce the networking scenario under consideration. Then, we propose several schemes for slicing-based network management in the planning stage, including dual time-scale planning, grid-based network planning, flexible binary slice zooming, and virtual slice separation.

3.2.1 System Model

Consider a wireless network with one MBS and M SBSs, as illustrated in Fig. 3.1. All the BSs share the same radio spectrum [146]. There are N slices in the physical network,

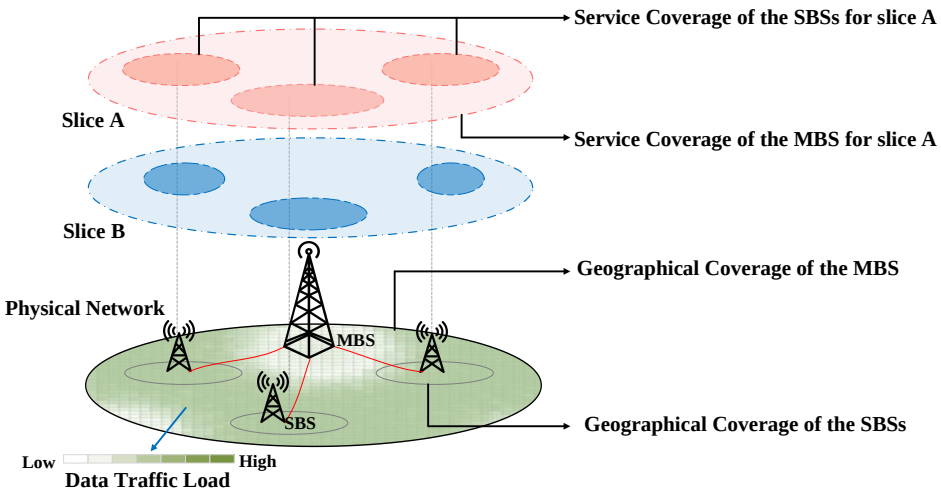


Figure 3.1: The network scenario.

corresponding respectively to N different services. Each service is accessible anywhere in the network coverage, and thus each slice spans over the entire network. The MBS and SBSs jointly provide each service across the network. The spatial coverage of a BS for a given service is referred to as the *service coverage* (SC) of that BS for the corresponding slice. The SC of any two different BSs do not overlap for any given slice. Therefore, for any slice, each BS solely serves all the MUTs within its SC for that slice.

Generally, in the operation stage, both SC determination and downlink transmission power control of BSs require *MUT-level information*, such as MUT location and data traffic volume [147, 148]. However, such MUT-level information is usually not available in the planning stage. The *aggregated data traffic* volume for a service requested by MUTs within a small area during a short time period, on the other hand, may be available through prediction [99, 149].[1] Adopting such an approach, SC determination and power control remain challenging due to three reasons. First, in the spatial dimension, both power control and SC determination depend on spatial DTD (i.e., the aggregated data traffic volumes of all small areas within the network coverage during a time period) and are correlated through interference among the BSs. An uneven spatial DTD can further

[1] A small area is relative to the geographical coverage of a BS. Considering that the network coverage consists of multiple small areas, we further define the spatiotemporal DTD of a slice as the aggregated data traffic volumes of all small areas in the network coverage during multiple short time periods. Thus, we determine SC and transmission power of BSs in the planning stage according to spatiotemporal DTDs of the slices.

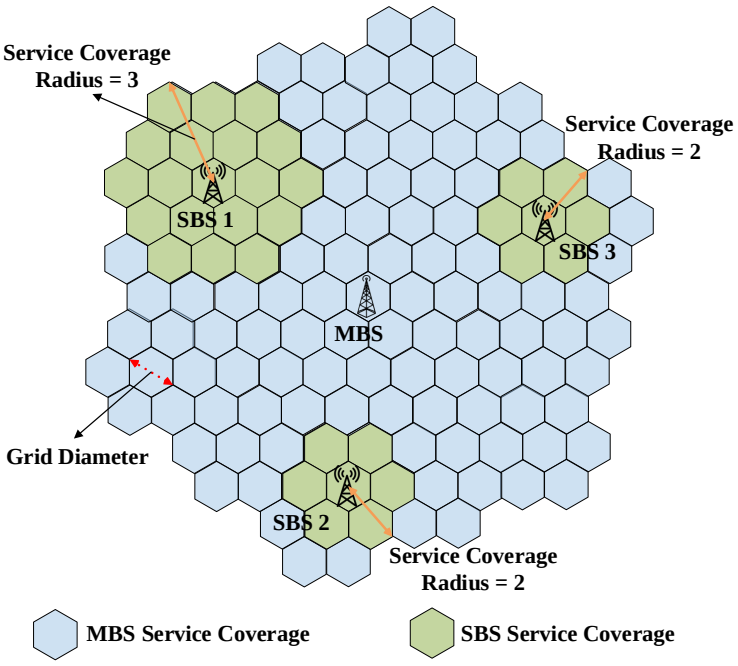


Figure 3.2: Grid-based network planning.

complicate the relation between SC determination and power control. Second, in the temporal dimension, the aggregated data traffic volume of every small area constantly changes among different short time periods, and determining the same power control for a long-term period does not provide enough flexibility to adapt to such temporal variation. Third, different slices can have different SINR requirements, and thus, two slices may need different SC and transmission power settings. With the objective of maximizing energy efficiency, enabling power control and SC determination for satisfying all SINR requirements is complicated, while considering slice-specific DTD. To address the above challenges, we propose SC determination and power control schemes in the planning stage, as detailed in the following subsections.

3.2.2 Grid-based Network Planning

The spatial DTD affects the decision of downlink transmission power control and SC determination. Specifically, due to the frequency reuse among BSs, the downlink transmission of any BS interferes with the downlink transmission associated with other BSs. Since each small area has different distances to all BSs, its location affects the interference from other BSs. Meanwhile, the aggregated data traffic volume of each small area determines the amount of its allocated radio spectrum and, thus, further affects the interference of its associated BS to the downlink transmission associated with other BSs. Therefore, for the same SINR requirement, we should differentiate the downlink transmission power for different small areas according to their locations relative to the BSs, and their aggregated data traffic volumes. For SC determination, adjusting the SC of any BS changes the association of small areas and could further change the aggregated data traffic volume of the BS. Consequently, the BS causes different interference to the downlink transmission associated with other BSs. Therefore, we should determine both transmission power and SC according to the spatial DTD.

To adapt to the spatial DTD, we propose *grid-based network planning*. Specifically, the coverage area of the considered network is divided into small spatial areas, i.e., grids, as shown in Fig. 3.2. We divide the whole coverage area of the network into I hexagon grids with the same grid diameter r. Without loss of generality, the locations of all the BSs are at the centers of the corresponding grids. The value of grid diameter r determines the network planning granularity in the spatial dimension. The temporal variation of the aggregated data traffic volume in each grid for every slice is assumed to be known in advance through prediction. The SC determination and transmission power allocation are based on the spatial DTD over the grids. We define the SC radius of a BS for any slice as the integer number of layers of grids covered by the BS for that slice. For example, as shown in Fig. 3.2, the SC radius of SBS 1 is 3, and the SC radius of SBS 2 is 2. Each BS individually allocates downlink transmission power for each grid within its SC, and the allocated power for different grids can be different. The proposed grid-based network planning has two advantages: First, it introduces flexibility to adapt to uneven spatial DTD through grid-based SC determination; Second, it increases the granularity of power control in the spatial dimension, which yields the potential for improving energy efficiency.

3.2.3 Dual Time-scale Planning

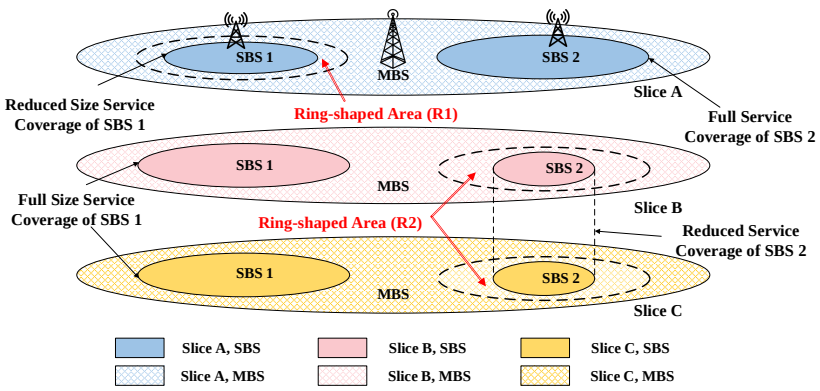
Aggregated data traffic volume of each grid varies temporally during each planning period. On the one hand, if a maximum or average aggregated data traffic volume of every grid over

each planning period is considered for planning-stage decisions (i.e., power control and SC determination), such temporal variations of data traffic are ignored, leading to service under or over-provisioning. On the other hand, frequent changing the planning-stage decisions to follow temporal variations of data traffic may result in excessive overhead. Therefore, there exists a trade-off between overhead and network performance, depending on the temporal granularity of network planning. To achieve a proper trade-off, we propose *dual time-scale planning*. We divide a planning period uniformly into T time intervals ($T > 1$). In each planning period, we determine the SC of all SBSs according to the predicted spatial DTDs in all time intervals. In each time interval, based on the determined SC, we allocate the downlink transmission power of all BSs for each individual grid based on the predicted DTD in the time interval. The reason is two-fold. First, as the SC of BSs determines MUT association in the operation stage, frequently adjusting the SC of BSs may frequently change the association of MUTs among different BSs. Second, we can allocate the downlink transmission power of BSs to the grids in the network coverage based on the specific SC of BSs, while, to determine the SC of BSs, we should compare the performance of different power allocations based on different SC. The complexity of determining the SC of BSs is higher than that of allocating transmission power. In this way, we take the temporal variations in a planning period into consideration without updating SC decisions.

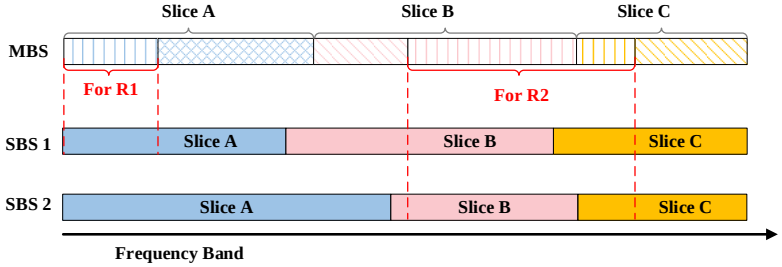
3.2.4 Flexible Binary Slice Zooming

Conventional wireless networks cannot provide service differentiation. Generally, the same BS coverage management and downlink transmission power control are conducted for serving all types of services, for example, cell-coverage based schemes, such as cell zooming [57, 150]. However, To satisfy differentiated service requirements and adapt to slice-specific spatiotemporal DTDs in the slicing-based network, differentiating SC and downlink transmission power for slices is crucial for optimizing the overall network performance. Based on the concept of cell zooming, we propose the idea of *slice zooming*. In specific, the SC and allocated downlink transmission power of every SBS can be different for slices. Compared to the cell-coverage based schemes for conventional networks, slice zooming can provide better performance by introducing higher flexibility in SC management and downlink transmission power control.

An example of slice zooming is shown in Fig. 3.3(a), two SBSs support three slices, and each SBS uses different SC for the three slices. SBS 1 provides smaller SC for slice A, and larger SC for slice B and slice C. As a result, there is a ring-shaped geographical area, i.e., area R1 in the figure. Similarly, a ring-shaped geographical area circling SBS 2 also exists. In such the ring-shaped area, some slices are served by an SBS, and other slices



(a) An illustration of slice zooming.



(b) Frequency band allocation in slice zooming.

Figure 3.3: Slice zooming.

are served by the MBS. Therefore, if the SBS and the MBS use the same frequency band for different slices in such areas, the two BSs may significantly interfere with each other in the ring-shaped area. To avoid such interference, we orthogonally allocate frequency bands among the BSs in ring-shaped areas, with details given as follows:

- For serving different slices, each BS should use different frequency bands. For example, in Fig. 3.3, SBS 1 should allocate different frequency bands for slices A, B, and C;
- In any ring-shaped area, the corresponding SBS and the MBS must use different frequency bands for different slices. For example, in Fig. 3.3, the frequency band for slice A in ring shaped area R1 used by the MBS should be different from the

frequency bands for slices A, B, and C, used by SBS 1.

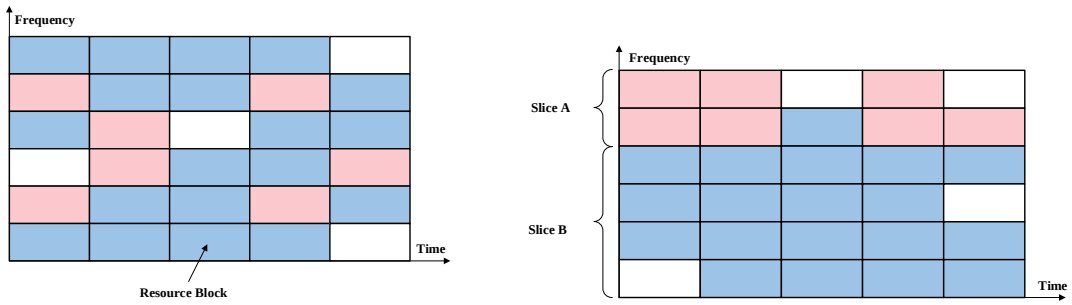
Following the above rules, for any slice, the frequency band allocated to the ring-shaped area should be a portion of frequency bands allocated to the corresponding SBS for that slice. In Fig. 3.3(b), we show an example of frequency band allocation corresponding to the SC of the BSs in Fig. 3.3(a). The frequency bands that the MBS allocates to slice A in ring-shaped area R1 must be a portion of the frequency bands that SBS 1 allocates to slice A. Similarly, the frequency bands that the MBS allocates to slices B and C in ring-shaped area R2 must be a portion of the frequency bands that SBS 2 allocates to slices B and C. It can be seen that slice zooming introduces flexibility to provide different SC for different slices but imposes extra constraints for frequency band allocation. The number of constraints increases with the number of slices. For example, if the SC for slices can be set arbitrarily, and an SBS has k different possible SC, the total number of constraints may increase as many as $k(k+1)/2$. Thus, slice zooming can result in high computation complexity. To provide flexibility without creating excessive constraints, we propose the idea of *flexible binary slice zooming* for SC management:

- For any slice, the SC radius of any SBS is not arbitrary but binary, i.e., either full or reduced;
- For any SBS, the spatial area of the full SC is the same for all slices and constrained by the maximum physical coverage of SBSs. Similarly, the spatial area of the reduced SC is the same for all slices. As a result, the ring-shaped area circling an SBS for any slice also has the same size;
- For different SBSs, the size of full SC can be different, depending on the DTD. Similarly, the size of reduced SC for different SBSs can be different.

The case shown in Fig 3.3(a) is in fact the proposed flexible binary slice zooming scheme. The size of the SC for slices B and C at SBS 2 is reduced and identical. The size of the SC for slice A at SBS 2 is full. The flexible binary slice zooming strikes a balance between flexibility and computation complexity.

3.2.5 Virtual Slice Separation

In the operation stage, downlink transmission power allocation to individual MUTs is based on the interference from the downlink transmission of different BSs, which is determined by



(a) Actual RB usage in the operation stage.

(b) Virtual slice separation in the planning stage for the resource block usage in (a).

Figure 3.4: Virtual Slice Separation.

the MUTs' locations and real-time scheduled spectrum resource blocks (RBs). However, due to the lack of individual MUT-level information in the planning stage, the model of interference in the operation stage cannot be used directly in the planning stage. To address this issue, we introduce the scheme of *virtual slice separation*. In specific, the number of RBs allocated to each slice is determined in the planning stage, while, in the operation stage, the specific sets of RBs are determined for serving individual MUTs in different slices according to the allocated number of RBs for slices. Accordingly, scheduling each RB at any BS for any slice in the operation stage can enable flexible RB multiplexing and improve RB utilization. This supports both orthogonal and non-orthogonal RB scheduling among different BSs. In Fig. 3.4(a), The RBs allocated to slice A and slice B corresponds to the virtual slice separation as shown in Fig. 3.4(b).

3.3 Problem Formulation

In this section, we give the problem formulation based on the system model and proposed planning schemes.

Let indexes $1, 2, \cdots, M$ and index 0 denote the SBSs and the MBS, respectively. Denote the set of all BSs, the set of all slices, the set of all gird indexes, and the set of all time intervals in a planning period by $\mathcal{M} = \{0, 1, 2, \cdots, m, \cdots, M\}$, $\mathcal{N} = \{1, 2, \cdots, n, \cdots, N\}$, $\mathcal{I} = \{1, 2, \cdots, i, \cdots, I\}$, and $\mathcal{T} = \{1, 2, \cdots, t, \cdots, T\}$, respectively. Denote the set of grids within the SC of BS m for slice n by $\mathcal{I}_{m,n}$. Let $d_{m,i}$ and $d_{m,m'}$ denote the distance between BS m and grid i for slice n and the distance between BS m and m', respectively.

Meanwhile, we denote the SC radius of SBS m for slice n by $L_{m,n} \in \{1, 2, \cdots, L_{\max}\}$, where $L_{\max}$ is the maximum radius. The data traffic of slice n generated at grid i ($i \in \mathcal{I}_{m,n}$) is served by SBS $m \in \mathcal{M}\backslash\{0\}$ if $d_{m,i} \leq L_{m,n}$. Otherwise, the data traffic of slice n generated at grid i ($i \in \mathcal{I}_{0,n}$) is served by the MBS.

According to the proposed flexible binary slice zooming, we use a binary indicator $a_{m,n}$ to indicate whether the SC of SBS m for slice n is full or reduced. Denote the radius of full and reduced SC of SBS m by L_m^{f} and L_m^{r}, respectively, where $L_m^{\mathrm{f}}, L_m^{\mathrm{r}} \in \{1, 2, \cdots, L_{\max}\}$. For slice n, $L_{m,n} \in \{L_m^{\mathrm{f}}, L_m^{\mathrm{r}}\}$. The SC of SBS m for slice n can be written as:

$$L_{m,n} = \begin{cases} L_m^{\mathrm{f}}, & a_{m,n} = 1; \\ L_m^{\mathrm{r}}, & a_{m,n} = 0. \end{cases} \tag{3.1}$$

Given the differentiated SC for different slices, we denote the set of grids in the ring-shaped area circling SBS m but covered by the MBS for slice n by $\mathcal{R}_{m,n}$. Given slices n and n' such that $a_{m,n} = 0$ and $a_{m,n'} = 1$, the data traffic in grid i for slice n is served by the MBS, if grid i is in the ring-shaped area circling SBS m (i.e., $i \in \mathcal{R}_{m,n}$, $m \in \mathcal{M}\backslash\{0\}$, and $L_{m,n} < d_{m,i} \leq L_{m,n'}$).

Denote the data traffic volume of slice n generated at grid i during time interval t by $w_{i,n}^t$. We use η_n to represent the required RBs for the data traffic of slice n per unit data traffic. As mentioned in section 3.2.5, the number of RBs allocated by the MBS to a ring-shaped area for all slices should be no more than that allocated by the corresponding SBS for all slices. The constraint can be written as the following inequality:

$$\sum_{n \in \mathcal{N}} \sum_{i \in \mathcal{R}_{m,n}} w_{i,n}^t \eta_n \leq \sum_{n \in \mathcal{N}} \sum_{i \in \mathcal{I}_{m,n}} w_{i,n}^t \eta_n, \ \forall m \in \mathcal{M}\backslash\{0\}, \ t \in \mathcal{T}, \tag{3.2}$$

where the left-hand side represents the number of RBs allocated to serve the grids within the ring-shaped area circling SBS m for all slices, and the right-hand side is the number of RBs allocated to all slices served by SBS m.

Frequency bands are reused among the BSs outside of the ring-shaped areas. For example, in Fig 3.3, the frequency bands used by SBS 1 to serve slice A can be the same as the frequency bands used by SBS 2 to serve slice A. We use $b_{i,n,i',n'}^t$ to indicate whether the downlink transmission to grid i for slice n interferes with the downlink transmission to grid i' for slice n' during time interval t, given by:

$$b_{i,n,i',n'}^t = \begin{cases} 0, & \forall i \in \mathcal{I}_{m,n}, i' \in \mathcal{R}_{m,n'}, a_{m,n} = 1, a_{m,n'} = 0; \\ 0, & \forall i \in \mathcal{R}_{m,n'}, i' \in \mathcal{I}_{m,n}, a_{m,n} = 1, a_{m,n'} = 0; \\ 1, & \text{otherwise.} \end{cases} \tag{3.3}$$

With the flexibility of the RB assignment and multiplexing brought by virtual slice separation, any RB may be used by any slice and at any BS. As a result, all RBs are used about equally likely. Let C denote the number of all RBs which can be allocated by each BS. We define resource block utilization ratio as $\theta_{i,n}^t$, which represents the number of RBs allocated to grid i for slice n during time interval t over the total number of RBs at any BS, as follows:

$$\theta_{i,n}^t = \frac{w_{i,n}^t \eta_n}{C}, \quad \forall i \in \mathcal{I}, n \in \mathcal{N}. \tag{3.4}$$

Denote the planning-stage downlink transmission power at grid i for serving slice n over time interval t by $p_{i,n}^t$. Considering that the data traffic load and the allocated transmission power may be different for different grids, we calculate the total transmission power of each BS in the planning stage statistically [151]. Denote the total downlink transmission power of BS m for slice n during time interval t by $P_{m,n}^t$, calculated as follows:

$$P_{m,n}^t = \sum_{i \in \mathcal{I}_{m,n}} \theta_{i,n}^t p_{i,n}^t, \quad \forall m \in \mathcal{M}, n \in \mathcal{N}. \tag{3.5}$$

The total downlink transmission power of each BS must satisfy the following constraints:

$$\sum_{n \in \mathcal{N}} P_{m,n}^t \leq \begin{cases} p_{\text{MBS}}, & \text{for } m = 0; \\ p_{\text{SBS}}, & \forall m \in \mathcal{M} \backslash \{0\}, \end{cases} \tag{3.6}$$

where p_{MBS} and p_{SBS} are the maximum downlink transmission power of the MBS and SBSs, respectively.

Let use $m_{i,n} \in \mathcal{M}$ to represent the index of BS which serves grid i for slice n. Denote the average downlink channel gain between BS $m_{i,n}$ and grid i over the duration of time interval t by $h_{i,n}^t$. For any BS $m_{i,n}$ and grid i such that $d_{m_{i,n},i} \leq L_{m,n}$, the SINR of the channel for grid i and slice n during time interval t is given in the following equation:

$$\gamma_{i,n}^t = \frac{p_{i,n}^t h_{i,n}^t}{N_0 + \sum_{m' \in \mathcal{M} \backslash \{m_{i,n}\}} \sum_{n' \in \mathcal{N}} \sum_{i' \in \mathcal{I}_{m',n'}} b_{i,n,i',n'}^t P_{m',n'}^t h_{m',i'}^t}. \tag{3.7}$$

Denote the energy consumption of BS m for serving slice n during time interval t by $E_{m,n}^t$, which is given by:

$$E_{m,n}^t = \tau P_{m,n}^t, \quad \forall m \in \mathcal{M}, n \in \mathcal{N}, \tag{3.8}$$

where τ is the duration of a time interval. Denote the SINR requirement of slice n by $\gamma_n^{\min}$. The power control should satisfy the following SINR requirements of slices:

$$\gamma_{i,n}^t \geq \rho \gamma_n^{\min}, \quad \forall i \in \mathcal{I}_{m,n}, \tag{3.9}$$

where ρ is a weight to flexibly scale the minimum required SINR level based on the feedback from the operation stage.[2]

We then formulate a problem to determine the planning-stage SC of SBSs and downlink transmission power of all BSs according to the given spatiotemporal DTD of all slices. We maximize the energy efficiency of the BSs and satisfy the diverse SINR requirements of slices. Let λ_n be the weight for the energy efficiency of slice n. Our objective is to maximize the weighted sum of energy efficiency of all slices. We first denote $\mathbf{p} = \left[p_{1,1}^1, \cdots, p_{i,n}^t, \cdots, p_{I,N}^T\right]^\top$, $\mathbf{L}_\mathrm{f} = \left[L_1^\mathrm{f}, \cdots, L_m^\mathrm{f}, \cdots, L_M^\mathrm{f}\right]^\top$, $\mathbf{L}_\mathrm{r} = \left[L_1^\mathrm{r}, \cdots, L_m^\mathrm{r}, \cdots, L_M^\mathrm{r}\right]^\top$, and $\mathbf{a} = \left[a_{1,1}, \cdots, a_{m,n}, \cdots, a_{M,N}\right]^\top$. The corresponding optimization problem can be formulated as follows:

$$\text{P1:} \max_{\{\mathbf{p},\mathbf{L}_\mathrm{f},\mathbf{L}_\mathrm{r},\mathbf{a}\}} \sum_{n\in\mathcal{N}} \frac{\lambda_n}{\sum_{t\in\mathcal{T}}\sum_{m\in\mathcal{M}} E_{m,n}^t} \cdot \frac{\sum_{t\in\mathcal{T}}\sum_{m\in\mathcal{M}}\sum_{i\in\mathcal{I}_{m,n}} w_{i,n}^t}{\sum_{t\in\mathcal{T}}\sum_{m\in\mathcal{M}}\sum_{i\in\mathcal{I}_{m,n}} w_{i,n}^t \eta_n} \tag{3.10a}$$

$$\text{s.t. } (3.2), (3.6), (3.9), \tag{3.10b}$$

$$d_{m,m'} \geq L_m^\mathrm{f} + L_{m'}^\mathrm{f}, \ \ \forall m \neq m', \ m, m' \in \mathcal{M}\backslash\{0\}, \tag{3.10c}$$

$$a_{m,n} \in \{0,1\}, \ \ \forall m \in \mathcal{M}, n \in \mathcal{N}, \tag{3.10d}$$

$$p_{i,n}^t > 0, \ \ \forall p_{i,n}^t \in \mathbb{R}, \tag{3.10e}$$

$$L_\mathrm{max} > L_m^\mathrm{r} \geq L_m^\mathrm{f} > 0, \ \ \forall L_m^\mathrm{r}, L_m^\mathrm{f} \in \mathbb{Z}. \tag{3.10f}$$

In problem P1, the optimization variables include power control through $\mathbf{p}$ and SC through $\mathbf{L}_\mathrm{f}$, $\mathbf{L}_\mathrm{r}$, and $\mathbf{a}$. We maximize the energy efficiency of all BSs in (3.10a). Constraint (3.10c) ensures that the SC of SBSs do not overlap. Problem P1 is a combinatorial optimization problem, which is difficult to solve by conventional optimization methods [152]. First, a large number of variables need to be determined. Specifically, there are $N \times I \times T$ variables for power control and $N \times M$ variables for SC determination.[3] Second, the power control and SC determination are mutually dependent. To solve this problem, we propose an unsupervised learning-assisted approach in the next section.

[2]The SINR in the planning stage, i.e., $\gamma_{i,n}^t$ is a reference value over the duration of a time interval, which may not represent the exact SINR level in the operation stage. Thus, we allow a feedback mechanism to change the SINR in the planning stage by adjusting the weight ρ based on real-time SINR in the operation stage.

[3]The number of combinations of SC of all SBSs for all slices is up to $M^{2^N \cdot (L_\mathrm{max})^2}$.

3.4 Unsupervised Learning-assisted Solution Search

As mentioned in Section 3.2.3, planning-stage downlink transmission power control and SC determination are mutually dependent, and finding appropriate SC is more challenging than determining the power. Therefore, in this section, we propose a unsupervised learning-assisted approach to solve problem P1 in two steps. In the first step, we design an iterative algorithm based on unsupervised learning to determine the SC of SBSs. In the second step, given the SC of all SBSs for all slices, we prove that the problem of power control is convex and find the grid-based power control solution for each slice. Since the second step is easier, we first present the closed-form power control solution and then find the SC.

3.4.1 Transmission Power Determination

As determining downlink transmission power of BSs is coupled by the inter-cell interference, we cannot allocate power to the girds within the SC of any BS independently. Thus, given the SC of all SBSs for all slices, i.e., $\mathbf{L}_{\mathrm{f}}, \mathbf{L}_{\mathrm{r}}, \mathbf{a}$, we can rewrite the problem of power allocation as follows:

$$\text{P2:} \max_{\{\mathbf{p}\}} \sum_{n\in\mathcal{N}} \frac{\lambda_n}{\sum_{t\in\mathcal{T}}\sum_{m\in\mathcal{M}} E^t_{m,n}} \cdot \frac{\sum_{t\in\mathcal{T}}\sum_{m\in\mathcal{M}}\sum_{i\in\mathcal{I}_{m,n}} w^t_{i,n}}{\sum_{t\in\mathcal{T}}\sum_{m\in\mathcal{M}}\sum_{i\in\mathcal{I}_{m,n}} w^t_{i,n}\eta_n} \tag{3.11a}$$

$$\text{s.t. } (3.6), (3.9), (3.10\text{e}). \tag{3.11b}$$

The downlink transmission power of all BSs for each grid needs to be determined for each time interval while the power control for different time intervals is independent. Therefore, it is sufficient to solve Problem P2 for just one time interval. In any time interval, the power control depends on slice-specific spatial DTDs. We first introduce vector $\mathbf{W}^t_n = \left[w^t_{1,n}, \cdots, w^t_{i,n}, \cdots, w^t_{I,n}\right]$ to represent the spatial DTD across all grids for slice n during time interval t and define its corresponding diagonal matrix as $\mathbf{w}^t_n = \mathrm{diag}(w^t_{1,n}, \cdots, w^t_{i,n}, \cdots, w^t_{I,n})$. Denote the spatiotemporal DTD of all slices in the network by $\mathbf{W} = \left[\mathbf{W}^1_1, \cdots, \mathbf{W}^t_n, \cdots, \mathbf{W}^T_N\right] \in \mathcal{W}$, where $\mathcal{W}$ is the set of all possible spatiotemporal DTDs. spatiotemporal

Theorem 1. *The optimal power, i.e.,* $\mathbf{p}^t = \left[p^t_{1,1}, \cdots, p^t_{i,n}, \cdots, p^t_{I,N}\right]^\top$*, during time interval*

t is given by:

$$\mathbf{p}^t = \rho \left(\mathbf{\Theta}^t - \rho \begin{bmatrix} \gamma_1^{\min} \mathbf{\Omega}_{1,1}^t \mathbf{w}_1^t & \dots & \gamma_1^{\min} \mathbf{\Omega}_{1,n'}^t \mathbf{w}_n^t & \dots & \gamma_1^{\min} \mathbf{\Omega}_{1,N}^t \mathbf{w}_N^t \\ \dots & \dots & \dots & \dots & \dots \\ \gamma_n^{\min} \mathbf{\Omega}_{n,1}^t \mathbf{w}_1^t & \dots & \gamma_n^{\min} \mathbf{\Omega}_{n,n'}^t \mathbf{w}_n^t & \dots & \gamma_n^{\min} \mathbf{\Omega}_{n,N}^t \mathbf{w}_N^t \\ \dots & \dots & \dots & \dots & \dots \\ \gamma_N^{\min} \mathbf{\Omega}_{N,1}^t \mathbf{w}_1^t & \dots & \gamma_N^{\min} \mathbf{\Omega}_{N,n'}^t \mathbf{w}_n^t & \dots & \gamma_N^{\min} \mathbf{\Omega}_{N,N}^t \mathbf{w}_N^t \end{bmatrix} \right)^{-1} \begin{bmatrix} \gamma_1^{\min} N_0 \\ \vdots \\ \gamma_n^{\min} N_0 \\ \vdots \\ \gamma_N^{\min} N_0 \end{bmatrix}, \tag{3.12}$$

where $\mathbf{\Theta}^t = \mathrm{diag}(h_{1,1}^t, \cdots, h_{i,n}^t, \cdots, h_{I,N}^t)$*, and*

$$\mathbf{\Omega}_{n,n'}^t = \begin{bmatrix} b_{1,n,1,n'}^t h_{1,n'}^t & \cdots & b_{1,n,i',n'}^t h_{i',n'}^t & \cdots & b_{1,n,I,n'}^t h_{I,n'}^t \\ \vdots & \ddots & \vdots & & \vdots \\ b_{i,n,1,n'}^t h_{1,n'}^t & \cdots & b_{i,n,i',n'}^t h_{i',n'}^t & \cdots & b_{i,n,I,n'}^t h_{I,n'}^t \\ \vdots & & \vdots & \ddots & \vdots \\ b_{I,n,1,n'}^t h_{1,n'}^t & \cdots & b_{I,n,i',n'}^t h_{i',n'}^t & \cdots & b_{I,n,I,n'}^t h_{I,n'}^t \end{bmatrix}_{I \times I}. \tag{3.13}$$

Proof. See Appendix 6.2. □

3.4.2 Local Optimum SC Search

Using the closed-form power control solution, we propose a searching algorithm for determining the SC of each SBS iteratively, given in Algorithm 1. Denote the energy efficiency of the network by ϕ. We first define a vector $\mathbf{L}_m = [L_{m,1}, \cdots, L_{m,n}, \cdots, L_{m,N}]$ to represent the SC of SBS m for all slices. Define the set of all possible values of vector $\mathbf{L}_m$ as $\mathcal{S}_m$, which includes all possible combinations of the SC of SBS m for all slices. We introduce a vector $\mathbf{g} = [g_1, \cdots, g_m, \cdots, g_M]$, where g_m is the maximum energy efficiency obtained by searching the set $\mathcal{S}_m$ for the SC of SBS m given the SC of other SBSs. We initialize the SC of all SBSs, i.e., $\mathbf{L}_\mathrm{f}$, $\mathbf{L}_\mathrm{r}$, and $\mathbf{a}$, and randomly select an SBS to start searching with a nonzero value for g. Then, we iteratively search for the SC of every SBS (one SBS in an iteration) for all slices and update the SC of any SBS for all slices if the new SC can result in a higher energy efficiency while satisfying constraints (3.10c) and (3.10e). The energy efficiency of all BSs either remains the same or increases after each update. The search stops if the energy efficiency cannot be improved by updating the SC of any SBS. Since the result of the LOSCS algorithm depends on the initial SC of SBSs, Algorithm 1 converges to a local optimum.

Algorithm 1: Local Optimum SC Search (LOSCS)

Input: $\mathbf{W}^{\mathrm{cu}}$
Initialize: $\phi = 0$, Randomly set $\hat{\mathbf{L}}_{\mathrm{f}}$, $\hat{\mathbf{L}}_{\mathrm{r}}$, $\hat{\mathbf{a}}$, m, and $\mathbf{g} \neq 0 \cdot \mathbf{e}$
while $\mathbf{g} \neq \phi \cdot \mathbf{e}$ **do**
  for $j = 1 : |\mathcal{S}_m|$ **do**
    $\hat{L}_m^{\mathrm{r}}$, $\hat{L}_m^{\mathrm{f}}$, $\hat{a}_{m,n} \leftarrow$ Select the vector $\mathbf{L}_m$ with index j in the set $\mathcal{S}_m$;
    if *constraints (3.10c) and (3.10e) are not satisfied* **then**
      Continue;
    else
      Obtain $\mathbf{p}$ according to Theorem 1;
      $\hat{\phi} \leftarrow$ Calculate the energy efficiency of all BSs given $\mathbf{W}^{\mathrm{cu}}$, $\hat{\mathbf{L}}_{\mathrm{f}}$, $\hat{\mathbf{L}}_{\mathrm{r}}$, $\hat{\mathbf{a}}$, and $\mathbf{p}$;
      if $\hat{\phi} > \phi$ **then**
        ϕ, $g_m \leftarrow \hat{\phi}$;
        $\mathbf{L}_{\mathrm{f}}$, $\mathbf{L}_{\mathrm{r}}$, $\mathbf{a} \leftarrow \hat{\mathbf{L}}_{\mathrm{f}}$, $\hat{\mathbf{L}}_{\mathrm{r}}$, $\hat{\mathbf{a}}$;
      else
        Continue;
      end
    end
  end
  $m \leftarrow$ Randomly select the index of SBS, i.e., f, from the set $\mathcal{F} = \{f : g_f \neq \phi\}$;
end
Output: $\mathbf{L}_{\mathrm{f}}$, $\mathbf{L}_{\mathrm{r}}$, $\mathbf{a}$, and ϕ

3.4.3 Unsupervised Learning-assisted SC Search

It is possible to find a better SC solution starting from a local optimum. For example, simulated annealing [153] could be used to generate random solutions based on the given local optimum. Then, the random solutions can be compared with the local optimum to identify potential better solutions. However, such an approach may not always be effective because it generates candidate solutions randomly [154]. We adopt a data-driven method to find candidate SC solutions by leveraging historical data and refine the initial solution based on the found candidate SC solutions. Specifically, the SC solutions obtained for historical and similar DTD instances can be used as candidate SC solutions since similar DTD instances may lead to the similar optimums of SC solutions.[4] Based on this idea,

[4]DTD instance is a realization of DTD.

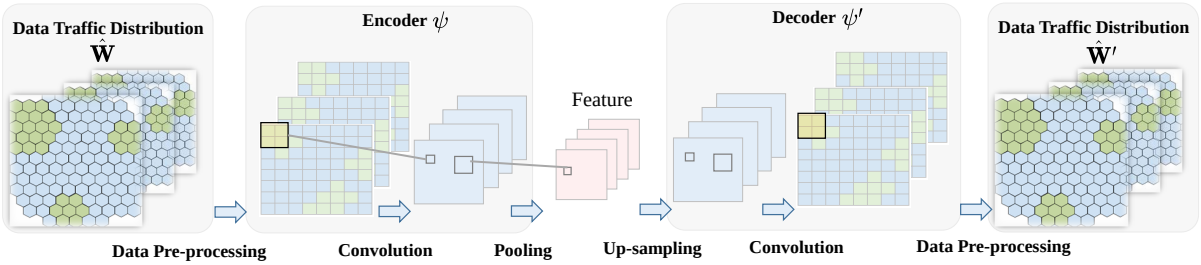


Figure 3.5: The architecture design of the convolutional neural network.

we propose a unsupervised learning-assisted SC search (ULSCS) algorithm including two components, i.e., dimensionality reduction and recommendation.

A. Dimensionality Reduction

Since the data traffic in each grid is a continuous parameter, the probability that two DTD instances are exactly the same is very low. Thus, we exploit similar DTD instances to help generate potential solutions for comparison with local optimum solutions found using Algorithm 1. As aforementioned, each DTD consists of three dimensions (temporal, spatial, and slice), and each dimension may have a large set of variables. Consequently, directly comparing DTD instances is impractical. Therefore, we first reduce the dimensionality of DTD and obtain a vector with the reduced dimension, named *key features*, which still contains the most information of the DTD. For dimensionality reduction and feature extraction, some model-driven approaches, e.g., principal component analysis, have been investigated [155]. However, since DTD is high-dimensional in this problem, there is no guarantee on the effectiveness and complexity of conventional model-driven approaches. Therefore, we apply auto-encoder model, i.e., a deep unsupervised learning approach, to extract the key features of each DTD instance. Fig. 3.5 shows the architecture design of the proposed deep unsupervised learning based on a convolutional neural network, which includes three parts, i.e., data preprocessing, encoder, and decoder. The data preprocessing part realizes the transformation of DTD between a grid-based format and a matrix format. The encoder has several convolutional and pooling layers to reduce the dimensionality of the preprocessed DTD and generate the corresponding key features. The decoder, with a mirror architecture of the encoder, can reconstruct a DTD according to the key features. Training the parameters of the neural network dose not require labeled dataset since the auto-encoder approach can generate their own labels from the training data, i.e., DTD instances. In the training phase, the goal is to let the input DTD instance be identical

with the reconstructed DTD passing through the encoder and the decoder. Once the parameters of the convolutional neural network are well-trained, only the encoder function is leveraged in Algorithm 2 to extract key features, while the decoder function is used only in the training phase [37].

We denote the extracted key features by $\mathbf{O} \in \mathcal{O}$, and then define two mapping functions: 1) encoder function $\psi : \mathcal{W} \to \mathcal{O}$; and 2) decoder function $\psi' : \mathcal{O} \to \mathcal{W}$. We leverage convolutional neural networks to approximate the two mapping functions, i.e., $\mathbf{O} = \psi(\mathbf{W}; \vartheta)$ and $\mathbf{W}' = \psi'(\mathbf{O}; \vartheta')$, parametrized by ϑ and ϑ', respectively. The matrix $\mathbf{W}'$ represents a re-constructed DTD according to the key features. To accurately approximate the mapping functions, the optimal parameters of the convolutional neural networks, denoted by ϑ_* and $\vartheta_*^{'}$, are obtained from the following equation:

$$\{\vartheta_*, \vartheta_*^{'}\} = \arg\min_{\{\vartheta, \vartheta'\}} F(\mathbf{W}, \mathbf{W}') = F(\mathbf{W}, \psi'(\psi(\mathbf{W}; \vartheta); \vartheta')), \tag{3.14}$$

where $F(\mathbf{W}, \mathbf{W}')$ is a loss function. Define the set of historical information entries as Υ, including the data regarding DTD instances, the corresponding SC solutions obtained by the LOSCS algorithm, and the resulting values of energy efficiency. Let $|\Upsilon|$ denote the number of historical information entries. The data regarding DTD instances are used in training the parameters of convolutional neural networks, i.e., ϑ and ϑ'. The parameters ϑ_* and $\vartheta_*^{'}$ can be obtained via the gradient descent method with the goal of minimizing the difference between the DTD and reconstructed DTD [37].

B. Recommendation

At the beginning of each planning window, we calculate the similarity between the DTD instance in the upcoming planning window, denoted by $\mathbf{W}_a$, and a historical DTD instance contained in set Υ, denoted by $\mathbf{W}_b$, as follows:

$$D(\mathbf{W}_a, \mathbf{W}_b) = \frac{\psi(\mathbf{W}_a; \vartheta_*) \cdot \psi(\mathbf{W}_b; \vartheta_*)}{\|\psi(\mathbf{W}_a; \vartheta_*)\| \|\psi(\mathbf{W}_b; \vartheta_*)\|} = \frac{\mathbf{O}_a \cdot \mathbf{O}_b}{\|\mathbf{O}_a\| \|\mathbf{O}_b\|}, \tag{3.15}$$

where $\mathbf{O}_a = \psi(\mathbf{W}_a; \vartheta_*)$ denotes the features of DTD instance $\mathbf{W}_a$, and $\mathbf{O}_b = \psi(\mathbf{W}_b; \vartheta_*)$ denotes the features of DTD instance $\mathbf{W}_b$. A certain number of SC solutions, which the corresponding DTD instances are the most similar, are selected as the potential candidate solutions for refining the obtained initial solution in the upcoming planning window. The set of reference historical information entries including the selected DTD instances, potential candidate solutions, and the resulting values of energy efficiency, is denoted by $\Upsilon_{\text{re}} \subseteq \Upsilon$. The number of reference historical information entries is denoted by υ.

Algorithm 2: Unsupervised Learning-assisted SC Search (ULSCS)

1 **Input:** ϑ_*, $\mathbf{W}_a$, Υ, and υ
2 Calculate the similarity D between $\mathbf{W}_a$ and historical DTD instances in Υ by (3.15);
3 $\Upsilon_{\rm re} \leftarrow$ Select υ reference entries with most similar DTD from Υ;
4 ϕ, $\mathbf{L}_{\rm f}$, $\mathbf{L}_{\rm r}$, $\mathbf{a} \leftarrow$ Execute Algorithm 1 given $\mathbf{W}_a$ and random $\mathbf{L}_{\rm f}$, $\mathbf{L}_{\rm r}$, and $\mathbf{a}$;
5 **for** $j = 1 : \upsilon$ **do**
6 $\quad \mathbf{L}_{\rm f}^{\rm re}$, $\mathbf{L}_{\rm r}^{\rm re}$, $\mathbf{a}^{\rm re} \leftarrow$ Obtain from the j-th entry in $\Upsilon_{\rm re}$;
7 $\quad \mathbf{L}_{\rm f}$, $\mathbf{L}_{\rm r}$, $\mathbf{a} \leftarrow \mathbf{L}_{\rm f}^{\rm re}$, $\mathbf{L}_{\rm r}^{\rm re}$, $\mathbf{a}^{\rm re}$;
8 $\quad \phi'$, $\mathbf{L}'_{\rm f}$, $\mathbf{L}'_{\rm r}$, $\mathbf{a}' \leftarrow$ Execute Algorithm 1 given $\mathbf{W}_a$, $\mathbf{L}_{\rm f}$, $\mathbf{L}_{\rm r}$, and $\mathbf{a}$;
9 $\quad$ **if** $\phi' > \phi$ **then**
10 $\quad\quad \phi$, $\mathbf{L}_{\rm f}$, $\mathbf{L}_{\rm r}$, $\mathbf{a} \leftarrow \phi'$, $\mathbf{L}'_{\rm f}$, $\mathbf{L}'_{\rm r}$, $\mathbf{a}'$;
11 $\quad$ **else**
12 $\quad\quad$ **Continue**;
13 $\quad$ **end**
14 **end**
15 Add $\mathbf{W}_a$, ϕ, $\mathbf{L}_{\rm f}$, $\mathbf{L}_{\rm r}$, and $\mathbf{a}$ to Υ;
16 **Output:** $\mathbf{L}_{\rm f}$, $\mathbf{L}_{\rm r}$, $\mathbf{a}$, and ϕ

Algorithm 2 presents the detailed procedures of leveraging historical information to refine the initial solution obtained by Algorithm 1. Given a well-trained neural network with parameter ϑ_* and the DTD instance $\mathbf{W}_a$ in the upcoming planning window, we first extract the key features and calculate the similarity between the DTD instance $\mathbf{W}_a$ and all historical DTD instances contained in the set of historical information entries, i.e., Υ (Line 2). υ reference historical information entries with the highest similarity are selected (Line 3). For the DTD instance $\mathbf{W}_a$, an initial solution, i.e., ϕ, $\mathbf{L}_{\rm f}$, $\mathbf{L}_{\rm r}$, $\mathbf{a}$, is obtained based on Algorithm 1. In Line 5-14, we set the SC solution contained in each reference historical information entry as the initial setting of SC in Algorithm 1 and run the algorithm to find a refined solution. By comparing the performance of the refined solution with that of the initial solution, we choose the coverage solution with the better performance as the final solution. At the end of each planning window, data regarding the DTD instance, the corresponding SC solutions, and the resulting energy efficiency can be added to the set Υ. The iteration terminates when all SC solutions contained in the set $\Upsilon_{\rm re}$ have been used as initial settings. The effectiveness of the proposed algorithm is expected to increase with the number of historical information entries and the number of historical reference information entries. Since the computation time of inferring deep neural networks is low, the computation time of running Algorithm 2 mainly depends on running Algorithm 1 and comparing the refined solution and the initial solution. As a result, the computation time

Table 3.1: Simulation Parameters

Parameter	Value	Parameter	Value
N	2	T	2
$\left[\gamma_1^{\min}, \gamma_2^{\min}\right]$	[20, 10]	$[\lambda_1, \lambda_2]$	[0.55, 0.45]
H_{MBS}	8 m	H_{SBS}	4 m
ρ	1	N_0	-174 dBm/Hz

of running Algorithm 2 is linear to the number of historical information entries and the number of historical reference information entries, respectively.

3.5 Performance Evaluation

In this section, extensive simulation results are presented to demonstrate the performance of the proposed schemes. We begin from introducing the simulation settings and benchmark schemes. Then, we show the resulting simulation results.

3.5.1 Simulation Settings

The radius of the MBS's SC is 1,500 m, and the maximum radius of SBSs' SC is 800 m. According to a propagation model in 3GPP standard [156], channel gain $h_{i,n}^t$ between BS $m_{i,n}$ and grid i for slice n with distance $d_{m,i}$ (km) can be approximated as follows:

$$h_{i,n}^t(\text{dB}) = 40(1 - 4 \times 10^{-3} H) \log_{10} d_{m,i} - 18 \log_{10} H + 21 \log_{10} f_{\text{c}} + 80(\text{dB}), \qquad (3.16)$$

where f_{c} is the carrier frequency in MHz, and H represents the antenna height. For the MBS and the SBSs, the antenna heights are denoted by H_{MBS} and H_{SBS}, respectively. Other simulation parameters are listed in Table 3.1.

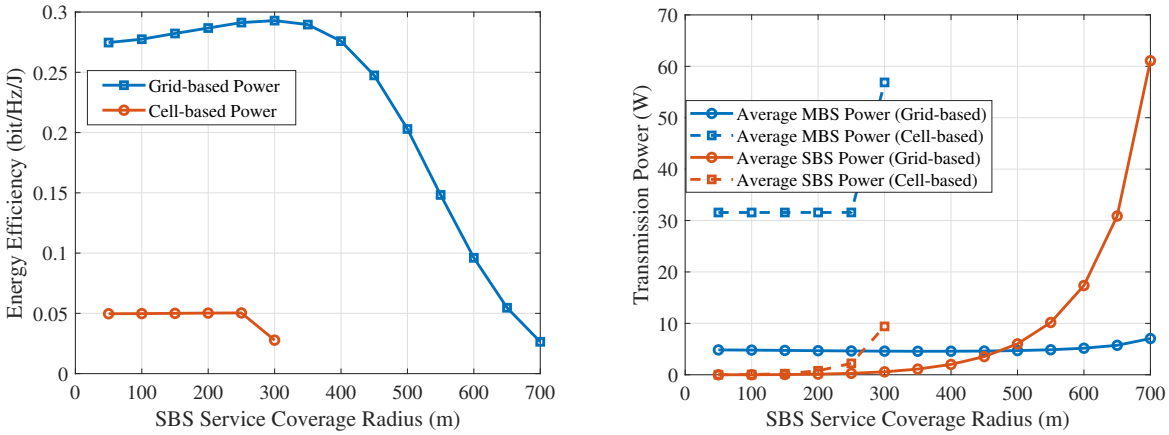
The implementation of the convolutional neural network in Fig. 3.5 is as follows. The encoder contains 2 convolutional layers with 16 and 32 neurons, respectively. The kernel size and stride is set as (3, 3) and 2 for both convolutional layers, respectively. Each convolutional layer is followed by a pooling layer. ReLU is used as the activation function after each layer, and the cross-entropy is used as the loss function F. The dimension of the key feature is 8. The architecture of the decoder is the reverse of that of the encoder. We adopt the Adam optimizer to train neural networks. To evaluate the performance of

the designed deep convolutional neural network, we select up to 8,000 DTD instances and divide them into two data sets, i.e., a training data set with 7,800 instances and a test data set with 200 instances.

The following two benchmark schemes are adopted:

- *Cell-based power control:* For each BS, the same transmission power is used for all grids within its SC;
- *Cell zooming:* For each SBS, its SC for different slices are the same.

3.5.2 Performance of Grid-based Power Control



(a) Energy efficiency versus SC radius.

(b) Average downlink transmission power versus SC radius.

Figure 3.6: Comparison between the proposed grid-based power control and cell-based power control.

In this subsection, we investigate the performance of the proposed grid-based power control in a simple scenario with one MBS, one SBS, and one slice. In Fig. 3.6(a), we compare the proposed grid-based power control and the cell-based power control. As shown in this figure, the proposed grid-based power control achieves higher energy efficiency due to its higher spatial granularity. By contrast, the cell-based power control cannot satisfy the SINR constraint beyond a 300 m SC radius of the SBS. Moreover, we compare the transmission power of the two power control schemes in Fig. 3.6(b). For the grid-based

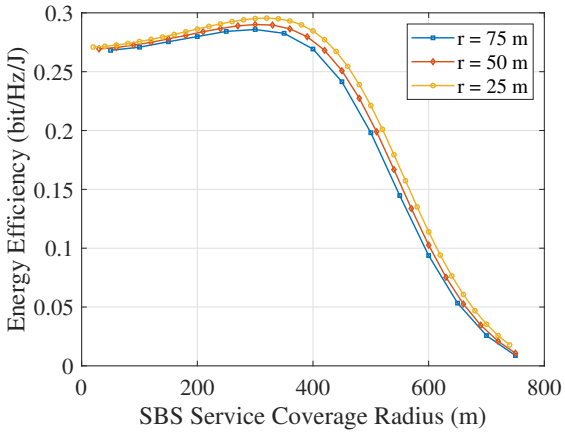


Figure 3.7: The impact of spatial granularity on energy efficiency.

scheme, we average the BS transmission power of all girds. The figure shows that the average transmission power of both the SBS and the MBS is higher with the cell-based transmission power. The difference between the two schemes is significant when the SBS SC radius is low.

Since the grid diameter r determines the network planning granularity in the spatial dimension, we investigate the impact of r. Fig. 3.7 shows the energy efficiency given different grid diameters. From this figure, we can make two observations. Firstly, for all gird diameters, the energy efficiency increases when the SBS SC radius is smaller than 350 m, and decreases when the SBS SC radius is larger than 350 m. Secondly, grid-based power control with a smaller grid diameter leads to higher energy efficiency. As the grid diameter decreases, the network is divided into more grids, and the power control can be allocated with higher spatial granularity to suit the spatial DTD.

3.5.3 Performance of Flexible Binary Slice Zooming

In this subsection, we investigate the performance of the proposed flexible binary slice zooming and compare it with that of the cell zooming scheme. As shown in Fig. 3.8, we give an example of different DTDs of two slices. In Fig. 3.9(a), we compare the energy efficiency of flexible binary slice zooming based on the grid-based power control with that of two cell-based schemes averaged over 200 DTD instances. Two observations can be made from this figure. Firstly, grid-based power control can achieve a better performance compared to cell-based power control, and the performance gain increases with the number

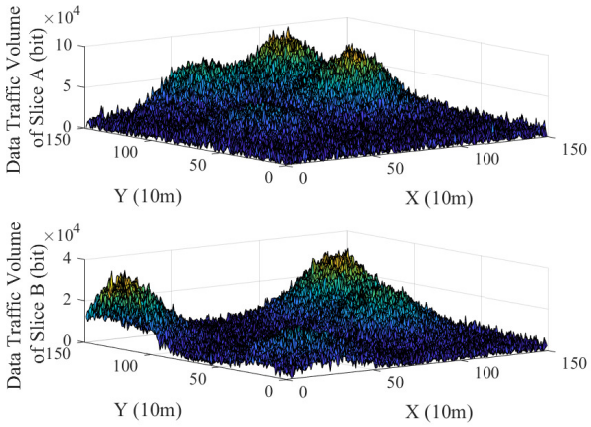
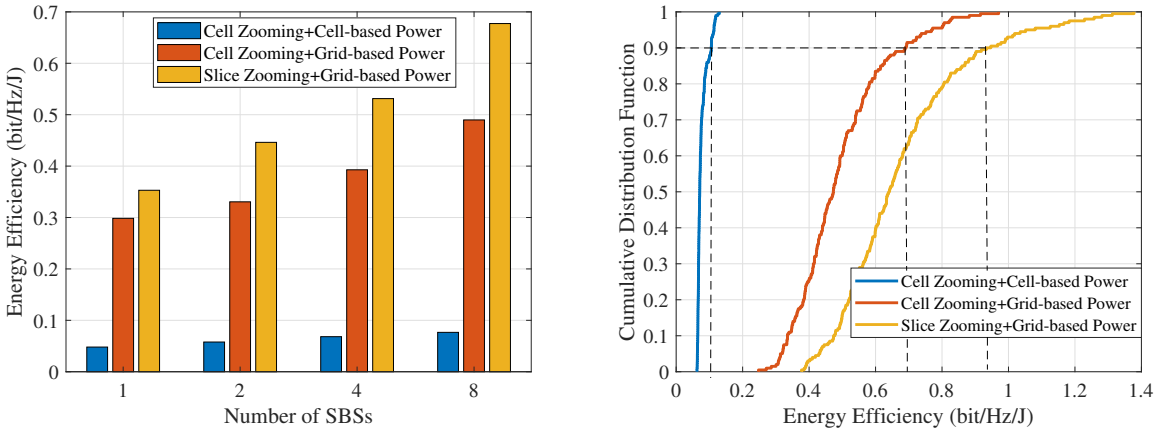


Figure 3.8: An example DTD in the case of two slices.



(a) Average Energy Efficiency versus the number of SBSs.

(b) The cumulative distribution function of energy efficiency.

Figure 3.9: The performance of flexible binary slice zooming with different DTD instances.

of SBSs. Secondly, the proposed flexible binary slice zooming significantly improves the energy efficiency compared to cell zooming. Moreover, the performance gap between flexible binary slice zooming and cell zooming increases with the number of SBSs. Fig. 3.9(b) shows the cumulative distribution function of the energy efficiency performance of all three schemes from 200 different DTDs and 8 SBSs. We can see that the proposed flexible binary slice zooming provides higher energy efficiency for most DTD instances. This is because

Table 3.2: The Impact of Grid Diameter.

Grid Parameter (m)	150	100	75	50	25
Energy Efficiency (bit/Hz/J)	0.5756	0.6211	0.6532	0.6773	0.6891
Computation Time (Sec)	14.85	47.63	102.70	351.49	1782.32

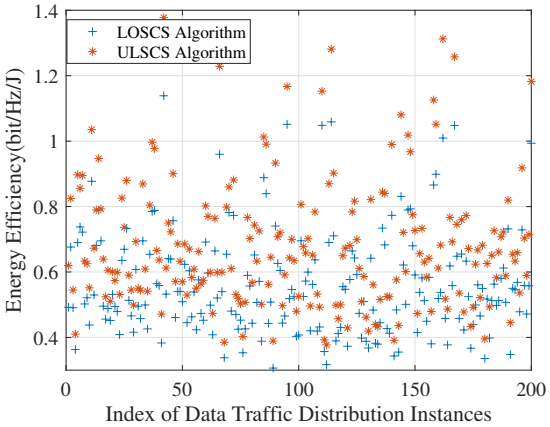
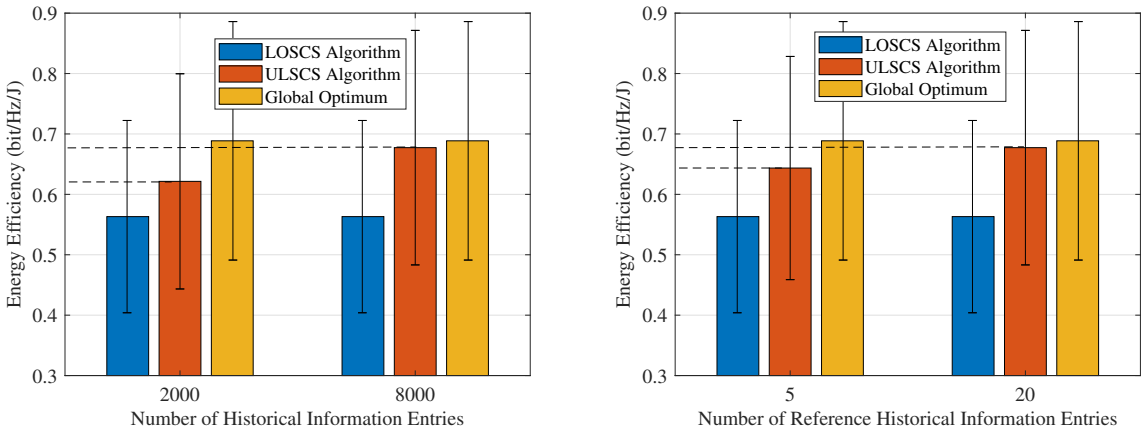


Figure 3.10: Energy Efficiency comparison between the LOSCS algorithm and the ULSCS algorithm (averaged over 200 DTD instances).

that flexible binary slice zooming enables SC determination with higher granularity in the slice dimension.

Next, we simulate the impact of grid diameter in the case of 8 SBSs, respectively. Table. 3.2 shows the impact of the grid diameter on the network energy efficiency and the computation time. As the grid diameter decreases, the network energy efficiency improves in all cases. The improvement is significant when the grid diameter is large than 50 meters. The computation time increases when the number of SBS increases and the grid diameter decreases. This happens because the number of variables increases with the total number of grids. The computation time of matrix operations and the number of iterations increases with the number of variables. Therefore, to obtain an appropriate grid diameter for network planning, we need to balance the energy efficiency and the computation time.



(a) The impact of the number of historical information entries.

(b) The impact of the number of reference historical information entries.

Figure 3.11: Average energy efficiency performance improvement of the ULSCS algorithm with different sizes of Υ and values of υ, respectively.

3.5.4 Performance of the ULSCS Algorithm

In this subsection, we compare the performance of the proposed ULSCS algorithm and the LOSCS algorithm and investigate the impact of the number of historical information entries, i.e., the size of Υ, and reference historical information entries, i.e., the value of υ, respectively. We consider eight SBSs, one MBS, and two slices and adopt 200 DTD instances to evaluate the performance. In Fig. 3.10, we can see that the performance of the ULSCS algorithm is better than that of the LOSCS algorithm for most of the DTD instances. Then, Fig. 3.11(a) shows the average energy efficiency of the ULSCS algorithm given different numbers of historical information entries, i.e., different sizes of Υ, over the 200 DTD instances. In this figure, the performance gap between the ULSCS algorithm and the LOSCS algorithm increases with the number of historical information entries. In addition, the performance of the ULSCS algorithm is approximately the same as the optimum global value. This is because more similar DTD instances and reference solutions could be found with more historical information entries. Fig. 3.11(b) shows the impact of the number of reference historical information entries, i.e., the value of υ. The performance gap between the ULSCS algorithm and the LOSCS algorithm increases with υ, and the ULSCS algorithm can approach the global optimum with sufficient reference historical information entries.

3.6 Summary

In this chapter, we have investigated a planning-stage SC determination and downlink transmission power control for network slicing. The considered network features DTD across the spatial, temporal, and slice dimensions, while each slice has a unique SINR requirement. To achieve fine-grained planning, we have proposed the ideas of grid-based network planning, dual time-scale planning, and flexible binary slice zooming. Following these ideas, a network planning problem with the objective of energy efficiency maximization is formulated. To solve this combinatorial problem with a large number of variables, we have developed a novel unsupervised learning-assisted algorithm to determine the SC and power control for all BSs. The proposed algorithm starts from an initial local optimum and searches for a better solution, and its performance can approach the global optimum with sufficient historical information. The ideas and solutions developed in this work provide insights for network planning problems in general.

Chapter 4

Planning-stage Resource Reservation for Multi-tier Computing

In this chapter, we investigate planning-stage resource reservation to support a compute-intensive service with taking into account the mobility of MUT. As a promising computing paradigm, multi-tier computing is expected to support a increasing number of computing tasks for compute-intensive applications. MUTs can offload their computing tasks to the computing servers deployed at different tiers of networks, i.e., the core networks, gateways, and BSs. Each computing server requires communication, computing, and storage resources to execute computing tasks. The mutual dependency among different resources and the impact of MUT mobility create challenges in proactive resource reservation. Our research objective is to minimize the usage of multi-network resources through adaptively reserving adequate network resources for dynamic computing demands,. To this end, we develop a hybrid data-model-driven network planning scheme with two elements, i.e., multi-resource reservation and resource reservation re-configuration. First, DTs are designed for collecting MUT status data, based on which MUTs are grouped according to their mobility patterns. Second, an optimization algorithm is proposed to customize resource reservation for different groups to satisfy their different resource demands. Last, a meta-learning-based approach is developed to re-configure resource reservation for balancing the network resource usage and the re-configuration cost. Simulation results demonstrate that the proposed network planning scheme outperforms benchmark schemes by using less network resources and incurring lower re-configuration costs.

4.1 Background and Motivations

The NGWN are expected to support a wide range of compute-intensive applications [1]. A large portion of these applications are *stateful*, meaning that context data is required to execute computing tasks [41, 42]. For example, augmented reality applications require volumetric media objects or holograms, as the context data, to process video segments for MUTs. The prevalent mobile edge computing (MEC) paradigm provides a solution to supporting computing applications with low offloading delay but has limitations in supporting stateful applications [17]. Specifically, edge servers close to MUTs generally have limited storage capacity to store all context data of stateful applications. Moreover, even if the context data could be fully stored at an edge server, limited communication coverage and a relatively small number of MUTs served by the edge server would degrade storage resource utilization.

To address the above limitations, both the industry and the academia have started looking into the collaboration of servers [157, 158]. Extending from MEC, *multi-tier computing* integrates multiple servers deployed at the core network, gateways, access points, and other locations in the network for executing computing tasks from MUTs. Servers at different tiers have diverse features in terms of resource capacity and service coverage [18]. Specifically, servers deployed at the core network and gateways have larger service coverage and more abundant resources than servers deployed at access points. Through coordinating servers at different tiers, multi-tier computing can exploit different features of servers to support computing applications, especially stateful ones, in the NGWN.

Network planning, as an important part of network management, can facilitate the coordination of servers at different tiers. Network planning consists of resource reservation and resource reservation re-configuration [1]. Resource reservation refers to proactively reserving network resources for satisfying the upcoming resource demands from MUTs. Resource reservation re-configuration refers to timely re-configuring resource reservation decisions to adapt to time-varying resource demands and dynamic network environments. Network planning for supporting stateful applications faces four challenges. First, computing, storage, and communication resource reservation for stateful applications is tightly coupled, yielding existing resource reservation solutions for supporting stateless applications inapplicable. Second, the requests for context data may vary across the network, rendering both computing task assignment and storage resource reservation dependent on specific servers and MUT mobility patterns. Third, information regarding individual MUT status, e.g., MUT mobility, is unavailable at the time of network planning, yet such MUT-level information can be useful for accurately calculating the amount of network resources needed for supporting the applications [106]. Fourth, re-configuring computing and stor-

age resource reservation for stateful applications in a dynamic network environment yields additional costs from computing service interruption, which complicates resource reservation re-configuration [159]. Addressing the above challenges is important to accurate and adaptive network planning for supporting stateful applications in the NGWN.

Recently, the digital twin paradigm has started attracting attention as a potential solution to advancing network management for the NGWN [3]. The concept of digital twins (DTs) originates from product life-cycle management in industry, where a DT is a synchronized virtual replica of a physical object [33, 34, 160]. For the NGWN, DTs can be introduced to represent individual MUTs. Each DT consists of a *MUT data profile* that describes the corresponding MUT, including the MUT's mobility, service demands, and QoS satisfaction, and *DT functions* for data acquisition, processing and analysis [3]. DTs bring three benefits to network planning. First, historical data contained in DTs can be used to predict MUT status in the upcoming time interval, which can, in turn, facilitate customized resource reservation for highly diversified MUTs and enable fine-grained network planning. Second, data indicating the performance of network planning can be collected based on DTs, which can provide a foundation for resource reservation re-configuration in network planning to adapt to a highly dynamic network environment. Third, DTs should acquire extensive and well-organized data that can be used to explore and exploit hidden network characteristics, thereby facilitating effective data-driven network planning approaches to enhancing network performance. Due to the above benefits, DTs can be potentially designed and exploited to improve the granularity, adaptivity, and intelligence of network planning in the NGWN.

In this chapter, we design a network planning scheme for supporting stateful applications in the scenario of multi-tier computing. Our research objective is to find out the minimum amount of network resources (including computing, storage, and communication) needed for supporting the application and also balance the resource usage and the cost from resource reservation re-configuration in a dynamic network environment. To achieve this objective, we propose a DT-empowered network planning framework with the following two elements: group-based multi-resource reservation and closed-loop resource reservation re-configuration. First, we design DTs for individual MUTs to characterize their status and group them based on their mobility patterns. We propose an algorithm based on matching theory and particle swarm optimization to address the coupling relation among computing, storage, and communication in resource reservation. The proposed method enables customized resource reservation for satisfying different resource demands of MUT groups with different mobility patterns. Second, we develop a Meta-learning-based approach for resource reservation re-configuration to cope with the dynamic network environment. The main contributions of this chapter are the followings:

- We propose a novel network planning framework to facilitate fine-grained resource reservation based on MUT data contained in DTs;
- We address a challenging multi-resource reservation problem for supporting stateful applications in multi-tier computing;
- We develop an automated closed-loop approach to re-configure resource reservation in a dynamic network environment for balancing the network resource usage and the cost from re-configuring resource reservation.

The remainder of this chapter is organized as follows. Section 4.2 describes the considered network scenario, the proposed DT-empowered framework, and system model. Section 4.3 formulates the network planning problem for multi-tier computing. Sections 4.4 and 4.5 introduce the proposed solutions for resource reservation and resource reservation re-configuration, respectively. Section 4.6 presents the simulation results, followed by the summary in Section 4.7.

4.2 Network Scenario and System Model

In this section, we first introduce the considered network scenario of multi-tier computing. Then, we propose a DT-empowered network planning framework to support stateful applications and present the corresponding system model.

4.2.1 Network Scenario

The considered scenario is shown in Fig. 4.1. Computing servers are deployed at different locations, including: (i) base stations (BSs); (ii) network aggregation points (NAPs), such as gateways; and (iii) the core network (CN) [17]. Each server can build one dedicated virtual machine (VM) to execute computing tasks from MUTs for the stateful application. The service coverage area of a server at a BS (S-BS) is the BS's communication coverage. The service coverage area of a server at a NAP (S-NAP) is the union of the communication coverage of all the BSs connected to it. For example, S-NAP 1 connects to S-BS 1 and S-BS 2, and the communication coverage area of S-NAP 1 consists of the two green cells as shown in Fig. 4.1. The server at the CN (S-CN) can provide computing service to all MUTs in the considered network. The service coverage areas of servers at the same tier of the network do not overlap. MUTs generate computing tasks and offload their computing tasks to a server when located in its service coverage area. We assume that the computing

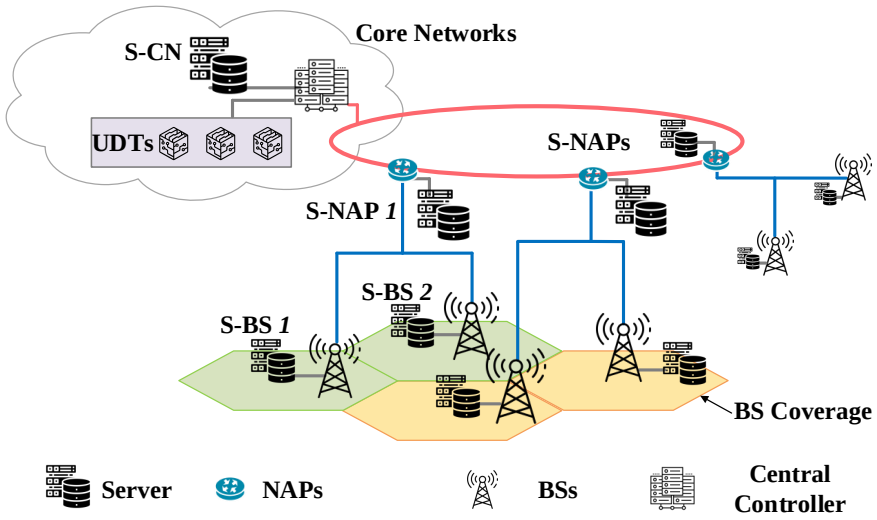


Figure 4.1: The considered scenario of multi-tier computing.

task generation at all MUTs corresponds to an identical statistical process due to the same stateful application. The VM at the server then executes the offloaded computing tasks and sends computing results back to the MUTs. Denote the S-CN by e^{cn}, and let $\mathcal{E}^{\text{bs}}$ and $\mathcal{E}^{\text{nap}}$ denote the set of S-BSs and the set of S-NAPs, respectively. Let $\mathcal{E} = \mathcal{E}^{\text{bs}} \cup \mathcal{E}^{\text{nap}} \cup \{e^{\text{cn}}\}$ represent the set of all servers in the network. Denote the set of BSs by $\mathcal{B} = \{1, 2, \cdots, B\}$, and let $\mathcal{E}_b \subset \mathcal{E}$ denote the set of servers that include BS $b \in \mathcal{B}$ in their service coverage areas.

We focus on resource reservation to support stateful applications, which requires context data for executing computing tasks. The application uses a fixed set of context data, and the popularity of different data chunks in this set is different. Moreover, the popularity of each data chunk can vary at different BSs across the network, i.e., the popularity of context data chunks is location-dependent. Therefore, if a MUT moves into the coverage of a different BS, its requests for context data may also change, creating a dependence of its resource demand on its mobility. A MUT that connects to more BSs within a time interval is considered to have higher mobility. MUTs with similar mobility patterns can be grouped together in resource reservation due to their similar need for context data.

The time duration of interest is partitioned into K time intervals of length τ (e.g., 5 to 10 minutes per time interval). Denote the set of time intervals by $\mathcal{K} = \{1, 2, \cdots, K\}$. A central controller in the CN maintains the DTs for MUTs and makes resource reservation

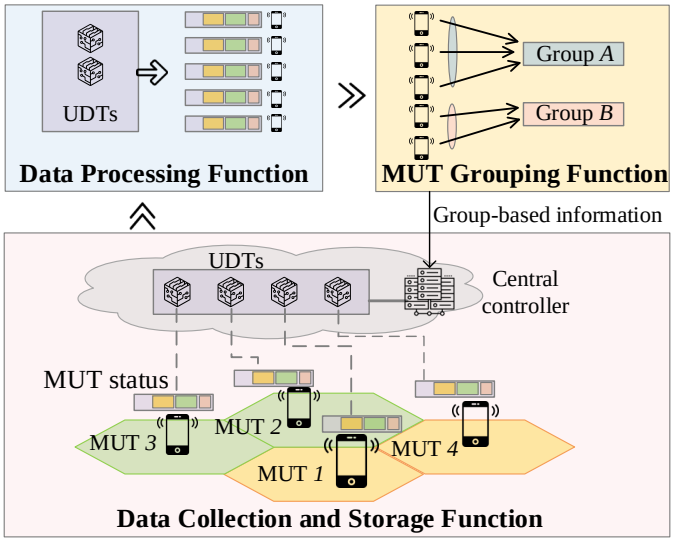


Figure 4.2: The designed UDTs used for network planning.

decisions proactively for the stateful application at the beginning of each time interval. We define vector $\boldsymbol{\phi}_{d,k}$ to represent the mobility pattern of MUT d in time interval $k \in \mathcal{K}$, where the bth element of $\boldsymbol{\phi}_{d,k}$ represents the time duration in which MUT d is in the coverage of BS b during the time interval. Table I lists the important symbols for easy reference.

4.2.2 DT-empowered Network Planning

In this subsection, we present the idea of DT-empowered network planning with our specific design of DTs, which is a succession and development of the framework in [3]. A DT is created for an individual MUT, called UDT, which consists of a MUT data profile and UDT functions. As shown in Fig. 4.2, UDTs are located at the CN, and MUT data profiles are maintained and updated by the central controller via UDT functions. Specifically, each MUT data profile is a well-organized set of MUT data. In this work, the data attributes of each UDT consist of the mobility of the corresponding MUT, including the MUT's location and velocity, as well as the information of each computing task from the MUT, including context data requirement, input and output data size, computing workloads, and resource demands. There are three UDT functions used to manage and analyze MUT data profiles for network planning, as follows:

- **Data collection and storage function** – MUT data required for network planning are collected from individual MUTs via BSs or offered by service providers. Data regarding MUT mobility, e.g., MUT locations, can be uploaded by individual MUTs periodically, as specified in 5G. Data regarding services, e.g., computing workloads and required context data, can be obtained from service providers or computing servers [17]. The collected MUT data are stored in the corresponding UDTs;

- **Data processing function** – At the beginning of time interval k, the mobility patterns of each MUT in the past T time intervals, i.e., $\boldsymbol{\phi}_{d,k'}, \forall k' \in [k-T, k-1]$, are obtained based on the MUT data in the corresponding UDT. Then, the historical mobility patterns are used to predict the mobility pattern of each MUT, i.e., $\boldsymbol{\phi}_{d,k}$, in the subsequent time interval. Other data prediction, e.g., spatial task distributions and requested context data, based on historical data in UDTs is also conducted via this function;

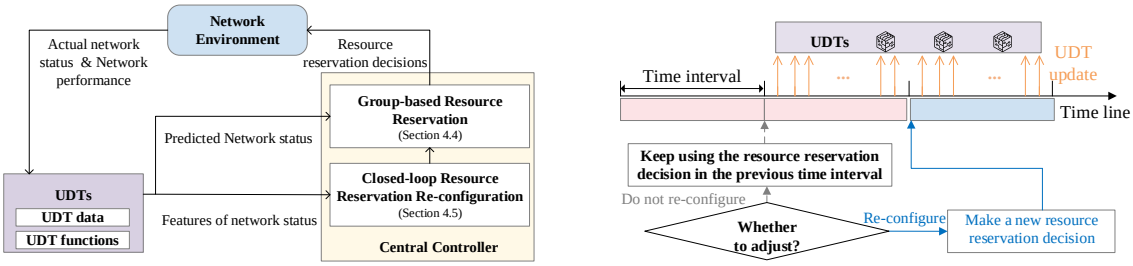
- **MUT grouping function** – We group MUTs based on their predicted mobility patterns, i.e., $\{\boldsymbol{\phi}_{d,k}, \forall d\}$, at the beginning of each time interval. MUTs from the same group have similar network resource demands for executing computing tasks due to their similar mobility patterns, allowing for an accurate approximation of resource demands for MUTs from each group. Based on MUT grouping, network resources can be reserved accurately to achieve fine-grained network planning. [1]

The DT functions are used to manage and analyze MUT data profiles to empower network planning. However, they do not make network planning decisions but only provide information for network planning.

Based on UDTs, we propose a novel network planning framework as shown in Fig. 4.3, which includes two core elements: group-based resource reservation and closed-loop resource reservation re-configuration.

- **Group-based resource reservation:** Based on MUT grouping, we reserve storage and computing resources accurately to satisfy the resource demands for UEs in each group in the upcoming time interval. Denote the set of groups by $\mathcal{N} = \{1, 2, \cdots, N\}$. Let $x_{b,k}^n$ denote the number of computing tasks generated by the MUTs who are associated with group n and are in the coverage of BS b during time interval k. We refer to matrix $\mathbf{x}_k^n = [x_{b,k}^n]_{\forall b \in \mathcal{B}}$ as *spatial task distribution* of group n in time

[1] Note that the mechanism of MUT grouping, as well as the data attributes used for grouping, can be customized based on the data contained in UDTs in different network scenarios. The number of groups depends on the trade-off between granularity and complexity.



(a) Workflow. (b) Time line.

Figure 4.3: An illustration of DT-empowered network planning.

interval k. Since it is impossible to know the actual value of $\mathbf{x}_k^n$ at the beginning of time interval k, $\mathbf{x}_k^n$ is predicted based on historical data contained in UDTs. We use superscript "$\sim$", e.g., $\tilde{x}$, to represent the predicted values of x. Given spatial task distributions, we propose an algorithm to address the storage and computing resource reservation problem with highly coupled variables. The detail of group-based resource reservation is discussed in Section 4.4.

- **Closed-loop resource reservation re-configuration:** Since spatial task distributions may change across time intervals, re-configuring the reserved resources on servers to adapt to dynamic computing demands is necessary but yields additional cost from re-configuring resource reservation. We design a closed-loop approach to re-configure resource reservation re-configuration for balancing the network resource usage and the cost from re-configuring resource reservation, and the time line is shown in Fig. 4.3(b). Specifically, at the beginning of each time interval, the central controller identifies whether to re-configure resource reservation with a proposed algorithm. If the resource reservation needs to be re-configured for the subsequent time interval, the controller will make a new resource reservation decision; Otherwise, the controller will keep using the resource reservation from the previous time interval. At the end of each time interval, the data regarding network performance are collected based on UDTs. The detail of closed-loop resource reservation re-configuration is discussed in Section 4.5I.

4.2.3 Computing Model

The overall load of computing tasks assigned to each server during a time interval affects network resource reservation on the server. Denote the load of computing tasks generated by MUTs from group n in the coverage of BS b and assigned to server e during time interval k by $f^n_{b,e,k}$. The relation between computing task assignment and spatial task distribution is given by

$$\sum_{e\in\mathcal{E}_b} f^n_{b,e,k} = \tilde{x}^n_{b,k},\ \forall b \in \mathcal{B}, n \in \mathcal{N}, k \in \mathcal{K}. \tag{4.1}$$

Let $m^n_{e,k}$ denote the overall load of computing tasks generated by MUTs from group n and assigned to server e during time interval k,

$$m^n_{e,k} = \sum_{b\in\mathcal{B}_e} f^n_{b,e,k},\ \forall e \in \mathcal{E}, n \in \mathcal{N}, k \in \mathcal{K}, \tag{4.2}$$

where $\mathcal{B}_e$ denotes the set of BSs within the service coverage of server e. In Eq. (4.2), the fine granularity of computing task assignment, reflected through $f^n_{b,e,k}$, helps determine the computing load at each server accurately.

Since we consider one specific stateful application, the computing tasks for this application are assumed to have (approximately) the same input data size (in bits), computing workload (in CPU cycles per bit), and result data size (in bits).[2] Let α, β, and γ denote the average input data size, the average computing workload, and the average result data size, respectively. Each server can reserve a certain amount of computing resource (in CPU cycles per second) for VMs to execute computing tasks. Let $c^n_{e,k}$ denote the amount of computing resource of server e reserved for group n during time interval k, and define $\mathbf{c}_k = [c^n_{e,k}]_{\forall e\in\mathcal{E}, n\in\mathcal{N}}$ as the computing resource reservation decision in time interval k. For time interval k, the computing resource reservation should satisfy the following constraint:

$$\sum_{n\in\mathcal{N}} c^n_{e,k} \le C_e,\ \forall e \in \mathcal{E}, k \in \mathcal{K}, \tag{4.3}$$

where C_e denotes the maximum computing resource at server e that can be utilized for the stateful application. We assume that the computing resources on the S-CN, i.e., $e = e^{\text{cn}}$, is sufficient for executing all computing tasks. The time that server e takes for executing

[2]The proposed framework can be straightforwardly extended to handle the case of computing tasks with different input data sizes, computing workloads, and result data sizes by leveraging DTs to collect such information for each computing task. The problem-solving approach (including the proposed algorithms) remains applicable in such a case.

the computing tasks assigned to it from group n during time interval k should satisfy the following requirement:

$$\frac{m_{e,k}^{n}\alpha\beta}{c_{e,k}^{n}} \leq \tau^{\mathrm{p}},\ \forall e \in \mathcal{E}, n \in \mathcal{N}, k \in \mathcal{K}, \tag{4.4}$$

where τ^{p} denotes the maximum tolerable computation time.[3] Due to (4.4), computing resource reservation and computing task assignment for each server are mutually dependent.

Let $\epsilon_{e,k}^{\mathrm{c}}$ represent the overall computing resource usage for executing all computing tasks assigned to server e in time interval k, which is computing load-dependent. Based on [161, 162], we adopt a linear model of computing resource usage as follows:

$$\epsilon_{e,k}^{\mathrm{c}} = \sum_{n \in \mathcal{N}} \varepsilon_{e}^{\mathrm{c}} m_{e,k}^{n},\ \forall e \in \mathcal{E}, k \in \mathcal{K}, \tag{4.5}$$

where $\varepsilon_{e}^{\mathrm{c}}$ is the computing resource usage for executing each computing task at server e.

4.2.4 Storage Model & Remote Access

Different from stateless applications, the execution of a computing task for stateful applications requires the corresponding context data. If the context data is not in the storage of the server, the server should download the context data from a remote server, thereby yielding additional communication resource usage. We model the storage and the additional communication resource usage for the stateful application in this subsection.

Denote the amount of the reserved storage resource (in bits) of server e during time interval k and the storage capacity (in bits) of server e by $g_{e,k}$ and G_e, respectively. The value of $g_{e,k}$ should satisfy the following constraint:

$$g_{e,k} \leq G_e,\ \forall e \in \mathcal{E}\backslash\{e^{\mathrm{cn}}\}, n \in \mathcal{N}, k \in \mathcal{K}. \tag{4.6}$$

We assume that the storage resources on the S-CN, i.e., $e = e^{\mathrm{cn}}$, is sufficient for storing all context data. Define $\mathbf{g}_k = [g_{e,k}]_{\forall e \in \mathcal{E}\backslash\{e^{\mathrm{cn}}\}}$ as the storage resource reservation decision for all S-NAPs and S-BSs in time interval k. Based on the model of storage resource usage

[3] We do not consider queuing delay in the planning stage because we do not assume a particular computing task arrival pattern. For modeling the queueing delay, the task arrival pattern must be known *a priori*. However, such a pattern is usually unavailable in practice, while assuming a particular arrival pattern can oversimplify the scenario.

in [163], the resource usage for reserving storage resource at server e in time interval k, denoted by $\epsilon^{\mathrm{s}}_{e,k}$, is given by

$$\epsilon^{\mathrm{s}}_{e,k} = \varepsilon^{\mathrm{s}}_{e} g_{e,k}, \ \forall e \in \mathcal{E}, k \in \mathcal{K}, \tag{4.7}$$

where $\varepsilon^{\mathrm{s}}_{e}$ represents the per bit resource usage for reserving storage resource at server e.

Let $\mathcal{I} = \{1, 2, \cdots, I\}$ represent the set of all chunks of context data in the library for the stateful application, where I denotes the number of chunks in the set $\mathcal{I}$. All chunks of context data for the stateful application have the same data structure and thus identical data size (in bits), denoted by L. Executing each computing task requires one chunk of context data in the set $\mathcal{I}$ [164]. Denote the request ratio of chunk i in the coverage of BS b in time interval k by $p^{i}_{b,k}$, i.e., the load of computing tasks requesting chunk $i \in \mathcal{I}$ over the load of all computing tasks generated in the coverage of BS b in time interval k. The value of $p^{i}_{b,k}$ may be different in the coverage of different BSs and may vary across different time intervals. Let $\mathbf{p}_k = [p^{i}_{b,k}]_{\forall b \in \mathcal{B}, i \in \mathcal{I}}$ denote the chunk request ratio profile in time interval k, which can be obtained via prediction based on historical data contained in UDTs [165].

The overall data volume of all chunks may be much larger than the storage capacities of S-BSs and S-NAPs. Each S-BS and S-NAP can only store some chunks of context data for executing computing tasks prior to the beginning of each time interval. The S-CN stores all chunks of context data. Let $\mathcal{I}_{e,k} \subseteq \mathcal{I}$ denote the set of chunks stored at server e in time interval k. Given the amount of reserved storage resource on server e, i.e., $\mathbf{g}_k$, the number of chunks in set $\mathcal{I}_{e,k}$ is $|\mathcal{I}_{e,k}| = \lfloor g_e / L \rfloor$ where $\lfloor \cdot \rfloor$ represents the floor function.

Since our focus is storage resource reservation, we follow the hierarchical storage policy in [166,167] to determine the set of stored chunks on each S-BS and S-NAP, i.e., $\mathcal{I}_{e,k}$, based on the *effective request ratio* of chunks. Define the effective request ratio of chunk i for executing the computing tasks assigned to server e in time interval k as $q^{i}_{e,k}$, i.e., the load of computing tasks requesting chunk $i \in \mathcal{I}$ over the load of all computing tasks generated in the service coverage of server e in time interval k. In multi-tier computing, the stored chunks on S-NAPs and the S-CN may not be requested for computing task executing when the chunks are stored on S-BSs for the sake of reducing communication resource usage. As a result, the effective request ratio of a chunk for an S-NAP depends on not only the chunk request ratio profile, i.e., $\mathbf{p}_k$, but also the set of chunks stored on S-BSs and computing task assignment, which is difficult to determine.

For simplicity, we make two following assumptions to estimate the effective request ratio for each S-NAP. First, S-BS e located at BS b stores $|\mathcal{I}_{e,k}|$ chunks with largest $p^{i}_{b,k}$. Second, given set $\mathcal{I}_{e,k}$ for any S-BS, the computing tasks requesting a chunk stored on the S-BS are assigned to the S-BS as much as possible when not violating the computing

resource capacity of the S-BS [167]. Based on the two assumptions, the value of $q^i_{e,k}$ for an S-BS equals to the request ratio of chunk i in the coverage of the corresponding BS in time interval k, i.e., $p^i_{b,k}$; The value of $q^i_{e,k}(\mathbf{g}_k, \mathbf{p}_k)$ for an S-NAP can be estimated given $\mathbf{g}_k$ and $\mathbf{p}_k$, which is detailed in Appendix 6.2. Each S-BS and S-NAP sorts the chunks in a non-increasing order based on the effective request ratio, i.e., $q^i_{e,k}(\mathbf{g}_k, \mathbf{p}_k)$, and stores the most requested $|\mathcal{I}_{e,k}|$ chunks.

When the chunk required for executing a computing task is not found in the storage of an S-BS, the S-BS will first attempt to download the chunk of context data from the S-NAP covering it and resort to the S-CN in the case that the S-NAP does not have the chunk of context data either. When the chunk required for executing a computing task is not found in the storage of an S-NAP, the server will download the context data from the S-CN. For any server, accessing another server remotely and downloading context data from the remote server yields additional communication resource usage, referred to as remote access of context data [168].

Denote the data volume (in bits) of remotely accessing context data for each computing task by L^{re}, the value of which may be different from the value of L due to headers used by transmission protocols. Let $f^{n,i}_{b,e,k}$ denote the number of computing tasks that require chunk i and are generated by group n in the coverage of BS b and assigned to server e during time interval k. The relation between $f^n_{b,e,k}$ defined in Eq. (4.1) and $f^{n,i}_{b,e,k}$ is $f^n_{b,e,k} = \sum_{\mathcal{I}} f^{n,i}_{b,e,k}$. Define the computing task assignment in time interval k as $\mathbf{f}_k = [f^{n,i}_{b,e,k}]_{\forall b\in\mathcal{B},\forall e\in\mathcal{E}_b,n\in\mathcal{N},i\in\mathcal{I}_{e,k}}$. For S-BS e, the communication resource usage for downloading context data from S-NAP e' covering S-BS e for executing computing tasks from group n during time interval k is given by:

$$\begin{aligned} v^{\mathrm{re}}_{n,e,k} = {} & L^{\mathrm{re}}\xi^{e'}_e \sum_{i\in\overline{\mathcal{I}_{e,k}}\cap\mathcal{I}_{e',k}} \sum_{b\in\mathcal{B}_e} f^{n,i}_{b,e,k} + \\ & L^{\mathrm{re}}\xi^{e^{\mathrm{cn}}}_e \sum_{i\in\overline{\mathcal{I}_{e,k}}\cup\overline{\mathcal{I}_{e',k}}} \sum_{b\in\mathcal{B}_e} f^{n,i}_{b,e,k},\ \forall e\in\mathcal{E}^{\mathrm{bs}}, \end{aligned} \tag{4.8}$$

where coefficients $\xi^{e'}_e$ and $\xi^{e^{\mathrm{cn}}}_e$ represent the communication resource usage for downloading per bit context data to S-BS e from S-NAP e' and the S-CN, i.e., e^{cn}, respectively. In Eq. (4.8), the first term represents the communication resource usage for server e to access the context data remotely from server e' at the NAP, and the second term represents the communication resource usage for server e to access the context data remotely from S-CN e^{cn}. For S-NAP e, the communication resource usage for downloading context data

from the S-CN during time interval k is as follows:

$$v_{n,e,k}^{\mathrm{re}} = L^{\mathrm{re}} \xi_e^{e^{\mathrm{cn}}} \sum_{i \in \overline{\mathcal{I}_{e,k}}} \sum_{b \in \mathcal{B}_e} f_{b,e,k}^{n,i}, \ \forall e \in \mathcal{E}^{\mathrm{nap}}, \tag{4.9}$$

where $\xi_e^{e^{\mathrm{cn}}}$ denotes the communication resource usage for downloading per bit context data to S-NAP e from the S-CN, i.e., e^{cn}.

4.2.5 Communication Model

Generally, communication resource usage for uploading the input data and downloading the result of a computing task involves two parts: (i) the resource usage for the wireless communication between the MUT and the connected BS; and (ii) the resource usage for the wired communication between the BS and a server. In the considered scenario, each MUT is associated with only one BS. Regardless of which computer server deployed in the multi-tier network is selected for executing the computing task, the resource consumption for the wireless communication between the associated BS and the MUT for uploading input data and downloading computing results is a constant. As a result, the wireless communication does not affect the solution of the resource reservation problem. Since S-BS and BSs are co-located, there is no additional communication resource usage if any computing task is processed at an S-BS.

Denote the maximum communication resource usage of server $e \in \mathcal{E}^{\mathrm{nap}} \cup \{e^{\mathrm{cn}}\}$ for uploading input data and downloading result data by V_e^{up} and V_e^{down}, respectively. Let $\eta_{b,e}$ denote the coefficient representing the communication resource usage for uploading and downloading per bit data between BS b and server e. The communication resource usage for uploading input data and downloading computing results between server e and BS b during time interval k are given by:

$$v_{b,e,k}^{\mathrm{up}} = \alpha \eta_{b,e} \sum_{n \in \mathcal{N}} \sum_{i \in \mathcal{I}} f_{b,e,k}^{n,i}, \ \forall e \in \mathcal{E} \backslash \{e^{\mathrm{bs}}\}, b \in \mathcal{B}_e, k \in \mathcal{K}, \tag{4.10}$$

and

$$v_{b,e,k}^{\mathrm{down}} = \gamma \eta_{b,e} \sum_{n \in \mathcal{N}} \sum_{i \in \mathcal{I}} f_{b,e,k}^{n,i}, \ \forall e \in \mathcal{E} \backslash \{e^{\mathrm{bs}}\}, b \in \mathcal{B}_e, k \in \mathcal{K}, \tag{4.11}$$

respectively.

4.3 Problem Formulation

We formulate the problem of planning-stage resource reservation for multi-tier computing to support stateful applications in this section. We aim to find out the minimum amount of network resources needed for supporting the application while balancing the resource usage and the cost from re-configuring resource reservation in the presence of network dynamics.

Let $\mathbf{r}_k = [\mathbf{c}_k, \mathbf{g}_k, \mathbf{f}_k]$ denote the resource reservation decision, including computing resource reservation, storage resource reservation, and computing task assignment, in time interval k. Variable $\mathbf{r}_k$ is determined at the beginning of time interval k. Given $\mathbf{r}_k$, the overall network resource usage in time interval k, denoted by $\Delta_k(\mathbf{r}_k)$, can be obtained based on Eqs. (4.5), (4.7), (4.10), and (4.11), as follows:

$$\Delta_k(\mathbf{r}_k) = \sum_{e \in \mathcal{E}} w^{\mathrm{c}} \epsilon^{\mathrm{c}}_{e,k} + w^{\mathrm{s}} \epsilon^{\mathrm{s}}_{e,k} + w^{\mathrm{o}}(v^{\mathrm{re}}_{e,k} + v^{\mathrm{co}}_{e,k}), \tag{4.12}$$

where $v^{\mathrm{co}}_{e,k} = \sum_{b \in \mathcal{B}} (v^{\mathrm{up}}_{b,e,k} + v^{\mathrm{down}}_{b,e,k})$, and w^{c}, w^{s}, and w^{o} are the weights of the computing, storage, and communication resource usage, respectively. Since the spatial task distribution may vary across time intervals, even if a resource reservation decision can minimize instantaneous resource usage in Eq. (4.12) in time interval k, the same decision may not minimize the overall network resource usage in the subsequent time intervals. Re-configuration is required for the resource reservation to adapt to the dynamic spatial task distribution, while re-configuring resource reservation yields additional cost, e.g., the cost from vertical scaling of VM [159]. Denote by O^{v} the cost from re-configuring resource reservation.

Let $a_k \in \{0, 1\}$ indicate whether the resource reservation in time interval k should be re-configured or not, which is determined at the beginning of time interval k. If $a_k = 0$, the controller should make new resource reservation decision for time interval k; Otherwise, the controller should keep using the resource resource from time interval $k-1$. Define $\tilde{\mathbf{x}}_k = [\tilde{x}^n_{b,k}]_{\forall b \in \mathcal{B}, n \in \mathcal{N}}$ as spatial task distributions of groups, referred to as group-based spatial task distribution. Let function $\mathbf{r}_k = \psi(\tilde{\mathbf{x}}_k, \mathbf{p}_k)$, representing that the resource reservation decision is made according to the group-based spatial task distribution and the chunk request profile in time interval k. The value of r_k, resource reservation in time interval k, evolves as follows:

$$\mathbf{r}_k = (1 - a_k)\psi(\tilde{\mathbf{x}}_k, \mathbf{p}_k) + a_k \mathbf{r}_{k-1}, \ \forall k \in \mathcal{K}. \tag{4.13}$$

Given the value of a_k, the cost from re-configuring resource reservation in time interval k, denoted by o^{v}_k, is as follows:

$$o^{\mathrm{v}}_k = (1 - a_k)O^{\mathrm{v}}, \ \forall k \in \mathcal{K}. \tag{4.14}$$

The problem of minimizing the long-term network resource usage and the cost from re-configuring resource reservation over K time intervals, is formulated as follows:

$$\text{P0:} \min_{\mathbf{a},\mathbf{R}} \ \sum_{k\in\mathcal{K}} \frac{\Delta_k(\mathbf{r}_k) + \lambda o_k^{\mathrm{v}}}{\sum_{b\in\mathcal{B}}\sum_{n\in\mathcal{N}} \tilde{x}_{b,k}^n} \tag{4.15a}$$

$$\text{s.t. } (4.3), (4.4), (4.6) \tag{4.15b}$$

$$\sum_{b\in\mathcal{B}_e} v_{b,e,k}^{\mathrm{up}} \le V_e^{\mathrm{up}}, \ \forall e \in \mathcal{E}\setminus\mathcal{E}^{\mathrm{bs}}, k\in\mathcal{K}, \tag{4.15c}$$

$$\sum_{b\in\mathcal{B}_e} v_{b,e,k}^{\mathrm{down}} \le V_e^{\mathrm{down}}, \ \forall e \in \mathcal{E}\setminus\mathcal{E}^{\mathrm{bs}}, k\in\mathcal{K}, \tag{4.15d}$$

$$\sum_{e\in\mathcal{E}} v_{n,e,k}^{\mathrm{re}} \le V_n^{\mathrm{re}}, n\in\mathcal{N}, k\in\mathcal{K}, \tag{4.15e}$$

$$c_{e,k}^n \in \mathbb{R}^+, \ \forall e\in\mathcal{E}, n\in\mathcal{N}, k\in\mathcal{K}, \tag{4.15f}$$

$$g_{e,k} \in \mathbb{R}^+, \ \forall e\in\mathcal{E}\setminus\{e^{\mathrm{cn}}\}, n\in\mathcal{N}, k\in\mathcal{K}, \tag{4.15g}$$

$$f_{b,e,k}^{n,i} \in \mathbb{N}, \ \forall b\in\mathcal{B}, e\in\mathcal{E}, n\in\mathcal{N}, k\in\mathcal{K}, \tag{4.15h}$$

$$a_k \in \{0,1\}, \ \forall k\in\mathcal{K}, \tag{4.15i}$$

where V^{re} denotes the maximum communication resource usage of remote access for one computing task, $\mathbf{a} = [a_k]_{\forall k\in\mathcal{K}}$, $\mathbf{R} = [\mathbf{r}_k]_{\forall k\in\mathcal{K}}$, and λ is the weight balancing the network resource usage and the cost from re-configuring resource reservation. $\mathbb{R}^+$ represents the set of positive real numbers, and $\mathbb{N}$ represents the set of natural numbers. Constraint (4.15e) limits the communication resource usage of remote access averaged over all computing tasks from each group to be less than V_n^{re}. The solution of (4.15) provide a lower bound on the resources needed to support the stateful application, taking into account the resource reservation re-configuration cost in network planning. If the network resources are sufficient, more resources can be reserved for the application for better service quality. In addition, in a practical network supporting multiple applications, statistical multiplexing among resources reserved for different applications can be implemented for high resource utilization.

Solving Problem P0 is challenging due to the following two reasons: (i) For each time interval, determining the value of $\mathbf{r}_k$ is a mix-integer optimization problem, and the variables of $\mathbf{f}_k$ and $\mathbf{g}_k$ are mutually dependent; (ii) determining the value of a_k is a sequential decision making problem, and the decision at any time interval affects the subsequent decisions. To solve Problem P0, we decouple it into two problems. We propose algorithms for resource reservation in each time interval, which are presented in Section 4.4, and a learning-based approach for re-configuring resource reservation over multiple time intervals, which is presented in Section 4.5.

4.4 Group-based Resource Reservation

In this section, we design an algorithm to enable group-based resource reservation in each time interval.

When $a_k = 1$, the controller keeps using the resource reservation decision from time interval $k-1$. For when $a_k = 0$, we formulate the group-based multi-resource reservation problem in time interval k, given the predicted spatial task distributions $\tilde{\mathbf{x}}_k$, as follows:

$$\begin{aligned} \text{P1:} \min_{\mathbf{r}_k} \ & \frac{\Delta_k(\mathbf{r}_k)}{\sum_{b\in\mathcal{B}}\sum_{n\in\mathcal{N}}\tilde{x}_{b,k}^n} \\ \text{s.t. } & (4.3), (4.4), (4.6), (4.15\text{c-h}). \end{aligned} \tag{4.16}$$

Problem P1 is a combinatorial optimization problem, and variables $\mathbf{f}_k$ and $\mathbf{g}_k$ are still coupled. We first address computing task assignment, i.e., $\mathbf{f}_k$, and computing resource reservation, i.e., $\mathbf{c}_k$, given a storage resource reservation decision, i.e., $\mathbf{g}_k$. Then, we leverage particle swarm optimization to find the solution of storage resource reservation. The solution of (4.16) corresponds to group-specific resource reservation, and the total amount resources to be reserved for all groups can be calculated accordingly. The reserved resources can be multiplexed among different groups.

4.4.1 Computing Task Assignment and Computing Resource Reservation

Since multiple computing tasks can be assigned to the same server, assigning computing tasks to servers is many-to-one matching. We first transform the many-to-one matching into a one-to-one matching. Specifically, we create several *virtual servers* to represent each physical server. Denote the number of virtual servers for S-NAP or S-BS e by N_e. The value of N_e is the maximum number of computing tasks that can be assigned to server e while not violating the constraints of resource usage, i.e., constraints (4.3), (4.15c), and (4.15d). The number of virtual servers for physical server e is given by:

$$N_e = \begin{cases} \min\{\lfloor\frac{\tau^{\mathrm{p}}C_e}{\alpha\beta}\rfloor, \lfloor\frac{V_e^{\mathrm{up}}}{\sum_{\mathcal{B}_e} v_{b,e,k}^{\mathrm{up}}}\rfloor, \lfloor\frac{V_e^{\mathrm{down}}}{\sum_{\mathcal{B}_e} v_{b,e,k}^{\mathrm{down}}}\rfloor\}, \text{ if } e \in \mathcal{E}^{\mathrm{nap}}; \\ \lfloor\frac{\tau^{\mathrm{p}}C_e}{\alpha\beta}\rfloor, \text{ if } e \in \mathcal{E}^{\mathrm{bs}}, \end{cases} \tag{4.17}$$

where $\lfloor\cdot\rfloor$ represents the floor function. For an S-NAP, i.e., $e \in \mathcal{E}^{\mathrm{nap}}$, the value of N_e in Eq. (4.17) is the minimum value among the maximum number of computing tasks that can

be executed, i.e., $\lfloor \frac{\tau^{\mathrm{p}} C_e}{\alpha\beta} \rfloor$, the maximum number of computing tasks that can be uploaded, i.e., $\lfloor \frac{V_e^{\mathrm{up}}}{\sum_{\mathcal{B}_e} v_{b,e,k}^{\mathrm{up}}} \rfloor$, and the maximum number of computing tasks that can be downloaded, i.e., $\lfloor \frac{V_e^{\mathrm{down}}}{\sum_{\mathcal{B}_e} v_{b,e,k}^{\mathrm{down}}} \rfloor$; and for an S-BS, i.e., $e \in \mathcal{E}^{\mathrm{bs}}$, the value of N_e in Eq. (4.17) is the maximum number of computing tasks that can be executed, i.e., $\lfloor \frac{\tau^{\mathrm{p}} C_e}{\alpha\beta} \rfloor$. We let the number of the corresponding virtual servers for the S-CN in time interval k be $\sum_{b\in\mathcal{B}} \sum_{n\in\mathcal{N}} \tilde{x}_{b,k}^n$, i.e., the load of all computing tasks in time interval k, to guarantee that all computing tasks can be processed. Each virtual server is assigned at most one computing task, and assigning computing tasks to virtual servers becomes one-to-one matching.

Given the amount of storage resource reserved on S-BSs and S-NAPs, the sets of stored context data chunks, i.e., $\mathcal{I}_{e,k}$, on all S-BSs and S-NAPs are determined based on the hierarchal storage policy described in Section 4.2. Denote by $D_{b,e,k}^{n,i}$ the network resource usage, including resource usage from computing, uplink communication, downlink communication, and remote access, for executing a computing task that requests chunk i and is generated by a MUT from group n in the coverage of BS b during time interval k at server e. The calculation of $D_{b,e,k}^{n,i}$ consists of two parts. The first part is the resource usage from computing, uplink communication, and downlink communication, which is not related to the sets of stored chunks, while the second part is the communication resource usage from remote access, which depends on the sets of stored chunks. Denote by $W_{b,e} = w^{\mathrm{c}} \varepsilon_e^{\mathrm{c}} + w^{\mathrm{o}}(\alpha+\gamma)\eta_{b,e}$ the sum of resource usage from computing, uplink communication, and downlink communication used to execute a computing task generated in the coverage of BS b at server e. According to the communication resource usage for remote access, the calculation of $D_{b,e,k}^{n,i}$ is categorized into the following four cases:

$$D_{b,e,k}^{n,i} = \begin{cases} W_{b,e} + w^{\mathrm{o}} L^{\mathrm{re}} \xi_e^{e'}, \text{ if } e \in \mathcal{E}^{\mathrm{bs}}, e' \in \mathcal{E}^{\mathrm{nap}}, i \in \overline{\mathcal{I}_{e,k}} \cap \mathcal{I}_{e',k}; \\ W_{b,e} + w^{\mathrm{o}} L^{\mathrm{re}} \xi_e^{e^{\mathrm{cn}}}, \text{ if } e \in \mathcal{E}^{\mathrm{bs}}, e' \in \mathcal{E}^{\mathrm{nap}}, i \in \overline{\mathcal{I}_{e,k}} \cup \overline{\mathcal{I}_{e',k}}; \\ W_{b,e} + w^{\mathrm{o}} L^{\mathrm{re}} \xi_e^{e^{\mathrm{cn}}}, \text{ if } e \in \mathcal{E}^{\mathrm{nap}}, i \notin \mathcal{I}_{e,k}; \\ W_{b,e}, \text{ otherwise.} \end{cases} \tag{4.18}$$

In Eq. (4.18), if chunk i is not stored on S-BS e but stored on S-NAP e', i.e., $i \in \overline{\mathcal{I}_{e,k}} \cap \mathcal{I}_{e',k}$, the communication resource usage for S-BS e to remotely access S-NAP e' for one computing task is $w^{\mathrm{o}} L^{\mathrm{re}} \xi_e^{e'}$; If chunk i is not stored on S-BS e or S-NAP e', i.e., $i \in \overline{\mathcal{I}_{e,k}} \cup \overline{\mathcal{I}_{e',k}}$, the communication resource usage for S-BS e to remotely access the S-CN for one computing task is $w^{\mathrm{o}} L^{\mathrm{re}} \xi_e^{e^{\mathrm{cn}}}$; If chunk i is not stored on S-NAP e, i.e., $i \notin \mathcal{I}_{e,k}$, the communication resource usage for S-NAP e to remotely access the S-CN for one computing task is $w^{\mathrm{o}} L^{\mathrm{re}} \xi_e^{e^{\mathrm{cn}}}$; Otherwise, chunk i is stored on server e, and no communication resource is used for remote access.

Algorithm 3: MCLA Algorithm

Input: $\mathcal{L}_{u,k}^{\text{server}}, \forall u \in \mathcal{U}$
Initialization: $\mathcal{U}$, $\mathcal{U}^{\text{not}} = \mathcal{U}$, $\mathcal{T}$, $j = 0$;
while $|\mathcal{T}| = 0$ **do**
    $j = j + 1$;
    for $u \in \mathcal{U}^{not}$ **do**
        Select the first computing task in preference list $\mathcal{L}_{u,k}^{\text{server}}$ as the proposal from virtual server u;
        Remove the selected computing task from preference list $\mathcal{L}_{u,k}^{\text{server}}$;
    end
    Adopt the dynamic programming in [169] to select proposals from the new proposals in iteration j and adjust the matched proposals in iteration $j-1$ for minimizing the objective function in (4.19) while satisfying constraint (4.15e).
    Remove the matched virtual servers from $\mathcal{U}^{\text{not}}$;
    Add the unmatched virtual servers to $\mathcal{U}^{\text{not}}$;
    Remove the matched computing tasks from $\mathcal{T}$ based on the matching result in iteration j;
end
$\mathbf{f}_k \leftarrow \mathbf{z}_k$;
Output: $\mathbf{f}_k$

Denote the set of virtual servers and the set of computing tasks by $\mathcal{U}$ and $\mathcal{T}$, respectively. Given computing task $t \in \mathcal{T}$ and virtual server $u \in \mathcal{U}$, we can determine the corresponding values of (n, i, b) of the computing task and physical server e. Let $D_k^{(u,t)}$ represent the sum of computing and communication resource usage for executing computing task t at virtual server u in time interval k, which can be calculated via Eq. (4.18) based on the corresponding values of (n, i, b) and physical server e. We introduce variable $z_k^{(u,t)} \in \{0, 1\}$ to indicate whether to assign computing task t to virtual server u in time interval k or not and define $\mathbf{z}_k = [z_k^{(u,t)}]_{\forall u \in \mathcal{U}, t \in \mathcal{T}}$. If computing task t is assigned to virtual server u, $z_k^{(u,t)} = 1$; Otherwise, $z_k^{(u,t)} = 0$. Finding the solution of $\mathbf{f}_k$ in Problem P1 can be transformed into finding the solution of $\mathbf{z}_k$, as follows:

$$\begin{aligned} \text{P2:} \min_{\mathbf{z}_k} \ & \sum_{u \in \mathcal{U}} \sum_{t \in \mathcal{T}} D_k^{(u,t)} z_k^{(u,t)} \\ \text{s.t. } & (4.15e), \\ & z_k^{(u,t)} \in \{0, 1\}, \ \forall k \in \mathcal{K}. \end{aligned} \tag{4.19}$$

We propose a matching-based computing task assignment (MCLA) algorithm to select a virtual server for each computing task, as shown in Algorithm 3.

We construct the preference list $\mathcal{L}_{u,k}^{\text{server}}$ for virtual server u in time interval k, which is a vector containing the indexes of all computing tasks that can be assigned to virtual server u in time interval k, sorted by the value of $D_k^{(u,t)}$ in a non-decreasing order. In each iteration, the controller checks the current preference list for each virtual server and selects the first computing task in the preference list as the proposal from the virtual server. After that, the selected computing task is removed from the virtual server's preference list. The controller selects a proposal for each computing task to minimize the objective function in Problem P2 while satisfying constraint (4.15e), which is a 0-1 Knapsack problem. Let $\mathcal{U}^{\text{not}} \subset \mathcal{U}$ denote the set of virtual servers that are not yet matched. A dynamic programming approach in [169] is adopted to adjust the matching result, i.e., selecting proposals for virtual servers in the set $\mathcal{U}^{\text{not}}$ and adjusting the matched proposals for virtual servers in the set $\mathcal{U} \setminus \mathcal{U}^{\text{not}}$ for minimizing the objective function in Problem P2 while satisfying constraint (4.15e). Then, set $\mathcal{U}^{\text{not}}$ is updated accordingly after each iteration. A computing task should be re-proposed for each virtual server in the set $\mathcal{U}^{\text{not}}$ in the next iteration. The matching process terminates when all computing tasks in the set $\mathcal{T}$ are successfully matched. Based on the matching result, i.e., $\mathbf{z}_k$, we can determine the computing task assignment decision, i.e., $\mathbf{f}_k$.

Given $\mathbf{f}_k$, the numbers of computing tasks assigned to the servers, i.e., $\mathbf{m}_k$, are determined based on Eq. (4.2). Then, computing resource reservation in time interval k is determined according to Eq. (4.3), which is:

$$\mathbf{c}_k = \frac{\alpha\beta}{\tau^{\text{p}}}\mathbf{m}_k, \ \forall k \in \mathcal{K}. \tag{4.20}$$

The time complexity of Algorithm 3 depends on the number of iterations of the outer loop for matching (Lines 5 - 13) and the time complexity of solving the knapsack problems in each iteration (Line 9). For the outer loop, the time complexity of solving the matching problem is $O(ZX_k)$, where $Z = \sum_{e\in\mathcal{E}} N_e$ and $X_k = \sum_{n\in\mathcal{N}} \sum_{b\in\mathcal{B}} x_{b,k}^n$ denote the number of all virtual servers and the number of all computing tasks in time interval k, respectively [170]. In each iteration, we should solve a knapsack problem to satisfy constraint (4.15e) for every group. The time complexity of the adopted dynamic programming approach for group n is $O(V_n^{\text{re}} X_k^n)$, where $X_k^n = \sum_{b\in\mathcal{B}} x_{b,k}^n$ denotes the number of all computing tasks from group n in time interval k [169]. Therefore, the time complexity of Algorithm 1 is $O(ZX_k \sum_{n\in\mathcal{N}} V_n^{\text{re}} X_k^n)$.

4.4.2 Storage Resource Reservation

Determining the amount of storage resources reserved on S-NAPs and S-BSs is a combinatorial optimization problem. Therefore, finding the globally optimal solution is challenging [171]. Evolutionary heuristics, specifically particle swarm optimization (PSO), is leveraged to achieve the local optima of the problem. Based on PSO, we propose a resource reservation (RR) algorithm to solve Problem P1. In the proposed RR algorithm, we leverage a number of particles, referred to as the particle swarm, where the position of each particle corresponds to the solution of storage resource reservation, i.e., $\mathbf{g}_k$. In each iteration, each particle moves within the solution space while adjusting its position and speed dynamically based on [171]. After repeating such an iteration multiple times, the positions of all particles can converge to the same position, which is the found solution of storage resource reservation [172]. Given the storage resource reservation, the values of $\mathbf{f}_k$ and $\mathbf{c}_k$ can be determined based on Section 4.4.1.

The detailed procedures of the RR algorithm are introduced in Algorithm 4. Since the RR algorithm applies to any time interval, we omit subscript "k" in the rest of this subsection. We define the the solution space of the storage resource reservation problem as $\mathbb{F}$, where the possible solution of $\mathbf{f}_k$, i.e., particles' positions, should satisfy constraint (4.6). Denote the set of particles and the position of particle y in the l th iteration by $\mathcal{Y}$ and $\mathbf{g}_y^{(l)}$, respectively. Let $\hat{\mathbf{g}}_y$ and $\mathbf{g}^*$ denote the best position of particle y and the best position among all particles' positions, i.e., the particle swarm's best position, up to the l th iteration, respectively. Accordingly, let $\Delta_y^{(l)}$, $\hat{\Delta}_y$, and Δ^* denote the value of Δ in Problem P1 given position $\mathbf{g}_y^{(l)}$, $\hat{\mathbf{g}}_y$, and $\mathbf{g}^*$, respectively. Based on [172], the speed of particle $y \in \mathcal{Y}$ in the l th iteration, denoted by $\mathbf{s}_y^{(l)}$, evolves as follows:

$$\begin{aligned}\mathbf{s}_y^{(l)} = \varsigma \mathbf{s}_y^{(l-1)} &+ \varsigma_1 \varphi_1 \left(\hat{\mathbf{g}}_y - \mathbf{g}_y^{(l-1)}\right) \\ &+ \varsigma_2 \varphi_2 \left(\mathbf{g}^* - \mathbf{g}_y^{(l-1)}\right),\end{aligned} \tag{4.21}$$

where parameter ς is the weight for each particle to keep its speed from the previous iteration. Parameters ς_1 and ς_2 are cognitive and social coefficients for learning from each particle's own best position and the particle swarm's best position up to the current iteration, respectively, and both are positive random variables for exploring the solution space [172]. The position of particle y, i.e., $\mathbf{g}_y^{(l)}$, in the l th iteration is given by:

$$\mathbf{g}_y^{(l)} = \mathbf{g}_y^{(l-1)} + \mathbf{s}_y^{(l)}. \tag{4.22}$$

If a particle moves out of the solution space, the particle is replaced by a new particle with a random position in the solution space $\mathbb{F}$. In this way, the positions of all particles are guaranteed to satisfy constraint (4.6).

Algorithm 4: RR Algorithm

1 **Input:** ς, ς_1, ς_2, φ_1, φ_2, $l^{\max}$
2 **Initialization:** $l = 0$, $\mathbf{s}_y^{(0)} = \mathbf{0}$, $\mathbf{g}_y^{(0)} \in \mathbb{F}$, $\forall y \in \mathcal{Y}$;
3 $\mathbf{f}_y^{(0)}, \forall y \in \mathcal{Y} \leftarrow$ Calculate by Algorithm 3 given $\mathbf{g}_y^{(0)}$;
4 $\hat{\Delta}_y, \forall y \in \mathcal{Y} \leftarrow$ Calculate by (4.12) given $\mathbf{c}_y^{(0)}$, $\mathbf{g}_y^{(0)}$, and $\mathbf{f}_y^{(0)}$;
5 $\hat{\mathbf{g}}_y, \forall y \in \mathcal{Y} \leftarrow \mathbf{g}_y^{(0)}, \forall y \in \mathcal{Y}$;
6 $\Delta^* \leftarrow \min\left\{\hat{\Delta}_y, \forall y \in \mathcal{Y}\right\}$;
7 $\mathbf{g}^* \leftarrow \hat{\mathbf{g}}_{y'}$ where $y' = \arg\min_y \left\{\hat{\Delta}_y, \forall y \in \mathcal{Y}\right\}$;
8 **while** $l \leq l^{max}$ **do**
9 **for** $y \in \mathcal{Y}$ **do**
10 **if** *constraint* (4.6) *is not satisfied* **then**
11 $\mathbf{g}_y^{(l)} \leftarrow$ Select a position in $\mathbb{F}$ randomly;
12 **end**
13 $\mathbf{f}_y^{(l)} \leftarrow$ Calculate by Algorithm 3 given $\mathbf{g}_y^{(l)}$;
14 $\Delta_y^{(l)} \leftarrow$ Calculate by Eq. (4.12) given $\mathbf{c}_y^{(l)}$, $\mathbf{g}_y^{(l)}$, and $\mathbf{f}_y^{(l)}$;
15 **if** $\Delta_y^{(l)} < \hat{\Delta}_y$ **then**
16 $\hat{\mathbf{g}}, \hat{\Delta}_y \leftarrow \mathbf{g}_y^{(l)}, \Delta_y^{(l)}$;
17 **end**
18 **if** $\Delta_y^{(l)} < \Delta^*$ **then**
19 $\mathbf{g}^*, \Delta^* \leftarrow \mathbf{g}_y^{(l)}, \Delta_y^{(l)}$;
20 **end**
21 $\mathbf{s}_y^{(l+1)}, \mathbf{g}_y^{(l+1)} \leftarrow$ Update by Eqs. (4.21) and (4.22), respectively;
22 **end**
23 **end**
24 $\mathbf{f}^*, \mathbf{c}^* \leftarrow$ Calculate by Algorithm 3 and Eq. (4.20) given $\mathbf{g}^*$, respectively;
25 **Output:** $\mathbf{g}^*$, $\mathbf{f}^*$, $\mathbf{c}^*$, Δ^*

Algorithm 4 shows the detail of the proposed RR algorithm. Denote the maximum number of iteration by $l^{\max}$. Line 2 initializes all particles with the positions in the solution space $\mathbb{F}$. Line 3 to Line 4 obtain the value of Δ in Problem P1, i.e., $\hat{\Delta}_y$, given the position of particle y, i.e., $\mathbf{g}_y^{(l)}$. Line 5 to Line 7 find the best solution among all particles based on the value of $\hat{\Delta}_y$. Line 10 to Line 21 update the position of each particle based on Eqs. (4.21) and (4.22) in each iteration. The outputs of the RR algorithm include the solution of Problem P1, i.e., $\mathbf{g}^*$, $\mathbf{f}^*$, and $\mathbf{c}^*$, and the value of Δ^*.

4.5 Meta Learning based Resource Reservation Re-configuration

In the preceding section, we solve the network resource reservation problem for one time interval. In this section, we determine the value of $\mathbf{a} = [a_k]_{\forall k \in \mathcal{K}}$ to re-configure resource reservation decisions among K time intervals.

We formulate the sub-problem of resource reservation re-configuration based on Problem P0, as follows:

$$\text{P3:} \min_{\mathbf{a}} \sum_{k \in \mathcal{K}} \frac{\Delta_k(\mathbf{r}_k) + \lambda o_k^{\text{v}}}{\sum_{b \in \mathcal{B}} \sum_{n \in \mathcal{N}} \tilde{x}_{b,k}^n} \tag{4.23a}$$

$$\text{s.t. (4.15i).} \tag{4.23b}$$

We define $k^\star$ as the time interval when the latest resource reservation was re-configured up to time interval k. In time interval $k^\star$, $a_{k^\star} = 0$, $a_j = 1$ for $j \in [k^\star + 1, \cdots, k-1]$ and $k^\star < k$. The relation between k and $k^\star$ is given by:

$$k^\star = \begin{cases} k, \text{ if } a_k = 0; \\ k^\star, \text{ otherwise.} \end{cases} \tag{4.24}$$

Problem P3 is a sequential decision making problem, which can be solved by reinforcement learning (RL) based methods [103]. However, RL-based methods cannot be applied directly to solve Problem P3. Resource reservation re-configuration is owing to the difference of *network status*, i.e., spatial task distributions and chunk request ratio profiles in this work, in different time intervals. Identifying the difference between network status can potentially improve the learning efficiency of RL-based methods.

We propose a Meta-learning-based resource reservation re-configuration (MetaR^3) approach to solve Problem P3, as given in Algorithm 5. At the end of each time interval, data regarding spatial task distributions, chunk request ratio profiles, indicator a_k and the network performance are collected and stored based on UDTs. Meta learning is adopted to capture the *similarity* between network status in two time intervals k and $k^\star$, and the policy of resource reservation decision re-configuration is learned based on the captured similarity by using RL. Based on the learned policy of resource reservation decision re-configuration, the value of a_k can be determined. If $a_k = 0$, resource reservation decision is re-configured at the beginning of time interval k.

Two components are underlying the proposed MetaR^3 approach: (i) capturing the similarity between network status during two different time intervals, and (ii) re-configuring resource reservation in a closed-loop manner, which are presented in the following two subsections, respectively.

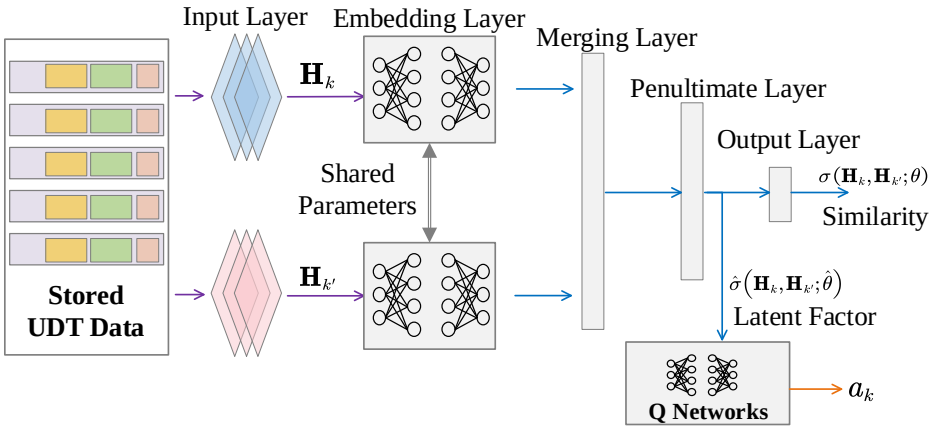


Figure 4.4: The proposed MetaR^3 approach.

4.5.1 Similarity Capture

Denote the network status in time interval k by $\mathbf{h}_k = [\mathbf{x}_k, \mathbf{p}_k]$. The value of $\mathbf{h}_k$ is unavailable at the beginning of time interval k. Therefore, we use the data regarding spatial task distributions and chunk request ratio profiles in past T time intervals contained in UDTs as the features of the network status in time interval k, denoted by $\mathbf{H}_k = [\mathbf{h}_{k-T}, \cdots, \mathbf{h}_{k-1}]$. Based on $\mathbf{H}_k$, the value of a_k is determined. If $a_k = 0$, $\mathbf{H}_k$ can be used to predict the network status, i.e., $\mathbf{h}_k$, for making resource reservation decision. Define the similarity between the network status in different time intervals k and k' as $\sigma(\mathbf{H}_k, \mathbf{H}_{k'})$.

We leverage Meta learning with siamese neural networks to approximate the value of $\sigma(\mathbf{H}_k, \mathbf{H}_{k'})$ [173], as illustrated in Fig. 4.4. Let θ and $\hat{\theta}$ denote the parameters of the whole siamese neural networks and the parameters of the siamese neural networks without the output layer, respectively. The inputs of the siamese neural networks are the features of network status, i.e., $\mathbf{H}_k$ and $\mathbf{H}_{k'}$. The siamese neural networks have two outputs, i.e., the output of the whole siamese neural network and the output of the penultimate layer. The output of the whole siamese neural network represents the value of similarity, denoted by $\sigma(\mathbf{H}_k, \mathbf{H}_{k'}; \theta)$, and the output of the penultimate layer represents the *latent factors* of similarity, denoted by $\hat{\sigma}(\mathbf{H}_k, \mathbf{H}_{k'}; \hat{\theta})$.

Training siamese neural networks should be based on labeled data. Define $\varrho(\mathbf{H}_k, \mathbf{H}_{k'}) \in$

$\{0, 1\}$ as a *label* of the features of network status in two time intervals k and k', given by:

$$\varrho(\mathbf{H}_k, \mathbf{H}_{k'}) = \begin{cases} 1, \text{ if } \Delta_k(\mathbf{r}_k) - \Delta_k(\mathbf{r}_{k'}) > \lambda O^{\mathrm{v}}; \\ 0, \text{ otherwise.} \end{cases} \tag{4.25}$$

In Eq. (4.25), $\Delta_k(\mathbf{r}_k)$ denotes the network resource usage in time interval k if the resource reservation is re-configured, i.e., $a_k = 0$. $\Delta_k(\mathbf{r}_{k'})$ denotes the network resource usage in time interval k if the resource reservation is not re-configured, i.e., $a_k = 1$, and the resource reservation decision from time interval k' is used, i.e., $\mathbf{r}_{k'}$. If $\Delta_k(\mathbf{r}_k) - \Delta_k(\mathbf{r}_{k'}) > \lambda O^{\mathrm{v}}$, the network status in two time interval k and k' are considered to be "similar"; Otherwise, the network status in the two time intervals are considered to be "not similar". The features of the network status in any two time intervals and the corresponding value of $\varrho(\mathbf{H}_k, \mathbf{H}_{k'})$ are referred to as a labeled data entry. The goal of training the siamese neural networks is to let $\sigma(\mathbf{H}_k, \mathbf{H}_{k'}; \theta)$ approximate label $\varrho(\mathbf{H}_k, \mathbf{H}_{k'})$ by using extensive labeled data entries. The parameters of the siamese neural networks, i.e., θ, are obtained by minimizing the following loss function via gradient descent [173]:

$$\begin{aligned} \theta^* = \arg\min_{\{\theta\}} & \varrho(\mathbf{H}_k, \mathbf{H}_{k'}) \log\left(\sigma(\mathbf{H}_k, \mathbf{H}_{k'}; \theta)\right) + \\ & (1 - \varrho(\mathbf{H}_k, \mathbf{H}_{k'})) \log\left(1 - \sigma(\mathbf{H}_k, \mathbf{H}_{k'}; \theta)\right). \end{aligned} \tag{4.26}$$

The approximated value of similarity, i.e., $\sigma(\mathbf{H}_k, \mathbf{H}_{k'}; \theta)$, can indicate whether the features of network status in two time intervals are similar or not. However, its information on how much a difference between network status is insufficient for re-configuring resource reservation.[4] Therefore, we use the latent factors of similarity, i.e., $\hat{\sigma}(\mathbf{H}_k, \mathbf{H}_{k'}; \hat{\theta})$, to determine the value of a_k.

4.5.2 Closed-loop Resource Reservation Re-configuration

We leverage deep Q learning with deep neural networks, named Q networks, to determine the value of a_k given the latent factors of similarity. The state and action in time interval k are $\hat{\sigma}(\mathbf{H}_k, \mathbf{H}_{k^\star}; \hat{\theta})$ and a_k, respectively. For simplicity, let $\hat{\sigma}_k$ denote $\hat{\sigma}(\mathbf{H}_k, \mathbf{H}_{k^\star}; \hat{\theta})$ in the rest of this section. Define a Q-value function to represent the discounted long-term resource usage and cost of making decision a_k in state $\hat{\sigma}_k$, given by:

$$Q(\hat{\sigma}_k, a_k) = \sum_{k=1}^{K} \rho^k \frac{\Delta_k + \lambda o_k^{\mathrm{v}}}{\sum_{b \in \mathcal{B}} \sum_{n \in \mathcal{N}} \tilde{x}_{b,k}^n}. \tag{4.27}$$

[4]The output layer in the siamese neural networks is used for training the siamese neural networks.

Algorithm 5: MetaR^3 Approach

1 **Input:** ρ
2 **Initialization:** θ, $\hat{\theta}$, ϑ, $k^\star$, $\mathbf{H}_1$, $\hat{\sigma}_1$
3 **for** $k = 1, \cdots, K$ **do**
4 $\quad \sigma_k \leftarrow$ Obtain $\hat{\sigma}(\mathbf{H}_k, \mathbf{H}_{k^\star}; \hat{\theta})$;
5 $\quad a_k \leftarrow$ Determine by Eq. (4.28);
6 $\quad \mathbf{r}_k \leftarrow$ Determine by Eq. (4.13);
7 $\quad \Delta_k(\mathbf{r}_k)$, $\hat{\sigma}_{k+1} \leftarrow$ Implement $\mathbf{r}_k$ for time interval k;
8 $\quad \theta$, $\hat{\theta}$, $\vartheta \leftarrow$ Train parameters via Eqs. (4.26) and (4.29);
9 $\quad k^\star \leftarrow$ Update by Eq. (4.24);
10 **end**
11 **Output: a**

where $\rho \in (0, 1)$ is the discount factor. In state $\hat{\sigma}_k$, a_k can be determined based on the Q-values as follows:

$$a_k = \arg \max_{a \in \{0,1\}} Q(\hat{\sigma}_k, a), \ \forall k \in \mathcal{K}. \tag{4.28}$$

The Q network with parameter ϑ is used to approximate the Q-value function for learning the policy of resource reservation re-configuration. The parameters ϑ of the Q networks are obtained by minimizing the following loss function via gradient descent [40]:

$$\begin{aligned} \vartheta^* = \arg \min_{\{\vartheta\}} \Big| & \frac{\Delta_k + \lambda o_k^{\mathrm{v}}}{\sum_{b \in \mathcal{B}} \sum_{n \in \mathcal{N}} \tilde{x}_{b,k}^n} + \rho \max_a Q(\hat{\sigma}_{k+1}, a; \vartheta) \\ & - Q(\hat{\sigma}_k, a_k; \vartheta) \Big|^2. \end{aligned} \tag{4.29}$$

We summarize the workflow of the MetaR^3 approach in Algorithm 5. Line 4 to Line 5 determine a_k at the beginning of time interval k based on the predicted network status obtained from UDTs. Given a_k, Line 6 to Line 7 determine and implement the resource reservation decision. At the end of the time interval, data regarding network performance, actual spatial task distributions, and chunk request ratio profiles are collected and stored in UDTs. Given the stored historical data, the siamese neural networks and Q networks can be trained to adapt to dynamic spatial task distribution and chunk request ratio profile in a closed-loop manner.

4.6 Performance Evaluation

4.6.1 Simulation Settings

The simulated multi-tier network consists of 1 S-CN, 2 S-NAPs, and 4 to 10 S-BSs. In the network, there are 600 MUTs with different trajectories within the networks. Based on the average time within the coverage of each BS for each MUT, these MUTs are grouped into 2 to 4 groups. The input data size, computing workload, and the size of computing results of each computing task are set to 2 MB, 4 Megacycles/s, and 15 MB, respectively [138]. The network resource usage and the resource capacity of servers at the same tier can be different. The average network resource usage, average resource capacity of servers, and other parameters are listed in Table 4.1.

Table 4.1: Simulation Parameters

Parameter	Value	Parameter	Value
G^{bs}, G^{nap}	0.9, 2 Gigacycles/s	C^{bs}, C^{nap}	0.75, 1.5 GB
$\xi^{\text{nap}}_{\text{bs}}$, $\xi^{\text{cn}}_{\text{bs}}$	$3.5 * 10^{-9}$, $2.5 * 10^{-9}$	$\xi^{\text{cn}}_{\text{nap}}$	$6 * 10^{-9}$
$\varepsilon^{\text{c}}_{\text{bs}}$, $\varepsilon^{\text{c}}_{\text{nap}}$,$\varepsilon^{\text{c}}_{\text{cn}}$	1, 1, 1	I	20
$\varepsilon^{\text{s}}_{\text{bs}}$, $\varepsilon^{\text{s}}_{\text{nap}}$,$\varepsilon^{\text{s}}_{\text{cn}}$	0.8, 0.5, 1	L	0.15 GB
η^{nap}, η^{cn}	$5 * 10^{-9}$, $9 * 10^{-9}$	τ^{p}	0.5 s
w^{s}, w^{c}, w^{o}	$0.5 * 10^{-7}$, 1, 1	λ	12

For the siamese neural networks illustrated in Fig. 4.4, we use 3 fully connected layers with (64, 64, 32) neurons as a embedding layer. The features of network status $\mathbf{H}_k$ and $\mathbf{H}_{k'}$ are fed to two embedding layers separately, each with the same structure. The merging layer merges the outputs of the two embedding layers based on Euclidean distance, followed by the penultimate layer with 16 neurons and the output layer with 1 neuron. For the Q networks, we adopt 4 fully connected layers with 128, 512, 128, 32 neurons, respectively. We adopt the RMSprop optimizer and the Adam optimizer for training the siamese neural networks and the Q networks, respectively.

4.6.2 Performance of Group-based Resource Reservation

The convergence performance of the proposed RR algorithm for group-based resource reservation is shown in Fig. 4.5. Given the same spatial task distribution and configuration of all servers, we conduct the simulation with 2, 8, and 32 particles for 30 iterations. The

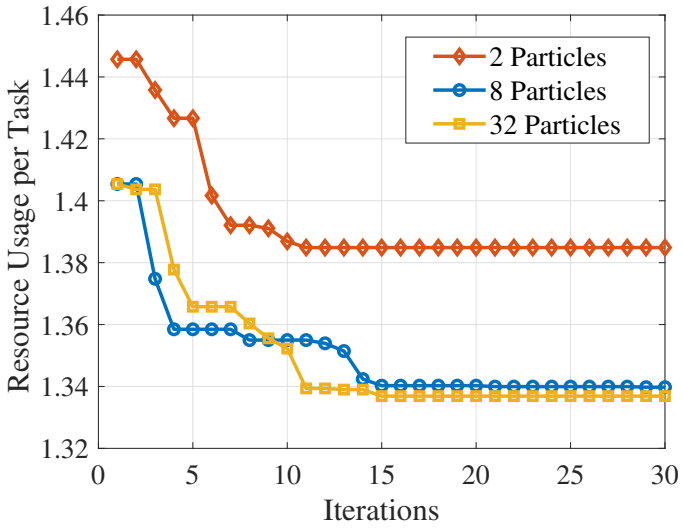
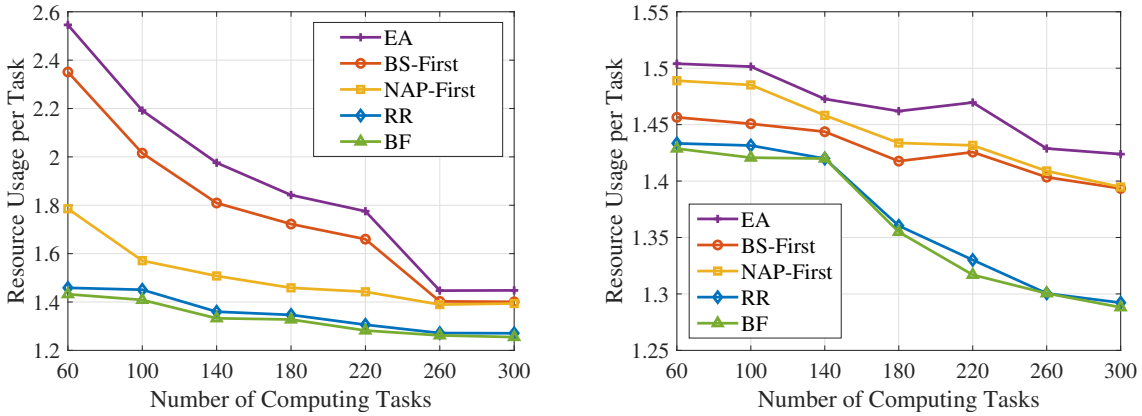


Figure 4.5: Convergence performance of the proposed RR algorithm.

proposed algorithm converges after 10, 14, and 16 iterations, respectively. With more particles, the algorithm achieves better performance at the cost of computation complexity.

Fig. 4.6(a) and Fig. 4.6(b) compare the network resource usage per computing task, versus the load of computing tasks, of the proposed RR algorithm and the benchmark algorithms. We adopt four benchmark algorithms: (1) BS-first, which assigns computing tasks to S-BSs and reserves storage and computing resources on S-BSs first as much as possible, then on S-NAPs, and last on the S-CN. (2) NAP-first, which assigns computing tasks to S-NAPs and reserves storage and computing resources on S-NAPs first as much as possible, then on S-BSs, and last on the S-CN. (3) EA, which assigns computing tasks and reserves storage and computing resources on S-BS, S-NAP, and S-CN with equal priority. (4) BF, which is a brute force algorithm to find the global optimum. We have the three following observations. First, for both even and uneven task distribution, the network resource usage per computing task of the RR algorithm is close to the global optimum and much lower compared to the benchmark algorithms (expect the BF algorithm). Second, the network resource usage per computing task decreases as the load of computing tasks generated in the network increases. This is because the storage resource usage per task decreases as the load of computing tasks requesting the same stored chunk increases. Third, the performance gap between the RR algorithm and the benchmark algorithms (expect the BF algorithm) under uneven spatial task distribution is larger than under even spatial task distribution. With uneven spatial task distribution, the optimal resource reservation may



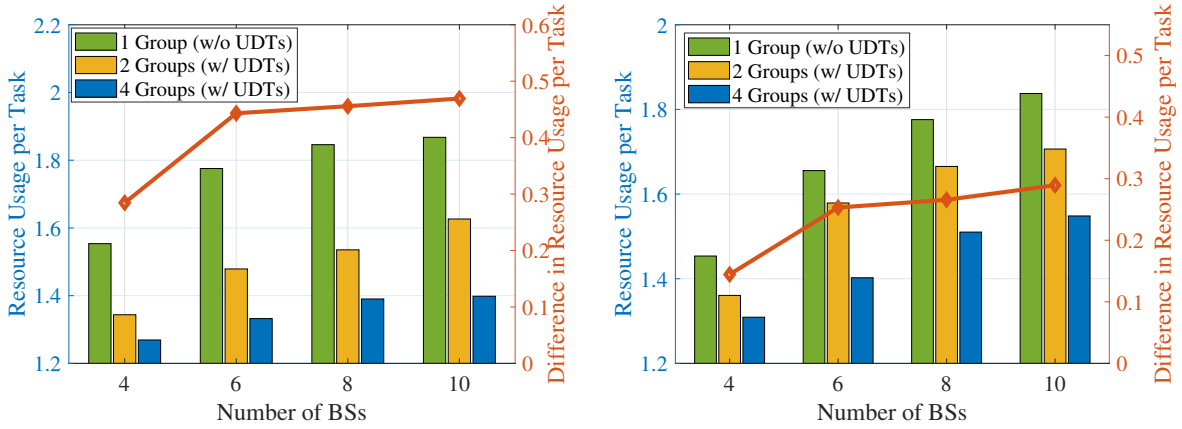
(a) Uneven spatial task distribution.

(b) Even spatial task distribution.

Figure 4.6: Network resource usage per computing task under even and uneven spatial task distributions.

be different for servers at the same tier. The RR algorithm can differentiate resource reservation decisions for different servers at the same tier using group-based spatial task distribution.

Figures 4.7(a) and 4.7(b) show the network resource usage per computing task versus the number of BSs and the number of groups in different scenarios. Specifically, the heterogeneous scenario means that the network resource usage for executing a computing task and resource capacity of servers at the same tier are different. The homogeneous scenario means that the network resource usage for executing a computing task and resource capacity of servers at the same tier are identical. The scheme labeled as "1 Group (w/o UDTs)" represents resource reservation without UDTs, and the schemes labeled as "n Groups (w/ UDTs)" represent the proposed RR algorithm with n groups. We have the following two observations. First, group-based resource reservation with UDTs outperforms resource reservation without UDTs in both homogeneous and heterogeneous scenarios. Without UDTs, group-based spatial task distribution are unknown for resource reservation and computing task assignment. As a result, schemes without UDTs should reserve more resources to satisfy constraint (4.15e), i.e., each group's communication resource usage for remote access. With more groups, the group-based resource reservation can achieve better performance at the cost of computation complexity and data management for UDTs. Second, the curve with diamond markers represents the performance gap between 4-group based resource reservation with UDTs ("4 Group (w/ UDTs)") and resource reservation



(a) Heterogeneous settings.

(b) Homogeneous settings.

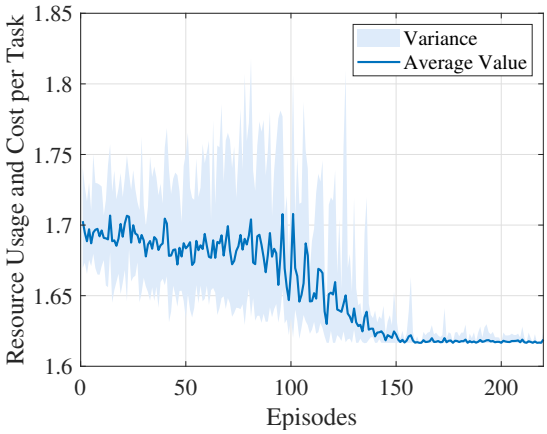
Figure 4.7: Network resource usage per computing task in heterogeneous and homogeneous scenarios.

without UDTs ("1 Group (w/o UDTs)"). In both homogeneous and heterogeneous scenarios, the effectiveness of group-based resource reservation increases with the number of BSs.

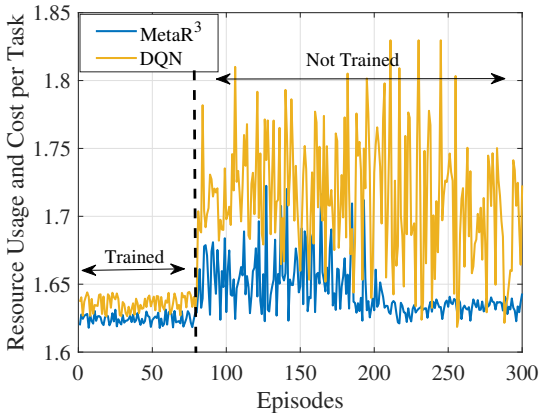
4.6.3 Performance of Resource Reservation Re-configuration

We create a training dataset that includes spatial task distributions in 80 time intervals. Conducting the simulation for all 80 spatial task distributions in the training dataset is referred to as one episode. We conduct 10 simulations on the dataset, and each simulation includes 220 episodes. In Fig. 4.8(a), the smooth solid line is the average result over 10 simulations, while the spikes in the background represent the corresponding variance. Fig. 4.8(a) shows that the proposed MetaR^3 algorithm can converge and find a policy of resource reservation re-configuration given a fixed network environment.

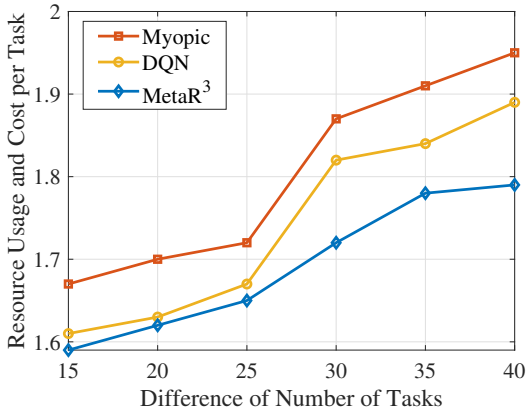
In Fig. 4.8(b), we compare the convergence performance of the MetaR^3 approach with that of a deep Q Learning (DQN) based algorithm, labeled as "DQN". In DQN, the group-based spatial task distribution is used as the state to determine the value of a_k. We create two datasets with different spatial task distributions. One training dataset is used to train the neural networks in advance, and one evaluation dataset is used to evaluate the convergence performance of MetaR^3. The evaluation dataset reveals network status from unknown network environments. Note that MetaR^3 keeps training the siamese neural



(a) Convergence performance of MetaR^3.



(b) Performance Comparison between MetaR^3 and DQN.



(c) The impact of network dynamics.

Figure 4.8: Performance of MetaR^3 in the weighted sum of network resource usage and cost from re-configuring resource reservation per computing task.

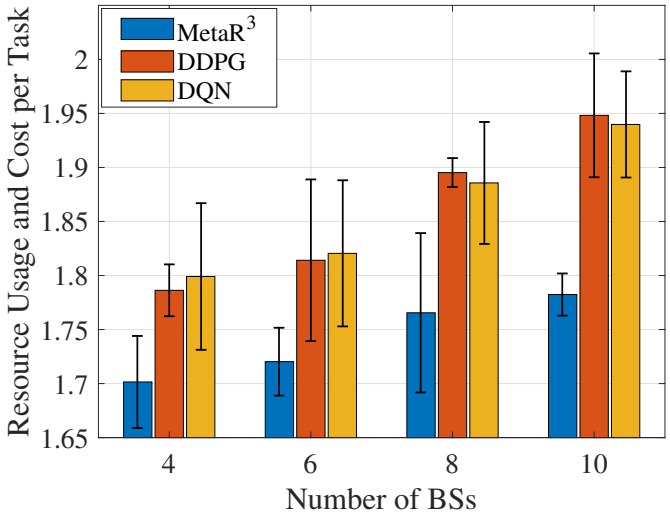


Figure 4.9: Resource usage and cost per computing task of the MetaR^3, DDPG, and DQN-based algorithms.

networks and the Q networks in unknown network environment due to the closed-loop re-configuration of resource reservation. We observe that MetaR^3 achieves lower network resource usage and lower cost from re-configuring resource reservation per computing task, and also converges in fewer episodes in unknown network environment compared with the DQN algorithm. This is because MetaR^3 captures the similarity of network status, instead of learning the variation of network status, to determine a_k even though the current network status is unknown.

Figure 4.8(c) shows the performance in the weighted sum of the network resource usage and the cost from re-configuring resource reservation per computing task versus the average difference in the load of computing tasks in adjacent two time intervals. When the average difference in the load of computing tasks in adjacent two time intervals increases, spatial task distribution changes faster. The benchmark algorithm, labeled as "Myopic", determines whether to re-configure resource reservation or not in each time interval without considering the long-term impact. We observe that the performance gaps between MetaR^3 and "DQN" and "Myopic" algorithms increase with the average difference in the load of computing tasks in adjacent two time intervals since the similarity capture features of MetaR^3 can reduce the state space for finding a good policy of resource reservation re-configuration in dynamic network environments, which improves learning efficiency.

In Fig. 4.9, we compare the performance of the proposed MetaR^3 algorithm with that of

two popular RL algorithms, i.e., deep deterministic policy gradient (DDPG)-based (labeled as "DDPG") and DQN-based algorithms in different network environments [103, 174]. Specifically, we conduct simulations of the three algorithms with different numbers of BSs and average the resource usage and cost per computing task over three independent simulations. We can observe that the proposed MetaR^3 algorithm outperforms the DDPG- and DQN-based algorithms. This is because both DDPG- and DQN-based algorithms use network status as states. When the network status has a large dimensionality and network environments are highly dynamic, finding the optimal resource reservation re-configuration policy is challenging for the DDPG and DQN-based algorithms. In contrast, the proposed MetaR^3 algorithm can capture the similarity of network status in consecutive time intervals. Since the similarity is low-dimensional, using similarities as states has advantage on finding a proper policy of resource reservation re-configuration, particularly in complicated network environments. Therefore, the proposed MetaR^3 algorithm achieves better network performance than DDPG- and DQN-based algorithms.

4.7 Summary

In this chapter, we have designed DT-empowered network planning for supporting stateful applications in multi-tier computing and proposed two approaches to enable group-based multi-resource reservation and closed-loop resource reservation re-configuration. Our study focuses on minimizing the long-term network resource usage and the cost from re-configuring resource reservation. The results have demonstrates that DT-empowered network planning can support MUTs with diverse characteristics and adapt to dynamic network environments. In addition, the Meta-learning-based approach can exploit data contained in DTs to facilitate closed-loop network planning. Overall, we have demonstrated the essential role that DTs can play in network planning for the NGWN.

Chapter 5

Operation-stage Computing Task Scheduling in SAGIN

In this chapter, We investigate operation-stage computing resource allocation in the heterogeneous space-air-ground integrated networks (SAGIN). In remote areas, terrestrial BS has limited communication coverage for providing computing service for all MUTs. Satellites can provide global coverage, but satellite communication has a long propagation delay. An unnamed aerial vehicle (UAV) is deployed to fly around MUTs and collect their computing tasks, while scheduling the collected computing tasks to be processed at the UAV locally or offloaded to the nearby terrestrial BSs or the remote satellite. The energy budget of the UAV, intermittent connectivity between the UAV and terrestrial BSs, and dynamic computing task arrival pose challenges in computing task scheduling. Our research objective is to design a computing task scheduling policy for minimizing the delay of computing task offloading and processing in SAGIN. To achieve the objective, we first formulate the online computing scheduling in the dynamic network environment as a constrained Markov decision process. Then, we develop a risk-sensitive DRL approach in which a risk value is used to represent energy consumption that exceeds the budget. By balancing the risk value and the reward from delay minimization, the UAV can explore the task scheduling policy to minimize task offloading and processing delay while satisfying the UAV energy constraint. Simulation results show the proposed data-driven resource allocation approach reduces the delay of task offloading and processing while not exceeding the UAV's energy budget compared with conventional schemes.

5.1 Background and Motivations

Equipped with advanced embedded monitoring and data collection technologies, MUTs, such as high definition cameras, object detectors, and meteorological sensors, play vital roles in a myriad of applications and services [175]. Specifically, MUTs can be deployed to monitor and sense the environment, offering new opportunities for industrial automation, intelligent transportation management, etc. There are two typical applications of delay-oriented computing services: intelligent urban transportation management and automated surface mining in suburban areas. For intelligent transportation management, on-board cameras and road-side sensors can reliably detect incidents, such as traffic signal violations, stopped vehicles, and on-road pedestrians. By leveraging deep learning-based image processing techniques, vehicle and pedestrian behaviors can be predicted to prevent potential traffic accidents in advance [1]. Rapidly processing the collected image can save more time in reacting to the complicated transportation scenarios, which enhances the road safety by preventing the transportation emergency. For automated surface mining, a large number of cameras and visual sensors are deployed in the active areas of the drill rigs to assess rock composition and collect environment information (e.g., humidity and temperature). The analytics results of input image/video from these MUTs can help achieve automated drilling control [4]. In this case, lower delay of image/video analytic can enable more accurate automated surface mining control. Generally, such computing services are delay-oriented which should be processed rapidly to adapt to highly dynamic input.

To support the aforementioned services, ubiquitous delay-oriented computing tasks become prevailing on MUTs, resulting in a surging demand for computing capability [176]. Due to the limited computing capability of MUTs, executing these delay-oriented tasks locally, such as on-camera image/video processing, can inflict unacceptable service delay and be detrimental to the service lifespan of MUTs [177]. Edge computing has been proposed as a de-facto paradigm to support compute-intensive computing services. Within this paradigm, MUTs can offload computing tasks to nearby terrestrial base stations (BSs), which can not only reduce the latency of task execution, but also save the power consumption of MUTs [178]. However, purely relying on offloading to terrestrial BSs is hard to guarantee the performance of computing service robustly. On the one hand, the MUTs are usually power constrained, which cannot support long-distance transmission for task offloading, especially when the BSs are sparsely deployed or unavailable nearby (e.g., automated mining applications) [138]. On the other hand, the physical computing resources on BSs are scarce and somewhat insufficient, but the MUTs' computing tasks arrive dynamically with possible bursty conditions (e.g., intelligent transportation applications), which can result in computing resource shortage and deteriorate delay performance [179] [180].

As a remedy to these limitations, satellites and unmanned aerial vehicles (UAVs) are considered as promising complements to enhance the terrestrial network. For satellites, many research and industrial efforts have been devoted to the commercialization of the low earth orbit (LEO) satellite constellation, such as SpaceX and OneWeb [8], which can provide ubiquitous services with acceptable propagation delay (e.g., about 6.44 ms) [9,181]. For UAVs with flexible deployment and agile management, they have been widely utilized in military and civil applications to provide on-demand communication and computing resources [10]. Besides, the 3rd Generation Partnership Project (3GPP) is also investigating on non-terrestrial networks and specifying novel architectures to complement terrestrial cellular networks [181]. Since satellite, UAV, and BS can complement each other, the integration of them, namely the space-air-ground integrated network (SAGIN), is proposed as a promising network architecture for the NGWN to serve the a large number of MUTs with delay-oriented service requirements [138], [121].

In this chapter, considering the low transmit power and short-distance communication range of MUTs, we propose a delay-orientated task scheduling (DOTS) scheme in SAGIN to process computing tasks in real time. We adopt a UAV (installed with dedicated MUT communication interface such as LoRa and NB-IoT [182], [183]) as the "flying scheduler" to communicate with MUTs and collect their computing tasks. As the UAV can move sufficiently close to MUTs, the distance between MUTs and the UAV can be significantly reduced, which not only saves the MUTs' power consumption and prolongs the service lifespan, but also guarantees the transmission reliability [6]. Then, the UAV makes task scheduling decisions in real time, i.e., processing locally, offloading to a nearby BS or the remote LEO satellite constellation.[1] Particularly, the UAV needs to offload tasks as soon as possible when it serves an excessive number of computing tasks, due to the limited computing capability [27]. In addition, the UAV should make decisions in real time to keep the pace of dynamic link conditions and computing task arrival. Therefore, how to obtain an efficient scheduling policy of processing computing tasks at appropriate SGAIN components is a crucial issue, which is quite challenging due to the following three reasons. First, with a large number of MUTs, task arrivals are dynamic and may be bursty, and even unknown *a priori*, which poses a real-time requirement for the scheduling policy. Second, UAV, BSs, and LEO satellites have differentiated features in terms of communication and computing capability. As a result, the scheduling policy should select appropriate SAGIN components for task processing in accordance with their features. Third, in the scheduling policy, both the current energy consumption and the energy reservation for future arrived tasks should be considered. The UAV needs to comply with the UAV energy capacity by making sequential task scheduling decisions.

[1]Note that the UAV can be installed with two communication interfaces, one for cellular BSs and the other for the LEO satellite constellation in SAGIN [184].

To tackle the above challenges, we formulate the online scheduling problem as a constrained Markov decision process (CMDP) to minimize the time-averaged task processing delay while taking the UAV energy capacity (consumed by communication and computing) into consideration. Inspired by the advantage of reinforcement learning (RL) methods in tackling the uncertainty and dynamics, we design a novel deep risk sensitive RL algorithm to deal with the formulated CMDP problem. The core idea is to define a risk function to capture whether the UAV energy capacity constraint is violated. Thus, satisfying the constraint is transformed into minimizing the risk. Afterward, we replace the typical Q-value function by the sum of two Q-value functions. The former Q-value function evaluates the long-term delay for different state-action pairs, and the latter accounts for the long-term risk. Based on the designed Q-value function, the scheduling policy can be learned by leveraging RL methods. Meanwhile, instead of constructing a space-costly Q-value table caused by the high dimensional state representation, we leverage parameterized deep neural networks (DNNs) to approximate the Q-value function. In addition, we add a filter layer after fully connected layers to exclude unavailable actions at different states. Extensive simulations are conducted, which show that the proposed deep RL-based DOTS scheme can achieve a lower time-average task processing delay while satisfying the UAV energy capacity constraint compared to that of benchmark schemes. The main contributions of this chapter are three-fold:

- We propose a computing task scheduling scheme named DOTS for delay-oriented computing services in SAGIN, where a UAV flies along a trajectory to collect computing tasks and make real-time scheduling decisions.
- We formulate an integer non-linear optimization problem with uncertainty to minimize the time-averaged task processing delay under the UAV energy capacity constraint. As the UAV location and task backlog evolve in an ergodic way, we reformulate the online computing task scheduling problem as a CMDP.
- We design a novel deep risk-sensitive RL algorithm to address the CMDP problem, where a risk function is defined to indicate whether the UAV energy consumption violates the constraint. Besides, we leverage DNNs to implement the proposed deep RL-based algorithm in the DOTS scheme.

The remainder of this chapter is organized as follows. The SAGIN architecture and computing task scheduling models are described in Section 5.2. The problem formulation is provided In Section 5.3. The designed DOTS scheme to make the online scheduling decision is presented in Section 5.4. Section 5.5 presents the simulation results of DOTS, followed by the summary in Section 5.6.

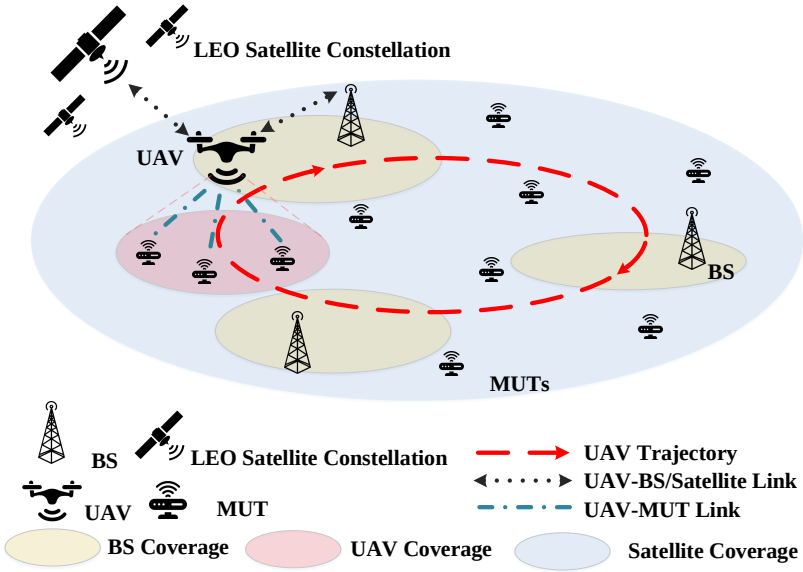


Figure 5.1: The network model.

5.2 System Model

In this section, we first introduce the proposed DOTS scheme in SAGIN architecture, and then describe the computing, communication, and energy consumption models for computing task offloading.

5.2.1 The SAGIN Architecture and the DOTS Scheme

As shown in Fig. 5.1, the UAV flies along a trajectory to collect delay-oriented computing tasks from MUTs.[2] As the rotary-wing UAV can hover in the air, and fly with a low height sufficiently close to MUTs, we adopt the rotary-wing UAV to collect the computing tasks [186]. Taking the computing functionality of the UAV [10], BSs [132], and LEO satellites [138] into account in the SAGIN, the UAV can schedule computing tasks on three different destination network components, i.e., processing tasks on the UAV locally, offloading to the nearby BS, or offloading the LEO satellite constellation. Let indexes $1, 2, \ldots, N$, and 0 denote the LEO satellite constellation and the BSs, respectively. Then,

[2]The UAV trajectory is assumed to be planned in advance since the UAV trajectory design has been well studied in many previous works [185, 186].

the set of the network components that computing tasks can be offloaded to (i.e., N BSs and the LEO satellite constellation) is denoted by $\mathcal{N} = \{0, 1, 2, \ldots, N\}$. Due to the UAV's limited on-board battery capacity, the computing capability at the UAV is limited [10]. The UAV cannot process all computing tasks alone, and thus some computing tasks can be offloaded to BSs or the LEO satellite constellation. BSs and the LEO satellite constellation have different characteristics. The BS has high computing capacity, while its coverage area is limited. The LEO satellite constellation can always cover the area and act as a complementary offloading solution for terrestrial networks, while the propagation delay of the UAV-satellite link cannot be neglected. Therefore, computing tasks should be scheduled appropriately to different destination network components in SAGIN to reduce the service delay.

We adopt the discrete epoch-based system with an equal time duration of τ in each epoch. In epoch t, the location of the deployed UAV is denoted by l_t. As the UAV flies along the trajectory, the set of available offloading destination network components also varies at different locations, which is denoted by $\mathcal{L}_t \subseteq \mathcal{N}$. Supposing that multiple computing tasks can be offloaded from the UAV in each epoch, only one offloading destination (i.e., a BS or the satellite) can be chosen. In summary, the UAV collects and schedules computing computing tasks according to the following steps in each epoch:

1) The UAV collects tasks from MUTs and locally processes their tasks within the computing queue. The collected tasks that have not been processed or offloaded will wait in the computing queue at the UAV.

2) The UAV can offload a certain number of computing tasks from the computing queue to a BS or the satellite. The offloaded tasks that have not been forwarded will wait in the forwarding queue at the UAV.

3) Newly arrived tasks from MUTs are stored in the computing queue at the UAV. Once the computing queue is full, newly arrived tasks will be dropped.

4) The UAV flies to the next location along the predefined trajectory, and continues to collect computing tasks.

As shown in Fig. 5.2, an exemplary work flow of the DOTS scheme in SAGIN is illustrated. In epoch 1, four tasks are collected, one of which is processed locally at the UAV, and three of which are offloaded to BS and moved into the forwarding queue. In epoch 2, the UAV cannot move new tasks into the forwarding queue due to the uncompleted task forwarding. Only one task is processed locally at the UAV, and all tasks in the forwarding queue are transmitted. In epoch 3, two tasks are offloaded to the satellite and moved into the

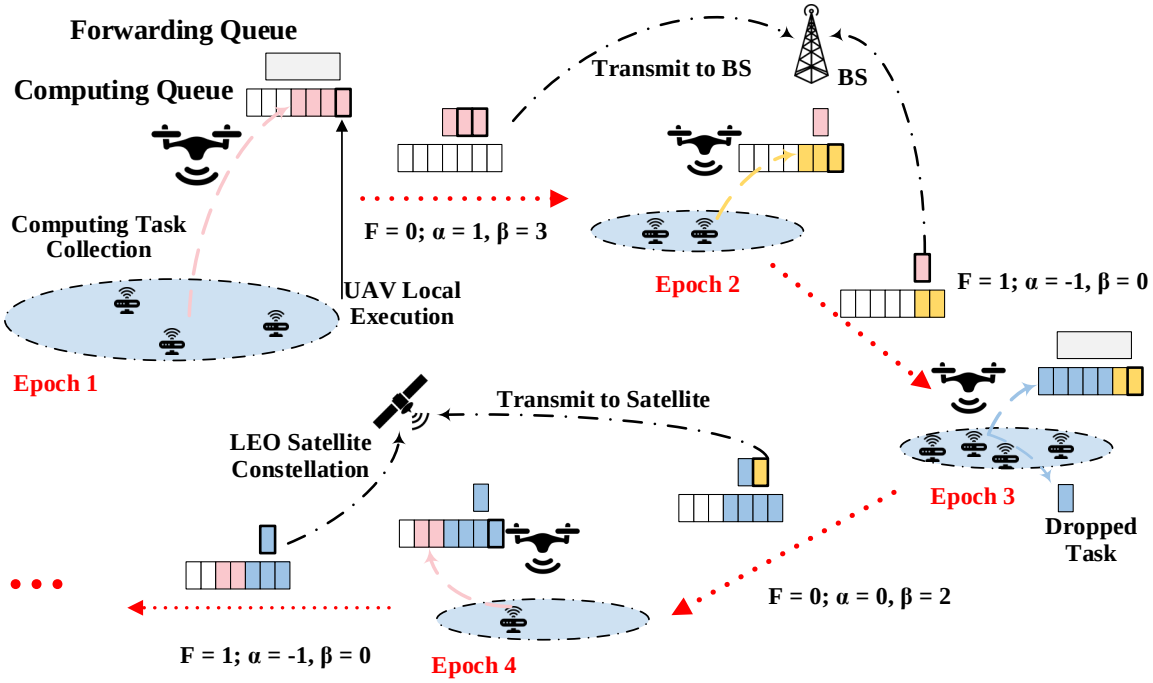


Figure 5.2: The illustration of the DOTS scheme in SAGIN, where different colors of tasks are used to distinguish the collection in different epochs.

forwarding queue. In epoch 4, all tasks can only be executed locally at the UAV. The details of the scheme are introduced in the following subsections.

5.2.2 Computing Model

In general, we adopt a tuple (ϕ, γ) to model a computing task [138]. Here, ϕ represents the input data size (in bits) of a computing task, and γ (in central processing unit (CPU) cycles per bit) indicates the computing workload of the task, i.e., how many CPU cycles are required to process one bit input data.[3] Note that task uploading is the key point of scheduling policy at the UAV in the considered scenario, and the downloading of the computing result can be ignored in this work.[4] For instance, MUTs upload images for analysis

[3]In practice, the computing workload is measured via conducting the same computing task with the same software of experimental platform in multiple times [187].

[4]Generally, the results of computing tasks cannot be immediately fed back to MUTs by the same UAV due to the mobility. In practical system, many UAVs can be deployed along different pre-defined

and download text messages as the output, and the uploaded data size is much larger than that of downloaded data [176]. As the UAV can offload tasks to either the nearby BS or the remote LEO satellite, or execute tasks locally, the corresponding computing delay is analyzed in the next.

A. Task Offloading

Denote the task offloading decision by α_t in epoch t, i.e., the offloading destination network components in epoch t. The UAV offloads the tasks to the satellite when $\alpha_t = 0$, or offloads tasks to BS n when $\alpha_t = n, \forall n \neq 0$. Denote by $\beta_t \leq \beta^{\max}, \beta_t \in \mathbb{N}$ the number of offloaded tasks in epoch t, where $\beta^{\max}$ is the maximal number of tasks that can be forwarded by the UAV in each epoch. Meanwhile, due to the occupation of the communication interface, we assume that new tasks cannot be forwarded if the offloading process of the last task is not completed. Let binary variable F_t indicate whether collected computing tasks on the UAV can be offloaded or not. Fig. 5.2 illustrates an example of the task forwarding. When $F_t = 0$, the UAV can offload tasks in epoch t since the channel is not occupied (i.e., $\alpha_t \in \mathcal{L}_t, \beta_t \leq \beta^{\max}$). $F_t = 1$ represents that the UAV cannot offload new tasks since a certain number of tasks are waiting to be transmitted in the forwarding queue (i.e., $\alpha_t = -1, \beta_t = 0$).

Denote by the computing capabilities (in CPU cycles per second) of BS n and the satellite by $f_n, n \neq 0$ and f_0, respectively. The computing delay of all β_t tasks at offloading destination network component n is given by:

$$d_1(\alpha_t, \beta_t) = \frac{\beta_t \phi \gamma}{f_{\alpha_t}}, \ \alpha_t \in \mathcal{L}_t, \tag{5.1}$$

where f_{α_t} represents the computing capability of offloading destination network component α_t.

B. Local Processing

Since the computing capability of the UAV is limited, the collected tasks may not be processed locally or offloaded completely at the UAV within an epoch. We assume that the remaining tasks wait to be scheduled in the computing queue at the UAV. As a result, the delay of processing task locally at UAV includes two parts, i.e., local computing delay

trajectories, and the result of computing tasks can be relayed via UAV-UAV links [188]. Therefore, MUTs can receive results as long as they are covered by UAVs.

and queuing delay. To model the computing queue, we first denote the unaccomplished task backlog at the beginning of epoch t by $H_t \in [0, \rho]$, where ρ is the maximum length of the computing queue. Then, given unaccomplished task backlog H_t and the number of offloaded tasks β_t, the number of queuing tasks O_t in epoch t within the computing queue is given by:

$$O_t = \max\left\{H_t - \lfloor\frac{f_{\mathrm{U}}\tau}{\phi\gamma}\rfloor - \beta_t, 0\right\}, \tag{5.2}$$

where f_{U} is the computing capability (in CPU cycles per second) of the UAV, and $\lfloor f_{\mathrm{U}}\tau/\phi\gamma\rfloor$ is the greatest integer less than the number of tasks executed by the UAV in epoch t. Given the number of newly collected tasks M_t from the MUTs, the unaccomplished task backlog H_{t+1} can be updated at the end of epoch t as follows:

$$H_{t+1} = \min\left\{O_t + M_t,\ \rho\right\}, \tag{5.3}$$

where $\min\{\cdot\}$ is the function to return the smallest value. Then, the delay of local task execution at the UAV can be calculated as the following equation:

$$d_2(\alpha_t, \beta_t) = \frac{\min\left\{\lfloor\frac{f_{\mathrm{U}}\tau}{\phi\gamma}\rfloor, H_t\right\}\phi\gamma}{f_{\mathrm{U}}} + O_t\tau, \tag{5.4}$$

where $\min\{\lfloor\frac{f_{\mathrm{U}}\tau}{\phi\gamma}\rfloor, H_t\}\phi\gamma/f_{\mathrm{U}}$ is the local computing delay within each epoch, and $O_t\tau$ is the queuing delay of all O_t tasks waiting in the computing queue.

5.2.3 Communication Model

We suppose two communication interfaces are equipped in this work [189], i.e., one for LEO satellites, and the other for BSs. Each of them uses different spectrum bands, which leads to no interference between BSs and the satellite [190]. In the following, the transmission delay of offloading tasks to the satellite and the BSs are discussed in detail.[5]

A. Offload to Satellite

Currently, the wireless communications between an LEO satellite and terrestrial users are enabled by Ka or Ku frequency band, the channel condition of which is mainly impacted

[5]Considering the flexibility of the UAV, it can fly sufficiently close to the MUTs such that the condition of UAV-MUT links is Line-of-sight (LoS). Since the LoS communications can achieve high data rate [10, 191], the transmission delay of UAV-MUT links is neglected.

by the communication distance and the rain attenuation (rain fading) [190]. Supposing the meteorological environment remains stationary during the computing task collection, the channel gain of the UAV-satellite link is mainly determined by the distance between the UAV and the satellite. Generally, the moving distance of the UAV (e.g., the maximum flight distance of the UAV is about 2 km) is much shorter than the altitude of the satellite (e.g., the LEO satellites are with an altitude of 200 km to 2,000 km), which results in the negligible variation of the distance between the UAV and the satellite. Therefore, the channel gain h of the UAV-satellite link can be assumed to be the same with the location of UAV. Then, the data rate of the UAV-satellite link in epoch t denoted by r_{α_t} is given by:

$$r_{\alpha_t} = W_{\mathrm{S}} \log_2 \left(1 + \frac{P_{\mathrm{S}} \cdot |h|^2}{\sigma_{\mathrm{S}}^2}\right), \quad \alpha_t = 0, \tag{5.5}$$

where W_{S} is the channel bandwidth of the UAV-satellite link, P_{S} is the transmission power of UAV-satellite link, and σ_{S}^2 indicates the power of noise. Due to the long distance between the LEO satellite and the UAV, the propagation delay cannot be ignored, which is denoted by d_{S}. Thus, given offloading decision α_t and offloaded task number β_t, transmission delay of offloading tasks to the satellite can be calculated as following equation:

$$d_3(\alpha_t, \beta_t) = \frac{\beta_t \phi}{r_{\alpha_t}} + d_{\mathrm{S}}, \quad \alpha_t = 0. \tag{5.6}$$

B. Offload to BS

Denote by $K_{\alpha_t}, \alpha_t \neq 0$ the duration that UAV will stay in the coverage of BS n since epoch t. As the UAV needs to guarantee that the forwarding process of all β_t tasks can be completed before the UAV flies out of the BS's coverage, the number of forwarded tasks β_t satisfies the following constraint:

$$\arg\min_k \left(\sum_{i=t}^{t+k} r_{\alpha_i} \tau \geq \beta_t \phi\right) \leq K_{\alpha_t}, \quad \alpha_t \in \mathcal{L}_t, \alpha_t \neq 0, \tag{5.7}$$

which means that the transmission time of β_t tasks is shorter than the duration that the UAV stays in the BS's coverage. Notice that duration K_{α_t} can be known *a priori* for the deployed UAV as it depends on the BSs' location and the UAV trajectory [138].

Given the pathloss of the UAV-BS link PL, data rate r_{α_t} of the UAV-BS n link can be calculated as

$$r_{\alpha_t} = W_{\mathrm{B}} \log_2 \left(1 + \frac{P_{\mathrm{B}} \cdot 10^{\frac{PL}{10}}}{\sigma_{\mathrm{B}}^2}\right), \quad \alpha_t \neq 0, \tag{5.8}$$

where W_{B} indicates the channel bandwidth of UAV-BS link, P_{B} represents the transmission power of from the UAV to a BS, and σ_{B}^2 indicates the power of the background noise. Denote by d_3 the transmission delay of offloading tasks to the BS, which is given by:

$$d_3(\alpha_t, \beta_t) = \frac{\beta_t \phi}{r_{\alpha_t}}, \quad \alpha_t \in \mathcal{L}_t, \alpha_t \neq 0, \tag{5.9}$$

where α_t and β_t represent offloading destination and offloaded task number, respectively.

5.2.4 Energy Consumption Model

Generally, UAV energy consumption includes propulsion energy, communication-related energy, and computing-related energy. Since UAV propulsion energy is mainly depends on different trajectories and aircraft parameters, it can be considered as a constant in our work [186]. Thus, we aim to guarantee the remaining components of energy consumption, i.e., computing-related and communication-related energy, do not exceed the UAV energy capacity. Denote by e_{o} the communication-related energy caused by the transmission of tasks, which can be calculated as follows:

$$e_{\mathrm{o}}(\alpha_t, \beta_t) = \begin{cases} P_{\mathrm{S}} d_4(\alpha_t, \beta_t), & \alpha_t = 0 \\ P_{\mathrm{B}} d_4(\alpha_t, \beta_t), & \alpha_t \in \mathcal{L}_t, \alpha_t \neq 0. \end{cases} \tag{5.10}$$

Meanwhile, processing computing task on the UAV also consumes energy, which depends on the computing workload of the computing task and the computing capability of the UAV. Denoted by e_{l} the computing-related energy, which can be expressed as follows:

$$e_{\mathrm{l}}(\alpha_t, \beta_t) = \min\left\{H_t \phi \gamma, f_{\mathrm{U}} \tau\right\} \cdot \xi \left(f_{\mathrm{U}}\right)^2, \tag{5.11}$$

where ξ indicates the effective switched capacitance determined by the chip architecture [138]. Denote by E_t the cumulative energy consumption in epoch t. Given the communication-related and computing-related energy consumption, the cumulative energy consumption can be calculated as the following equation:

$$E_t = E_{t-1} + e_{\mathrm{o}}(\alpha_t, \beta_t) + e_{\mathrm{l}}(\alpha_t, \beta_t). \tag{5.12}$$

The cumulative energy consumption can be leveraged to evaluate whether the UAV satisfies the energy capacity or not.

5.3 Problem Formulation

We aim to minimize the long-term delay of all computing tasks while satisfying the UAV energy consumption constraint. The total delay of all tasks in epoch t can be calculated as follows:

$$D_t = \begin{cases} \dfrac{\beta_t \phi \gamma}{f_{\alpha_t}} + \dfrac{\min\left\{\lfloor \frac{f_{\mathrm{U}}\tau}{\phi\gamma} \rfloor, H_t\right\}\phi\gamma}{f_{\mathrm{U}}} + O_t\tau + \dfrac{\beta_t\phi}{r_{\alpha_t}} + d_{\mathrm{S}}, & \alpha_t = 0 \\ \dfrac{\beta_t \phi \gamma}{f_{\alpha_t}} + \dfrac{\min\left\{\lfloor \frac{f_{\mathrm{U}}\tau}{\phi\gamma} \rfloor, H_t\right\}\phi\gamma}{f_{\mathrm{U}}} + O_t\tau + \dfrac{\beta_t\phi}{r_{\alpha_t}}, & \alpha_t \neq 0, \end{cases} \tag{5.13}$$

where both the computing delay and the transmission delay are included. Let $\boldsymbol{\alpha} = \{\alpha_t, \forall t\}$ and $\boldsymbol{\beta} = \{\beta_t, \forall t\}$ denote the set of task offloading decisions and the number of offloaded tasks in each epoch, respectively. As link availability and task arrival are highly dynamic, we concentrate on minimizing the time-averaged delay of all tasks. The delay minimization problem can be formulated as follows:

$$\text{P1:} \min_{\{\boldsymbol{\alpha},\boldsymbol{\beta}\}} \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} D_t \tag{5.14a}$$

$$\text{s.t. } (5.7), \tag{5.14b}$$

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \left[e_{\mathrm{o}}(\alpha_t, \beta_t) + e_{\mathrm{l}}(\alpha_t, \beta_t)\right] \leq \varepsilon, \tag{5.14c}$$

$$\alpha_t \leq N, \alpha_t \in \boldsymbol{\mathcal{L}}_t, \tag{5.14d}$$

$$\beta_t \leq \beta_{\max}, \beta_t \in \mathbb{N}, \tag{5.14e}$$

where (5.14a) is the objective that minimizes the time-average delay of all collected tasks over T epochs, and (5.14b) limits the offloading destinations and the number of offloading tasks. (5.14c) restricts the time-averaged energy consumption of the UAV where ε is the UAV energy capacity. (5.14d) and (5.14e) constrain task offloading decisions and the numbers of offloaded tasks, respectively. Problem P1 is an integer nonlinear optimization problem with unknown number of newly collected tasks in each epoch, which is difficult to solve. Considering the UAV location and the backlog of unaccomplished task in the computing queue evolve in an ergodic way, we adopt the stationary decision to address this problem, which is time-invariant and only depends on the current system status. Therefore, the problem can be reformulated as a Markov decision process (MDP) for a stationary decision which is the optimal in the ergodic system [39].

We define a tuple $\mathcal{M} := \langle \boldsymbol{S}, \boldsymbol{A}, \boldsymbol{P}, \boldsymbol{C}, \boldsymbol{\Pi} \rangle$ to model the MDP, which is a sequential decision-making process. Specifically, $\boldsymbol{S}$ represents the set of states. $\boldsymbol{A}$ is the set of actions. $\boldsymbol{P} := \boldsymbol{S} \times \boldsymbol{A} \times \boldsymbol{S} \to \mathbb{R}$ is set of state transition probabilities. $\boldsymbol{C} := \boldsymbol{S} \times \boldsymbol{A} \to \mathbb{R}$ indicates the cost function. $\boldsymbol{\Pi}$ is the policy that is a decision rule mapping from a state $\boldsymbol{s} \in \boldsymbol{S}$ to an action $\boldsymbol{a} \in \boldsymbol{A}$. Meanwhile, $C(\boldsymbol{s}, \boldsymbol{a})$ is defined as the cost when the system stays in state $\boldsymbol{s}$ with adopting action $\boldsymbol{a}$. For the aforementioned problem, the states, actions, and cost in an MDP model are formulated as follows:

- **State** – In epoch t, a tuple denoted by $\boldsymbol{s}_t = (l_t, F_t, H_t, E_t), \boldsymbol{s}_t \in \boldsymbol{S}$ is used to describe the system state, where l_t, F_t, H_t, E_t represent UAV location, the number of offloaded tasks in the forwarding queue, the unaccomplished task backlog in the computing queue and the cumulative energy consumption, respectively;

- **Action** – An action is made based on the current state, and the decision is denoted by a tuple $\boldsymbol{a}_t = (\alpha_t, \beta_t), \boldsymbol{a}_t \in \boldsymbol{A}$ in epoch t, where α_t is used to indicate offloading destination, and β_t denotes the number of the offloaded tasks;

- **Cost** – Considering an intuitive policy that the UAV does not offload tasks and keep the queue full, and almost all newly arrived tasks will be dropped. In such case, although the cost (delay) can be minimized, an excessive number of dropped tasks lead to practical infeasibility. To minimize the cost while avoiding the excessive task dropping, a penalty Λ_t is introduced as follows:

$$\Lambda_t = \lambda \max\left(M_t + O_t - \rho, 0\right), \tag{5.15}$$

  where $\max\left(M_t + O_t - \rho, 0\right)$ represents the excessive number of the newly collected tasks will be dropped, and λ is a constant penalty weight. With the objective of minimizing long-term delay of all computing tasks, the cost function can be defined as $C(\boldsymbol{s}_t, \boldsymbol{a}_t) = D_t + \Lambda_t$, where Λ_t is the penalty to avoid excessive drop of computing tasks;

- **Policy** – Denote by $\boldsymbol{\pi}$ the stationary policy, which means that state $\boldsymbol{s}_t$ is assigned with action $\boldsymbol{a}_t$ and this action will be chosen whenever the system stays in this state.

Therefore, MDP based delay-oriented tasks scheduling problem can be formulated as

follows:

$$\text{P2:} \min_{\boldsymbol{\pi}} \lim_{T\to\infty} \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} C_t(\boldsymbol{s}_t, \boldsymbol{a}_t)\middle|\boldsymbol{\pi}\right] \tag{5.16a}$$

$$\text{s.t. } (5.14\text{b}), (5.14\text{d}), (5.14\text{e}) \tag{5.16b}$$

$$\lim_{T\to\infty} \mathbb{E}\left[\frac{E_t}{T}\middle|\boldsymbol{\pi}\right] \leqslant \varepsilon, \tag{5.16c}$$

where (5.16a) represents the expected average cost and expected energy consumption. Problem P1 is transformed into problem P2 to find the optimal policy $\boldsymbol{\pi}$ with respect to a cost $C_t(\boldsymbol{s}_t, \boldsymbol{a}_t)$ for choosing action $\boldsymbol{a}$ at state $\boldsymbol{s}$, which minimizes the expected average cost. Above problem P2 is a CMDP problem, which is a typical MDP problem with additional constraints. Solving such a CMDP problem with uncertainty is challenging. On the one hand, typical MDP problems are well-investigated, which can be solved by iterative methods by finding a deterministic policy, such as the policy iteration and the value iteration. However, these methods for MDP cannot cope with the CMDP problem since constraints and the objective cannot be optimized simultaneously. On the other hand, although CMDP problems with the known transition probability can be solved simply via a linear programming method, the linear programming method cannot address the CMDP problem with uncertainty, since transition probability $P(H_{t+1}|H_t)$ is unknown due to the uncertainty of the arrived task number.

5.4 Deep Risk-sensitive RL Algorithm

In this section, we first introduce the preliminary of RL methods. Afterward, by tailoring the typical RL methods, we propose the deep risk-sensitive RL algorithm to address problem P2. Finally, we present the details of DNN-based implementation of the proposed algorithm.

5.4.1 Preliminary

In problem P2, since the objective is to find policy $\boldsymbol{\pi}$ that chooses appropriate actions at different states to minimize the long-term cost (delay), which consists of the immediate cost (generated in the current epoch) and the future cost (generated in the following epochs) for each state-action pair. Because the future cost is related to both the current scheduling action and the actions in the following epochs, it is challenging to model the relationship

between the current action and the future cost, particularly in the case with unknown state transition probability. Therefore, the discounted cost model is designed to balance the immediate cost and the future cost for each state-action pair, which is calculated as $\sum_{t=0}^{\infty} \varsigma^t C(\boldsymbol{s}_t, \boldsymbol{a}_t)$ [39]. Note that the discount factor, denoted by $\varsigma \in [0, 1]$ is to prevent the long-term cost from going to negative infinity.

Then, to measure the long-term cost starting from state $\boldsymbol{s}$ under policy $\boldsymbol{\pi}$, a *value function* is defined to determine the value of expected long-term discounted cost when the system is at state $\boldsymbol{s}$. Denote by $V_{\boldsymbol{\pi}}(\boldsymbol{s})$ the value function, which is given by:

$$V_{\boldsymbol{\pi}}(\boldsymbol{s}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \varsigma^t C(\boldsymbol{s}_t, \boldsymbol{a}_t) | \boldsymbol{\pi}, \boldsymbol{s}_0 = \boldsymbol{s}\right]. \tag{5.17}$$

Based on Eq. (5.17), a *Q-value function* is defined to further evaluate state-action pairs, which is denoted by $Q_{\boldsymbol{\pi}}(\boldsymbol{s}_t, \boldsymbol{a}_t)$. Such the Q-value function measures the expected long-term discounted cost that the system may get from being at state $\boldsymbol{s}$, following policy $\boldsymbol{\pi}$ and choosing action $\boldsymbol{a}$, which is given by:

$$Q_{\boldsymbol{\pi}}(\boldsymbol{s}_t, \boldsymbol{a}_t) = C(\boldsymbol{s}_t, \boldsymbol{a}_t) + \sum_{\boldsymbol{s}_{t+1}} \varsigma P(\boldsymbol{s}_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_t) V_{\boldsymbol{\pi}}(\boldsymbol{s}_{t+1}). \tag{5.18}$$

With the objective of the cost minimization, we choose the minimum Q-value as the optimal Q-value, which is denoted by $Q^*_{\boldsymbol{\pi}}(\boldsymbol{s}, \boldsymbol{a}) = \min_{\boldsymbol{\pi}} Q_{\boldsymbol{\pi}}(\boldsymbol{s}, \boldsymbol{a})$.

Generally, due to the unknown state transition probability, the basic idea behind model-free RL methods is temporal difference (TD) learning, i.e., the current approximation of Q-value function (which might not be accurate) can be leveraged to update the estimated value for the following states [40]. The mechanism of the RL methods allows the UAV to iteratively update and approximate the Q-value function and then choose actions based on the approximated Q-value function. Therefore, RL methods can learn online and interact with the environment simultaneously, which is suitable for the considered case with unknown task arrival. Denote by $\boldsymbol{a}^* = \arg\min_{\boldsymbol{a}_t \in \boldsymbol{A}} Q_{\boldsymbol{\pi}}(\boldsymbol{s}_t, \boldsymbol{a}_t)$ the greedy action which acquires the optimal Q-value. The Q-value can be updated based on the following TD backup equation:

$$Q'_{\boldsymbol{\pi}}(\boldsymbol{s}_t, \boldsymbol{a}_t) = Q_{\boldsymbol{\pi}}(\boldsymbol{s}_t, \boldsymbol{a}_t) + \eta \left[C(\boldsymbol{s}_t, \boldsymbol{a}_t) + \varsigma Q_{\boldsymbol{\pi}}(\boldsymbol{s}_{t+1}, \boldsymbol{a}^*)\right], \tag{5.19}$$

where the learning rate denoted by η is to determine how much newly acquired cost should be accepted to adjust the evaluation of Q-value function. Note that $0 < \eta < 1$ is a constant value in the learning process. The convergence of such RL methods based on Q-value iteration has been proved, i.e., the Q-values converge to the optimal Q-values [39].

Conventional RL methods update Q-values based on a Q-value table, i.e., all state-action pairs are listed in a table, and each pair is updated iteratively and independently. However, tabular methods require a large memory to store all state-action pairs, which increases exponentially with the state and action space [40]. Due to the curse of dimensionality in the considered scenario (e.g., a large number of UAV locations, the large size of the computing queue backlog), conventional tabular RL methods cannot be applied practically. To deal with the aforementioned problem, instead of tabular methods, DNN is adopted to approximate Q-value function. Let ϑ be the parameters of DNN, which includes neural network weights and biases. Denote by $Q_{\boldsymbol{\pi}}(\boldsymbol{s}_t, \boldsymbol{a}_t; \vartheta)$ the DNN-based Q-value function, which is updated by minimizing the following loss function:

$$L(\vartheta) = |C(\boldsymbol{s}_t, \boldsymbol{a}_t) + \varsigma Q_{\boldsymbol{\pi}}(\boldsymbol{s}_{t+1}, \boldsymbol{a}^*; \vartheta) - Q_{\boldsymbol{\pi}}(\boldsymbol{s}_t, \boldsymbol{a}_t; \vartheta)|^2, \tag{5.20}$$

where $L(\vartheta)$ is named as the TD error. Similar to tabular RL methods, DNN-based RL methods can also allow the UAV to iteratively update the DNN-based Q-value function and then choose actions based on the approximated DNN-based Q-value function in an online manner.

5.4.2 Deep Risk-Sensitive RL Algorithm

In problem P2, apart from the objective of cost minimization, there is an extra constraint of energy capacity that needs to be satisfied. However, since the energy consumption is not a component of the cost function, conventional RL methods mentioned above cannot satisfy the constraint in problem P2. Therefore, we propose a deep risk-sensitive RL algorithm to deal with the CMDP problem. Specifically, in addition to the cost function, an extra risk function is defined to capture whether the UAV energy consumption in the current epoch violates the UAV energy capacity constraint, and then a corresponding Q-value function is defined to evaluate the value of risk. Therefore, the algorithm has two Q-value functions, i.e., one Q-value function to evaluate the cost and the other Q-value function to evaluate the risk. Afterward, the proposed deep risk-sensitive RL algorithm updates two different Q-value functions independently and chooses the action based on the sum of two Q-value functions.

Define the set of error states as $\Phi \subseteq \boldsymbol{S}$. An error state $\boldsymbol{s}_t \in \Phi$ represents the energy consumption of the UAV in epoch t exceeds the UAV energy capacity, i.e., $E_t > \varepsilon t$. Then, to measure how much consumed energy that exceeds the UAV energy capacity when the system is at state $\boldsymbol{s}$ choosing action $\boldsymbol{a}$, we denote the risk function by $R(\boldsymbol{s}_t, \boldsymbol{a}_t)$, which is

given by:

$$R(\boldsymbol{s}_t, \boldsymbol{a}_t) = \begin{cases} |E_t - \varepsilon t|, & \text{if } \boldsymbol{s}_t \in \Phi \\ 0, & \text{otherwise.} \end{cases} \tag{5.21}$$

The value of risk that are at a non-error state is zero, and the value of risk at an error state is equivalent to the exceeding part of the energy consumption. Consequently, if the current state of the system is an error state, the following states will also be error states with the increased value of risk. To satisfy the UAV energy capacity constraint in problem P2, the value of risk at each state should be zero. Thus, we transform the goal that keeps the energy consumption below the energy capacity into the goal that minimize the risk. Note that the risk minimization is not equivalent to energy consumption minimization since the energy consumption minimization is not the objective of this problem.

Similar to the aforementioned cost minimization, the risk minimization can be achieved by using another Q-value function, which is operated separately. Based on the discounted risk, we define the expected long-term discounted risk as the value function $\bar{V}_{\boldsymbol{\pi}}(\boldsymbol{s})$ of state $\boldsymbol{s}$ under policy $\boldsymbol{\pi}$, which is given by:

$$\bar{V}_{\boldsymbol{\pi}}(\boldsymbol{s}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \bar{\varsigma}^t R(\boldsymbol{s}_t, \boldsymbol{a}_t) | \boldsymbol{\pi}, \boldsymbol{s}_0 = \boldsymbol{s}\right], \tag{5.22}$$

where $\bar{\varsigma}$ is the discount factor for the discounted risk. Then, to measure the expected long-term discounted risk that the UAV may get from being at state $\boldsymbol{s}$, following policy $\boldsymbol{\pi}$ and choosing action $\boldsymbol{a}$, the corresponding Q-value function, $\bar{Q}_{\boldsymbol{\pi}}(\boldsymbol{s}_t, \boldsymbol{a}_t)$, is defined as follows:

$$\bar{Q}_{\boldsymbol{\pi}}(\boldsymbol{s}_t, \boldsymbol{a}_t) = R(\boldsymbol{s}_t, \boldsymbol{a}_t) + \sum_{\boldsymbol{s}_{t+1}} \bar{\varsigma} P(\boldsymbol{s}_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_t) \bar{V}_{\boldsymbol{\pi}}(\boldsymbol{s}_{t+1}). \tag{5.23}$$

Based on the TD learning, the Q-value function of risk can also be estimated based on the following equation:

$$\bar{Q}'_{\boldsymbol{\pi}}(\boldsymbol{s}_t, \boldsymbol{a}_t) = \bar{Q}_{\boldsymbol{\pi}}(\boldsymbol{s}_t, \boldsymbol{a}_t) + \bar{\eta}\left[R(\boldsymbol{s}_t, \boldsymbol{a}_t) + \bar{\varsigma}\bar{Q}_{\boldsymbol{\pi}}(\boldsymbol{s}_{t+1}, \bar{\boldsymbol{a}}^*)\right], \tag{5.24}$$

where greedy action $\bar{\boldsymbol{a}}^* = \arg\min_{\boldsymbol{a}_t \in \boldsymbol{A}} \bar{Q}_{\boldsymbol{\pi}}(\boldsymbol{s}_t, \boldsymbol{a}_t)$ is adopted to acquire the optimal Q-value, and $\bar{\eta}$ is the learning rate for the risk minimization. As $\bar{\boldsymbol{a}}^*$ and $\boldsymbol{a}^*$ are two different greedy actions based on different goals, i.e., cost minimization and risk minimization, the chosen actions may not be the same at each state. However, only one action can be selected when each state is reached. Thus, we need to design a new Q-value function to combine two goals. which is given by:

$$Q^{\delta}_{\boldsymbol{\pi}}(\boldsymbol{s}_t, \boldsymbol{a}_t) = Q_{\boldsymbol{\pi}}(\boldsymbol{s}_t, \boldsymbol{a}_t) + \delta \bar{Q}_{\boldsymbol{\pi}}(\boldsymbol{s}_t, \boldsymbol{a}_t), \tag{5.25}$$

where δ is a weight parameter to balance two different goals. If δ is fixed, Q^{δ} forms a standard Q-value function of state-action pair with respect to the new reward $C + \delta R$, which is same as the Q-value function in typical RL methods [192] [193]. Specifically, when $\delta = 0$, $Q^{\delta} = Q$, the minimization of the weighted sum of the cost and the risk leads to the optimal policy for cost minimization, which is same as the cost minimization without constraints. When δ tends to infinity, the minimization of the weighted sum of the cost and the risk leads to the optimal policy for the risk minimization. As the adaption of δ provides a method to find the space of feasible polices, δ can be adjusted to produce the optimal policy to minimize the cost while satisfying the constraint. Therefore, there exists the optimal deterministic policy for the designed new Q-value function, and the convergence of proposed deep risk-sensitive RL algorithm can be guaranteed if discount factors ς and $\bar{\varsigma}$ are equivalent [40].

Due to the curse of dimensionality, we adopt DNN to approximate the Q-value function of risk as the approximation of the DNN-based Q-value function of cost. Denote by $\bar{Q}_{\boldsymbol{\pi}}(\boldsymbol{s}_t, \boldsymbol{a}_t)$ the DNN-based Q-value function of risk, where $\bar{\vartheta}$ is the parameter of the corresponding neural network. The update of DNN-based Q-value function of risk is the same as that of Q-value function of cost in Eq. (5.20). As shown in Algorithm 1, we propose a two-cycle algorithm to minimize the cost while minimizing the risk, i.e., learn the appropriate parameters of DNNs in the inner cycle, and search the appropriate weight parameter to balance two goals in the outer cycle. The former is shown from line 4 to line 20, and each inner cycle is named as an *iteration*. In one iteration, the DNN parameters of $Q_{\boldsymbol{\pi}}(\boldsymbol{s}_t, \boldsymbol{a}_t; \vartheta)$ and $\bar{Q}_{\boldsymbol{\pi}}(\boldsymbol{s}_t, \boldsymbol{a}_t; \bar{\vartheta})$ are updated separately and iteratively. The searching in the outer cycle is shown from line 3 to line 21. Each outer cycle is named as an *episode*. In each outer cycle, the optimal weight parameter δ is updated according to the energy consumption, which is shown from line 16 to line 20. Based on whether the energy consumption in the current episode satisfies the constraint, weight parameter δ is increased or decreased with a fixed step size denoted by Δ. The partial detail of Algorithm 1 is introduced in the next subsection.

5.4.3 DNN-based Implementation

Instead of constructing space-costly Q-tables in conventional RL methods, we implement the proposed algorithm by approximating the Q-value function via DNNs. However, directly replacing the Q-table by a DNN model meets several challenges, e.g., unavailable actions at each state cannot be deleted simply by DNNs due to the "black-box" characteristic of DNN. Therefore, we should design the DNN model to fit the proposed algorithm, the details of which are introduced as follows. As shown in Fig. 5.3, four significant modules

Algorithm 6: Deep Risk-Sensitive RL Algorithm

Initialize: ε, replay memory D; ϑ, ϑ', $\bar{\vartheta}$, $\bar{\vartheta}'$; state $\boldsymbol{s}_0$; step size Δ; δ;
for $k = 1, 2, 3, \cdots, K$ **do**
  for $t = 1, 2, 3, \cdots, T$ **do**
    Choose $\boldsymbol{a}_t$: select a random action with probability ϵ, or select $\arg\min_{\boldsymbol{a}} \left[Q(\boldsymbol{s}_t, \boldsymbol{a}; \vartheta) + \delta \bar{Q}(\boldsymbol{s}_t, \boldsymbol{a}; \bar{\vartheta})\right]$ with probability $1 - \epsilon$;
    Perform action $\boldsymbol{a}_t$ and observe cost C_t, risk R_t and next state $\boldsymbol{s}_{t+1}$;
    Store transition $(\boldsymbol{s}_t, \boldsymbol{a}_t, C_t, R_t, \boldsymbol{s}_{t+1})$ in D;
    Sample random mini-batch of transitions $(\boldsymbol{s}_j, \boldsymbol{a}_j, C_j, R_j, \boldsymbol{s}_{j+1})$ from D;
    Set $y_j = C_j + \varsigma \min_{\boldsymbol{a}'}(Q'(\boldsymbol{s}_{j+1}, \boldsymbol{a}'; \vartheta'))$;
    Set $\bar{y}_j = R_j + \bar{\varsigma} \min_{\boldsymbol{a}'}(\bar{Q}'(\boldsymbol{s}_{j+1}, \boldsymbol{a}'; \bar{\vartheta}'))$;
    Perform a gradient descent step on $\mathbb{E}_{(\boldsymbol{s}_j, \boldsymbol{a}_j, C_j, R_j, \boldsymbol{s}_{j+1}) \sim U(D)}[(y_j - Q(\boldsymbol{s}_j, \boldsymbol{a}_j; \vartheta))^2]$ with respect to ϑ;
    Perform a gradient descent step on $\mathbb{E}_{(\boldsymbol{s}_j, \boldsymbol{a}_j, C_j, R_j, \boldsymbol{s}_{j+1}) \sim U(D)}\left[(\bar{y}_j - \bar{Q}(\boldsymbol{s}_j, \boldsymbol{a}_j; \bar{\vartheta}))^2\right]$ with respect to $\bar{\vartheta}$;
    Set $\vartheta' = \vartheta$, and $\bar{\vartheta}' = \bar{\vartheta}$;
  end
  if $\frac{E_T}{T} > \varepsilon$ **then**
    $\delta \leftarrow \delta + \Delta$;
  else
    $\delta \leftarrow \delta - \Delta$;
  end
end
Output: DNN models with parameters ϑ and $\bar{\vartheta}$, and weight parameter δ

are introduced, i.e., DNN replacement, filter layer design, experience replay, and ϵ-greedy selection.

A. DNN Replacement

For a more stable training, we adopt two DNNs to estimate a Q-value function, i.e., one for a target network and the other for a prediction network. The target network has the same DNN architecture as the prediction network but with frozen parameters. For every certain number of iterations, the parameters from the prediction network are copied to the target network, and this procedure is called DNN replacement. Since the TD error

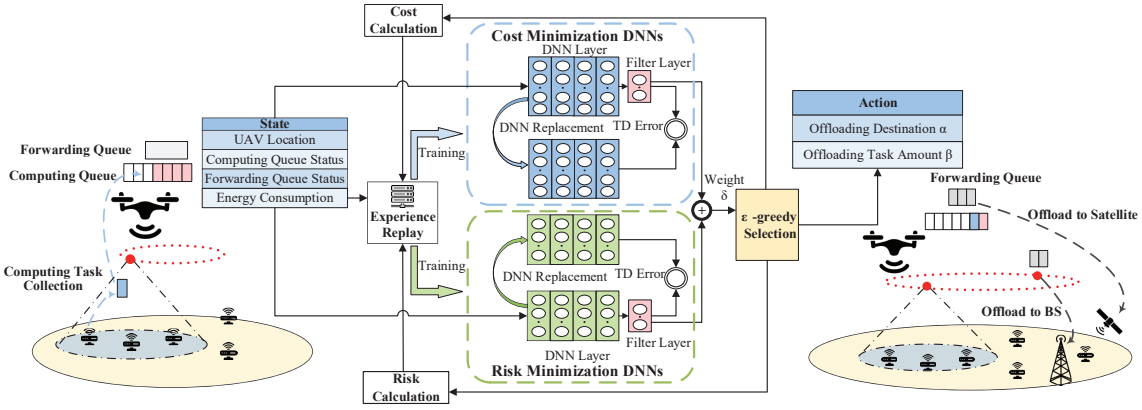


Figure 5.3: An overview of the deep RL-based DOTS scheme.

is used as the loss function in DNN backpropagation to approximate Q-value function by DNNs, the backpropagation requires the output gradient of DNN with respect to weights for input epoch t, and this gradient needs to be saved until we have the new TD error at epoch $t+1$. Thus, there always exists a predicted value for epoch $t+1$ when we compute gradient at epoch t. If we use the same DNN to calculate the predicted value (e.g., $Q_{\boldsymbol{\pi}}(\boldsymbol{s}_t, \boldsymbol{a}_t; \vartheta)$) and the target value (e.g., $C(\boldsymbol{s}_t, \boldsymbol{a}_t) + \varsigma Q_{\boldsymbol{\pi}}(\boldsymbol{s}_{t+1}, \boldsymbol{a}^*; \vartheta)$), the DNN can become destabilized in the feedback loops between the target value and the predicted value [40]. Considering cost minimization and risk minimization are independent, we leverage two DNNs to approximate Q-value function of cost, and another two DNNs to estimate Q-value function of risk, which are shown in Fig. 5.3.

B. Filter Layer Design

We adopt a filter layer to exclude the outputs of unavailable actions. In the considered problem, the available action set at different states is different. For example, the UAV can only offload tasks to the nearby BSs, and thus available action set $\mathcal{L}_t$ changes with the location of UAV l_t. However, since the output size of a fully connected layer in DNN is fixed, the number of Q-value outputs from DNN cannot be changed according to the various number of actions in the available set. As a result, the unavailable actions are included in the DNN-based approximation of Q-value, which is incorrect. Furthermore, constraint (5.7) needs to be guaranteed and requires the various available action set at different states. Thus, we adopt a binary coding in the filter layer, which can select available action depending on the current state. Then, to exclude unavailable actions, the

Q-value of these actions can be increased (i.e., add a constant to the original Q-value, which is a hyper-parameter depending on the magnitude of Q-values). These actions are excluded since only the minimal Q-value is selected to feed into the loss function. As shown in Fig. 5.3, a filter layer is added to help the target network exclude invalid actions and output real Q-values.

C. Experience Replay

Considering the high correlation between continuous states in this scenario (e.g., cumulative energy consumption E_t is highly correlated with E_{t+1} due to the accumulative sum), DNN can be easily over-fitting if high correlation data is fed. Furthermore, the DNN is required to not only learn from current interaction with the environment but also a more varied array of past experiences (e.g., past task arrival pattern). To this end, experience replay is utilized to store experiences including state transitions, costs, risks, and actions, which are necessary to perform the proposed deep risk-sensitive RL. As shown in Fig. 5.3, the replay memory, denoted by D, is used to store experience, and mini-batches of experiences are fed to train DNNs. In Algorithm 1, mini-batches of experience $(\boldsymbol{s}_j, \boldsymbol{a}_j, C_j, R_j, \boldsymbol{s}_{j+1}) \sim U(D)$ are uniformly draw at random from the replay memory to update DNNs. This technique has the following merits: 1) reducing the correlation among experiences in updating DNNs, 2) reusing the previous state transitions to avoid catastrophic forgetting, and 3) increasing learning efficiency with mini-batches and learning stability.

D. ϵ-Greedy Selection

To learn how to react to all possible states in the environment, it must be exposed to as many as possible states. The UAV needs to explore different energy consumption and the number of tasks in the buffer. However, the UAV needs to exploit the exposed experiences to learn a decent task scheduling policy, which conflicts the experience exploration. Thus, the proposed learning policy should deal with such an exploration and exploitation trade-off. To deal with this problem, the ϵ-greedy selection approach is leveraged to balance the trade-off. The UAV selects the action based on approximated Q-value function most of the time, but occasionally chooses the action randomly. In the realization of Algorithm 1, parameter ϵ is an adjustable parameter which determines the probability of taking a random action, rather than the action based on the Q-value function.

5.5 Performance Evaluation

In this section, extensive simulations are carried out to evaluate the proposed deep RL-based DOTS scheme. Specifically, we first elaborate on the simulation settings, and benchmark strategies. Afterward, the overall performance evaluation of the proposed scheme is conducted.

5.5.1 Simulation Settings

In the experiments, locations of MUTs follow a uniform distribution [138]. The computing task arrival is set to follow a Poisson distribution with arrival rate μ, which is unknown *a priori* for the UAV. Referring to well-studied UAV trajectory design algorithm [186], a UAV is dispatched. The UAV flies along with main areas of MUTs, which can be more effective to accommodate the computing service demand. The UAV trajectory is generated by the VISSIM which is a simulation tool in transportation research [121]. The altitude of the UAV is set to 10 m, and the size of computing queue ρ is set to 20. Additionally, by adopting the pathloss (in dB) model of UAV communication in [138], the pathloss of UAV-BS links is given by:

$$PL\left(x,\theta\right)=10A_0\log\left(x\right)+B_0\left(\theta-\theta_0\right)e^{\frac{\theta_0-\theta}{C_0}}+\eta_0, \tag{5.26}$$

where x represents the distance between the UAV and a BS, and θ is the corresponding vertical angle. Both x and θ can be obtained based on UAV location l_t and the BS location. Due to the mobility of the deployed UAV, x and θ vary over different locations. Parameters A_0, θ_0, B_0, C_0 and η_0 in Eq. (5.26) are configured as 3.04, -3.61, -23.29, 4.14, and 20.7, respectively [138]. Meanwhile, the LEO satellite connection is always available for the UAV. The Weibull-based channel model is adopted to model the rain attenuation of UAV-satellite links [194]. Other simulation parameters are listed in Table 5.1.

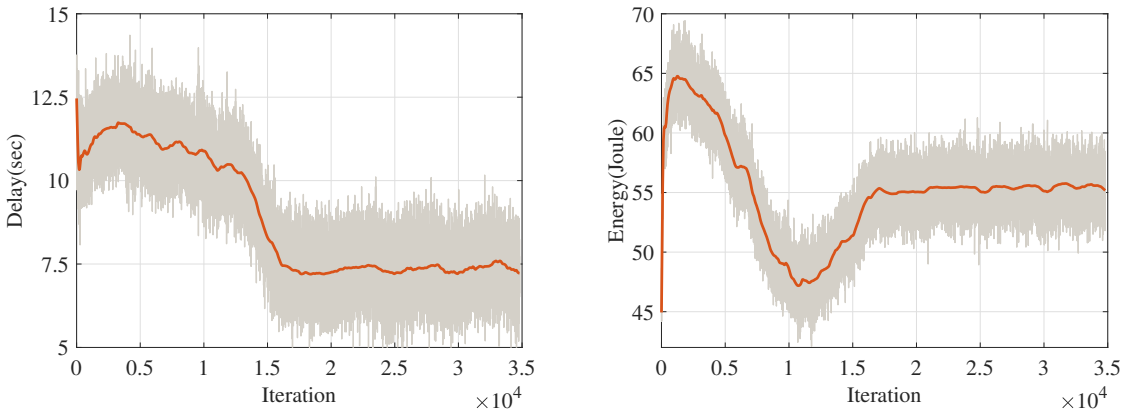
Table 5.1: Simulation Parameters

Parameter	Value	Parameter	Value
N	5	f_{U}	1 Gigacycle/s
ϕ	5 MB	f_0	5 Gigacycle/s
γ	25 cycles/bit	$f_n, n \neq 0$	10 Gigacycle/s
W_{B}	3 MHz	N_0	-174 dBm/Hz
W_{S}	2 MHz	P_{B}	1.6 W
P_{S}	5 W	ξ	10^{-28}
d_{S}	6.44 ms	$\beta^{\max}$	7

The proposed DNN-based scheme is implemented via Python 3.7 and Tensorflow open-source machine learning library. The training of DNNs is conducted with a NVDIA 1660 Ti GPU. The DNN of cost minimization includes four fully-connected hidden layers with (256, 128, 128, 64) neurons, and the DNN of risk minimization includes four fully-connected hidden layers with (512, 256, 128, 128) neurons, respectively. ReLU function is adopted as the activation function to realize nonlinear approximation after the fully connected layers. Additionally, L2 regularization is used to reduce the possibility of DNN over-fitting. Meanwhile, Adam optimizer is adopted in the DNN training. In each episode, the behavior policy during training is ϵ-greedy with ϵ increases linearly from 0 to 0.9995 over 35,000 iterations.

Benchmark schemes adopted in this computing task scheduling problem are introduced below:

- **Random probabilistic configuration (RPC)** – In this scheme, the random policy is adopted, which means that actions are selected randomly in different states. All available actions are selected with the same probability;
- **Sampling-based probabilistic configuration (SPC)** – In this scheme, the probability of available actions on each state is fixed. Based on a large number of historical sampling experiments, the probability of different actions is configured to meet the UAV energy capacity.



(a) Delay versus the number of iterations

(b) Energy consumption versus the number of iterations

Figure 5.4: Convergence performance of the proposed deep RL-based DOTS scheme in one episode.

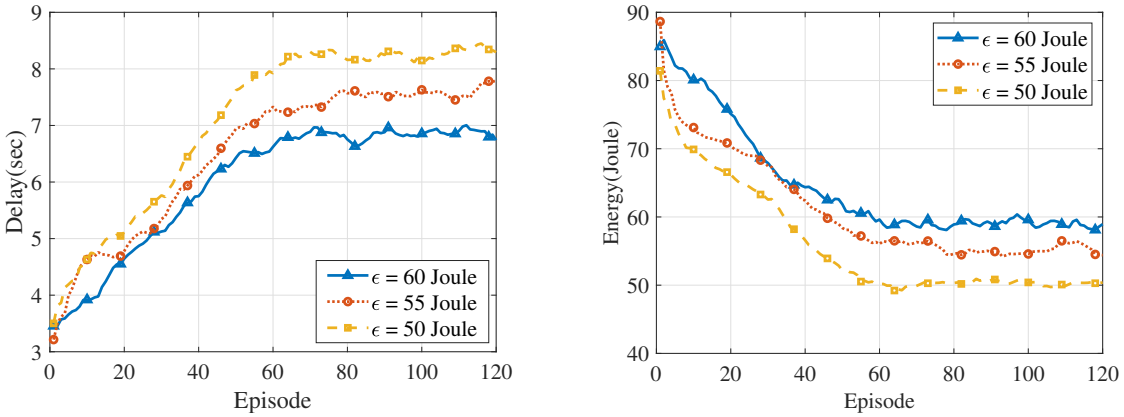
5.5.2 Simulation Results

We show the simulation results of our proposed algorithm from two parts. Firstly, we evaluate the convergence performance of the proposed deep RL-based DOTS scheme. Secondly, we compare the performance of the proposed deep RL-based DOTS scheme with other benchmark schemes.

A. Convergence Performance

The convergence performance of the two-cycle structure of the proposed algorithm is shown in this subsection, i.e., the convergence performance of the inner cycle in Fig. 5.4 and that of the outer cycle in Fig. 5.5.

Fig. 5.4(a) shows the convergence performance of the delay and the energy consumption in the inner cycle (in one episode), respectively, where the orange line is the moving average results of the previous 100 iterations. It can be seen that the delay converges after 16,000 iterations, when UAV energy capacity ε is set to 55 Joule. However, the convergence trends of delay and energy consumption vary differently due to the differentiated functions of the cost and the risk. Specifically, the delay performance gradually decreases and converges after around 16,000 iterations, while the energy consumption performance



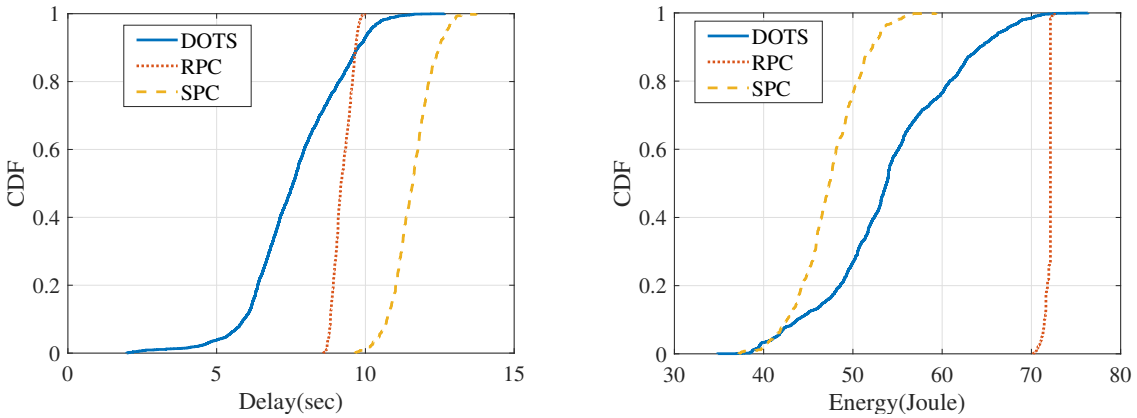
(a) Delay versus the number of episodes

(b) Energy consumption versus the number of episodes

Figure 5.5: Convergence performance of the proposed deep RL-based DOTS scheme.

exhibits a turning point at around the 11,000 th iteration. Compared to the simple policy of minimizing the risk, e.g., the UAV can offload fewer tasks to reduce energy consumption intuitively, the policy of minimizing the cost is related to both the task arrival and the policy of minimizing the risk. As a result, as shown in Fig. 5.4(b), from iteration 0 to iteration 11,000, the policy of minimizing the risk has been well learned, while the learning process of cost minimization is still ongoing as shown in Fig. 5.4(a). After 16,000 iterations, the policy of delay minimization is learned while the energy consumption is approximately equivalent to the UAV energy capacity.

The convergence performance of delay and energy consumption in the outer cycle are shown in Fig. 5.5(a) and Fig. 5.5(b), respectively. To evaluate the convergence performance of the proposed DOTS scheme, we adopt different values of the UAV energy capacity, i.e., 50 Joule, 55 Joule, and 60 Joule. It can be seen that the average delay and average energy consumption converge after 70 episodes, where one episode consists of 35,000 iterations. Both the average delay and the average energy consumption oscillate at the beginning of the learning process due to the inaccurate weight parameter δ, which takes time to approach to the optimal weight parameter. In Fig. 5.5(a), we can observe that the average delay of the learned policy decreases as the increase of the UAV energy capacity of the UAV, which happens since more energy can be consumed by the UAV to offload more tasks to either the BS or the satellite. In Fig. 5.5(b), the impact of energy consumption is shown on different energy consumption capacities. As expected, the energy consumption



(a) The CDF of delay performance

(b) The CDF of energy consumption

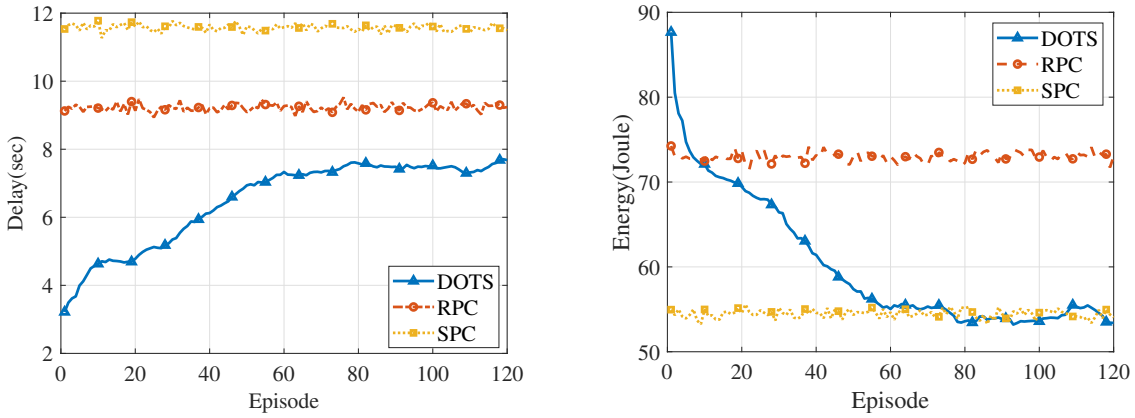
Figure 5.6: Cumulative distribution functions (CDFs) of delay and energy consumption.

of different cases is approximately equivalent to the pre-set energy consumption capacities. Therefore, based on aforementioned convergence performance, the DOTS scheme can work well in scenarios with different energy consumption capacities.

B. Performance Comparison

To compare DOTS with benchmark schemes, we plot cumulative distribution functions (CDFs) of delay and energy consumption in Fig. 5.6(a) and Fig. 5.6(b), respectively. Note that average delay and energy consumption are calculated for the period that UAV flies back to the same destination along the same trajectory. Considering the dynamics of task arrival, we show the delay and energy consumption performance over 1,000 flights. We can see that DOTS is able to enhance the performance that the delay in 90 % flights which is below 9 seconds. Meanwhile, 60 % flights satisfy energy capacity of $\varepsilon = 55$ Joule. The RPC scheme cannot guarantee the UAV energy capacity constraint. Although the SPC scheme can satisfy the energy capacity, the delay of most flights is longer than 8.5 seconds. Therefore, the proposed DOTS scheme can work efficiently in different task arrival scenarios.

Figures 5.7(a) and 5.7(b) show the delay and energy consumption performance under DOTS, RPC, and SPC schemes, where the energy capacity is set to $\varepsilon = 55$ Joule. In the simulation, we set the probability of offloading tasks in the SPC scheme to satisfy energy capacity 55 Joule. It can be seen that the DOTS scheme and the SPC scheme are



(a) (b)

Figure 5.7: Performance of delay and energy consumption.

able to guarantee the UAV energy capacity constraint. However, the delay performance of the SPC scheme is worse than DOTS before 40 episodes, and the RPC scheme is always worse than the DOTS scheme. At the beginning of the learning process, the delay can be minimized, but the UAV energy capacity is exceeded. Due to the un-tuned weight δ at the beginning of the learning process, the goal of the policy is to minimize the cost. With the learning episode increasing, the policy of risk minimization can be found. Therefore, after 60 episodes, the delay-minimized policy is learned without exceeding the UAV energy capacity. Compared with the other two schemes, the proposed scheme has the lowest time-averaged task processing delay when the optimal policy has been learned.

Figure 5.8 shows the offloading proportion under different policies with $\varepsilon = 55$ Joule. The action proportion of SPC and RPC schemes is similar, as both of them are based on probabilistic selection. However, RPC cannot guarantee the UAV energy capacity constraint. Although the SPC scheme can bound the energy consumption, SPC selects actions based on the historical experience, and thus it cannot learn to schedule proper number of tasks in different scenarios according to the future information. Particularly, the SPC scheme and the RPC scheme may offload the tasks at inappropriate states (e.g., low data rate), in which task offloading to other BSs or satellite should be suppressed and wait for more appropriate states (e.g., high data rate). Unlike the benchmark schemes, the proposed DOTS scheme can make the UAV offload a certain number of tasks to BSs when they are covered by BS, and offload to the satellite when it is out of the BS coverage. As offloading tasks to the satellite is an important complementary solution for offloading tasks

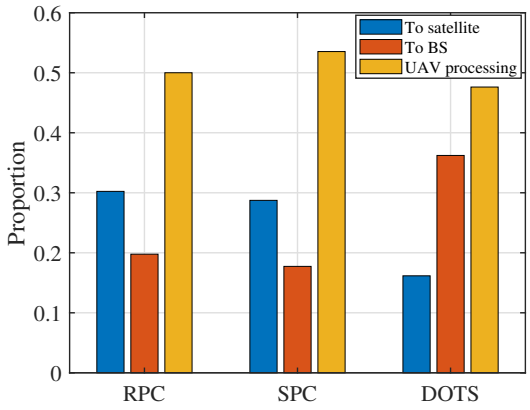


Figure 5.8: Offloading proportion under different policies for $\varepsilon = 55$ Joule.

to BSs, it effectively reduces the queuing delay when the UAV is out of the BS coverage. Therefore, the RL-based DOTS scheme can schedule the optimal number of tasks to BS or satellite according to the learned knowledge, such as the task arrival pattern.

5.6 Summary

In this chapter, we have proposed a novel computing task scheduling scheme named DOTS in SAGIN, where a UAV is dispatched to collect tasks from MUTs and then make online scheduling decisions to process the tasks. Considering the limited UAV energy capacity and the dynamics of task arrival, we have formulated the online scheduling problem as a CMDP. With the objective of minimizing the long-term average delay without violating the constraint, we have designed the deep risk-sensitive RL algorithm to make online task scheduling decisions. Extensive simulation results have demonstrated that the deep RL-based DOTS scheme can significantly reduce the delay of processing computing tasks while satisfying the UAV energy capacity constraint. The proposed scheme can provide low-latency computing services and extend the service lifespan for a large number of MUTs with limited power supply.

Chapter 6

Conclusions and Future Works

In this chapter, we summarize the main results and contributions of this thesis and present our future research directions.

6.1 Main Research Contributions

In this thesis, we have investigated data-driven network management for the NGWN. In specific, three data-driven schemes have been designed to facilitate network management in the planning and operation stages for different services in the NGWN. First, an unsupervised learning-assisted scheme has been designed to enable planning-stage service coverage and power control for different communication services. Then, a meta learning-based scheme has been designed to enable planning-stage resource reservation for a stateful compute-intensive service. Last, a reinforcement learning-based scheme has been designed to enable operation-stage computing task scheduling for a compute-intensive service in SAGIN. The main contributions of this thesis are summarized as follows.

1. A planning-stage service coverage and power control scheme has been proposed with three novel designs, including flexible binary slice zooming, dual time-scale planning, and grid-based network planning. The scheme introduces flexibility to differentiate the service coverage and downlink transmission power of the same BS for different network slices while improving the temporal and spatial granularity of network planning. With the proposed scheme, we develop an unsupervised learning-assisted method to solve the complicated network planning problem in two steps. In the first step, we derive the optimal solution of power control based on optimization theory

and design a heuristic algorithm to obtain initial solutions of service coverage. In the second step, we adopt unsupervised learning to find similar historical solutions and use the historical solution to refine the initial solutions obtained by the model-driven algorithm. The work provides a flexible and fine-grained solutions for multiple communication services while satisfying their different QoS requirements.

2. A planning-stage resource reservation scheme has been proposed with two elements consisting of multi-resource reservation and resource reservation re-configuration. We group MUTs based on their mobility patterns and enable group-based resource reservation, and develop an automated closed-loop approach to re-configure resource reservation in a dynamic network environment for balancing the network resource usage and the cost from re-configuring resource reservation. With the proposed scheme, we design a meta-learning-based method to solve the complex optimization problem. We decouple the problem into three subproblems and adopt the matching theory, particle swarm optimization, and meta learning to solve them. The work provides a data-driven framework to facilitate fine-grained resource reservation based on data contained in DTs and intelligent resource reservation re-configuration to adapt to dynamic spatial-temporal service demands for supporting stateful compute-intensive applications.

3. An operation-stage computing task scheduling scheme has been proposed for compute-intensive services in SAGIN. Considering the energy budget of UAV, we formulate the online computing scheduling in the dynamic network environment as a constrained Markov decision process. Then, we develop a risk-sensitive reinforcement learning approach in which a risk value is used to represent energy consumption that exceeds the energy budget. By balancing the risk value and the reward from delay minimization, we can explore the task scheduling policy to minimize task offloading and processing delay while satisfying the UAV energy constraint. With this scheme, the computing resources from different network components, i.e., terrestrial BSs, UAVs, and satellites, are coordinated efficiently and adaptively to improve service performance.

In summary, this thesis has investigated data-driven network management for the NGWN. Based on the advantages of addressing complicated network management problems and adapting to highly dynamic and complex network environments, we leverage data-driven methods to complement conventional model-driven network management. All the three research issues in the thesis have concentrated on the design of data-driven network management solutions to improve flexibility, fine-granularity, and adaptivity. The

proposed schemes and theoretical analysis can provide valuable guidelines for implementing data-driven network management in the NGWN.

6.2 Future Works

For the future research, there are some interesting and promising research directions listed as follows:

1. **User-centric Network Management** – Individual MUTs may have different requirements, e.g., Quality-of-Experience (QoE). For example, for Metaverse and immersive AR/VR applications, different MUTs have diverse hardware, heterogeneous connections, and different immersive scenarios. As a result, optimal service provision depends on individual MUTs, such as their social connection, hardware, and network conditions. To this end, differentiation and customization for MUTs can be important for many applications in the future. However, existing network management schemes are mostly from the perspective of service, i.e., service-centric, which have limited flexibility to incorporate MUT-level differentiation and customization. The main reasons can be summarized from two aspects. First, MUT-level network management necessitates a customized MUT data profile to collect and manage data characterizing MUTs (e.g., QoE performance) and the situation of MUTs (e.g., location and activity). However, the systematic design of creating and maintaining such MUT data profile lacks in existing network management schemes. Second, even if extensive MUT data can be provided easily, utilizing such MUT data to customize network management and service provision is still not straightforward. For example, user-level network planning is fine-grained for improving resource utilization but results in extreme overhead from control and signaling. Making the right trade-offs in different circumstances for each MUT is challenging. While user-centric network management has benefits for both MUTs and networks, the system design of user-centric network management calls for further investigation.

2. **Network Management for AI Services** – With the rapid development of AI in different research areas, AI can function as a service, namely AI services. Designing and optimizing network management schemes to facilitate AI services, also summarized as "Networking for AI", become a hot topic. In contrast to conventional services, AI services have the following two features. First, AI services have unique performance metrics, e.g., accuracy and convergence rate, in addition to conventional performance metrics for networking and communication. Such unique performance

of AI services mainly depends on the provided data. However, conventional works on network management do not focus on data management that is used for improving service performance. Data could be envisioned as a type of network resource in network management. Second, different AI algorithms can be adopted for the same AI service. For example, AI algorithms with different types of DNNs can be used for object detection, which results in multiple options in network management for AI services. Since different options may yield different network resource usage, including computing and storage, choosing an appropriate AI algorithm for each AI service is required. The resulting network management for AI services becomes more complicated than conventional compute-intensive services. Due to the distinct features of AI services, some innovative designs of network management are important for supporting AI services.

References

[1] X. Shen, J. Gao, W. Wu, K. Lyu, M. Li, W. Zhuang, X. Li, and J. Rao, "AI-assisted network-slicing based next-generation wireless networks," *IEEE Open J. Veh. Technol.*, vol. 1, no. 1, pp. 45–66, 2020.

[2] W. Tong and P. Zhu, "6G: The next horizon," *Cambridge Univ. Press*, 2022.

[3] X. Shen, J. Gao, W. Wu, M. Li, C. Zhou, and W. Zhuang, "Holistic network virtualization and pervasive network intelligence for 6G," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 1, pp. 1–30, 2022.

[4] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 869–904, 2020.

[5] Huawei, "Touching an intelligent world," 2019, [Online]. Available: https://www.huawei.com/minisite/giv/Files/whitepaper_en_2019.pdf.

[6] N. Kato, Z. M. Fadlullah, F. Tang, B. Mao, S. Tani, A. Okamura, and J. Liu, "Optimizing space-air-ground integrated networks by artificial intelligence," *IEEE Wireless Commun.*, vol. 26, no. 4, pp. 140–147, 2019.

[7] GSMA, "The mobile economy," 2020, [Online]. Available: https://www.gsma.com/mobileeconomy/wp-content/uploads/2020/03/GSMA_MobileEconomy2020_Global.pdf.

[8] E. Buchen, "Small satellite market observations," 2015.

[9] B. Di, L. Song, Y. Li, and H. V. Poor, "Ultra-dense LEO: Integration of satellite access networks into 5G and beyond," *IEEE Wireless Commun.*, vol. 26, no. 2, pp. 62–69, 2019.

[10] Q. Wu, Y. Zeng, and R. Zhang, "Joint trajectory and communication design for multi-UAV enabled wireless networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 2109–2121, 2018.

[11] C. Zhou, H. He, P. Yang, F. Lyu, W. Wu, N. Cheng, and X. Shen, "Deep RL-based trajectory planning for AoI minimization in UAV-assisted IoT," in *Proc. IEEE WCSP*, 2019.

[12] N. Zhang, S. Zhang, P. Yang, O. Alhussein, W. Zhuang, and X. Shen, "Software defined space-air-ground integrated vehicular networks: Challenges and solutions," *IEEE Commun. Mag.*, vol. 55, no. 7, pp. 101–109, 2017.

[13] Ł. Budzisz, F. Ganji, G. Rizzo, M. A. Marsan, M. Meo, Y. Zhang, G. Koutitas, L. Tassiulas, S. Lambert, B. Lannoo *et al.*, "Dynamic resource provisioning for energy efficiency in wireless access networks: A survey and an outlook," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 2259–2285, 2014.

[14] S. Wang and C. Ran, "Rethinking cellular network planning and optimization," *IEEE Wireless Commun.*, vol. 23, no. 2, pp. 118–125, 2016.

[15] W. Ni, I. B. Collings, J. Lipman, X. Wang, M. Tao, and M. Abolhasan, "Graph theory and its applications to future network planning: Software-defined online small cell management," *IEEE Wireless Commun.*, vol. 22, no. 1, pp. 52–60, 2015.

[16] M. Chraiti, A. Ghrayeb, C. Assi, N. Bouguila, and R. A. Valenzuela, "A framework for unsupervised planning of cellular networks using statistical machine learning," *IEEE Trans. Commun.*, vol. 68, no. 5, pp. 3213–3228, 2020.

[17] S. Kekki *et al.*, "MEC in 5G networks," *ETSI white paper*, vol. 28, pp. 1–28, 2018.

[18] K. Wang, W. Chen, J. Li, Y. Yang, and L. Hanzo, "Joint task offloading and caching for massive MIMO-aided multi-tier computing networks," *IEEE Trans. Commun.*, vol. 70, no. 3, pp. 1820–1833, 2022.

[19] X. Li, M. Samaka, H. A. Chan, D. Bhamare, L. Gupta, C. Guo, and R. Jain, "Network slicing for 5G: Challenges and opportunities," *IEEE Internet Comput.*, vol. 21, no. 5, pp. 20–27, 2017.

[20] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888–1906, 2013.

[21] F. Li, D. Yu, H. Yang, J. Yu, H. Karl, and X. Cheng, "Multi-armed-bandit-based spectrum scheduling algorithms in wireless networks: A survey," *IEEE Wireless Commun.*, vol. 27, no. 1, pp. 24–30, 2020.

[22] W. Wu, C. Zhou, M. Li, H. Wu, H. Zhou, N. Zhang, X. S. Shen, and W. Zhuang, "AI-native network slicing for 6G networks," *IEEE Wireless Commun.*, vol. 29, no. 1, pp. 96–103, 2022.

[23] H. Chergui and C. Verikoukis, "Offline SLA-constrained deep learning for 5G networks reliable and dynamic end-to-end slicing," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 350–360, 2020.

[24] S. Hu, Y.-C. Liang, Z. Xiong, and D. Niyato, "Blockchain and artificial intelligence for dynamic resource sharing in 6G and beyond," *IEEE Wireless Commun.*, vol. 28, no. 4, pp. 145–151, 2021.

[25] Q. Ye, W. Zhuang, S. Zhang, A. Jin, X. Shen, and X. Li, "Dynamic radio resource slicing for a two-tier heterogeneous wireless network," *IEEE Trans. Veh. Technol.*, vol. 67, no. 10, pp. 9896–9910, 2018.

[26] H. Yu, J. Yang, and C. Fung, "Fine-grained cloud resource provisioning for virtual network function," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 3, pp. 1363–1376, 2020.

[27] Z. Zhou, J. Feng, C. Zhang, Z. Chang, Y. Zhang, and K. Huq, "SAGECELL: Software-defined space-air-ground integrated moving cells," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 92–99, 2018.

[28] A. Zappone, M. Di Renzo, and M. Debbah, "Wireless networks design in the era of deep learning: Model-based, AI-based, or both?" *IEEE Trans. Commun.*, vol. 67, no. 10, pp. 7331–7376, 2019.

[29] D. Huang, B. He, and C. Miao, "A survey of resource management in multi-tier web applications," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1574–1590, 2014.

[30] T. M. Mitchell and T. M. Mitchell, *Machine learning.* McGraw-hill New York, 1997, vol. 1, no. 9.

[31] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[32] C. Janiesch, P. Zschech, and K. Heinrich, "Machine learning and deep learning," *Electronic Markets*, vol. 31, no. 3, pp. 685–695, 2021.

[33] Q. Yu, J. Ren, H. Zhou, and W. Zhang, "A cybertwin based network architecture for 6G," in *Proc. IEEE 6G SUMMIT*, Levi, Finland, 2020.

[34] F. Tao, H. Zhang, A. Liu, and A. Nee, "Digital twin in industry: State-of-the-art," *IEEE Trans. Ind. Informat.*, vol. 15, no. 4, pp. 2405–2415, 2018.

[35] T. Hastie, R. Tibshirani, and J. Friedman, "Overview of supervised learning," in *The elements of statistical learning*. Springer, 2009, pp. 9–41.

[36] H. Wu, F. Lyu, C. Zhou, J. Chen, L. Wang, and X. Shen, "Optimal UAV caching and trajectory in aerial-assisted vehicular networks: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 12, pp. 2783–2797, 2020.

[37] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. Lew, "Deep learning for visual understanding: A review," *Neurocomputing*, vol. 187, pp. 27–48, 2016.

[38] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[39] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[40] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[41] S. Shillaker and P. Pietzuch, "Faasm: Lightweight isolation for efficient stateful serverless computing," in *Proc. USENIX ATC*, 2020, Boston, MA, USA.

[42] D. Barcelona-Pons, M. Sánchez-Artigas, G. París, P. Sutra, and P. García-López, "On the FaaS track: Building stateful distributed applications with serverless architectures," in *Proc. ACM/IFIP Middleware*, 2019, Davis, CA, USA.

[43] Y. Jin, *Multi-objective machine learning*. Springer Science & Business Media, 2006.

[44] E. Amaldi, A. Capone, and F. Malucelli, "Planning UMTS base station location: Optimization models with power control and algorithms," *IEEE Trans. Wireless Commun.*, vol. 2, no. 5, pp. 939–952, 2003.

[45] K. Tutschku, "Demand-based radio network planning of cellular mobile communication systems," in *Proc. IEEE INFOCOM*, 1998.

[46] H. Ghazzai, E. Yaacoub, M.-S. Alouini, Z. Dawy, and A. Abu-Dayya, "Optimized LTE cell planning with varying spatial and temporal user densities," *IEEE Trans. Veh. Technol.*, vol. 65, no. 3, pp. 1575–1589, 2015.

[47] A. Guo and M. Haenggi, "Spatial stochastic models and metrics for the structure of base stations in cellular networks," *IEEE Trans. Wireless Commun.*, vol. 12, no. 11, pp. 5800–5812, 2013.

[48] A. A. Khalek, L. Al-Kanj, Z. Dawy, and G. Turkiyyah, "Optimization models and algorithms for joint uplink/downlink UMTS radio network planning with sir-based power control," *IEEE Trans. Veh. Technol.*, vol. 60, no. 4, pp. 1612–1625, 2011.

[49] J. Lyu, Y. Zeng, R. Zhang, and T. J. Lim, "Placement optimization of UAV-mounted mobile base stations," *IEEE Commun. Lett.*, vol. 21, no. 3, pp. 604–607, 2016.

[50] E. Kalantari, H. Yanikomeroglu, and A. Yongacoglu, "On the number and 3D placement of drone base stations in wireless cellular networks," in *Proc. IEEE VTC-Fall*, 2016.

[51] M. Mozaffari, A. T. Z. Kasgari, W. Saad, M. Bennis, and M. Debbah, "Beyond 5G with UAVs: Foundations of a 3D wireless cellular network," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 357–372, 2018.

[52] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Efficient deployment of multiple unmanned aerial vehicles for optimal wireless coverage," *IEEE Commun. Lett.*, vol. 20, no. 8, pp. 1647–1650, 2016.

[53] R. I. Bor-Yaliniz, A. El-Keyi, and H. Yanikomeroglu, "Efficient 3-D placement of an aerial base station in next generation cellular networks," in *Proc. IEEE ICC*, 2016.

[54] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Drone small cells in the clouds: Design, deployment and performance analysis," in *Proc. IEEE GLOBECOM*, 2015.

[55] J. C. Cheung, M. A. Beach, and J. P. McGeehan, "Network planning for third-generation mobile radio systems," *IEEE Commun. Mag. IEEE Wireless Comm*, vol. 32, no. 11, pp. 54–59, 1994.

[56] Q. Ye, B. Rong, Y. Chen, M. Al-Shalash, C. Caramanis, and J. G. Andrews, "User association for load balancing in heterogeneous cellular networks," *IEEE Trans. Wireless Commun.*, vol. 12, no. 6, pp. 2706–2716, 2013.

[57] A. Khamesi and M. Zorzi, "Energy harvesting and cell zooming in K- tier heterogeneous random cellular networks," *IEEE Trans. Green Commun. Netw.*, vol. 2, no. 1, pp. 63–73, 2018.

[58] G. Giambene, T. Bourgeau, H. Chaouchi *et al.*, "Iterative multi-level soft frequency reuse with load balancing for heterogeneous LTE-A systems," *IEEE Trans. Wireless Commun.*, no. 2, pp. 924–938, 2017.

[59] D. Kim, T. Park, S. Kim, H. Kim, and S. Choi, "Load balancing in two-tier cellular networks with open and hybrid access femtocells," *IEEE/ACM Trans. Netw.*, vol. 24, no. 6, pp. 3397–3411, 2016.

[60] S. Wang, W. Zhao, and C. Wang, "Budgeted cell planning for cellular networks with small cells," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4797–4806, 2014.

[61] Y. Wu, P. A. Chou, Q. Zhang, K. Jain, W. Zhu, and S.-Y. Kung, "Network planning in wireless Ad Hoc networks: A cross-layer approach," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 1, pp. 136–150, 2005.

[62] M. M. Hasan, S. Kwon, and J.-H. Na, "Adaptive mobility load balancing algorithm for LTE small-cell networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 4, pp. 2205–2217, 2018.

[63] I. Siomina and D. Yuan, "Analysis of cell load coupling for LTE network planning and optimization," *IEEE Trans. Wireless Commun.*, vol. 11, no. 6, pp. 2287–2297, 2012.

[64] K. I. Pedersen, B. Soret, S. B. Sanchez, G. Pocovi, and H. Wang, "Dynamic enhanced intercell interference coordination for realistic networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 7, pp. 5551–5562, 2015.

[65] S. N. T.-c. Chiueh and S. Brook, "A survey on virtualization technologies," *Rpe Report*, vol. 142, 2005.

[66] S. Meier, B. Virun, J. Blumert, and M. T. Jones, "IBM systems virtualization: Servers, storage, and software," *IBM Redbook, May*, 2008.

[67] R. P. Goldberg, "Survey of virtual machine research," *Comput.*, vol. 7, no. 6, pp. 34–45, 1974.

[68] L. Peterson and T. Roscoe, "The design principles of planetlab," *ACM SIGOPS operating systems review*, vol. 40, no. 1, pp. 11–16, 2006.

[69] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Communications Surveys Tutorials*, vol. 20, no. 3, pp. 2429–2453, 2018.

[70] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, and I. Seskar, "GENI: A federated testbed for innovative network experiments," *Computer Networks*, vol. 61, pp. 5–23, 2014.

[71] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[72] A. Voellmy and J. Wang, "Scalable software defined network controllers," in *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, 2012, pp. 289–290.

[73] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, 2015.

[74] 3GPP, "Technical specification group services and system aspects; enhancement of dedicated core networks selection mechanism; (Release 14)," Tech. Rep. 3GPP TR 23.711 V14.0.0, Sep. 2016.

[75] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Commun. Mag.*, vol. 53, no. 2, pp. 90–97, 2015.

[76] Q. Zhang, F. Liu, and C. Zeng, "Adaptive interference-aware VNF placement for service-customized 5G network slices," in *Proc. IEEE INFOCOM*, 2019.

[77] T. Taleb, M. Bagaa, and A. Ksentini, "User mobility-aware virtual network function placement for virtual 5G network infrastructure," in *Proc. IEEE ICC*, 2015.

[78] O. Alhussein and W. Zhuang, "Robust online composition, routing and NF placement for NFV-enabled services," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1089–1101, 2020.

[79] L. M. M. Zorello, M. Sodano, S. Troia, and G. Maier, "Power-efficient baseband-function placement in latency-constrained 5G metro access," *IEEE Trans. Green Commun. Netw.*, 2022.

[80] D. Harutyunyan, N. Shahriar, R. Boutaba, and R. Riggio, "Latency and mobility-aware service function chain placement in 5G networks," *IEEE Trans. Mobile Comput.*, 2020.

[81] K. Qu, W. Zhuang, Q. Ye, X. Shen, X. Li, and J. Rao, "Dynamic flow migration for embedded services in SDN/NFV-enabled 5G core networks," *IEEE Trans. Commun.*, vol. 68, no. 4, pp. 2394–2408, 2020.

[82] D. Bhamare, R. Jain, M. Samaka, and A. Erbad, "A survey on service function chaining," *Journal of Network and Computer Applications*, vol. 75, pp. 138–155, 2016.

[83] I. Jang, D. Suh, S. Pack, and G. Dán, "Joint optimization of service function placement and flow distribution for service function chaining," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2532–2541, 2017.

[84] H.-J. Einsiedler, A. Gavras, P. Sellstedt, R. Aguiar, R. Trivisonno, and D. Lavaux, "System design for 5G converged networks," in *2015 European Conference on Networks and Communications (EuCNC)*. IEEE, 2015, pp. 391–396.

[85] K. Mahmood, T. Mahmoodi, R. Trivisonno, A. Gavras, D. Trossen, and M. Liebsch, "On the integration of verticals through 5G control plane," in *2017 European Conference on Networks and Communications (EuCNC)*. IEEE, 2017, pp. 1–5.

[86] 3GPP, "Technical specification group services and system aspects; procedures for the 5G systems (5GS); stage 2 (Release 16)," Tech. Rep. 3GPP TR 23.711 V16.6.0, Sep. 2020.

[87] I. Da Silva, G. Mildh, A. Kaloxylos, P. Spapis, E. Buracchini, A. Trogolo, G. Zimmermann, and N. Bayer, "Impact of network slicing on 5G radio access networks," in *Proc. IEEE EuCNC*, 2016.

[88] R. Ferrus, O. Sallent, J. Pérez-Romero, and R. Agusti, "On 5G radio access network slicing: Radio interface protocol features and configuration," *IEEE Commun. Mag.*, vol. 56, no. 5, pp. 184–192, 2018.

[89] S. E. Elayoubi, S. B. Jemaa, Z. Altman, and A. Galindo-Serrano, "5G ran slicing for verticals: Enablers and challenges," *IEEE Commun. Mag.*, vol. 57, no. 1, pp. 28–34, 2019.

[90] S. T. Arzo, R. Bassoli, F. Granelli, and F. H. Fitzek, "Study of virtual network function placement in 5G cloud radio access network," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 4, pp. 2242–2259, 2020.

[91] P. Muñoz, O. Sallent, and J. Pérez-Romero, "Self-dimensioning and planning of small cell capacity in multitenant 5G networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 5, pp. 4552–4564, 2018.

[92] T. Taleb, D. E. Bensalem, and A. Laghrissi, "Smart service-oriented clustering for dynamic slice configuration," in *Proc. IEEE GLOBECOM*, 2019.

[93] Y. Sun, S. Qin, G. Feng, L. Zhang, and M. A. Imran, "Service provisioning framework for RAN slicing: User admissibility, slice association and bandwidth allocation," *IEEE Trans. Mobile Comput.*, vol. 20, no. 12, pp. 3409–3422, 2020.

[94] W. Wu, N. Chen, C. Zhou, M. Li, X. Shen, W. Zhuang, and X. Li, "Dynamic RAN slicing for service-oriented vehicular networks via constrained learning," *IEEE J. Sel. Areas Commun.*, pp. 1–16, 2020, DOI: 10.1109/JSAC.2020.3041405.

[95] M. Li, J. Gao, C. Zhou, X. S. Shen, and W. Zhuang, "Slicing-based artificial intelligence service provisioning on the network edge: Balancing AI service performance and resource consumption of data management," *IEEE Veh. Technol. Mag.*, vol. 16, no. 4, pp. 16–26, 2021.

[96] X. Yu, P. Navaratnam, and K. Moessner, "Resource reservation schemes for IEEE 802.11-based wireless networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 3, pp. 1042–1061, 2012.

[97] Y. He, J. Sun, X. Ma, A. V. Vasilakos, R. Yuan, and W. Gong, "Semi-random backoff: Towards resource reservation for channel access in wireless LANs," *IEEE/ACM Trans. Netw.*, vol. 21, no. 1, pp. 204–217, 2012.

[98] J.-Y. Chang and H.-L. Chen, "Dynamic-grouping bandwidth reservation scheme for multimedia wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 10, pp. 1566–1574, 2003.

[99] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "DeepCog: Optimizing resource provisioning in network slicing with AI-based capacity forecasting," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 361–376, 2019.

[100] Y. Zhang, S. Bi, and Y.-J. A. Zhang, "Joint spectrum reservation and on-demand request for mobile virtual network operators," *IEEE Trans. Commun.*, vol. 66, no. 7, pp. 2966–2977, 2018.

[101] N. Reyhanian, H. Farmanbar, and Z.-Q. Luo, "Resource reservation in backhaul and radio access network with uncertain user demands," *IEEE Trans. Veh. Technol.*, 2022.

[102] S. Hu, W. Shi, and G. Li, "CEC: A containerized edge computing framework for dynamic resource provisioning," *IEEE Trans. Mobile Comput.*, pp. 1–15, 2022, to be published, doi: 0.1109/TMC.2022.3147800.

[103] J. Zhang, S. Chen, X. Wang, and Y. Zhu, "Dynamic reservation of edge servers via deep reinforcement learning for connected vehicles," *IEEE Trans. Mobile Comput.*, pp. 1–13, 2021, to be published, doi: 10.1109/TMC.2021.3123135.

[104] H. Yin, X. Zhang, H. H. Liu, Y. Luo, C. Tian, S. Zhao, and F. Li, "Edge provisioning with flexible server placement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 1031–1045, 2016.

[105] Z. Zhou, S. Yu, W. Chen, and X. Chen, "CE-IoT: Cost-effective cloud-edge resource provisioning for heterogeneous IoT applications," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8600–8614, 2020.

[106] H. Wu, J. Chen, C. Zhou, J. Li, and X. Shen, "Learning-based joint resource slicing and scheduling in space-terrestrial integrated vehicular networks," *J. Commun. Netw.*, vol. 6, no. 3, pp. 208–223, 2021.

[107] Q. Xiang, H. Yu, J. Aspnes, F. Le, C. Guok, L. Kong, and Y. R. Yang, "Optimizing in the dark: Learning optimal network resource reservation through a simple request interface," *IEEE/ACM Trans. Netw.*, vol. 29, no. 2, pp. 571–584, 2021.

[108] V. Heorhiadi, M. K. Reiter, and V. Sekar, "Simplifying software-defined network optimization using SOL," in *USENIX NSDI*, 2016, pp. 223–237.

[109] Q. Xiang, J. J. Zhang, X. T. Wang, Y. J. Liu, C. Guok, F. Le, J. MacAuley, H. Newman, and Y. R. Yang, "Toward fine-grained, privacy-preserving, efficient multi-domain network resource discovery," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 8, pp. 1924–1940, 2019.

[110] E. L. Madruga and L. M. R. Tarouco, "Fault management tools for a cooperative and decentralized network operations environment," *IEEE J. Sel. Areas Commun.*, vol. 12, no. 6, pp. 1121–1130, 1994.

[111] M. I. Kamel, L. B. Le, and A. Girard, "LTE wireless network virtualization: Dynamic slicing via flexible scheduling," in *Proc. IEEE VTC-Fall*, 2014.

[112] B. Yang, L. Zhang, O. Onireti, P. Xiao, M. A. Imran, and R. Tafazolli, "Mixed-numerology signals transmission and interference cancellation for radio access network slicing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 8, pp. 5132–5147, 2020.

[113] V. Mancuso, P. Castagno, M. Sereno, and M. A. Marsan, "Slicing cell resources: The case of HTC and MTC coexistence," in *Proc. IEEE INFOCOM*, 2019.

[114] B. Han, V. Sciancalepore, D. Feng, X. Costa Perez, and H. D. Schotten, "A utility-driven multi-queue admission control solution for network slicing," in *Proc. IEEE INFOCOM*, 2019.

[115] S. D'oro, F. Restuccia, A. Talamonti, and T. Melodia, "The slice is served: Enforcing radio access network slicing in virtualized 5G systems," in *Proc. IEEE INFOCOM*, 2019.

[116] P. Caballero, A. Banchs, G. De Veciana, and X. Costa-Pérez, "Network slicing games: Enabling customization in multi-tenant mobile networks," *IEEE/ACM Trans. Netw.*, vol. 27, no. 2, pp. 662–675, 2019.

[117] P. Caballero, A. Banchs, G. De Veciana, and X. Costa-Perez, "Multi-tenant radio access network slicing: Statistical multiplexing of spatial loads," *IEEE/ACM Trans Netw.*, vol. 25, no. 5, pp. 3044–3058, 2017.

[118] G. Yu, Y. Jiang, L. Xu, and G. Y. Li, "Multi-objective energy-efficient resource allocation for multi-RAT heterogeneous networks," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 10, pp. 2118–2127, 2015.

[119] G. Tseliou, F. Adelantado, and C. Verikoukis, "Scalable RAN virtualization in multitenant LTE-A heterogeneous networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 8, pp. 6651–6664, 2016.

[120] Y. Xiao, M. Hirzallah, and M. Krunz, "Distributed resource allocation for network slicing over licensed and unlicensed bands," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2260–2274, 2018.

[121] S. Zhang, W. Quan, J. Li, W. Shi, P. Yang, and X. Shen, "Air-ground integrated vehicular network slicing with content pushing and caching," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 2114–2127, 2018.

[122] H. Wu, J. Chen, C. Zhou, W. Shi, N. Cheng, W. Xu, W. Zhuang, and X. S. Shen, "Resource management in space-air-ground integrated vehicular networks: SDN control and AI algorithm design," *IEEE Wireless Commun.*, vol. 27, no. 6, pp. 52–60, 2020.

[123] C. Zhou, H. Wu, M. He, W. Wu, N. Cheng, and X. Shen, "Adaptive access mode selection in space-ground integrated vehicular networks," in *Proc. IEEE GLOBECOM*, 2021.

[124] N. Van Huynh, D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz, "Optimal and fast real-time resource slicing with deep dueling neural networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1455–1470, 2019.

[125] H. Halabian, "Distributed resource allocation optimization in 5G virtualized networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 627–642, 2019.

[126] C. Liang, F. R. Yu, H. Yao, and Z. Han, "Virtual resource allocation in information-centric wireless networks with virtualization," *IEEE Trans. Veh. Technol.*, vol. 65, no. 12, pp. 9902–9914, 2016.

[127] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, 2015.

[128] H. He, H. Shan, A. Huang, Q. Ye, and W. Zhuang, "Edge-aided computing and transmission scheduling for LTE-U-enabled IoT," *IEEE Trans. Wireless Commun.*, vol. 19, no. 12, pp. 7881–7896, 2020.

[129] S. Pasteris, S. Wang, M. Herbster, and T. He, "Service placement with provable guarantees in heterogeneous edge computing systems," in *Proc. IEEE INFOCOM*, 2019, Paris, France.

[130] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2333–2345, 2018.

[131] Z. Kuang, L. Li, J. Gao, L. Zhao, and A. Liu, "Partial offloading scheduling and power allocation for mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6774–6785, Aug. 2019.

[132] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, 2016.

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

[134] Y. Sun, S. Zhou, and J. Xu, "EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2637–2646, 2017.

[135] I. Lera, C. Guerrero, and C. Juiz, "Availability-aware service placement policy in fog computing based on graph partitions," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3641–3651, 2018.

[136] L. Chen and J. Xu, "Budget-constrained edge service provisioning with demand estimation via bandit learning," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2364–2376, 2019.

[137] S. Yu, R. Langar, X. Fu, L. Wang, and Z. Han, "Computation offloading with data caching enhancement for mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11 098–11 112, 2018.

[138] N. Cheng, F. Lyu, W. Quan, C. Zhou, H. He, W. Shi, and X. Shen, "Space/aerial-assisted computing offloading for IoT applications: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117–1129, 2019.

[139] C. Zhou, W. Wu, H. He, P. Yang, F. Lyu, N. Cheng, and X. Shen, "Delay-aware IoT task scheduling in space-air-ground integrated network," in *Proc. IEEE GLOBECOM*. IEEE, 2019.

[140] A. Varasteh, S. Hofmann, N. Deric, M. He, D. Schupke, W. Kellerer, and C. M. Machuca, "Mobility-aware joint service placement and routing in space-air-ground integrated networks," in *Proc. IEEE ICC*, Shanghai, China, 2019.

[141] 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 Commun. Mag.*, vol. 56, no. 8, pp. 26–32, 2018.

[142] W. Saad, M. Bennis, and M. Chen, "A vision of 6G wireless systems: Applications, trends, technologies, and open research problems," *IEEE Netw.*, vol. 34, no. 3, pp. 134–142, 2020.

[143] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5G: Survey and challenges," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 94–100, 2017.

[144] Q. Liu, T. Han, N. Zhang, and Y. Wang, "Deepslicing: Deep reinforcement learning assisted resource allocation for network slicing," *arXiv preprint arXiv:2008.07614*, 2020.

[145] M. Li, J. Gao, L. Zhao, and X. Shen, "Deep reinforcement learning for collaborative edge computing in vehicular networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 4, pp. 1122–1135, 2020.

[146] 3GPP, "Technical specification group radio access network; Small cell enhancements for E-UTRA and E-UTRAN-physical layer aspects (Release 12)," Tech. Rep. 3GPP TR 36.872 V12.1.0, Dec. 2013.

[147] X. Li, L. Qian, and D. Kataria, "Downlink power control in co-channel macrocell femtocell overlay," in *Proc. IEEE CISS*, Baltimore, MD, USA, 2009.

[148] A. Osseiran, J. Monserrat, and P. Marsch, *5G mobile and wireless communications technology*. Cambridge University Press, 2016.

[149] W. Wang, C. Zhou, H. He, W. Wu, W. Zhuang, and X. Shen, "Cellular traffic load prediction with LSTM and gaussian process regression," in *Proc. IEEE ICC*, Dublin, Ireland, 2020.

[150] Z. Niu, Y. Wu, J. Gong, and Z. Yang, "Cell zooming for cost-efficient green cellular networks," *IEEE Commun. Mag.*, vol. 48, no. 11, pp. 74–79, 2010.

[151] 3GPP, "Technical specification group radio access network; study on Cell-specific Reference Signals (CRS) interference mitigation for homogenous deployments of LTE (Release 12)," Tech. Rep. 3GPP TR 36.363 V12.0.0, Dec. 2013.

[152] B. Korte and J. Vygen, *Combinatorial optimization*. Springer, 2011.

[153] P. Van Laarhoven and E. Aarts, "Simulated annealing," in *Simulated annealing: Theory and applications*. Springer, 1987.

[154] H. Valpola, "From neural PCA to deep unsupervised learning," in *Advances in independent component analysis and learning machines*. Elsevier, 2015.

[155] H. Abdi and L. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[156] 3GPP, "Technical specification group radio access network; 3D channel model for LTE (Release 12)," Tech. Rep. 3GPP TR 36.873 V1.0.0, Sep. 2013.

[157] M. Król, S. Mastorakis, D. Oran, and D. Kutscher, "Compute first networking: Distributed computing meets ICN," in *Proc. ACM ICN*, 2019, Macao SAR, China.

[158] IETF, "Framework of Compute First Networking (CFN)," Tech. Rep. draft-li-rtgwg-cfn-framework-00, 2019.

[159] K. Hwang, Y. Shi, and X. Bai, "Scale-out vs. scale-up techniques for cloud performance and productivity," in *Proc. IEEE CloudCom*, 2014, Singapore.

[160] C. Zhou, J. Gao, M. Li, X. Shen, and W. Zhuang, "Digital twin-empowered network planning for multi-tier computing," *J. Commun. Netw.*, 2022, to be appeared.

[161] M. Dayarathna, Y. Wen, and R. Fan, "Data center energy consumption modeling: A survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 732–794, 2015.

[162] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1619–1632, 2018.

[163] A. Ruiz-Alvarez and M. Humphrey, "A model and decision procedure for data storage in cloud computing," in *Proc. IEEE/ACM CCGRID*, 2012.

[164] J. Park, P. Chou, and J. Hwang, "Rate-utility optimized streaming of volumetric media for augmented reality," *IEEE J. Emerg. Sel. Top. Circuits Syst.*, vol. 9, no. 1, pp. 149–162, 2019.

[165] G. S. Paschos, G. Iosifidis, M. Tao, D. Towsley, and G. Caire, "The role of caching in future communication systems and networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1111–1125, 2018.

[166] J. Dai, Z. Hu, B. Li, J. Liu, and B. Li, "Collaborative hierarchical caching with dynamic request routing for massive content distribution," in *Proc. IEEE INFOCOM*, 2012, Orlando, FL, USA.

[167] K. Poularakis and L. Tassiulas, "On the complexity of optimal content placement in hierarchical caching networks," *IEEE Trans. Commun.*, vol. 64, no. 5, pp. 2092–2103, 2016.

[168] F. Zhang, G. Liu, X. Fu, and R. Yahyapour, "A survey on virtual machine migration: Challenges, techniques, and open issues," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 1206–1243, 2018.

[169] H. Kellerer, U. Pferschy, and D. Pisinger, "Multidimensional knapsack problems." Springer, 2004.

[170] L. Lovász and M. D. Plummer, *Matching theory*. American Mathematical Soc., 2009.

[171] M. Cygan, Ł. Kowalik, and M. Wykurz, "Exponential-time approximation of weighted set cover," *Inf. Process. Lett.*, vol. 109, no. 16, pp. 957–961, 2009.

[172] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, 2002.

[173] G. Koch, R. Zemel, R. Salakhutdinov *et al.*, "Siamese neural networks for one-shot image recognition," in *Proc. ICML*, 2015, Lille, France.

[174] A. Nouruzi, A. Zakeri, M. R. Javan, N. Mokari, R. Hussain, and S. A. Kazmi, "Online service provisioning in NFV-enabled networks using deep reinforcement learning," *IEEE Trans. Netw. Service Manag.*, pp. 1–14, 2022, to be published, doi:10.1109/TNSM.2022.3159670.

[175] F. Wang, J. Xu, and S. Cui, "Optimal energy allocation and task offloading policy for wireless powered mobile edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 19, no. 4, pp. 2443–2459, 2020.

[176] P. Yang, F. Lyu, W. Wu, N. Zhang, L. Yu, and X. Shen, "Edge coordinated query configuration for low-latency and accurate video analytics," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4855–4864, 2020.

[177] Z. Zhou, Q. Wu, and X. Chen, "Online orchestration of cross-edge service function chaining for cost-efficient edge computing," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 8, pp. 1866–1880, 2019.

[178] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.

[179] Y. Huo, X. Fan, L. Ma, X. Cheng, Z. Tian, and D. Chen, "Secure communications in tiered 5G wireless networks with cooperative jamming," *IEEE Trans. Wireless Commun.*, vol. 18, no. 6, pp. 3265–3280, 2019.

[180] F. Lyu, H. Zhu, N. Cheng, H. Zhou, W. Xu, M. Li, and X. Shen, "Characterizing urban vehicle-to-vehicle communications for reliable safety applications," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 6, pp. 2586–2602, 2020.

[181] 3GPP, "Technical specification group radio access network; study on new radio (NR) to support non-terrestrial networks (Release 15)," Tech. Rep. 3GPP TR 38.811 V15.2.0, Oct. 2019.

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

[183] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu, "Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system," *IEEE Trans. Comput.*, vol. 65, no. 12, pp. 3702–3712, 2016.

[184] M. Li, Y. Hong, C. Zeng, Y. Song, and X. Zhang, "Investigation on the UAV-to-satellite optical communication systems," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 2128–2138, 2018.

[185] F. Cheng, S. Zhang, Z. Li, Y. Chen, N. Zhao, F. R. Yu, and V. C. Leung, "UAV trajectory optimization for data offloading at the edge of multiple cells," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 6732–6736, 2018.

[186] Y. Zeng and R. Zhang, "Energy-efficient UAV communication with trajectory optimization," *IEEE Trans. Wireless Commun.*, vol. 16, no. 6, pp. 3747–3760, 2017.

[187] J. L. Hennessy and D. A. Patterson, *Computer architecture: A quantitative approach*. Elsevier, 2011.

[188] Z. Wang, L. Duan, and R. Zhang, "Adaptive deployment for UAV-aided communication networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 9, pp. 4531–4543, 2019.

[189] N. Zhang, H. Liang, N. Cheng, Y. Tang, J. W. Mark, and X. Shen, "Dynamic spectrum access in multi-channel cognitive radio networks," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 11, pp. 2053–2064, 2014.

[190] F. Vatalaro, G. E. Corazza, C. Caini, and C. Ferrarelli, "Analysis of LEO, MEO, and GEO global mobile satellite systems in the presence of interference and fading," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 2, pp. 291–300, 1995.

[191] N. Hosseini, H. Jamal, J. Haque, T. Magesacher, and D. W. Matolak, "UAV command and control, navigation and surveillance: A review of potential 5G and satellite systems," in *IEEE Aerospace Conference*, Big Sky, MT, USA, 2019.

[192] P. Geibel and F. Wysotzki, "Risk-sensitive reinforcement learning applied to control under constraints," *J. Artificial Intelligence Res.*, vol. 24, pp. 81–108, 2005.

[193] L. Xiao, H. Zhang, Y. Xiao, X. Wan, S. Liu, L.-C. Wang, and H. V. Poor, "Reinforcement learning based downlink interference control for ultra-dense small cells," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 423–434, 2020.

[194] S. A. Kanellopoulos, C. I. Kourogiorgas, A. D. Panagopoulos, S. N. Livieratos, and G. E. Chatzarakis, "Channel model for satellite communication links above 10GHz based on Weibull distribution," *IEEE Commun. Lett.*, vol. 18, no. 4, pp. 568–571, 2014.

Appendices

Appendix A

Proof of Theorem 1

For $\forall t \in \mathcal{T}$, we use ϕ^t to represent the energy efficiency of all BSs during time interval t. For simplicity, we define

$$\varsigma_n^t = \sum_{m \in \mathcal{M}} \sum_{i \in \mathcal{I}_{m,n}} \tau \theta_{i,n}^t p_{i,n}^t, \quad n \in \mathcal{N}, \tag{1}$$

and

$$\chi_n^t = \frac{\sum_{m \in \mathcal{M}} \sum_{i \in \mathcal{I}_{m,n}} w_{i,n}^t}{\sum_{m \in \mathcal{M}} \sum_{i \in \mathcal{I}_{m,n}} w_{i,n}^t \eta_n}, \quad n \in \mathcal{N}. \tag{2}$$

The energy efficiency of all BSs during time interval t is given by:

$$\phi^t = \sum_{n \in \mathcal{N}} \frac{\lambda_n \chi_n^t}{\varsigma_n^t}. \tag{3}$$

The Hessian matrix of the energy efficiency ϕ^t can be written as the following block matrix:

$$\nabla^2 \phi^t = \frac{\partial^2 \phi^t}{\partial p_{x,y}^t \partial p_{x',y'}^t} = \begin{bmatrix} \mathbf{A}_1^t & & & \mathbf{0} \\ & \ddots & & \\ & & \mathbf{A}_n^t & \\ & & & \ddots \\ \mathbf{0} & & & \mathbf{A}_N^t \end{bmatrix}_{IN \times IN}, \tag{4}$$

where block $\mathbf{A}_n^t$, $n \in \mathcal{N}$ is given by:

$$\mathbf{A}_n^t = \begin{bmatrix} \frac{2\lambda_n\chi_n^t\tau^2(\theta_{1,n}^t)^2}{(\varsigma_n^t)^3} & \cdots & \frac{2\lambda_n\chi_n^t\tau^2\theta_{1,n}^t\theta_{i,n}^t}{(\varsigma_n^t)^3} & \cdots & \frac{2\lambda_n\chi_n^t\tau^2\theta_{1,n}^t\theta_{I,n}^t}{(\varsigma_n^t)^3} \\ \vdots & \ddots & \vdots & & \vdots \\ \frac{2\lambda_n\chi_n^t\tau^2\theta_{i,n}^t\theta_{1,n}^t}{(\varsigma_n^t)^3} & \cdots & \frac{2\lambda_n\chi_n^t\tau^2(\theta_{i,n}^t)^2}{(\varsigma_n^t)^3} & \cdots & \frac{2\lambda_n\chi_n^t\tau^2\theta_{i,n}^t\theta_{I,n}^t}{(\varsigma_n^t)^3} \\ \vdots & & \vdots & \ddots & \vdots \\ \frac{2\lambda_n\chi_n^t\tau^2\theta_{I,n}^t\theta_{1,1}^t}{(\varsigma_n^t)^3} & \cdots & \frac{2\lambda_n\chi_n^t\tau^2\theta_{I,n}^t\theta_{i,n}^t}{(\varsigma_n^t)^3} & \cdots & \frac{2\lambda_n\chi_n^t\tau^2(\theta_{I,n}^t)^2}{(\varsigma_n^t)^3} \end{bmatrix}_{I\times I}. \tag{5}$$

If constraint (3.10e) is satisfied, ς_n^t is positive. In this case, the first-order leading principal minor of the Hessian matrix, i.e., $\frac{2\lambda_n\chi_n^t\tau^2(\theta_{1,n}^t)^2}{(\varsigma_n^t)^3}$, is nonnegative. Meanwhile, except for the first-order leading principal minor of the Hessian matrix, all leading principal minors is 0. As a result, the Hessian matrix is positive semidefinite when constraint (3.10e) is satisfied. Thus, when $\forall p_{i,n}^t > 0$, the function ϕ^t is convex.

The function ϕ^t increases with the decrease of allocated transmission power for all grids, while the allocated transmission power for all grids should satisfy the SINR constraints in Eq. (3.9). Consequently, due to the convexity of the function ϕ^t, the power control solution must exist on the boundary of the feasible domain. Thus, the optimal power control solution should satisfy Eq. (3.9) with equality, i.e.,

$$\frac{p_{i,n}^t h_{i,n}^t}{N_0 + \sum_{m'\in\mathcal{M}\setminus\{m_{i,n}\}}\sum_{n'\in\mathcal{N}}\sum_{i'\in\mathcal{I}_{m',n'}} b_{i,n,i',n'}^t P_{m',n'}^t h_{m',i'}^t} = \rho\gamma_n^{\min}. \tag{6}$$

Define $\mathbf{\Theta}^t = \mathrm{diag}(h_{1,1}^t, \cdots, h_{i,n}^t, \cdots, h_{I,N}^t)$ and $\mathbf{\Omega}_{n,n'}^t$ in Eq. (3.13). Denote by $\mathbf{p}^t = \left[p_{1,1}^t, \cdots, p_{i,n}^t, \cdots, p_{I,N}^t\right]^\top$ the downlink transmission power of all BSs for all girds during time interval t . Then we can rewrite Eq. (6) into a matrix format, as follows:

$$\mathbf{\Theta}^t \cdot \mathbf{p}^t = \rho \cdot \left(\begin{bmatrix} \gamma_1^{\min}\mathbf{\Omega}_{1,1}^t\mathbf{w}_1^t & \cdots & \gamma_1^{\min}\mathbf{\Omega}_{1,n'}^t\mathbf{w}_n^t & \cdots & \gamma_1^{\min}\mathbf{\Omega}_{1,N}^t\mathbf{w}_N^t \\ \vdots & \ddots & \vdots & & \vdots \\ \gamma_n^{\min}\mathbf{\Omega}_{n,1}^t\mathbf{w}_1^t & \cdots & \gamma_n^{\min}\mathbf{\Omega}_{n,n'}^t\mathbf{w}_n^t & \cdots & \gamma_n^{\min}\mathbf{\Omega}_{n,N}^t\mathbf{w}_N^t \\ \vdots & & \vdots & \ddots & \vdots \\ \gamma_N^{\min}\mathbf{\Omega}_{N,1}^t\mathbf{w}_1^t & \cdots & \gamma_N^{\min}\mathbf{\Omega}_{N,n'}^t\mathbf{w}_n^t & \cdots & \gamma_N^{\min}\mathbf{\Omega}_{N,N}^t\mathbf{w}_N^t \end{bmatrix}_{IN\times IN} \cdot \mathbf{p}^t + \begin{bmatrix} \gamma_1^{\min}N_0 \\ \vdots \\ \gamma_n^{\min}N_0 \\ \vdots \\ \gamma_N^{\min}N_0 \end{bmatrix}_{IN\times 1} \right), \tag{7}$$

Therefore, the closed-form optimal power during time interval t can be derived as Eq. (3.12).

Appendix B

Effective Request Ratio for S-NAPs

For S-BS e located at BS b, S-BS e sorts $|\mathcal{I}_{e,k}|$ chunks with largest values of $p^i_{b,k}$ in time interval k. Let $J^i_{e,k}$ denote the order of chunk i among the chunks with largest values of $p^i_{b,k}$ in set $\mathcal{I}_{e,k}$, and denote by $\mathcal{I}^{(J^i_{e,k})}_{e,k} \subseteq \mathcal{I}_{e,k}$ the set of $J^i_{e,k}$ chunks with largest values of $p^i_{b,k}$. We assume that the computing tasks requesting any chunk in $\mathcal{I}_{e,k}$ are assigned to the S-BS as much as possible while not violating the communication and computing resource capacities. Given different values of $g_{e,k}$ for S-BS e, the load of computing tasks assigned to S-BS e may be different. For S-BS e co-located with BS b, the overall load of computing tasks requiring any chunk in set $\mathcal{I}^{(J^i_{e,k})}_{e,k}$ is given by $\sum_{i\in\mathcal{I}^{(J^i_{e,k})}_{e,k}} \tilde{x}^{(i)}_{b,k}$ where $\tilde{x}^{(i)}_{b,k} = p^i_{b,k}\sum_{n\in\mathcal{N}} x^n_{b,k}$ is the load of computing tasks requiring chunk i in the coverage of BS b in time interval k. The load of computing tasks that request chunk $i \in \mathcal{I}_{e,k}$ in the coverage of BS b and are not assigned to S-BS e in time interval k, denoted by $P^i_{b,e,k}$, is as follows:

$$P^i_{b,e,k} = \begin{cases} 0, \text{ if } \sum_{i\in\mathcal{I}^{(J^i_{e,k})}_{e,k}} \tilde{x}^{(i)}_{b,k} \le M_e; \\ \min\left\{\tilde{x}^{(i)}_{b,k}, \sum_{i\in\mathcal{I}^{(J_{i,e,k})}_{e,k}} \tilde{x}^{(i)}_{b,k} - M_e\right\}, \text{ otherwise.} \end{cases} \tag{8}$$

where $M_e = \lfloor \frac{\tau^{\mathrm{p}} C_e}{\alpha\beta} \rfloor$ is the maximum load of computing tasks that can be assigned to S-BS e with satisfying the computing capacity. In Eq. (8), if S-BS e has sufficient computing resource for executing all computing tasks requiring chunk i, i.e., $\sum_{\mathcal{I}^{(J^i_{e,k})}_{e,k}} \tilde{x}^{(i)}_{b,k} \le M_e$, no computing task requiring chunk i needs to be assigned to an S-NAP or the S-CN; Otherwise, a certain load of computing tasks requiring chunk i cannot be processed at S-BS e, which should be assigned to an S-NAP or the S-CN.

The overall load of computing tasks that request chunk i and are not assigned to any S-BS e within the service coverage of S-NAP e' in time interval k is given by $\sum_{b\in\mathcal{B}_{e'}} P^i_{b,e,k}$. The effective request ratio of chunk i for S-NAP e' in time interval k is as follows:

$$q^i_{e',k} = \frac{\sum_{b\in\mathcal{B}_{e'}} P^i_{b,e,k}}{\sum_{b\in\mathcal{B}_{e'}}\sum_{\mathcal{N}} \tilde{x}^n_{b,k}}, \ \forall i \in \mathcal{I}, e' \in \mathcal{E}^{\mathrm{nap}}, k \in \mathcal{K}. \tag{9}$$

S-NAP e' stores $|\mathcal{I}_{e',k}|$ chunks with the largest values of $q^i_{e',k}$.