Task Offloading and Resource Allocation for 5G and Beyond Networks

YUE CAI

Doctor of Philosophy (Engineering)

Supervisor: Prof.Yonghui Li
Associate Supervisor: Dr.Peng Cheng, Prof.Branka Vucetic

A thesis submitted in fulfilment of
the requirements for the degree of
Doctor of Philosophy

School of Electrical & Information Engineering
Faculty of Engineering
The University of Sydney
Australia

9 March 2025

Statement of Originality and Scholarship

This is to certify that to the best of my knowledge, the content of this thesis is my own work. This thesis has not been submitted for any degree or other purposes.

I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.

I acknowledge that I received the Postgraduate Research Scholarship in Deep Learning-Based Low Earth Orbit (LEO) Satellites to cover my fees from 1st July 2022. Additionally, the research reported in this thesis was supported by the award of a Research Training Program scholarship.

This work has been supported by the SmartSat CRC, whose activities are funded by the Australian Government's CRC Program.

Signature:

Authorship Attribution Statement

The thesis contains material that I have previously published during my PhD study.

Chapter 3 of this thesis is published as

- **Y. Cai**, P. Cheng, Z. Chen, M. Ding, B. Vucetic and Y. Li, "Deep Reinforcement Learning for Online Resource Allocation in Network Slicing," in IEEE Transactions on Mobile Computing, vol. 23, no. 6, pp. 7099-7116, June 2024, doi: 10.1109/TMC.2023.3328950.
- **Y. Cai**, P. Cheng, Z. Chen, W. Xiang, B. Vucetic and Y. Li, "Dynamic Resource Allocation in Network Slicing with Deep Reinforcement Learning," GLOBECOM 2023 - 2023 IEEE Global Communications Conference, Kuala Lumpur, Malaysia, 2023, pp. 2955-2960, doi: 10.1109/GLOBECOM54140.2023.10437340.

Chapter 4 of this thesis is published as

- **Y. Cai**, P. Cheng, Z. Chen, W. Xiang, B. Vucetic and Y. Li, "Graphic Deep Reinforcement Learning for Dynamic Resource Allocation in Space-Air-Ground Integrated Networks," in IEEE Journal on Selected Areas in Communications, doi: 10.1109/JSAC.2024.3460086.

Chapter 5 of this thesis is submitted to IEEE/AMC Transaction on Networking.

As the only first author, I have made a substantial contribution to the above journal papers including but not limited to designing the study, deriving mathematical equations, programming and modelling, analysing the data and writing the drafts of the manuscripts. In addition to the statements above, in cases where I am not the corresponding author of a published item, permission to include the published material has been granted.

Signature:

As supervisor for the candidature upon which this thesis is based, I can confirm that the authorship attribution statements above are correct.

Signature:

To my loving husband Xuan Jiao
and
my beloved parents Yuxia Wang and Chunxiang Cai

Acknowledgements

I would like to express my great thanks to the helpful people around me. Without you, it is impossible for me to complete this thesis.

First of all, I would like to express my great thanks to my husband, Xuan Jiao. Thank you for always being there with me, encouraging me to face challenges, and providing me with solid support. It is you who spend countless days listening to my complaints, comforting me when I am upset, and providing me with the exact help I need. I cannot image life without you, and I am so honored to have you as my husband. I would also like to express my gratitude to my parents and parents-in-law. Without you, I will not be able to chase my dream. My mother, Yuxia Wang, thank you for being there with me when I am sad and listing to all my complaints. My father, Chunxiang Cai, thank you for your encouragement, which really eased the pressure.

I would like to express my huge thanks to my supervisor, Professor Yonghui Li, from the University of Sydney. You lead me to the academic world, provide support for me to achieve my goals, and guide me on the way to conquer the challenges encountered in my research. I would also like to express my gratitude to Professor Branka Vucetic from the University of Sydney. Thank you for your help and encouragement.

Furthermore, I would like to express my great gratitude to Dr.Peng Cheng from both La Trobe University and the University of Sydney. Thank you so much for being very patient and helpful. You spent countless days helping me solve problems encountered in my research, leading me the right way, listening to my concerns, and answering my questions. Without you, it would be impossible for me to finish my thesis. A special thanks to Dr.Zhou Chen from Data61, CSIRO. Thank you so much for your help in writing all the works and guiding me the right way to do research.

I would like to express my thanks to the University of Sydney for providing me with the opportunity to pursue a Doctorate degree and Smartsat CRC for providing me with the scholarship to focus on my studies without the need to worry about money.

Thank you all once again.

Abstract

The increasing number of users and service types have introduced challenges to existing communication networks. Specifically, more efficient resource allocation schemes are needed for 5G and beyond networks to meet the service level agreements (SLA) of each user and the quality of service (QoS) requirements for each service type. Furthermore, the integration of space and air communication segments into conventional terrestrial communication networks introduces rapid environment changes, necessitating low-complexity online resource allocation decision-making. This further complicates the design of resource allocation schemes.

In this thesis, we give a detailed analysis of the challenges encountered in the 5G and beyond networks in the field of resource allocation. To start with, we provide an introduction to the development of communication technologies and highlight the distinct features of 5G and beyond networks. This serves the purpose of providing the readers with a more general view of the cause behind the challenges encountered in current networks and understanding the importance of addressing these challenges.

To solve these challenges, we proposed three schemes in this thesis, with one focusing on the resource allocation in 5G radio access network (RAN) slicing and two on task offloading and resource allocation in space-air-ground integrated networks (SAGIN). Specifically, these schemes are proposed to satisfy the following requirements under different environment settings: low complexity, online decision-making ability, strong scalability, strong generalization ability, and ease of implementation. To achieve these goals, all schemes are formulated into a block structure where each block accomplishes a dedicated task. The design of each block depends on the function to be accomplished, and both optimization-based approaches and DRL-based approaches are utilized to formulate a block. Each block can be easily implemented into other schemes to achieve the same function as in our work, which greatly improves the generalization ability of our proposed schemes. Additionally, this block structure enables

the scheme to have each block designed separately. Several blocks can also be trained in parallel to reduce the complexity of the scheme.

In summary, this thesis provides an end-to-end review of the current challenges and achievements in communication networks. It begins by introducing the problems faced, the causes behind these issues, and the proposed solutions. Simulation results are presented to validate the performance of the proposed schemes, and a conclusion chapter highlights the key achievements in the field of resource allocation for 5G and beyond networks, along with potential directions for future research.

LIST OF ABBREVIATIONS

QOS: Quality of Service

SLA: Service Level Agreements

RAN: Radio Access Network

SAGIN: Space-Air-Ground Integrated Network

API: Application Programming Interfaces

SDN: Software Defined Network

NFV: Network Function Virtualization

VNF: Virtual Network Functions

NMS: Network Management Segments

TN: Transport Network

CN: Core Network

RB: Resource Blocks

MEC: Mobile Edge Computing

LEO: Low-Earth Orbit

UAV: Unmanned Aerial Vehicle

HAPS: High-Altitude Platform Station

MINLP: Mixed Integer Nonlinear Programming Problems

DL: Deep Learning

DRL: Deep Reinforcement Learning

TO: Throughput Oriented

DS: Delay Sensitive

DTT: Delay Throughput Tolerant

PW-DRL: Prediction-aided Weighted DRL

GDRL: Graphic DRL

UE: User Equipment

GFEN: Grap Feature Extraction Network

AEGN: Action Encoding and Generation Network

GF-DRL: Graph Fusion DRL

MLP: Multilayer Perception

GNN: Graph Neural Network

AMN: Action Mapping Network

POTS: Plain Old Telephone Services

FDMA: Frequency Division Multiple Access

RRM: Radio Resource Management

GSM: Global System for Mobile Communications

SIM: Subscriber Identity Module

TDMA: Time Division Multiple Access

GPRS: GSM General Paket Radio Service

UL: Uplink

DL: Downlink

FR: Full Rate

HR: Half Rate

EFR: Enhanced Full Rate

NSS: Network Switching Subsystem

BS: Base Station

BSS: Base Station Subsystem

NE: Network Element

BTS: Base Transceiver Station

BSC: Base Station Controller

MSC: Mobile Service Switching Center

MGW: Mobile Gateway

HLR: Home Location Register

VLR: Visitor Location Register

AC: Authentication Center

EIR: Equipment Identity Register

GGSN: Gateway GPRS Support Node

SGSN: Serving GPRS Support Node

3GPP: 3rd Generation Partnership Project

UMTS: Universal Mobile Telecommunication Network

W-CDMA: Wideband Code Division Multiple Access

CS: Circuit Switched

PS: Packet Switched

PDN: Packet Data Networks

PSTN: Public Switched Telephone Network

ITU: International Telecommunication Union

LTE: Long-Term Evolution

VoIP: Voice over IP

EPC: Evolved Packet Core

SAE: System Architecture Evolution

EPS: Evolved Packet System

OFDMA: Orthogonal Frequency-Division Multiple Access

OFDM: Orthogonal Frequency Division Multiplexing

SGW: Serving Gateway

MME: Mobile Management Entity

HSS: Home Subscriber Server

PGW: Package Data Network Gateway

PRB: Physical Resource Block

MBMA: Multimedia Broadcast and Multicast Service

DQN: Deep Q-learning Network

eMBB: Enhanced Mobile Broadband

DDoS: Distributed Denial-of-Service

MAMO: Management and Orchestration

O-RAN: Open RAN

VM: Virtual Machine

EC: Edge Cloud

DDQN: Deep Distributional Q-Network

A2C: Twin Actor Critic

LSTM: Long-Short-Term Memory

SE: Spectrum Efficiency

IoT: Internet of Thing

AO: Alternating Optimization

SAC: Soft Actor-Critic

PPO: Proximal Policy Optimization

DAG: Directed Adjacency Graph

GEO: Geostationary Earth Orbit

DDPG: Deep Deterministic Policy Gradient

QoE: Quality of Experience

TRPO: Trust Region Policy Optimization

MMPP: Markov Modulated Poisson Porcess

CNN: Convolutional Neural Network

MAC: Medium Access Control

ST: Short Timescale

LT: Long Timescale

ISL: Inter-Satellite Link

IHL: Inter-HAPS Link

LAGN: Latent Action Generation Network

MDP: Markov Decision Process

AMN: Action Mapping Network

MAML: Model Agnostic Meta Learning

:

List of Figures

List of Tables

Contents

CHAPTER 1

Introduction

The success of current 5G and beyond networks relies on efficient task offloading/resource allocation schemes. In this thesis, a comprehensive analysis of state-of-the-art task offloading/resource allocation schemes is presented. To mitigate the limitations of these schemes, three works are presented. The first one focuses on resource allocation in the 5G radio access network (RAN) slicing, which is a crucial part for the successful establishment of network slices. The second and the third works focus on task offloading/resource allocation in the Space-Air-Ground integrated network (SAGIN), which is the key solution for 6G to achieve global coverage. Each work is presented in detail, with its contributions highlighted at the beginning of chapters 4, 5, and 6. In this chapter, we first elaborate on the background of our research. Then, we introduce the driving factors behind our choice of this research direction, followed by the challenges faced by each of our works and their innovations. We also give a detailed discussion of the correlations and differences among our works, highlighting their impacts in this field of study. Finally, an outline that summarizes the content of the subsequent chapters is given.

1.1 Background

Starting from the first generation (1G), which is an analog communication system in the early 1980s, to the recently deployed 5G digital communication system, each generation deploys a larger bandwidth, higher frequency range, and higher transmission rate compared to the previous one [1]. Nowadays, 5G utilizes millimeter wave (mmWave) bands and can achieve a peak transmission rate of 20Gbps [2]. With this improvement and support of

other technologies, 5G can embrace a broader range of services like autonomous driving, 4k streaming, remote surgery, and massive IoT compared with 4G [3]. However, the requirements of some services are beyond the capabilities of 5G. For example, 5G cannot support global coverage and thus cannot provide seamless wireless access services, especially for rural areas, remote oceans, and mountains [4]. Furthermore, during disasters like volcanic eruptions and tornadoes, terrestrial BS can be easily destroyed, causing wireless network service disruptions, which is vital for efficient emergency and disaster management [5]. These requirements are to be explored and resolved in the 6G and the SAGIN is considered the key enabler [6]. In the following, we elaborate on the features of 5G and SAGIN.

1.1.1 5G Terrestrial Network

5G, deployed in 2019, is a service-oriented network that is designed to support different service types with distinct characteristics and quality of service (QoS) requirements [7]. It incorporates application programming interfaces (APIs) instead of predefined interfaces, which enhance system flexibility [8]. For example, network providers can easily create and deploy APIs to suit the need for new service types. Similar to 4G, 5G also separates the user plane and control plane, enabling the separation of scaling and centralized/distributed deployment [9].

The key enabler of 5G is network slicing, which takes advantage of various technologies like software defined network (SDN) and network function virtualization (NFV) [10, 11]. Each active network slice can manage its own resources, create and terminate virtual network functions (VNF), and report its status to the network management segments (NMS)[12]. It is designed to address the challenge of simultaneously satisfying the QoS requirements of different types of services [12]. The idea behind network slicing is to create multiple virtual network slices on a shared physical infrastructure where each network slice is designed to satisfy the QoS requirement of one type of service [13, 14]. Network slicing consists of three parts: the RAN slicing, the transport network (TN) slicing, and the core network (CN) slicing [15]. The RAN part performs the function of connecting user mobile devices to the network, allocating resources based on user requests, mapping virtualized resource blocks (RB) to

different network slices as well as managing and reporting physical link status [16, 17]. It is a very crucial part of network slicing as it determines how many RBs can each request utilize and directly influences the final QoS [18].

However, there is no such thing as a free lunch. Although network slicing is promising, it also brings a lot of challenges. We will elaborate on the challenges in two aspects: Resources and complexity.

Resources: To successfully establish a network slice, resources like transmission power, and bandwidth need to be provisioned [19]. In some networks that incorporate MEC, CPU frequency also needs to be provisioned for each slice [20]. As lots of services exist in the network at the same time, sufficient resource supply is needed to satisfy QoS requirements of all these services [21]. However, available bandwidth in 5G is very scarce with most of them already pre-allocated to other services. Additionally, more resources will inevitably induce higher costs [22, 17]. Therefore, it is not possible for a network to have sufficient resources. As the allocation of resources to one network slice will influence the available resources for others, the problem lies in how to efficiently allocate resources under limited resources [23].

Complexity: From the complexity perspective, algorithms used to establish network slices need to have low computational complexity [24]. This is because 5G encompasses services with stringent latency requirements, and the time available for establishing a network slice is very limited. However, the resource allocation problem is usually non-convex and np-hard which makes it hard to find a low complexity algorithm [25]. Furthermore, as the network environment changes rapidly, the algorithm should be able to adapt to this environment and have strong generalization ability [26, 27].

1.1.2 Space-Air-Ground Integrated Network

Satellite technology has been recognized as a powerful solution for achieving seamless global connection and satisfying the high availability requirement of telecommunication systems in disasters [28]. In recent years, the improved onboard processing capabilities of satellites have enabled their role transformation from mere transport relays that provide connections

for remote areas to in-orbit computing platforms [29]. Specifically, the integration of MEC with satellite networks prompts this shift of roles. This shift is also driven by the increasingly limited terrestrial computing and frequency band resources caused by an increasing number of users [30, 31]. With the number of satellites, especially low-earth orbit (LEO) satellites exploding, the computing capability of satellite networks will increase significantly in the future [32, 33, 34]. Therefore, the cooperation between satellite and terrestrial networks is now recognized as an attractive, cost-efficient way to improve the user experience and allow computing service to more users [35].

To address the cooperation, a framework called SAGIN is proposed and has been researched in many works [36]. The space segment is usually established by LEO satellites that are distributed at an altitude of 300-1500 km and interconnected by inter-satellite links (ISLs) [37]. The air segment can be established by unmanned aerial vehicles (UAVs) or high-altitude platform stations (HAPSs), which can also be equipped with MEC servers and storage hardware to perform computing-related tasks [38]. This SAGIN, by comprehensively considering the connections between ground, air, and space, can effectively utilize all available resources simultaneously, but it also introduces many challenges [39, 40]. Next, we will elaborate on the challenges in two aspects: Resources and complexity.

Resources: The resource distribution of SAGIN is very different from that of 5G [41]. To begin with, SAGIN consists of three segments-space, air, and ground segments-hence, available resources (RBs, frequency) are distributed in a three-dimensional domain, with each layer having different available resources [42]. Additionally, different from the 5G terrestrial network, the positions of space and air segments in SAGIN change rapidly, which results in varying network connection status and available resources [43]. When a task arrives, two steps need to be performed for task processing: task offloading and resource allocation [44]. Task offloading refers to the transfer of computing tasks from resource-constrained nodes to more capable ones within the network, such as from terrestrial devices to satellite or aerial platforms[36]. Resource allocation refers to the allocation of computational and frequency resources for the processing and transmission of tasks [45]. Efficient task offloading and resource allocation is crucial for the integration of space, air, and ground segments and is

critical for satisfying the service level agreement (SLA) [46]. However, both steps are hard to perform due to the complex interconnection between segments and the rapidly changing network environment.

Complexity: As SAGIN consists of a large number of segments and chain dynamically, some of the most critical problems are the complexity of performing resource allocation under varying environments and the difficulty in providing task offloading decisions due to large numbers of LEOs and UAVs or HAPSs [47]. As the network connection status changes from time to time, low-complexity task offloading and resource allocation strategies are crucial for making timely decisions [48]. However, the task offloading and resource allocation problems in SAGIN are usually mixed integer nonlinear programming problems (MINLP), which prevents the implementation of low-complexity algorithms.

1.2 Research Problems

In the previous section, we provided a detailed description of the 5G terrestrial network and SAGIN and evaluated the challenges from the perspective of resources and complexity. Specifically, the effectiveness of both 5G and SAGIN are constrained by resource allocation strategies. Furthermore, SAGIN requires a highly efficient and low-complexity task offloading scheme before performing resource allocation. Recently, the advancement in machine learning, including deep learning (DL) and deep reinforcement learning (DRL), has enabled the implementation of learning-based methods in solving problems encountered in communication networks. These learning-based methods are known for their ability to accommodate dynamically changing network environments with low complexity. When considering integrating learning-based approaches into 5G and SAGIN, a key question emerges: how can we leverage these methods to solve the resource allocation problem in 5G and the task offloading and resource allocation problem in SAGIN? This motivates us to dive into various task offloading and resource allocation problems and strategies. In the following, we elaborate on these research problems. The first research problem (Chapter 3) lies in the field of resource

allocation in 5G terrestrial networks. For recent works, the learning-based network slicing resource allocation schemes appear to suffer from one or more of the following deficiencies:

(1) The generation process for user requests is simplified into the Poisson process and provides no information to guide the design of the system.
(2) Joint power and RB allocation problem with constraints is a non-convex optimization problem and is proved to be NP-hard. Current existing algorithms cannot deal with long-term constraints, and long-term system stability is not guaranteed.
(3) Differences among services are usually not considered. Differences, including timescale difference, arrival process difference, and the underlying feature difference, are usually ignored to simplify the resource allocation problem.

Consequently, in Chapter 3, we try to mitigate these deficiencies and propose a novel resource allocation scheme for 5G.

The second research problem (Chapters 4 and 5) lies in the field of task offloading resource allocation in SAGIN. For recent works, the learning-based task of offloading resource allocation schemes has one or more of the following limitations.

(1) These schemes cannot handle both discrete and continuous decisions in an end-to-end manner. Original problems need to be separated into two subproblems and relaxed into convex, which leads to potential performance loss.
(2) These schemes suffer from inference performance degradation when the model mismatch between training and testing environments exists. As the variation of critical system parameters characterizes the SAGIN environment, it is highly likely that the testing and training environments differ from each other.
(3) These schemes do not take into account the inherent topological structures of the SAGIN environment. Loss of information will inevitably lead to slow convergence to suboptimal solutions and performance degradation.
(4) These schemes do not consider the correlations and graphical structure of each task, limiting the covered service types.

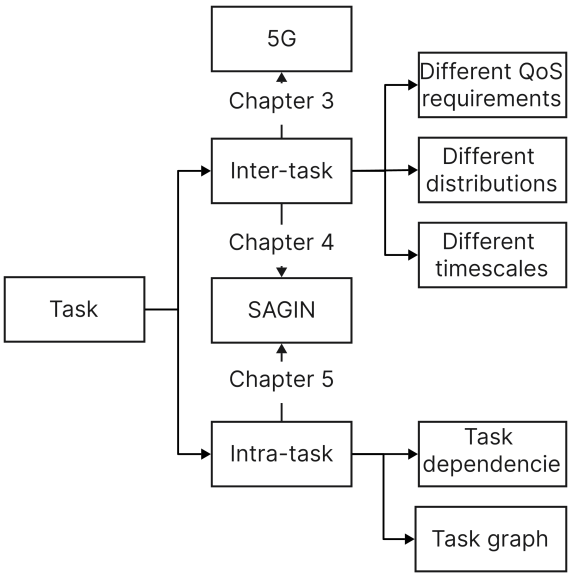


FIGURE 1.1. Differences and correlations among works.

(5) These schemes cannot handle varying numbers of LEO/HAPS and need to be retrained with training high complexity when the number changes.

Consequently, in Chapter 4, we focus on resolving points 1, 2, and 3, and in Chapter 5, points 1, 2, 4, and 5 are addressed.

Next, we elaborate on the differences and correlations among all these works, which is shown Fig. 1.1. Specifically, Chapters 3 and 4 focus on inter-task features, while Chapter 5 focuses on intra-task features. Inter-task features refer to the distinct features of different tasks and users. Thus, Chapters 3 and 4 focus on satisfying the QoS requirements for each user, adopting different distributions for each type of task, and having two timescales to reflect the distinct task features. Intra-task features refer to the correlations among subtasks inside each task. Thus, Chapter 5 focuses on the formulation of task dependencies and utilizing task graphs to model these dependencies. Additionally, Chapter 3 focuses on the resource allocation in the 5G network, while Chapters 4 and 5 on both task offloading and resource allocation in SAGIN. Table 1.1 summarizes the key points for each work.

TABLE 1.1. Summary of Key Points for Each Research Work

	Chapter 3	Chapter 4	chapter 5
Scope	5G	SAGIN	SAGIN
Task dependency	No	No	Yes
Future traffic prediction	Yes	Yes	No
Task offloading	No	Yes	Yes
Resource allocation	Yes	Yes	Yes
Two timescales	Yes	Yes	No
MIMO	No	Yes	No
Problem	Sequential decision-making	Sequential decision-making	Sequential decision-making
Method	DRL+Lyapunov	DRL	DRL
Constraint	Long-term and short-term	Short-term	Short-term
Hybrid action space	Yes	Yes	Yes
Graph	No	Yes	Yes
Graph fusion	No	No	Yes

1.3 Contributions

Work 1 is proposed to address the lack of focus on the time-sequential dynamic resource allocation in 5G RAN slicing and mitigate the limitations of state-of-the-art research. In Chapter 4, we design a novel time-sequential decision-making resource allocation framework tailored for 5G RAN slicing with two distinct features. Firstly, we consider a realistic setting where the arrival of user requests is random, and each request can occupy the allocated resources for a random period of time. Thus, the total available resources change dynamically over time, which results in complicated correlations between resource allocation actions. Secondly, in this framework, the online real-time generation of these correlated decisions is made with low complexity, considering both short-term and long-term system constraints. We formulate the resource allocation problem as a time-sequential decision-making problem, taking into account user priority, system limitations, and system stability. To obtain an efficient resource allocation scheme, we utilize DRL to directly learn the mapping between environment states and resource allocation actions. Our contributions are listed as follows.

(1) Our work encompasses three types of services: one referred to as throughput-oriented (TO), which has a high transmission rate requirement; one is delay-sensitive (DS), which has a stringent latency requirement; and one is delay-throughput-tolerant (DTT), which has a moderate transmission rate and relaxed latency requirement. Compared to the 5G RAN slicing models, the main features of this RAN slicing

include: 1) Each user request in TO, DTT, and DS slice follows a distinct distribution, and a wide range of distribution types are considered. 2) This model utilizes different priority values for different traffic types, where each type belongs to one slice. By addressing the priority of different slices, our proposed method can prioritize critical traffic under resource-stringent circumstances.

(2) Our work considers a broad range of factors to formulate the resource allocation problem. Specifically, the constraints for total available resources, the scalability of the system, two types of timescales, the priority for different users, and the possible fluctuation of the system (system instability). Specifically, a long-term constraint is applied to account for long-term system performance, a short-term constraint to address resource limitations, and virtual queues to account for system stability.

(3) The prediction-aided weighted DRL (PW-DRL) approach is proposed to solve the aforementioned problem by leveraging DRL and Lyapunov optimization. Furthermore, a prediction network, as part of PW-DRL, is proposed to extract features related to the dependence between current and future states.

The second work is proposed to address the following problems. Firstly, the SAGIN model considered in state-of-the-art research only considers a single timescale for all services, single antenna configurations, and a common channel model for all three segments. However, different types of services require varying timescales, and the distinct features of each segment should be explored. Secondly, these methods cannot effectively capture the dependencies within environmental state data featured by topological structures. This leads to slow convergence and performance degradation in terms of the reward function. Thirdly, these DRL-based approaches generate task offloading (discrete) and resource allocation decisions (continuous) separately. The original optimization problems are typically divided into two subproblems: one solved by DRL-based methods and the other by heuristic or optimization-based methods. This division increases the overall computational complexity and results in potential performance loss. Lastly, these approaches assume minor differences between deployment and training environments. In SAGIN, however, critical parameters like initial total resources may drastically change due to LEO and aerial platform movements, resulting in inference performance degradation. In Chapter 5, we focus on finding an efficient online low-cost

task offloading and resource allocation scheme in SAGIN, with the goal of minimizing the latency experienced by all users. However, performing efficient task offloading and resource allocation is a challenging task. This is because SAGIN functioning in a dynamic environment involves changing features such as channel states, LEO and aerial platform locations, and total available resources. This requires highly adaptive and nearly real-time resource allocation approaches to capture and closely follow the variations of these features. The main contributions of our work are listed as follows:

(1) We propose a SAGIN model that can more accurately reflect the real environment compared with other SAGIN models. It encompasses a wider range of services, each with a diverse priority, different antenna configuration schemes, different time lengths, and even different transmission models.
(2) We propose an encoding scheme to efficiently generate offloading decisions. This scheme can both reduce the dimension of the original offloading decisions and avoid the generation of non-practical ones. Additionally, with the implementation of this encoding scheme, a separate network can be formulated to generate decisions directly, which can significantly reduce the overall computational complexity of solving the proposed resource allocation problem. Furthermore, this scheme has strong generalization ability and can be easily applied to other resource allocation problems with the same benefits.
(3) We propose a graphic DRL (GDRL) approach that encompasses many new parts, each with a diverse functionality. Specifically, a state feature extraction network to extract features from different types of environment states, a latent action generation network to generate latent actions from the extracted state features, and an action mapping network that leverages the encoding scheme to generate resource allocation and offloading decisions directly. Each part features a modified encoder-decoder structure and can be easily adapted to other networks to leverage their advantages.
(4) We leverage meta-learning methods MAML and modify TRPO to construct a meta-learning-enabled TRPO. This modification enables the fast adaption of TRPO to a new environment encountered with only a few steps of gradient updates. Additionally,

it mitigates the slow retraining problem of DRL-based methods and can boost the performance of GDRL with low complexity.

Work 3 is proposed to mitigate the lack of focus in the time-sequential decision-making formula for SAGIN offloading/allocation. Specifically, many approaches have been proposed for SAGIN task offloading and resource allocation, which, in most cases, have been formulated as a one-shot non-convex optimization problem. This formulation focuses on optimizing a predesigned objective function, such as minimizing latency or maximizing throughput, within the constraints of available resources for each problem instance. There is no correlation and time-dependency between instances. It is worth noting that the one-shot optimization aforementioned only considers a static resource pool, ignoring how offloading/allocation decisions affect available resources across different problem instances. Consequently, it fails to capture the inherent dependencies over multiple timesteps and ignores long-term performance metrics like task completion ratio and resource utilization ratio. Additionally, the static offloading/allocation fails to account for dynamic correlations between tasks, where the execution of current tasks influences the available resources, and, in turn, affects the offloading and allocation decisions for future tasks. In Chapter 6, we focus on finding an efficient task offloading and resource allocation scheme that can maximize the number of completed tasks. Different from the second problem, this one is formulated considering the topological structures of the SAGIN environment and tasks. Furthermore, we consider task dependencies where each task consists of several subtasks, and the process of one subtask requires the result of another. Our contributions are listed as follows:

(1) We propose a dynamic SAGIN model that takes into account the correlation among subtasks and changes in user equipment (UE)/task graph structure. This can reflect a more realistic scenario and extend the covered service types.
(2) We propose a graph feature extraction network (GFEN) that incorporates hard and soft attention mechanisms. It can merge the UE graph and task graph and adapt to graph topology changes, which enhances DRL's feature extraction ability for graph-structured states.

(3) We propose an action encoding and generation network (AEGN) to enable the end-to-end generation of both task offloading and resource allocation decisions. It can significantly reduce the size of the mixed action space and provide fast online decisions.

(4) We leverage the reincarnating technique and propose a reincarnating graph fusion DRL (GF-DRL) to adapt to the changing number of UE/LEO/HAPS with low retraining complexity. We also theoretically proved that our proposed GF-DRL performs better than multilayer perception (MLP)-based DRL methods.

1.4 Thesis Outline

The remainder of this paper is formulated as follows.

Chapter 2 presents the development of communication networks and the importance of resource allocation in each generation.

Chapter 3 presents a literature review on resource allocation in 5G terrestrial networks and task offloading resource allocation in SAGIN. We explore a wide range of studies and highlight their limitations.

Chapter 4 elaborates on the resource allocation based on PW-DRL and Lyapunov optimization to solve the time sequential resource allocation problem in 5G. Specifically, both long-term and short-term constraints are considered to solve this problem, and services are categorized into three types, with each having distinct QoS requirements.

Chapter 5 focuses on the task offloading and resource allocation problem in SAGIN. Specifically, we proposed an online dynamic resource allocation method called GDRL. A graph neural network (GNN) based feature extraction network is proposed to capture features of both graph-structured and non-graph-structured states, and an autoencoder structure-based action mapping network (AMN) is proposed to enable end-to-end generation of both discrete and continuous actions.

Chapter 6 elaborates on the offloading and resource allocation problem in SAGIN when considering task dependencies. Specifically, we design a new low-complexity online scheme referred to as GF-DRL. Attention mechanisms are deployed to merge the state graph and task graph together, GNN is utilized to extract features from graph-structured data, and an action encoding and generation network is proposed to generate final offloading and resource allocation decisions.

Chapter 7 summarizes the contributions of this thesis and elaborates on possible future works.

CHAPTER 2

History

With the increasing number of users and communication applications, communication technologies have evolved from generation to generation [49]. In the following, each communication generation is described in detail.

2.1 The First Generation

In 1946, the United States Federal Communications Commission granted approval for AT&T to operate the first commercially available carborne telephony service. During the 1950s and 1960s, similar systems were proposed and implemented by various telephone administrations. These systems served as the prototypes for the first-generation communication system.

Then, in the early 1980s, due to the increasing number of subscribers, an urgent need for a communication network that can satisfy the mobile communication requirements emerged and attracted several interested parties. There are many international mobile communication systems at that time, and the best-known ones are NMT in Nordic countries, AMPS in North America, TACS in Europe, and J-TACS in Japan. These systems were entirely analog-based and designed to support plain old telephone services (POTS), which only provided voice communication along with basic supplementary services. It can be seen that for the first generation, each country has its own communication system, and global communication is not possible. It can only achieve a low data rate of around 2.4 kbps.

Different from other generations, 1G utilizes analog signals to transmit voice data, and frequency division multiple access (FDMA) is used for user access control. In 1G, the concept

of resource allocation is very straightforward. It relies on static radio resource management (RRM) methods, where all resources need to be planned manually. The main resources considered in 1G are as follows.

(1) Frequency channels: utilizing FDMA, the frequency band is divided into multiple discrete frequency channels. Each frequency channel can be used by one user to transmit voice data. In 1G, the frequency channels allocated to one user cannot be released until the call between two users ends.
(2) Cell planning: Cell planning can be treated as geographical resource allocation. In 1G, network operators need to determine the cell size, the cell place, the antenna orientations, and the transmission power setting of that cell in advance before constructing each cell. To avoid interference between users, frequency reuse patterns need to be planned for all cells, where the frequency channels allocated to one cell do not overlap with those of its neighboring cells.

In 1G, all resources need to be planned, and no dynamic scheduling/admission is allowed. The decisions of how frequency channels are allocated, how each user can access the network, and how cells are deployed were finalized during the network design phase. Thus, the resource allocation stays static over the network's operational lifetime, and changes can only be made by designing and deploying a new network, which is of high cost.

2.2 The Second Generation

The second generation (2G) is developed based on the global system for mobile communications (GSM), which is the first communication system that archives global connectivity. The GSM originally referred to group special mobile, which was proposed as a new communication standard for Europe only. However, the success of this standard globally requires the name to be changed to suit its deployment scope from Europe to global. The success of GSM is based on the following characteristics.

(1) It is an open standard with the ability for anyone in the world to access it directly.

(2) It has multiple standardized interfaces which allow vendors to collaborate with each other.
(3) It enables global connection with the consideration of roaming in the design of its network architecture and communication procedures.
(4) It supports encrypted transmission, which can significantly improve the performance of communication networks to protect private information compared with 1G.
(5) It integrates subscriber control by introducing the subscriber identity module (SIM).
(6) It implements the technology of frequency division multiple access (FDMA) and time division multiple access (TDMA) to utilize available frequency and time resources efficiently.
(7) It requires lower transmission power of the mobile phone, which enables the utilization of small mobiles due to smaller battery size.
(8) Compared with 1G, it utilizes digital transmission, which brings the benefits of high speech quality and enables possible signaling for different services.
(9) It first introduces the concept of an upgradable system and downward compatibility, which maximizes its flexibility and generalization ability.

To enable global access, the first step is to define a standardized air interface. In GSM, this air interface is referred to as the Um interface (unlimited mobility), where both GSM and the later extension of GSM general packet radio service (GPRS) share the same interface. To effectively utilize the frequency and time resources, frequency blocks assigned to uplink (UL) and downlink (DL) are divided into frequency bands with a bandwidth of 200 kHz. Additionally, each frequency band is further divided into eight channels referred to as time slots, which enables a maximum of eight users to transmit using the same frequency band. This separation of transmission in time while using the same band is called TDMA, and the eight timeslots formulate a TDMA frame.

After the frequency band and UL/DL timeslot number are allocated to one mobile station (MS), it can start to transmit speech and control signals. This process is called traffic channel (TCH) establishment. Before transmitting, the analog speech signal needs first to be digitized and coded to suit the channel conditions. There are three commonly used coding schemes

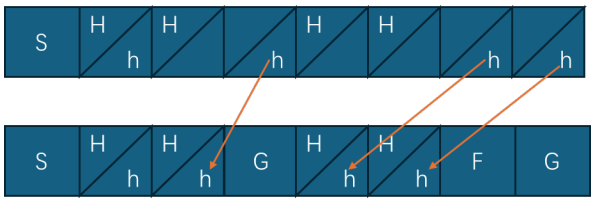


FIGURE 2.1. HR pairing procedure.

for GSM: full rate (FR), half rate (HR), and enhanced full rate (EFR). Specifically, FR and EFR transmit one timeslot after another, while HR transmits once for two timeslots. Thus, HR can enable the transmission of two MS for one channel. For HR, in order to use time resources efficiently, GSM implements a technology referred to as HR pairing, where two unpaired HR users will be forced to move to one channel. This can guarantee a minimum number of timeslots are used; thus, other types like FR and GSMR are not blocked due to HR. Fig. 2.1 shows this HR pairing procedure. It can be seen that after using HR paring, timeslots 4, 7, and 8 can now be utilized by GSMR and FR transmission.

The GSM was first proposed in 1982 at the European Conference of Postal and Telecommunications Administrations. After that, Phase 1 standardization of GSM1800 was finalized in 1990, and that for GSM900 in 1991. The first GSM phone call, conducted on the 1st of July 1991, marks the start of the utilization of GSM900 in Europe. In Phase 1, only FR speech is supported, and the data rate is very low, from 0.3 to 9.6 kbits/s. Soon after Phase 1, Phase 2 was proposed in 1995, which brought about a significant enhancement of Phase 1 GSM. Specifically, supplementary services like charging were supported in Phase 2, and HR was introduced. Additionally, in Phase 2, two concepts mentioned before were implemented to support Phase 1 GSM. Note that the implementation of Phase 2 signifies the complete standardization of GSM. After Phase 2, Phase 2+ represents several approaches to evolving the GSM standard. For example, the well-known GPRS is proposed in Phase 2+ to extend the package-switching capability of GSM. The GSM network is known as the public land mobile network (PLMN) and consists of several subsystems. The first major update of GSM is the separation of network exchange and user access. Specifically, the network exchange is the network for routing information from one point to another and is referred to as the network switching subsystem (NSS). The user access is the transceiver network, which is responsible

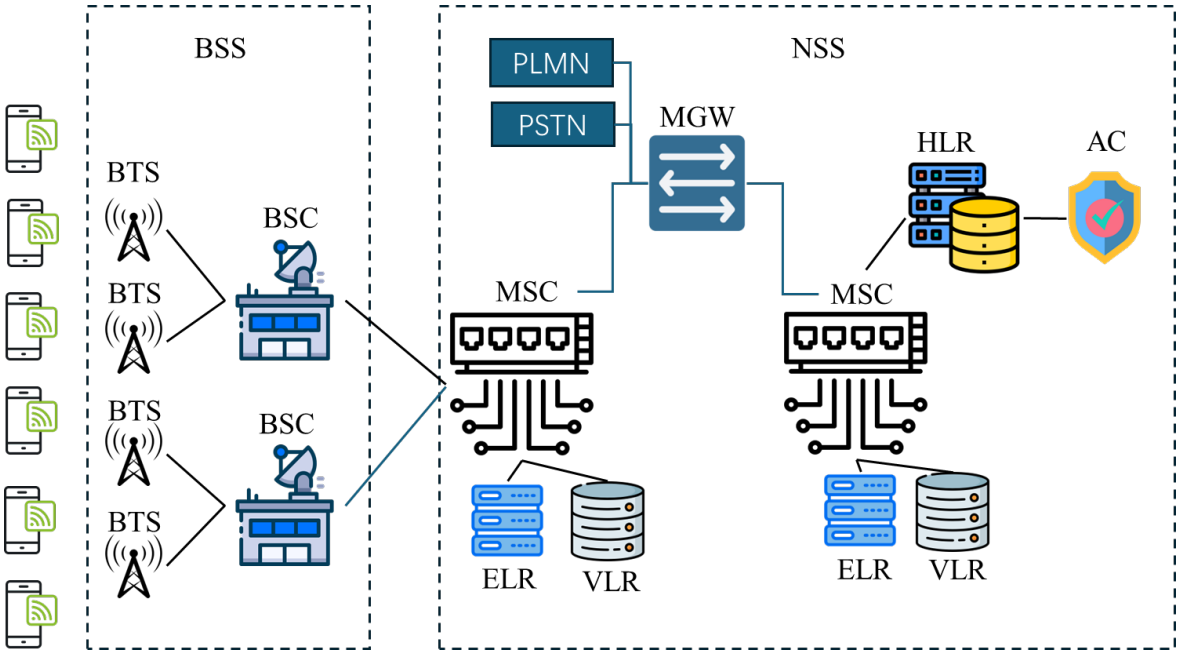


FIGURE 2.2. Network structure of GSM.

for the user's wireless access to the base station (BS), which is referred to as the base station subsystem (BSS). Fig. 2.2 shows the structure of the GSM network. Specifically, the GSM network consists of the following network elements (NEs).

(1) Base transceiver station (BTS): The communication between MS and PLMN is enabled by BTS. It serves a limited area (cell), and lots of BTSs are distributed around the country. The number of BTSs deployed in certain areas is determined by the number of subscribers (MS). The main tasks of a BTS are to perform channel coding, cipher/decipher, synchronize MS in both time and frequency, perform HF modulation, and monitor, evaluate, and optimize UL/DL transmission.

(2) Base station controller (BSC): BSC is implemented to address the limitation of BTS to handle information related to handover and keep track of MS. One BSC controls multiple BTS and can evaluate the strength of signals to determine when is the best time to perform handover for certain MS. The key responsibilities of BSC are: monitoring/measuring signal strength between MS and BTS, manage resource allocation for each BTS, and perform paging of MS.

(3) Mobile service switching center (MSC): MSC is the key part of NSS. The goal of NSS is to perform procedures that can establish connections between subscribers.The main functions of MSC are evaluating the dialing information, setting connections between subscribers, measuring and generating charging-related data, performing load sharing, and satisfying requirements of legal interception.

(4) Mobile gateway (MGW): MGW functions like a router that can connect one PLMN to other networks. It can be integrated into MSC.

(5) Home location register (HLR) and visitor location register (VLR): HLR and VLR together form the location register (LR) part of GSM. The main purpose of LR is to record the identity and location of each subscriber. Specifically, HLR records static user data like user telephone number, SIM card number, and current VLR area for that user. VLR records temporary user location data and receives requests from the authentication center for user authentication.

(6) Authentication center (AC): AC is the part in GSM that performs user authentication. It communicates with MSC and VLR to validate the identity of a certain subscriber.

(7) Equipment identity register (EIR): EIR is a database that stores equipment-related information. The main function of EIR is to supervise suspicious devices and track lost devices. It is an optional part of GSM and is usually not implemented due to high cost.

It can be noticed that BSS consists of BTS and BSC, and other parts constructed NSS. GSM can only support circuit switching where a dedicated virtual line is constructed between two subscribers. Time and frequency resources will be released only after their connection is terminated. Later on, GPRS was proposed, which added elements into the network (gateway GPRS support node (GGSN) and serving GPRS support node (SGSN)) to enable packet switching. Specifically, SGSN is similar to MSC, which performs packet switching inside one network, and GGSN interconnects different types of networks. The advantage of packet switching is that resources are not reserved for communication between two subscribers, which enhances the resource utilization efficiency compared with circuit switching.

After the introduction of the GSM/GPRS network structure, we now move to the resource allocation part for GSM/GPRS. Starting from GSM/GPRS, communication networks utilize digital signals instead of analog signals, and a new type of network, packet-switched network, is introduced. Different from the circuit-switched network, packet-switched networks do not construct a communication line dedicated to two users and reserve resources for the entire call. It is an IP-based network where messages are transmitted in the form of packets, and for communication between two subscribers, each packet can be transmitted following different paths. This enables a more efficient resource utilization of 2G compared with 1G. However, this flexibility also introduces complexity in resource management, and a new type of resource, network resource, is introduced. Note that in 2G, GSM is the circuit-switched network, and GPRS is the packet-switched network. Additionally, 2G utilizes both time division multiple access (TDMA) and FDMA, and the timeslot is also treated as a resource that can be allocated. In summary, resources to be allocated in the GSM/GPRS network can be categorized into two types, network resources and communication resources. The components of each type are listed below.

(1) Network resources: Network resources include resources that are related to the operation of the IP network. It consists of the IP address and the routing path. In 2G, IP version 4 (IPv4) is used, and each user needs to have their own IP address. The IP address is considered a network resource and needs to be allocated before the communication starts. Each path constructed between two routers/switches has a capacity limit, and only a limited number of packets can be transmitted through it. Network operators need to determine the path that each packet needs to follow, and the collection of paths is referred to as the routing table. Thus, the routing path is considered a network resource, and the routing table needs to be carefully designed for the packet-switched network to function normally.

(2) Communication resources: The utilization of TDMA in the GSM/GPRS network introduces an additional dimension in communication resource allocation: time. The transmission timeline is divided into multiple timeslots, with the basic communication resource block defined by a combination of one timeslot and one frequency. In 2G, DL and UL utilize different frequency bands, and the resource allocation for

them is managed separately. This means that one subcarrier can utilize a particular UL timeslot, and the corresponding timeslot in DL can be used by another subcarrier. Additionally, in the previous section, we mentioned the HR and FR transmission modes in GSM/GPRS. Users transmitting in HR mode can be paired together using the active HR pairing to utilize the time resource more efficiently.

Similar to 1G, 2G utilizes static RRC strategies, where the allocation of both network resources and communication resources is predefined at the design phase of the network. There are three main reasons of using static RRC stratiges in 2G. Firstly, the processing capability of each router/switch limited the implementation of high-complexity methods. Static RRC strategies are known for their low implementation complexity and do not pose a burden for these network devices. Secondly, the traffic patterns in 2G networks were relatively predictable, as they primarily supported voice communication and simple data services. Finally, the total allocable resources are limited. At that time, static RRC strategies are enough for managing available resources in 2G and satisfying the requirements of users.

2.3 The Third Generation

After GSM/GPRS, the 3rd generation partnership project (3GPP) produced specifications for the third generation (3G) in 1999 with the name of Universal Mobile Telecommunication Network (UMTS). 3GPP was established in 1998 with the goal of producing technical specifications for UMTS. There are many specifications for 3G from different organization parties. Here, we present those proposed by 3GPP, which are widely accepted worldwide.

Release 99 is the first release of UMTS, which was published in March 2000. It is based on the core network architecture of GSM/GPRS with a new air interface, utilizing the technology of wideband code division multiple access (W-CDMA). Each release includes multiple specifications, with each specification focusing on a different part of the telecommunication system. Here, terminologies used in UMTS are shown in Table 2.1.

TABLE 2.1. Existing Learning-based Works

Terms	Explanations
User equipment (UE)	3G system mobile terminal
Universal terrestrial radio access (UTRA/UTRAN)	The radio access network for 3G
Radio network controller (RNC)	The BSC for 3G
Node B	The BTS for 3G
Circuit switched (CS) domain	Corresponding to GSM
Packet switched (PS) domain	Corresponding to GPRS
Packet data networks (PDN)	PS-based network
Public switched telephone networks (PSTN)	CS-based network

As UMTS reused the core network architecture of GSM/GPRS, which is introduced in the previous subsection, the main difference lies in the access network. Fig. 2.3 compares the UTRAN with GSM access network. It can be seen that all NEs in GSM/GPRS access network were replaced with new NEs, and new interfaces are proposed to suit the specific needs of UTRAN. Note that in the previous section, we separate the GSM network into two subsystems. This does not mean that the access network (BSS) is separated from the core network (NSS). In fact, the separation of the access network from the core network is achieved in UMTS. This separation not only facilitates the integration and utilization of other advanced radio access technologies but also enables the effective implementation and operation of UTRAN. By decoupling these components, the system gains greater flexibility, allowing for smoother transitions and compatibility with future advancements in wireless communication technologies.

Another key distinction between UTRAN and GSM/GPRS is the partial separation of the user plane and the control plane. The reason for partial separation is that this separation only takes place in CN. In the access network, RNC handles control plane functions like mobility management, resource allocation, and user plane data forwarding. In the CN, the control signaling is handled by MSC for circuit-switched services and SGSN for packet-switched services. User traffic is forwarded to GGSN for packet-switched services and MGW for circuit-switched services. The partial separation limits the scalability as both planes are processed through a central controller.

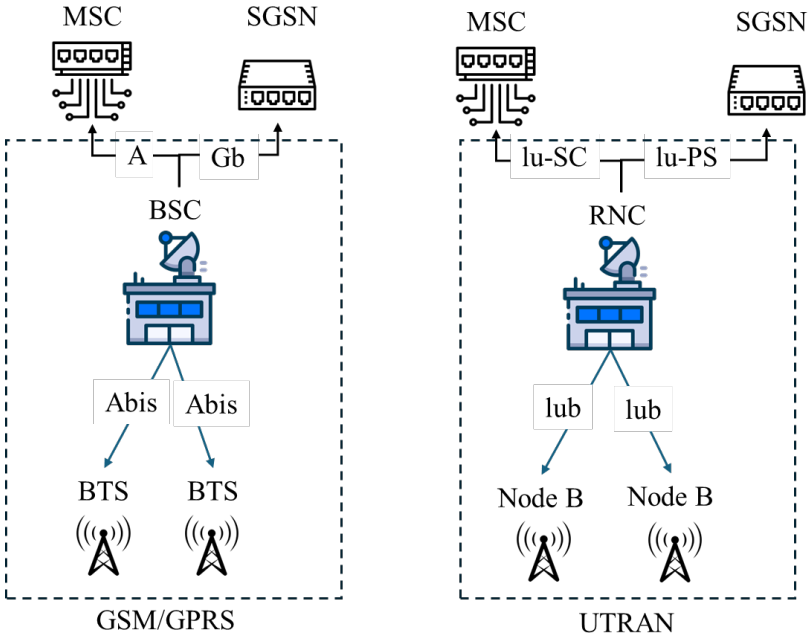


FIGURE 2.3. RAN comparison between UMTS and GSM/GPRS network.

In 3G, the amount of data traffic significantly increased, which is different from GSM, where speech-related traffic dominates the network. This shift from voice-dominant to data-dominate networks brings significant challenges to static resource planning methods, and the first attempt to allocate resources to users dynamically occurred. In UMTS, the major changes compared with GSM/GPRS are

(1) The ability to support transmission data rate up to 384 kbps
(2) The core network of UMTS is able to support both GSM/GPRS and UMTS RAN and the handover between GSM/GPRS and UMTS.
(3) Each subscriber may require bandwidth, and UMTS should be able to deliver the required bandwidth to each subscriber.
(4) UMTS should be able to deliver services with required QoS requirements.

It can be seen that to achieve the requirement of the last two points, a certain degree of dynamic resource allocation is needed. However, due to the limitation of the computing capability of hardware devices, heuristic methods are proposed to perform resource allocation. Specifically, some RRM algorithms are pre-configured in the network, and resource allocation

decisions are calculated based on these algorithms. These algorithms usually categorize users into different groups and allocate resources based on the defined allocation plan for that group. Due to the low complexity requirements, these algorithms are usually very simple and cannot cater to the varying network environments. Another difference in UMTS is the utilization of WCDMA instead of FDNA/TDMA in GSM/GPRS. The features of WCDMA are listed as follows.

(1) WCDMA can support higher transmission data rates compared with FDMA/TDMA. Thus, a large bandwidth of 5MHz is utilized in UMTS compared with 200 Khz used in GSM/GPRS.
(2) WCDMA utilizes only one frequency channel in theory.
(3) WCDMA utilizes RRM algorithms to control the overall quality of transmission.
(4) As the name suggested, different users utilize different codes to be separated from each other. In GSM/GPRS, users are separated by timeslot or frequency channels.

It can be noticed from the features of WCDMA that the resource to be allocated in UMTS is different from GSM/GPRS. Specifically, instead of allocating timeslots/channels to each user, codes are considered as communication resources in UMTS and need to be allocated to each user. In WCDMA, both scrambling and channelization codes need to be assigned to each user to compensate for the spreading phenomenon. Another resource that needs to be planned carefully is the transmission power of each user device. For WCDMA, the frequency reuse factor is 1, fast power control with high-level accuracy is more crucial than networks utilizing FDMA/TDMA. This is because all users in the same cell are using the same frequency and transmitting at the same time; user devices located near the base station can easily block other devices at the edge of the cell by overshooting them, leading to the effect referred to as the blocking effect. Thus, for both upline and downline, fast power control is required. One example gives a preview of how dynamic resource management is performed in UMTS. For example, in the UL, the BS measures the power received from user devices and calculates the SIR for that user. If the calculated value of SIR is larger than the pre-configured threshold value, BS asks user devices to reduce their transmission power. It

can be seen that this procedure can hardly be called as dynamic and a simple heuristic method is utilized to perform resource management.

2.4 The Forth Generation

The main driving factor for proposing the fourth generation (4G) is similar to that of 3G: the increasing number of users. This increase leads to an urgent need for larger system capacity, and there are three commonly used ways to achieve this. Firstly, the size of cells can be reduced to provide more network capacity. This is because the channel capacity in the cellular network is calculated as the maximum data transmission rate achieved by a single cell. By introducing more base stations and a smaller cell size, the total capacity of the network can be increased. Secondly, system capacity can be increased by increasing the allocated bandwidth managed by the International Telecommunication Union (ITU). However, only a finite number of bands are available in 2G and 3G, and most of them have been allocated to diverse applications like military usage. Additionally, the allocation of bandwidth is time-consuming and has high operational costs. Finally, communication technologies can be improved to support a larger system capacity. Specifically, new technology can help us to get closer to the theoretical channel capacity and utilize higher frequency bands with larger bandwidth. The third one is the main focus of 4G.

4G is also referred to as long-term evolution (LTE), and its development is also driven by three other issues introduced by the design of 2G and 3G as follows.

(1) Both 2G and 3G operators need to manage and maintain two core networks: the circuit-switched network and the packet-switched network. As mentioned in the previous section, the circuit-switched network is for voice calls only, and the packet-switched network is IP-based, which supports the transfer of data packets. However, the development of technologies like voice over IP (VoIP) enables the transmission of voice calls over the packet-switched network, and it is desirable to design a unified network with only packet-switched ability to reduce capital and operational costs.

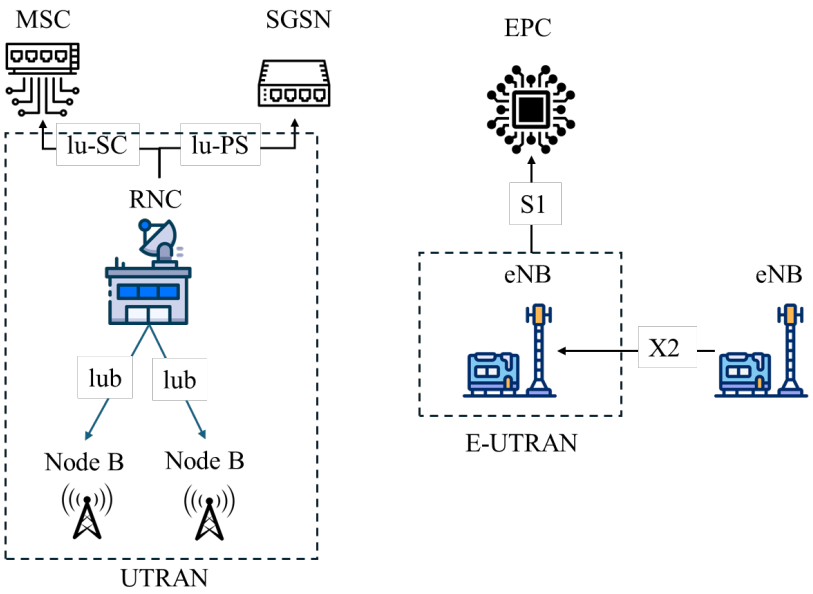


FIGURE 2.4. Network architecture comparison between UMTS and LTE

(2) Both 2G and 3G focus on voice calls with very limited other services. In 3G, the network-induced delay is about 100 milliseconds for data transmission, which is acceptable for voice calls and other supplementary applications. However, for applications like real-time interactive gaming, this delay is not acceptable. A network with the ability to achieve lower delay is required to support emerging service types.

(3) The complexity of UMTS and GSM have both become over complex over the years. This is because new features need to be added to the 2G and 3G networks, and backward compatibility also needs to be achieved for both networks. 4G can be considered as the new start and avoids the need to go through complicated documentation to improve system performance.

With these motivations in mind, we can now move to the architecture of LTE. In the following, we refer to 4G as LTE and 3G as UMTS. Fig. 2.4 shows the architecture comparison between UMTS and LTE, with the left-hand side showing the architecture for UMTS and the right-hand side for LTE. It can be noticed that in LTE, the evolved packet core (EPC) replaced the previous CS and PS. Specifically, EPC utilizes only packet-switched technologies, and the CS network has been discarded. EPC utilizes VoIP technology and transmits voice calls over the PS network. Additionally, it can be noticed that the LTE utilizes evolved UTRAN

(E-UTRAN) as its access network, which only consists of the eNodeB (eNB) and discarded the centralized controller RNC. eNBs can directly communicate with each other through the X1 interface and connect with EPC via the S1 interface. Originally, LTE only referred to the architecture change of the access network, and that for the core network is named as system architecture evolution (SAE). The combination of SAE and LTE is referred to as evolved packet system (EPS). However, LTE is now commonly used as the colloquial name for 4G.

The requirements of LTE are to achieve a peak data transmission rate of 100 Mbps DL and 50 Mbps UL, to achieve a latency of around five milliseconds for time-critical applications, to achieve optimal performance with the cell size of 5 km, to support degraded performance with the cell size of 100 km, and to support speeds of up to 350 km power hour. As WCDMA can only support data rates up to 14 Mbps DL and 5.7 Mbps UL, a new multiple access scheme, referred to as orthogonal frequency-division multiple access (OFDMA), is utilized in LTE to help achieve the desired peak data rates. OFDMA performs the same function as other multiple access techniques, allowing the BS to communicate with multiple users simultaneously. It is also utilized by other wireless networks like WLAN and WiMAX, which is beyond the scope of this thesis. OFDMA is developed based on orthogonal frequency division multiplexing (OFDM). OFDM separates a single data stream into multiple sub-streams, with each transmitted at a different frequency. Here, one frequency for transmission is referred to as a subcarrier in OFDM. OFDMA allows BS to allocate users to different subcarriers at different timeslots, and each block is formulated by one timeslot, and one frequency is referred to as one resource element. In OFDM, one resource unit is also called an OFDM symbol. In LTE, the smallest unit for resource allocation is the resource block, which is formulated by 7 consecutive timeslots and 12 consecutive subcarriers.

After introducing the technologies used in E-UTRAN, we then move to the EPC. Fig. 2.5 shows the architecture of EPC. We elaborate on each part of EPC in the following.

(1) Serving gateway (SGW): SGW is a router that performs the function of packet forwarding between E-UTRAN and the PDN gateway. Usually, one LTE contains many SGW, each has its own serving area, and each mobile is assigned to one SGW. Handover occurs when mobile is moved far enough from the SGW.

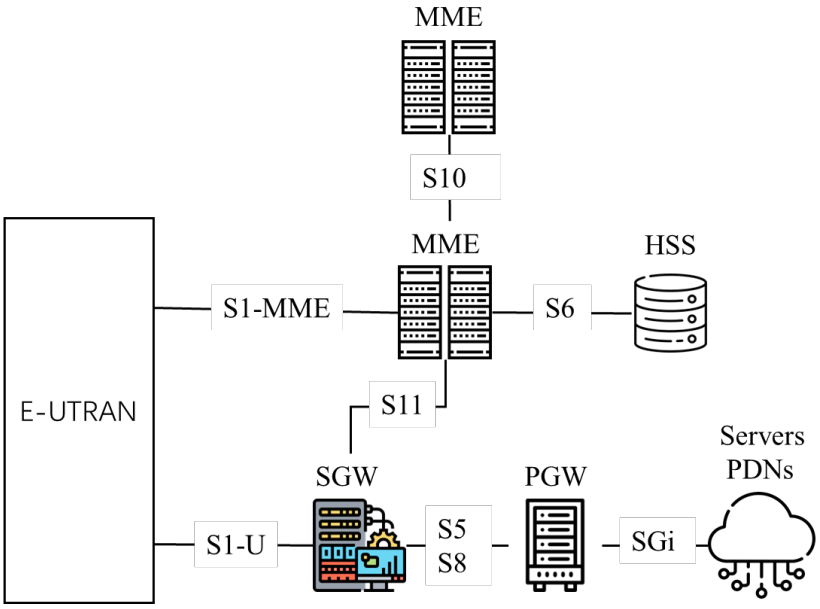


FIGURE 2.5. Network architecture of EPC

(2) Mobile management entity (MME): MME handles the control plane function like session establishment and management, location update and monitoring, assigning SGW to mobile, performing paging for mobiles, and monitoring current network QoS. One MME connected with multiple SWG.

(3) Home subscriber server (HSS): HSS performs similar functions as that of HLR, VLR, and AC in GSM/GPRS and UMTS. It stores mobile-related data, generates and manages authentication vectors to be used by MME, interfaces with the policy and charging rules function (PCRF) to provide subscriber-specific policies for data session management and charging, and supports roaming.

(4) Package data network gateway (PGW): PGW performs the function of packet routing between one PDN and another. It collects data related to charging and assigns IP addresses to mobile devices.

It can be noticed in Fig. 2.5 that two interfaces between E-UTRAN and EPC exist. The one connected to MME is S1-MME, which handles control plane information, and the other one connected to SGW is S1-U, which handles the user plane data forwarding. It can be seen that compared with UMTS, the full separation of the user plane and control plane is achieved by removing the central control NE RNC. This complete separation enables the independent

scaling of each plane and can move the user plane devices closer to the mobiles to allow low-latency transmission. This separation also enhances network flexibility, enabling the deployment of user plane devices in distributed locations to optimize resource utilization and reduce backhaul congestion.

In 4G, three main changes occurred. Firstly, the circuit-switched network is completely replaced by the packet-switched network. Previous voice data is transmitted over the packet-switched network utilizing the VoIP technique. Secondly, the user plane and the control plane are separate completely, and the user plane is brought closer to the user side. Finally, OFDM has been introduced in 4G to enable the system to achieve high data rates. These three changes enable the 4G network to serve the exploding number of users with high transmission data rate requirements and to transfer multimedia information like voices, images, and videos. In UMTS, two kinds of fading are caused by multipath propagation, time-dispersive fading and frequency-selective fading, which exist during the transmission of wide-band wireless signals. The capability of OFDM to transfer frequency-selective channels into independent flat-fading channels can significantly reduce the multipath propagation of wireless signals, thus reducing the two fadings mentioned above. In OFDM, two kinds of resources exist, like in GSM/GPRS, but with a finer granularity: time and frequency. Each user can be allocated one timeslot and one frequency channel (subcarrier), which is referred to as one resource element in OFDM. Due to the implementation of OFDM, starting from LTE, one key issue that communication systems must solve is the changing time-frequency channel status between users and BSs. Specifically, the real dynamic resource allocation of time and frequency resources must be performed in accordance with the channel status for both fast and slow-fading environments.

In LTE, the basic time-frequency allocation unit is referred to as a physical resource block (PRB), where each PRB consists of multiple consecutive resource units. Specifically, it consists of 180 kHz bandwidth and one timeslot (0.5 ms). By default, 180 kHz bandwidth is separated into 12 subcarriers and 24 subcarriers when multimedia broadcast and multicast service (MBMA) is utilized. This is because, normally, the frequency channel is set to 15 kHz, and it is set to 7.5 kHz when MBMA is utilized. Thus, the number of available PRB

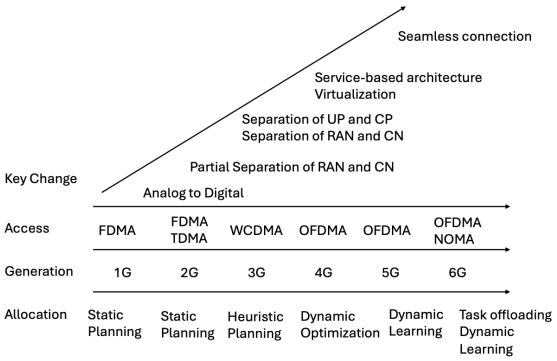


FIGURE 2.6. Development of resource allocation from 1G to 6G

depends on the allocated usable bandwidth. The size of this PRB is determined by several factors. We summarize the reasons as follows.

(1) During one PRB, the time status of the channel remains unchanged. One way of measuring the speed of the channel varies is by calculating the Doppler spread encountered when the speed of the devices travels at around 300 to 350 km/h under the carrier frequency of 2.7 GHz. The Doppler shift calculated under this circumstance is around 750 Hz, corresponding to 1.33 ms, which is much larger than 0.5 ms. Thus, one timeslot in PBR is set to 0.5 ms.

(2) The channel over different frequencies in a PRB should stay constant and not even vary at all under the worst-case delay spread conditions. Specifically, the worst-case delay spread is around 512 timeslots, which, in the frequency domain, corresponds to the main lobe of the sine wave with a width of around 120 kHz.

In conclusion, the allocation of time and frequency resources in LTE depends on the real-time channel status to maximize the throughput. In the previous generation, the channel status is not the main focus. As the channel significantly varies between timeslots, dynamic resource allocation is a must. Additionally, as two types of fading environments exist, there are two principles in performing resource allocation. Specifically, in a slow-fading environment, frequencies are utilized by users with good signals. In a fast-fading environment, the frequency channels are distributed among users. Fig. 2.6 shows the development of resource allocation from 1G to 6G and beyond.

CHAPTER 3

Literature Review

In the previous section, we introduced the development of communication networks from 1G to 5G and pointed out the importance of SAGIN in achieving the goals of future 6G networks. We also mentioned that to utilize the advantages of 5G and SAGIN, an efficient resource allocation scheme is essential. In the following, we will give a detailed literature review on state-of-the-art resource allocation schemes for both 5G and SAGIN. Note that in SAGIN, task offloading needs to be performed before the resource allocation. In the first section, we give a literature review of both the 5G core network and the 5G RAN network. A table that summarizes the goal, objective, and schemes used in each work is presented at the start of this section. Next, we give a literature review of task offloading/resource allocation schemes in SAGIN. These schemes are separated into optimization-based approaches and learning-based approaches depending on whether DL/DRL is the main part of their proposed methods. One table that summarizes the goal, objective, and schemes used in each work is presented at the beginning of this section, and another one that summarizes the key features of learning-based works is presented at the end of this section. This chapter is formulated with the goal of presenting a comprehensive review of current works and pointing out the main driving factors for our works.

3.1 Resource Allocation in 5G Terrestrial Network

5G terrestrial network plays an important role in modern communication networks to achieve the goal of simultaneous satisfaction of distinct QoS requirements for emerging service types. It leverages both new and well-developed technologies like NFV, SDN, and plane separation

and can satisfy SLA for both delay stringent services and high transmission rate services. As mentioned in the introduction, these advantages of 5G come with challenges, and one of the most critical ones is resource allocation for 5G RAN network slicing. In this section, we will give a literature review of strategies used for managing network slices in the 5G, with an emphasis on RAN slicing, which is the main topic of Chapter 3. The table shows the summary of all the works listed in this section.

TABLE 3.1. 5G Network Slicing Resource Allocation

Work	**Objective**	**Goal**	**Method**	**Part**
[50]	Delay, intra-slice isolation	Minimize delay	Optimization-based approach	CN
[51]	UE association, resource allocation, slice management, and VNF replacement	Perform signaling optimization	DQN	CN
[52]	User SLA optimization problem	Enhance eMBB SLA satisfaction level	Prioritized experience replay and pointer network SFC mapping	CN
[53]	Perform on-demand slice isolation	Mitigate DDoS attacks	Convex optimization	CN
[54]	Create and manage network slices	Evaluate 5G system performance	NFV, MIMO, OpenStack Tacker, free5GC	CN
[55]	Energy consumption	Minimize delay and energy	Convex optimization	RAN
[56]	Latency minimization in resource allocation	Perform service-aware baseband resource allocation	Convex optimization	RAN
[57]	Resource allocation for cross-modal communications	Meet transmission requirements	Optimization-based method	RAN
[58]	On-demand resource allocation for URLLC and eMBB	Maximize performance of URLLC services	Convex optimization	RAN
[59]	Find optimal frequency allocation	Minimize latency	genetic algorithm	RAN
[60]	Joint resource allocation for eMBB and URLLC service	Satisfy QoS requirements for both services	Optimziation-based approaches	RAN
[61]	Demand-aware resource allocation framework	Maximize SLA satisfaction ratio and SE	GAN-DDQN	RAN

TABLE 3.1. 5G Network Slicing Resource Allocation (Continued)

Name	Object	Goal	Method	Part
[62]	Demand-aware resource allocation framework	Maximize SLA satisfaction ratio and SE	LSTM-A2C	RAN
[63]	QoS-aware RAN slicing strategy	Maximize the weighted sum of service QoS and SE of slices	DQN	RAN
[64]	Communication, computing, and caching (3C) resource allocation	Maximize the utility of the network	A2C-DDPG	RAN
[65]	Resource allocation and task scheduling	Meet QoS requirements of URLLC and eMBB	Layered DRL	RAN
[66]	Resource allocation considering user QoS requirements	Maximize the utility	DDPG	RAN
[67]	Frequency resource allocation	Maximize spectral efficiency	DQN	RAN
[68]	Resource allocation and routing challenges	Maximize resource utilization	Multitask DRL with GCN	RAN

3.1.1 5G CN Slicing

There are two challenges exist in 5G CN slicing: satisfaction of end-to-end delay of each slice and intra-slice function isolation. Here, the intra-slice function isolation refers to the principle that different VNFs of one slice should be hosted in different servers to avoid the situation where the malfunction of one server impacts the whole slice. In [50], the authors formulated a MINLP problem and proposed a model for 5G core network slicing that can satisfy the end-to-end delay, intra-slice isolation, and finding the minimal delay path for slice establishment. There are three issues considered in this work. The first one is intra-slice isolation, which means the isolation of a single slice for VNF. Note that in this work, the authors do not take into account inter-slice isolation, which includes the high availability of design, physical isolation of different slices, and hardware and virtual machine-level isolation. The second one is the end-to-end delay. 5G network is known for its support for various real-time applications, and it is important to consider the end-to-end delay in performing 5G

core network slicing. Finally, this work highlights the problem of slice allocation in 5G core network by investigating different VNF placement approaches and finding the minimum delay path.

In [51], the authors proposed a joint UE association, resource allocation, slice management, and VNF replacement problem (TROCH) and utilized Dijkstra's algorithm together with the deep Q-learning network (DQN) to solve this problem. They proposed an end-to-end network slicing scheme where the number of slices and network functions dynamically change in accordance with a current number of users and their corresponding QoS requirements. The goal of this paper is to provide insight into the development of future-generation communication networks by performing signaling optimization of the core network. The proposed TROCH is a MINLP problem, and they decomposed it into two sub-problems: the link assignment problem and the ASSAIL problem. Specifically, the link assignment problem is solved by Dijkstra's algorithm, given the solution for the ASSAIL. These two problems are solved alternately until convergence. It is considered as converged if the difference of two consecutive solutions is within a predefined range.

In [52], the authors proposed a double DQN integrated with prioritized experience replay and pointer network SFC mapping method to solve the user service satisfaction level optimization problem. The authors introduced an end-to-end network slicing architecture with the goal of satisfying end-to-end service level agreements (SLA). This framework considers the upper-layer cross-domain orchestrator coordination problem and incorporates several subordinate domain controllers. They designed two latency equalization policies for this upper layer, and for the lower layer, they split the latency budget. The performance of their proposed method is validated by performing the joint optimization of improving the enhanced mobile broadband (eMBB) SLA level of satisfaction and, in the meantime, maximizing the accepted number of users in the network. They also introduced service satisfaction level (SSL) parameters to measure the performance and perform tradeoffs between optimizing user experience and accepting as many users as possible.

In [53], the authors proposed a mathematical model that can perform on-demand slice isolation with the constraint of end-to-end delay of 5G core network. This is considered as the

solution for mitigating the distributed denial-of-service (DDoS) attacks in 5G core network. Specifically, in traditional communication networks, DDoS can only affect services that are targeted,d and other services have low risk to be affected as they use different hardware. However, in 5G, the implementation of NFV makes it possible for multiple services to use a shared physical infrastructure. Thus, DDoS will have a more severe impact on 5G than on other networks. To mitigate this problem, slice isolation is very important. Instead of using traditional hardware isolation for slice isolation, the authors proposed the utilization of slice resource isolation to counter DDoS attacks. Additionally, the proposed solution considers both inter-slice and intra-slice isolation, with a guarantee of end-to-end delay to satisfy requirements of delay-sensitive services.

In [54], the authors leveraged open source software OpenStack Tacker and NCTU free5GC to create and manage network slices and compared multiple network slicing schemes. They also proposed two 5G core network slicing models utilizing technologies like NFV and management and orchestration (MAMO), one based on multiple network slicing models and another based on one-size-fits-all architecture. The main goal of this work is to evaluate how the changes in the architecture affect the 5G system performance. The experiment results show that the one based on the multiple network slicing model performs better than the one-size-fits-all design in terms of resource utilization and end-to-end latency with the same total available virtual and physical resources.

3.1.2 5G RAN Slicing

In this subsection, we consider the resource allocation strategies for RAN slicing. Many approaches have been proposed in this field, which mainly focus on solving one-shot resource allocation problems. Specifically, the one-shot resource allocation problem only considers the current timestep network status (e.g., current task status) and ignores the long-term system performance and correlations among allocation decisions for different timesteps. These approaches can be further classified as optimization-based and learning-based.

3.1.2.1 Optimized-based approaches

For the optimization-based approaches, in [55], the authors aim to minimize the end-to-end energy consumption in RAN slicing and propose a joint delay and energy consumption model for each slice. The proposed model takes into account the latency and energy consumption for three sites: the radio site, the transport site, and the cloud site of RAN. They proposed to solve a non-convex resource allocation problem by relaxing it using quadratic programming techniques into a convex one. It is then solved by a convex problem solver. The solution of this problem gives the communication and computation resources allocation for each slice. Additionally, in this work, how the 5G numerology will impact the resource allocation strategies and the minimization of energy consumption is investigated. It is validated that a trade-off between latency and energy consumption can be achieved by leveraging mixed numerology.

In [56], an iterative algorithm is proposed to minimize the end-to-end delays with fronthaul capacity-related constraints. In this work, the authors focus on resource allocation for the next RAN generation, open RAN (O-RAN), which can provide more flexibility with lower costs compared with the currently deployed RAN. The goal is to perform service-aware baseband resource allocation and investigate the activation procedure of VNF in O-RAN. Resources for optimization include the O-RAN radio unit, physical resource block, and transmission power. As it is np-hard to directly solve the original MINLP problem, the authors separated this problem into two subproblems and solve them alternately until convergence. Specifically, one subproblem is for resource allocation, and the other is for radio unit association. Additionally, to check the complexity of this method, the authors give a detailed analysis of the complexity and also analyze the feasibility region of its proposed problem, and another algorithm is proposed to fast-track this region.

In [57], a scheme that integrates network slicing and puncturing is proposed to perform efficient transmission resource allocation for cross-modal communications with audio-visual and tactile signals. The authors proposed a joint transmission scheme for different signals to solve the proposed resource allocation problem, where both networking slicing and puncturing

architecture are utilized in this scheme. With the implementation of two architectures, a more flexible resource allocation can be performed with the benefit of lower resource waist. As the puncturing system will bring some limitations to users, like missing content, the optimization problem is formulated to mitigate this problem. Specifically, the problem is formulated with the goal of meeting the transmission requirements of two types of signals and considering the transmission rate requirement and reliability requirement. The original problem is divided into two subproblems, one for video traffic resource allocation and another for puncturing tactile traffic. The first problem is solved using a channel-matching algorithm, and the second is solved by a puncturing resource allocation algorithm.

In [58], a modified version of the sparrow search is proposed to solve the resource allocation problem. The goal is to maximize the number of URLLC services supported by the gNB while satisfying eMBB service requirements. Specifically, the authors proposed a URLLC preemptive strategy that changes the arrival of URLLC services from just occupying one punctured resource (even allocation) to occupying resources for multiple mini-slots (uneven allocation). The reason behind this is that when the number of URLLC services becomes larger, the previous even allocation will result in some URLLC users not being supported while some improperly inserted into eMBB streams, interrupting eMBB services. Using an uneven multiple mini-slots preemptive strategy can solve this problem with less data rate loss for each user and can support more URLLC services. The problem is then formulated considering both the delay requirement of URLLC services and the influence of punctuation on eMBB services. The resource allocation of URLLC is performed on demand.

The authors in [59] utilized the genetic algorithm in finding the optimal frequency allocation in vehicular Ad-Hoc networks. It is designed to suit the needs of both mission-critical and best-effort traffic. In this work, a network slicing framework utilizing technologies like SDN, NFV, and edge computing is proposed to satisfy end-to-end QoS requirements for services in 5G VANETs. In this work, network slices are divided into mission-critical slices and non-critical slices. Mission-critical slice focuses on meeting ultra-reliability and low latency requirements of mission-critical applications like remote surgery. In this scenario, the virtual machines (VMs) are located at the edge cloud (EC), and the allocation of the slices is

performed on demand. Non-critical slice performs resource allocation in a best-effort manner, and VMs remain in a centralized cloud. Note that some functions that need to be performed by end-user devices are also placed in EC.

In [60], the authors considered the joint resource allocation for eMBB and URLLC service, and QoS requirements for both services are considered with priorities. They proposed a joint eMBB/URLLC scheduling problem and incorporated multiple models that can describe the relationship between eMBB rate loss caused by the puncturing of URLLC. Specifically, in the linear model, the rate loss of eMBB is proportional to the number of punctured timeslots. Under this model, the URLLC traffic can be uniformly distributed across every timeslot, and the eMBB traffic can be scheduled using the greedy method despite the non-linearity of their utility function. Another model is the convex model, which can represent a more general setting compared with a linear model. Under this model, the optimal allocation decision is hard to find. The authors simplified this problem into a mini slot-homogeneous scheduling problem where the URLLC allocation policy stays unchanged across mini slots.

However, these approaches require a large number of iterations to solve the resource allocation problem with high computational complexity. This prevents the online deployment of these methods. Additionally, these approaches need to solve the formulated problem again once the network parameters change (e.g., channel state information), leading to prohibitory high computational complexity and preventing the adaption of these approaches to the changing network environment.

3.1.2.2 Learning-based approches

For learning-based approaches, the authors in [61] proposed a demand-aware resource allocation framework with all base stations sharing the same total frequency and time resources. A modified version of deep distributional Q-network (DDQN), referred to as generative adversarial DDQN (GAN-DDQN), is utilized to solve the communication and computational resources allocation problem in RAN slicing. In GAN-DDQN, the generator network learns the distribution of action values by producing a dedicated number of particles. This design enabled GAN-DDQN to avoid the large variance induced by the varying network environment.

Additionally, the training process of GAN-DDQN is analyzed with the conclusion that the widely used system utility values will introduce unstable parameter updating. To concur with this problem, the reward-clipping algorithm is used where the initial system utility values are clipped into a set of constant values with adjustable thresholds. Furthermore, to mitigate the limitation of GAN-DDQN where only a small portion of particles can be generated, a dueling GAN-DDQN is proposed that is based on dueling DQN and discrete normalized advantage functions algorithm. Specifically, dueling GAN-DDQN distinguishes between state-value distributions and action-value distributions and introduces an advantage function to mitigate the varying network.

In [62], the authors proposed a demand-aware inter-slice resource allocation scheme where BSs share the same frequency and time resources. They integrated twin action critic (A2C) with long-short-term memory (LSTM), referred to as LSTM-A2C, in solving the RAN inter-slice resource management problem. The goal is to maximize the utility function, which is designed as the weighted sum of spectrum efficiency (SE) and SLA SSR, by provisioning available bandwidth for each slice. To avoid invalid allocation decisions, the minimum allocated bandwidth per slice is predefined as hyperparameters, and the allocated bandwidth must be an integer multiple of the minimum bandwidth. Additionally, in this network, the request patterns of user equipment (UE) dynamically, and the location of the UE also changes overtime. The temporal correlation of the requests is captured using LSTM, where the state is defined as a set of vectors where each entity represents the number of arrived packets for a certain time window.

In [63], the authors proposed a QoS-aware RAN slicing strategy with the goal of maximizing the weighted sum of service QoS and SE of slices. This strategy consists of multiple granularity of frequency resources and time to capture the dynamically changing network conditions. It contains two control layers, the upper-level controller and lower-level control, to focus on different degrees of granularity. The upper-level controller performs slice provisioning by constraining the total achievable data rate of all users in each slice, with the goal of maximizing SE. The lower-level controller controls the allocated resources to each user at a small timescale with the constraint obtained from the upper level. Thus, the configuration

decision made by the upper level will have a significant influence on the lower level. Also, the performance of the lower level at the current time step will influence the decision-making of the upper level in the next step. For conventional RL methods, it is hard to capture this kind of correlation. Thus, the authors integrated DQN with DDPG and proposed a multi-layer resource allocation scheme. Specifically, the lower-level control policy is generated based on DDPG and the upper-level control is performed by DQN.

In [64], a modified version of A2C-DDPG is proposed to solve the communication, computing, and caching (3C) allocation in RAN slicing. The authors proposed a framework that combines network slicing with multi-access edge computing, referred to as multi-access network slicing, and proposed a two-level resource allocation problem based on this framework. The main objective of the problem is to maximize the utility of the network and guarantee QoS's satisfaction with each service. The unique feature of this work is the utilization of twin actors to generate both slice-level and user-level continuous actions. The slice-level action refers to the resource allocation decisions of frequency, computing, and caching resources for each slice, and the user-level refers to that for each user. To achieve the goal, the reward function of A2C-DDPG is designed as the weighted sum of system utility, the difference between slice maximum service delay and the delay threshold for not satisfying its QoS requirement, the delay requirements related to content downloading, and the throughput requirements.

In [65], the authors proposed a joint resource allocation and task scheduling framework for the 5G Internet of Vehicles (IoV) network. The problem is formulated considering both inter-slice and inter-slice decision-making procedures, and the distinct features of two types of services, URLLC and eMBBS, are included in the formulation. In accordance with service types, network slices are divided into URLLC slices and eMBB slices. A heterogeneous Markov decision-making process (HMDP) is proposed to model the decision-making process, which consists of two layers. The utilization of two-layer HMDP significantly reduces the size of the action space and enables the separation of the original MDP problem into two sub-MDP problems: upper-layer MDP and lower-layer MDP. The upper-layer MDP models the inter-slice resource allocation and the lower-layer MDP is designed to account for intra-slice

task scheduling. To tackle these two subproblems, a layered DRL method is proposed, which contains three agents: one for upper-layer resource allocation and two for task scheduling.

In [66], a recurrent DDPG is proposed to cater to the end-to-end resource allocation considering user QoS requirements. The formulated problem takes into account dynamically changing user demands and the CSI uncertainties. Specifically, CSI uncertainties are caused by the random movement of the user and can significantly degrade the resource allocation performance if no countermeasures are deployed. The objective of this problem is to maximize the utility, which is defined as the difference between cost and revenue. Network slices are divided into eMBB slice, URLLC slice and mMTC slice, where eMBB slice represents services with high transmission data rate requirements, URLLC slice represents services with stringent latency requirements, and mMTC slice represents services with massive connection requirements. In this work, the main focus is the eMBB slice. Resources to be allocated include the number of subchannels in RAN, computing-related resources (CPU/RAM), and caching-related resources (storage).

In [67], a modified version of Q-learning is proposed to perform frequency resource allocation in a smart grid network. The goal is to maximize spectral efficiency while considering different user QoS requirements. Also, the authors classified power services into two types: elastic and real-time services, and two utility functions are proposed to model each type. According to the service types, network slices are divided into elastic slices and real-time slices. For the elastic services, no minimum bandwidth requirement exists due to its loose latency requirements. Services like distributed power and video surveillance can be categorized into elastic services. For real-time services, the guarantee that the network provides the lowest level of performance is essential, which is caused by their stringent latency requirements. Services like emergency communication and remote surgery can be categorized into real-time services.

Authors in [68] propose a multitask DRL method enhanced by GCN to solve resource allocation and routing challenges in C-RAN. The authors have proposed a framework for network slicing and routing, trying to unify network management and control. Unlike the traditional methods of slicing and routing separately, this approach dynamically allocates the resources and selects the optimal routes in real-time. It uses a differentiable pooling-based

GCN to capture the topological information of graph-structured network states for efficient policy generation in the multitask DRL model. The multitask DRL model divides the problem into parallel tasks: one for slicing and multiple for routing across network slices. The division reduces the action space while improving decision-making efficiency. Each task uses a shared structure in the DRL model, capturing a relationship between tasks and maximizing resource utilization. The model should be trained in an SDN controller that allows real deployment and monitoring in real time.

However, these approaches have the following limitations. 1) A common distribution like Poisson distribution is used to model the arrival process of user requests. In the real world, the arrival process for different types of user requests should be different. 2) DRL-based approaches can only generate discrete or continuous action and not both. The resource allocation problem are usually formulated as a mixed integer nonlinear programming problem (MINLP) and contains both discrete and continuous variables. Thus, DRL-based approaches need to separate/relax the original problem, which leads to system performance loss.

3.2 Task Offloading and Resource Allocation in SAGIN

SAGIN has been recognized as the key solution for 6G to achieve seamless global connection and satisfy the high availability requirement of telecommunication systems in disasters. SAGIN is a heterogeneous networking model that interconnects the space, air, and ground segments and can utilize communication/computing resources from all three layers. Recently, there has been a surge in the deployment of satellites; hence, abundant computational resources are present in the space segment, which can help alleviate current resource constraints. As mentioned in the introduction, one of the most important tasks for the integration of space, air, and ground segments in SAGIN is finding efficient task offloading and resource allocation. Many approaches have been proposed to solve the resource allocation and task offloading problem in SAGIN. They can be classified into optimization-based approaches and learning-based approaches. In the following, we present detailed research on current state-of-the-art

task offloading/resource allocation strategies for SAGIN. Table 3.2 summarizes all the works listed in this section.

TABLE 3.2. SAGIN Task Offloading and Resource Allocation

Work	**Objective**	**Goal**	**Method**
[69]	Distributed task offloading decision-making scheme	Minimize delay and energy consumption	Optimization-based approach
[70]	Task offloading decisions from terrestrial to space	minimize the total energy consumption of IoT devices	Sequential fractional programming and Lagrangian dual decomposition
[71]	Task offloading and communication/computation resource allocation	Minimize energy consumption	AO
[72]	Task offloading decisions in MEC-enabled SAGIN	Minimize the user-experienced latency	BSUM
[73]	association control, computing/frequency/transmission power allocation, and UAV deployment optimization	Minimize the maximum computation delay	BCD, SAC, and AO
[74]	Joint UAV trajectory planning resource allocation and task offloading	Minimize the energy consumption	BSUM
[75]	Task offloading for space/aerial-aided MEC network	Minimize long-term task completion delays	BnB and Lyapunov optimization
[76]	Task offloading and power control scheme	11	Optimization-based approaches
[77]	User scheduling, partial offloading decisions, computational resource and bandwidth allocation, and UAV trajectory planning	Minimize weighted energy consumption of ground users and UAVs	AO
[78]	UAV trajectory planning, resource allocation, and task offloading	Maximize energy efficiency	AO, SAC, and Dinkelbach algorithm
[79]	Frequency and transmission power allocation for	Maximize energy efficiency	Lagrange dual
[80]	Hybrid centralized and decentralized learning	Minimize the total learning cost	DQN
[81]	VMs computing resource allocation and task offloading	Minimize delay	DQN
[82]	Offloading decisions for multiple tasks	Minimize latency	A-PPO

TABLE 3.2. SAGIN Task Offloading and Resource Allocation (Continued)

Name	Object	Goal	Method
[83]	Resource allocation and task offloading	Minimize rental prices	Attention-enabled multi-agent PPO
[84]	Task offloading for SAGIVN	Maximize the number of tasks completed	DQN and convex optimization
[43]	Task offloading/resource allocation for cloud/edge SAGIN	Minimize energy consumption and latency	Collaborative multi-agent DRL
[85]	Traffic offloading	Minimize the delay of all packets	DFSAC
[86]	Task offloading in SAGIN vehicular networks	Minimize latency and energy consumption	Multi-agent DRL
[87]	Task offloading computational collaboration scheme	Minimize delay and maximize the number of completed tasks	DDPG
[88]	Task offloading/resource allocation with energy harvesting	Optimize system efficiency	PPO

For optimization-based approaches, in [69], the authors proposed a distributed decision-making scheme where Lyapunov optimization is used to divide the main problem into several sub-problems that can be solved by each satellite. The goal of this work is to minimize the weighted sum of delay and energy consumption by performing multi-hop task offloading and resource allocation. Specifically, the muti-hop task offloading refers to the offloading process where the offloading is performed along multi-hop paths. This can utilize the available resources for multiple satellites simultaneously. Note that this work does not consider services with stringent latency requirements. The authors formulated a multi-hop satellite peer offloading problem that takes into account the uncertainty of future workloads.

In [70], the authors proposed to use sequential fractional programming and Lagrangian dual decomposition to solve the computation offloading problem and minimize the total energy consumption of IoT devices. The proposed framework consists of the LEO satellites and remote internet-of-things (IoT) mobile devices (IMDs), where IMDs are incorporated with terrestrial-satellite terminals and can directly communicate with LEOs. The authors formulated the task offloading process into two stages with the goal of minimizing the total

energy consumption of all IMDs. According to the two-stage task offloading, the formulated problem is divided into two layers, where the lower layer is solved by sequential fractional programming with the goal of minimizing the latency encountered in the space segment, and the higher layer utilizes Lagrangian dual decomposition and is solved using convex optimization approaches.

In [71], the authors proposed to use the alternating optimization (AO) method to decompose the problem with the goal of minimizing the energy consumption of LEO into two sub-problems. These sub-problems are then solved by low-complexity iterative algorithms. However, these works only focus on space and terrestrial segments and no air segment is considered. In [72], the authors proposed to use the block successive upper-bound minimization (BSUM) method to find the optimal offloading decisions in MEC-enabled SAGIN. The goal is to minimize the user-experienced latency while considering the energy consumption constraint for UAV and LEO.

In [73], the authors exploited block coordinate descent (BCD) and successive convex approximation (SAC) and proposed to use AO to solve the joint UAV position optimization and resource allocation task offloading problem in SAGIN. The authors proposed an edge and cloud computing-assisted SAGIN framework where UAVs are edge servers that receive tasks from IoT devices and LEOs are relay platforms to transmit tasks to a cloud server. The proposed problem is a min-max problem with the goal of minimizing the maximum computation delay for all IoT devices. Specifically, it takes into account the scheduling for association control, computing/frequency/transmission power allocation, and UAV deployment optimization.

In [74], the authors proposed to use BSUM to solve the problem of joint UAV trajectory planning resource allocation and task offloading problem. The goal is to minimize the energy consumption of both IoT devices and UAV. In [75], an offloading scheme for space/aerial-aided MEC network is proposed, where Lyapunov optimization together with branch-and-bound (BnB) are employed to decompose the long-term problem into per-slot one. In this work, both LEO satellites and ground users can generate tasks and offload these tasks to space/air segments. Additionally, LEO can offload their tasks either to one-hop away air/space segments

or multi-hops away cloud computing centers. The problem is formulated with the goal of minimizing long-term task completion delays for both LEO and ground users, considering the dynamically changing SAGIN environment.

Authors in [76] proposed a joint task offloading and power control scheme, where task offloading is determined using an approximation algorithm with fixed power allocation, and the optimal power allocation is obtained by solving a closed-form optimization problem.

In [77], an alternating optimization approach is proposed where four sub-problems—user scheduling, partial offloading decisions, computational resource and bandwidth allocation, and UAV trajectory planning—are solved alternately until convergence. The goal of this work is to minimize the power consumption of ground users (GU) and UAVs, and the objective function is formulated as the weighted sum of GU power consumption and UAV power consumption. In this work, each UAV integrates an MEC server and can perform edge computing, and LEO serves as a relay node that GU can offload their tasks to cloud servers through multi-hop LEO communications. The problem is formulated as a non-convex MINLP problem, with feasibility verification and admission control integrated for overloaded network scenarios.

In [78], the authors incorporated an alternating optimization (AO)-based method, Dinkelbach algorithm, and successive convex approximation (SAC) to maximize energy efficiency and optimize UAV trajectory, resource allocation, and task offloading in SAGIN. They proposed a caching/MEC-assisted SAGIN framework for satisfying the requirements of extended reality-enabled IoT (XRI) applications. In this work, UAV is the air segment, and LEO is the space segment, each equipped with catching devices and MEC servers. To efficiently model the characteristics of XRI, four offloading procedures are considered in this work in accordance with the type of information (shared or private). For private data, it can only be offloaded to UAV/LEO and the shared data can be offloaded to LEO considering the caching status.

In [79], the authors proposed a matching game and Lagrange dual method to solve the energy maximization problem under the NOMA-enabled SAGIN IoT networks. In this work, Ldatellite serves as a relay platform to achieve a wide range of coverage, and UAV provides

dedicated coverage support for a small area. The problem is formulated as MINLP problem and divided into two subproblems, which are solved alternatively until final convergence. The EE is maximized by allocating frequency resources and transmission power to UAVs.

However, these approaches separate their formulated problems into multiple sub-problems and solve these sub-problems alternatively, which leads to performance loss. Furthermore, these approaches need to resolve their formulated problems when the SAGIN environment changes, leading to high computational complexity and low system scalability.

For learning-based approaches, in [80], the authors proposed a hybrid centralized and decentralized deep-Q network (DQN) method to minimize the total learning cost of the whole system in a cloud server-equipped satellite network. The proposed method can utilize the advantages of both centralized and decentralized learning, where distributed training is used when the long-distance transmission needs to be avoided, and centralized training is used when the transmission cost is low. In this work, the implementation of MEC in SAGIN and the combination of cost for transmission and processing are considered, which is a rare combination that can be found in other works. Also, the authors proposed mathematical models for the cost of three scenarios, distributed learning, centralized learning, and the combination of both, which takes into account exceptions of the 6G network.

In [81], the authors proposed a SAGIN model that integrates edge/cloud computing architecture to perform onboard task processing with the constraint of energy consumption and total available computing/communication resources. In this framework, only UAVs implemented MEC servers and can serve as edge nodes to process tasks. Satellites served as relay platforms to provide connections to a cloud server, which enables the offloading of tasks to the cloud. The goal of this paper is to minimize delay by allocating computing resources to virtual machines (VMs) and performing task offloading to edge/cloud servers. This problem is divided into two layers, where the lower layer is the task offloading/resource allocation for UAV edge servers only, and the upper layer is the task offloading/resource allocation for the whole SAGIN network. The former is formulated as a MILP problem, which is solved using a heuristic method, and the decision-making process for the latter is formulated as MDP, and DQN is proposed to solve this problem.

In [82], the authors proposed an attention-enabled proximal policy optimization (A-PPO) method to generate offloading decisions. This paper utilized UAV to collect tasks from IoT devices and perform the task offloading. Therefore, only air and space segments are considered. This work focuses on making offloading decisions for multiple tasks at the same time. These tasks are organized in a directed adjacency graph (DAG), where lower-priority tasks serve as the child node of higher-priority tasks for enabling the linear transformation of two-dimensional DAG. The proposed A-PPO utilizes advanced optimization and approximation techniques. It utilizes the first-order approximation and a simplified version of the surrogate objective function for the training of GRU-based RNN. Additionally, the adaptation of GRU-based RNN, which is an architecture that has proven to have good performance in handling sequential dependencies and complex temporal dynamics, can boost the performance of A-PPO by providing more accurate extracted features. Furthermore, the introduction of attention mechanisms enhances the model's ability to focus on critical task features and dependencies, which can improve task offloading/ resource allocation performance.

In [83], the authors separated the main problem into two subproblems, namely the resource allocation subproblem and the task offloading subproblem. An attention-enabled multi-agent PPO is proposed to solve the task offloading subproblem, and the resource allocation subproblem is relaxed into a convex problem and solved by convex optimization methods. The goal of this work is to minimize rental prices, and the problem is formulated as a MINLP problem. In [84], the authors proposed a slicing-based task offloading framework for the space-air-ground integrated vehicular network (SAGIVN) with dynamically changing slicing window length. Specifically, deep DQN is used to solve the task scheduling problem, a convex optimization method is used to solve the resource allocation problem, and a heuristic method is used to adjust the slicing window length. The goal of this work is to maximize the number of tasks completed.

In [43], the authors proposed a collaborative multi-agent DRL method to solve the task offloading and resource allocation problem in SAGIN. In this work, cloud/edge servers enabled SAGIN is considered where UAV is utilized as the air segments. The goal of this

work is to reduce energy consumption and latency. Here, tasks can only be generated by ground users and the correlation between tasks is molded as DAGs.

In [85], a differentiated federated reinforcement learning (DFRL) method is proposed to solve the traffic offloading problem in SAGIN. Specifically, multiple agents are utilized in this work, where each agent generates one specific traffic offloading policy. As the characteristics of each region in SAGIN are different, the problem is formulated as a decentralized partially observable MDP (DEC-POMDP), and DFRL is proposed to solve this problem. The DRL method used in DFRL is soft actor-critic (SAC), which is known for its ability to encourage exploration by adding an entropy term. The reward for DFRL is formulated as the weighted sum of network packet delay of all tasks in the network, and a joint target network of all agents is proposed, referred to as the global trend model. In this work, the air segment is constructed by UAV, and the space segment consists of two layers, the LEO and geostationary earth orbit satellites (GEO). The trajectory of UE is predefined as circular motion/random movement, and both LEO/GEO serve as relay nodes to forward tasks to cloud servers.

In [86], the authors integrate digital twin technology into SAGIN to address the difficulties of task offloading in vehicular networks (VN). Furthermore, a multi-agent DRL method is proposed to provide online task offloading decisions under several network constraints. The authors proposed a framework that combines digital twin (DT) with SAGIN with the objective of supporting URLLC in VN. The DT-SAGIN consists of two layers, the terrestrial digital twin layer and the non-terrestrial digital twin layer, both layers are formed in the cloud with high deployment flexibility. The goal of this work is to minimize latency and reduce energy consumption by performing task offloading and resource allocation with the constraint of total available resources and edge processing requirements. In this work, partial task offloading is performed, where each task can be offloaded to multiple LEOs/UAVs.

In [87], a dependent task offloading computational collaboration scheme based on deep deterministic policy gradient (DDPG) is proposed to solve the resource allocation problem in SAGIN with the consideration of task dependencies. This framework is formulated considering the distinct quality of experience (QoE) for different users, where the air segment is constructed by UAV and the space segment by LEO. Both UAV and LEO integrate MEC and

have the ability to process tasks offloaded from ground users. In this work, task dependencies are molded by DAG, and the SDN controller is utilized to control the process of multi-application migration. The goal is to minimize the delay of all tasks and maximize the number of completed tasks.

In [88], the authors formulated a task offloading and resource allocation problem considering energy harvesting and utilized proximal policy optimization (PPO) to solve this problem. Specifically, this work focuses on the utilization of SAGIN in solving the challenges faced in smart city deployments, like large-scale stable connection requirements and high energy consumption caused by rapid information exchange. SAGIN also integrates MEC with the goal of optimizing system efficiency. This network consists of three tires: the ground tier constructed by user devices (UDs), the air tier constructed by UAVs, and the space tile constructed by satellites. Both UDs and UAVs can generate computational tasks, and these tasks can be offloaded to UAVs and satellites for processing.

However, these DRL-based approaches have several limitations. Firstly, work [43, 85, 86, 88] does not take into account the correlation between tasks, which limits the type of services covered. In the work [87], task dependency is considered, but the graphical structural information is not explored, and only task offloading decisions are provided. Secondly, none of these methods account for the inherent topological structure of the SAGIN states. Since graph neural networks perform better with graph-structured data compared to MLPs, these methods are inefficient at capturing features in the SAGIN environment due to their reliance on MLP-based DRL. This results in slow convergence and performance degradation in terms of minimizing latency. Lastly, all these methods generate discrete and continuous actions separately. Specifically, they either separate the problem into two subproblems or generate one type of action at each agent, which leads to potential performance loss. Furthermore, the size of the discrete action space usually grows exponentially with the number of segments, none of the above methods provide a solution for decreasing the size of the action space to improve system scalability. Table3.3 shows the main focus of these learning-based works.

TABLE 3.3. Existing Learning-based Works

	[43]	[85]	[86]	[87]	[88]	[80]	[82]	[83]	[81]	[84]	[89]
Task dependency	✓			✓				✓			
Task graph	✓			✓			✓				
UE-HAPS/UAV-LEO graph		✓									✓
Graph fusion											
Hybrid action space	✓				✓		✓		✓		✓
Variable number of UE/HAPS/LEO											
Sequential decision-making problem									✓	✓	✓

3.3 Conclusion

For both 5G and SAGIN, state-of-the-art approaches have many limitations. These limitations can be categorized into two types: DRL-originated limitations and system model-originated limitations. Specifically, DRL-originated limitations include

(1) DRL approaches cannot handle long-term constraints.
(2) DRL approaches cannot generate both discrete and continuous actions in an end-to-end manner.
(3) DRL approaches cannot maintain performance when the difference between the testing and training environment is large.
(4) DRL approaches cannot adapt to changing dimensions of output actions.

To mitigate these limitations, state-of-the-art DRL approaches need to be modified. The system model-originated problems are

(1) The system model does not consider different distributions for different types of services.
(2) The system model is not comprehensive enough.
(3) The service types included in the system model are very limited.
(4) The system model is oversimplified and cannot reflect real work scenarios.

to mitigate these system model-originated limitations, a system model that can reflect real work scenarios needs to be formulated.

In the following Chapters 4, 5, and 6, we present three works that try to mitigate all these limitations mentioned above. To solve the DRL-originated limitations, we integrate additional elements into DRL approaches. Specifically, in Chapter 4, we integrate Lyapunov optimization into DRL to address long-term constraints. In Chapter 5, meta-learning is integrated into DRL to mitigate the difference between the testing and training environment. In Chapter 6, the reincarnating technique is utilized to reduce the training complexity of DRL approaches. To solve the system model-originated limitations, we proposed a more comprehensive system model compared with state-of-the-art models in each chapter.

CHAPTER 4

Resource allocation for network slicing in 5G network

In this chapter, we present the resource allocation strategy referred to as PW-DRL for 5G RAN slicing. This work focuses on inter-task features, as discussed in Chapter 1 (introduction), where the main problem lies in allocating resources to different requests while satisfying QoS for different services at the same time. In the following section, we first give a brief summary of the discussion related to 5G RAN slicing in Chapter 1. Then, we present the system model and formulate the resource allocation problem. After that, each part of the proposed PW-DRL is introduced, and simulation results validate its performance. Finally, we conclude this chapter.

An efficient 5G resource allocation approach is essential for ensuring optimal system performance, supporting the diverse requirements of users, and satisfying QoS requirements for emerging services. In Chapter 1, we presented three main reasons for the importance of efficient resource allocation approaches, which can be summarized as follows.

(1) Successful deployment of network slice requires efficient resource allocation: Network slicing is the key enabler in 5G to satisfy diverse QoS requirements of different services simultaneously. It is a technology not only utilized in 5G but also in other domains. When applying it to 5G, network slicing technologies can be categorized into RAN slicing, TN slicing, and CN slicing. In most works, TN slicing is integrated with CN slicing and is referred to as CN slicing. The RAN part performs the function of connecting user mobile devices to the network, allocating resources based on user requests, mapping virtualized resource blocks (RB) to different network slices, and managing and reporting physical link status. It plays a very important part in

establishing functional network slices. All these functions of RAN slicing require highly efficient resource allocation schemes. Here, resources to be allocated can be categorized into communication resources and network resources. Communication resources include the number of resource blocks, frequency bands, and transmission power. Network resources include the routing path, processing capabilities of each router, and IP addresses.

(2) 5G network is service-based, and the deployment of resources to each function affects the performance of 5G: 5G is built on a service-based architecture. Specifically, a traditional network has dedicated hardware to perform certain functions. For example, in 2G, the authentication of subscribers is performed by AC. In 5G, this authentication function is not performed by dedicated hardware but is performed by a function called Access and Mobility Management Function (AMF). Different functions interact with each other through predefined interfaces. These functions share the same physical infrastructure, and the allocation of computational/communication resources to these functions dynamically and efficiently ensures that the network adapts to varying traffic demands and service requirements. For example, services with low latency requirements need to be allocated with more network resources, and data-intensive applications require larger bandwidth. Additionally, efficient resource allocation can reduce the total resources needed thereby reducing the initial investment cost and operational costs.

(3) 5G network consists of a large number of users and diverse service types, which requires a highly adaptive resource allocation scheme: The 5G network consists of a vast number of users and different types of services. The requirements of services usually include three components: latency, transmission, and connectivity. For example, eMBB requires high transmission rates, URLLC requires low latency, and mMTC requires dense connectivity. An efficient resource allocation scheme is important for satisfying these requirements under limited available resources. Another reason for needing an adaptive resource allocation scheme is the variability in user mobility and network conditions.

Next, in the previous chapter, we present a detailed literature review on state-of-the-art resource allocation schemes for 5G. In general, these approaches can be categorized into optimization-based and learning-based approaches. However, these approaches suffer from the following limitations. For optimization-based approaches

(1) These approaches require a large number of iterations to solve the resource allocation problem with high computational complexity.
(2) These approaches need to solve the formulated problem again once the network parameters change (e.g., channel state information), leading to prohibitory high computational complexity and preventing the adaption of these approaches to the changing network environment.

Learning-based approaches are proposed to mitigate the limitations of optimization-based ones. In learning-based approaches, resource allocation decisions can be learned through dynamic and data-driven optimization. Specifically, learning-based approaches like DRL can generate resource allocation decisions by directly interacting with the environment and maximizing a predefined reward function. After training, the complexity of online deployment of learning-based approaches is low. However, these learning-based approaches also have limitations, which are listed below.

(1) A common distribution like the Poisson distribution is used to model the arrival process of user requests. In the real world, the arrival process for different types of user requests should be different.
(2) DRL-based approaches can only generate discrete or continuous action and not both. The resource allocation problem are usually formulated as a MINLP and contains both discrete and continuous variables. Thus, DRL-based approaches need to separate/relax the original problem, which leads to system performance loss.
(3) These learning-based approaches focus on solving one-shot optimization problems. Specifically, the one-shot optimization only considers a static resource pool, ignoring how allocation decisions affect available resources across different problem instances. Consequently, it fails to capture the inherent dependencies over multiple timesteps

and ignores long-term performance metrics like task completion ratio and resource utilization ratio.

To mitigate these limitations, we present the resource allocation strategy referred to as PW-DRL for 5G RAN slicing in this chapter.

4.1 System Model

4.1.1 Network Model

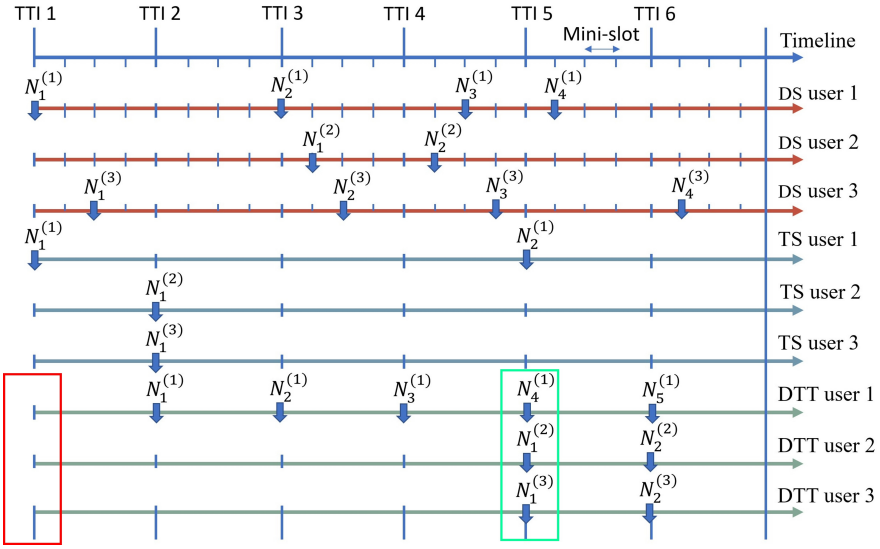
In this paper, we consider the downlink transmission with a base station (BS) and incorporate orthogonal frequency-division multiple access (OFDMA) to avoid user interference. A time-slotted model is employed as shown in Fig. 6.10a with the time horizon divided into equal-length transmission time intervals (TTIs) indexed by $\mathcal{T} = \{1, 2, \ldots, T\}$, each further divided into mini-slots denoted by $\mathcal{S} = \{1, 2, \ldots, S\}$. We also consider three types of services, namely throughput-oriented (TO), delay-sensitive (DS), and delay-throughput-tolerant (DTT), and define three network slices accordingly. Each network slices contains U_m users with $m \in \{TO, DS, DTT\}$. For each user, we denote the i-th request as $N_i^{(u)}$.

The resource allocation model is shown in Fig. 6.10c. Specifically, the resource units allocated to one TO request will not be released until the task is finished (usually several TTIs), that for DTT stay occupied for only one TTI, and that for DS request one mini-slot. Note that resource units cannot be allocated to other requests until released.

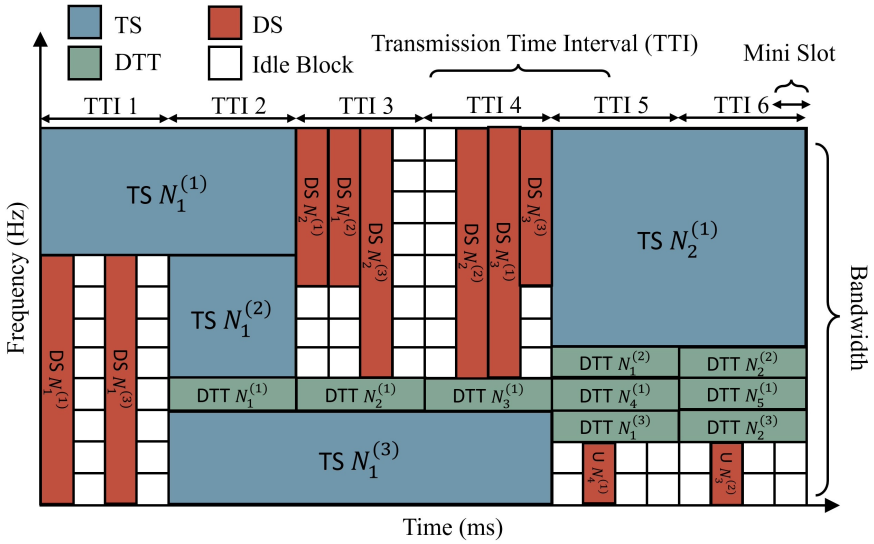
4.1.2 Channel Model

The block fading wireless channel can be divided into two parts, namely small-scale fading and large-scale fading. We assume Rayleigh fading for small-scale fading and path loss for large-scale fading. The mathematical representation for path loss is

$$L_p(d_u)(\mathrm{dB}) = L(d_0)(\mathrm{dB}) + 10nlog_{10}(\frac{d_u}{d_0}) + X_{\sigma_u}(\mathrm{dB}) \tag{4.1}$$



(A) The time-slotted model for requests.



(B) The time-slotted model for resource allocation.

FIGURE 4.1. The time-slotted model of RAN slicing

where X_{σ_u} is a Gaussian random variable with zero-mean and σ_u standard deviation, $L_p(d_u)$ represents the path loss for u-th user at distance d_u, $L(d_0)$ represents the path loss from the transmitter at position d_0 to the reference point and is calculated using free space path loss model $L(d_0) = \left(\frac{4\pi d_0}{\lambda}\right)^2$. Usually $d_0 = 100$ and λ represents the wavelength. For Rayleigh fading, the probability density function is

$$f(x) = \frac{2x}{\overline{P}} exp\left[-\frac{x}{\overline{P}}\right] \tag{4.2}$$

where $x > 0$ represents the envelope amplitude and $\overline{P}$ represents the mean power.

4.1.3 Throughput-Oriented Slice

The TO slice is designed to suit the needs of services with high data transmission rate requirement across a wide range of area. Three traffic distributions are selected to model the diverse traffic characteristics in TO slice and each distribution contains of: package size m and number of TTO intervals between two requests w.

In the first traffic distribution, m can be drawn from the following distribution.

$$f_1^m(x) = \frac{1}{\sqrt{2\pi}\sigma x} e^{\frac{-(\ln x - \mu)^2}{2\sigma^2}}, \;\; x > 0 \tag{4.3}$$

and w follows

$$f_1^w(t) = \lambda e^{\lambda t}, \;\; t \geq 0. \tag{4.4}$$

This distribution is chosen to reflect the features of FTP traffic. For FTP, its distribution is characterized by a heavy tail, rendering the truncated Lognormal Distribution as the appropriate choice.

In the second distribution, m and w can be selected based on

$$f_2^{(m,w)}(x \mid k_1, k_2, k_3) = \left(\frac{1}{k_2}\right)\left(1 + k_1 \frac{(x - k_3)}{k_2}\right)^{-1-\frac{1}{k_1}}, \tag{4.5}$$

where the hyperparameters k_1, k_2, k_3 represent the shape, scale, and threshold, respectively. Note that different values of k_1, k_2, k_3 are utilized for m and w. This distribution is chosen to reflect the features of video streaming traffic. Voice streaming traffic distribution tail follows the power law model, which makes the truncated Pareto distribution a good choice.

In the third traffic distribution, w is selected based on

$$\mathbf{P} = \begin{bmatrix} 1 - \varphi_1 & \varphi_1 \\ \varphi_2 & 1 - \varphi_2 \end{bmatrix}, \tag{4.6}$$

where φ_1 and φ_2 are predefined values that reflect the probability of changing one state to another, and m is a fixed value. Specifically, φ_1 and φ_2 represent the state transition

probability from state 1 (with a request) to state 2 (without a request) and state 2 to state 1, respectively. This distribution is chosen to reflect the characteristics of VoIP traffic. VoIP has an inherent pattern of alternating between voice bursts and silences, which makes a two-state Markov process with on and off states a suitable distribution.

The TO transmission rate can be represented as

$$r_u^{\text{TO}}(t) = \frac{m}{l_u^{\text{TO}}}, \tag{4.7}$$

where l_u^{TO} is user u's latency threshold. Accordingly, the corresponding resource units and power can be calculated based on

$$r_u^{\text{TO}}(t) = W_c \alpha_u(t) N_u^{\text{TO}}(t) \log_2 \left(1 + \frac{p_u^{\text{TO}}(t) g_u(t)}{\sigma_0^2}\right), \tag{4.8}$$

The power allocation, resource unit assignment, and user selection decision for the t-th time slot are represented by the power $p_u^{\text{TO}}(t)$, number $N_u^{\text{TO}}(t)$, and binary decision $\alpha_u(t)$, respectively. The channel gain is denoted by $g_u(t)$, and the standard deviation of the additive white Gaussian noise is σ_0.

4.1.4 Delay-Sensitive Slice

The DS slice is designed to support services with stringent latency and reliability requirements.

To model m and w for DS, we use a Markov-modulated Poisson process (MMPP).

Accordingly, the transmission rate is

$$r_u^{\text{DS}}(t_m) \approx \frac{W_c T_f}{\ln 2} \left\{ N_u^{\text{DS}}(t_m) \alpha_u(t_m) \ln \left[1 + \frac{a_u p_u^{\text{DS}}(t_m) g_u(t_m)}{N_0 W_c}\right] - \sqrt{\frac{1}{T_f W_c}} f_Q^{-1}\left(\epsilon_u^d\right) \right. \tag{4.9}$$

Here, t_m is the mini-slot number, ϵ_u^d is the error for decoding, the duration for one mini-slot is T_f, and finally the inverse Q function is f_Q^{-1}.

The corresponding resource units can be calculated as

$$N_u^{\mathrm{DS}}(t_m) = \frac{\frac{\ln 2 E_u^B(D_q^u, \epsilon_u^q)}{W_c T_f} + \sqrt{\frac{1}{T_f W_c}} f_Q^{-1}\left(\epsilon_u^d\right)}{\alpha_u(t_m) \ln\left[1 + \frac{a_u p_u^{\mathrm{DS}}(t_m) g_u(t_m)}{N_0 W_c}\right]}, \tag{4.10}$$

where $E_u^B(D_q^u, \epsilon_u^q)$ is the effective bandwidth with D_q^u the queuing delay and ϵ_u^q the queuing delay violation probability, which can be calculated as [90]

$$E_u^B(D_q^u, \epsilon_u^q) = \frac{m \ln\left(1/\epsilon_u^q\right)}{D_q^u \ln\left[\frac{T_f \ln\left(1/\epsilon_u^q\right)}{\lambda_u D_q^u} + 1\right]} (\mathrm{bits/s}), \tag{4.11}$$

where λ_u is a predefined parameter.

4.1.5 Delay-Throughput-Tolerant Slice

The DTT slice provides support for services with stable connection requirements.

The package size m follows

$$f^{\mathrm{DTT}}(t) = \frac{t^{\beta_1 - 1}(T-t)^{\beta_2 - 1}}{T^{\beta_1 + \beta_2 - 1} Beta(\beta_1, \beta_2)}. \tag{4.12}$$

The transmission rate is

$$r_u^{\mathrm{DTT}}(t) = \frac{m}{l_u^{\mathrm{DTT}}}, \tag{4.13}$$

where l_u^{DTT} equals to one TTI. Similar to TO request, the number of resource units is

$$N_u^{\mathrm{DTT}}(t) = \frac{r_u^{\mathrm{DTT}}(t)}{W_c \alpha_u(t) \log_2\left(1 + \frac{p_u^{\mathrm{DTT}}(t) g_u(t)}{\sigma_0^2}\right)}. \tag{4.14}$$

4.2 Problem Formulation

In this section, we present the formulated long-term resource allocation problem with short-term total available resources constraints (resource units and power). Specifically, the objective function is designed in accordance to the system model presented in Fig 4.1. For TO or DTT

user u, the objective function is formulated as

$$R_u^{\text{TO/DTT}}(t) = k_u \alpha_u^{\text{TO/DTT}}(t) N_u^{\text{TO/DTT}}(t), \tag{4.15}$$

where k_u represents the priority weight for user u. For delay sensitive (DS) user u, it is formulated as

$$R_u^{\text{DS}}(t_m) = k_u \alpha_u^{\text{DS}}(t_m) N_u^{\text{DS}}(t_m). \tag{4.16}$$

The tuple $(N_u^{\text{TO}}, \alpha^{\text{TO/DTT}}, R_u^{\text{TO/DTT}})$ only has a value at the beginning of each TTI and can be represented as

$$\begin{aligned}
N_u^{\text{TO/DTT}}(t_m) &= \begin{cases} N_u^{\text{TO/DTT}}(t) & t_m = S(t-1)+1 \\ 0 & \text{otherwise} \end{cases} \\
\alpha_u^{\text{TO/DTT}}(t_m) &= \begin{cases} \alpha_u^{\text{TO/DTT}}(t) & t_m = S(t-1)+1 \\ 0 & \text{otherwise} \end{cases} \\
R_u^{\text{TO/DTT}}(t_m) &= \begin{cases} R_u^{\text{TO/DTT}}(t) & t_m = S(t-1)+1 \\ 0 & \text{otherwise} \end{cases}
\end{aligned} . \tag{4.17}$$

Accordingly, the mini-slot reward is

$$R(t_m) = \sum_{u=1}^{U} R_u(t_m), \tag{4.18}$$

With the definition of this mini-slot reward, the long-term reward is

$$\lim_{T\to\infty} \frac{1}{ST} \sum_{t_m=1}^{ST} R(t_m), \tag{4.19}$$

based on which the resource allocation problem can be formulated as

$$\mathcal{P}_1: \quad \underset{\boldsymbol{\alpha},\mathbf{p}}{\text{maximize}} \ \lim_{T\to\infty} \frac{1}{ST} \sum_{t_m=1}^{ST} R(t_m) \tag{4.20}$$

$$\text{subject to } \sum_{u=1}^{U} \alpha_u(t_m) N_u(t_m) \leq N(t), \tag{4.21a}$$

$$\lim_{T\to\infty} \frac{1}{ST} \sum_{t_m=1}^{ST} \mathbb{E}[\boldsymbol{\alpha}^T(t_m)\mathbf{p}(t_m)] \leq P, \tag{4.21b}$$

$$\boldsymbol{\alpha}(t_m) \in \{0,1\}^U, \tag{4.21c}$$

Here, constraint (4.21a) indicates that the total allocated computational resources should be within the available resources. Constraint (4.21b) represents the long-term constraint of total allocated power should not exceed power threshold P. Constraint (4.21c) shows two ways to process each user request: accepted ($\alpha_u = 1$) or declined ($\alpha_u = 0$). We have the following observations.

(C1) Different from constraint (4.21a), constraint (4.21b) is a soft one. For DL transmission, the power constraint is not as stringent as the resource unit constraint. To stabilize the system and lower the computational complexity, an average power constraint is used instead. However, this long-term constraint (4.21b) is still difficult to be fulfilled due to the random nature of user request generation.

(C2) Knowing $\mathbf{p}(t_m)$, the problem of determining optimal $\boldsymbol{\alpha}(t_m)$ is simplified into winner determination problem (WDP). The auctioneer has an item (resource unit) and the item has several copies ($N(t_m)$) available. A number of bidders submit their bids including ($N_u(t_m)$) and the price for each item ($k_u N_u(t_m)$). The auctioneer chooses the best combination of bidders to maximize the received payment. As WDP is an NP-complete problem and is impossible to solve directly, it is hard to solve (4.20) even with the knowledge of $\mathbf{p}(t_m)$ using the optimization method.

(C3) Action $\boldsymbol{\alpha}(t)$ can explicitly influence the next timestep resources. For example, in Fig. 6.10c, TO user 1 request 1 $N_1^{(1)}$ arrives and is accepted at the beginning of TTI 1, the allocated resource units $N_1^{\text{TO}}(t)$ will be occupied for two TTIs, and cannot be allocated to other user requests. At the beginning of TTI 2, the number of available resource units is $N - N_1^{\text{TO}}(t)$. If TO 1 is not accepted at TTI 1, the available resource

units at TTI 2 will be N. In other words, the action of the auctioneer taken at previous timestep will affect the number of available units at current timestep.

In our problem, we need to consider both long-term and instantaneous constraints that are difficult to be fulfilled. Also, methods that can solve WDP problem are not suitable for our problem as for we also need to allocate power at the same time. Additionally, the time length for decision making is very short (one mini-slot), which requires fast decision making algorithm with low complexity. Conventional methods cannot tangle these difficulties.

4.3 Detail Description of PW-DRL

4.3.1 Motivation

To address the aforementioned challenges, we propose PW-DRL, which can directly generate both computational resource allocation decisions, $\boldsymbol{\alpha}(t_m)$ and power allocation decisions, $\mathbf{p}(t_m)$. Specifically, PW-DRL directly interacts with the environment and learns the mapping between states and actions in accordance with the designed reward function. It takes into account both short-term and long-term constraints. In the following, we discuss how each challenge affects the formation of PW-DRL.

(1) To handle (C1), the MINLP problem is separated into two subproblems: computational resource allocation and transmission power allocation. This can eliminate the need to search for a large hybrid action space.

(2) To address (C2), we leverage TRPO to learn the mapping between actions $\boldsymbol{\alpha}(t_m)$ and current observations $N(t_m)$. TRPO maximizes a surrogate objective function, which is constructed by an estimation of advantage function, under a KL divergence constraint, with monotonic improvement guarantee. We also adapt Lyapunov optimization method to generate $\mathbf{p}(t_m)$ given $\boldsymbol{\alpha}(t_m)$, where virtual queues are formulated to address long-term constraints.

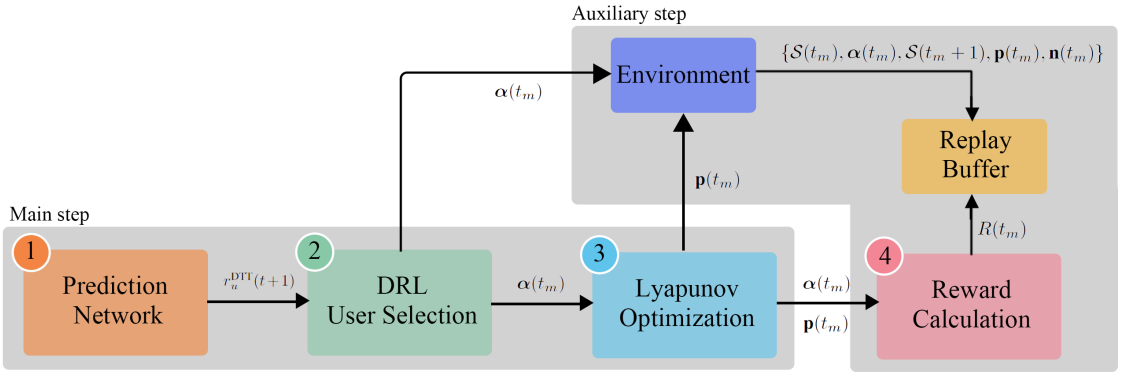


FIGURE 4.2. The architecture of the proposed PW-DRL.

(3) To address (C3), both online and offline prediction network is proposed to predict $r_u^{\mathrm{DTT}}(t+1)$ and $\lambda(t+1)$ with the knowledge of $r_u^{\mathrm{DTT}}(t)$ and $\lambda(t)$. Note that other variables cannot be predicted due to the lack of correlation among time steps.

4.3.2 Detailed description for each part of PW-DRL

4.3.2.1 Prediction Network (Step 1 in Fig. 4.2)

In previous section, we mentioned that in our proposed model, previous timestep power and computational resource allocation actions will influence current timestep available resources and implicitly influence current timestep actions. In order to capture this dependencies among actions across different timesteps, a prediction network consists of two parts is formulated, namely online and offline prediction network. Specifically, the online prediction network predicts the next timestep DTT user requests $r_u^{\mathrm{DTT}}(t+1)$ based on current timestep ones $r_u^{\mathrm{DTT}}(t)$. The offline prediction network provide the next timestep Markov state for DS user request generation based on previous ones. Both $r_u^{\mathrm{DTT}}(t)$ and Markov states have strong correlations across multiple timesteps, which guarantee the predictability of both variables. For $r_u^{\mathrm{DTT}}(t)$, this correlation arises from the periodic patterns of beta distribution. Note that other variables cannot be predicted due to the fast changing network environment. For example, the number of required resource blocks N_u, transmission rates (DS/TO user requests), and inter-arrival time between user requests w have no correlation accorss multiple

timesteps. The features of these variables are included in the environment states formulation and can be captured by TRPO directly.

4.3.2.2 DRL User Selection (Step 2 in Fig. 4.2)

In Step 2, DRL generates frequency resource allocation decisions $\boldsymbol{\alpha}(t_m)$ based on the prediction results from previous step $\mathbf{r}^{\mathrm{DTT}}(t+1))$ and current timestep transmission rate for all users $\mathbf{r}^*(t_m) = [r_1^*(t_m), r_2^*(t_m), \cdots, r_U^*(t_m)]$. The problem of generating $\boldsymbol{\alpha}(t_m)$ ($\mathcal{P}_1$) can be formulated as a Markov decision process (MDP) problem and which is defined by a tuple $(\mathcal{S}, \mathcal{A}, O, \zeta, \gamma, \rho_0)$. Here, $\mathcal{S}$ represents the state, $\mathcal{A}$ is the finite action set of action $\boldsymbol{\alpha}(t_m)$, and $O : o\left(\mathbf{s}(t_m+1) \mid \mathbf{s}(t_m), \boldsymbol{\alpha}(t_m)\right) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathcal{R}$ is the transition probability distribution. In addition, $\rho_0 : \mathcal{S} \to \mathcal{R}$ represents the distribution of the initial state, ζ the reward, and $\gamma \in (0, 1)$ the discount factor. The state and action set are designed as follows.

(1) *State*: The state set is $\mathcal{S} = \{N(t), \mathbf{r}^*(t_m), \mathbf{r}^{\mathrm{DTT}}(t+1)\}$, where $N(t)$ represents the number of resource units, $\mathbf{r}^*(t_m)$ represents transmission rate, and $\mathbf{r}^{\mathrm{DTT}}(t+1)\}$ represents the prediction results for next timestep DTT user transmission rate.

(2) *Action*: The action set is the collection of all combination of $\boldsymbol{\alpha}$. It can be represented as $\mathcal{A} = \{0, 1\}^U$ where U is the total number of users. For example, $\mathcal{A} = \{[0, 0], [0, 1], [1, 0], [1, 1]\}$ for $U = 2$, and the number of actions in the action set can be represented as $\mathcal{A} = \{[0, 0], [0, 1], [1, 0], [1, 1]\}$ with $|\mathcal{A}| = 2^U = 4$.

We also set the distribution for ρ_0 and π_0 as uniform distributions.

In this paper, TRPO is utilized as the DRL method to learn the mapping between state and action. The reasons for choosing TRPO instead of other DRL methods are listed as follows.

(1) TRPO has monotonic policy improvement guarantee while other DRL methods do not. This feature of TRPO can guarantee the next timestep policy is a better one compared with previous timestep, and resulting in a more stable convergence.

(2) TRPO incorporates the advantage function A_π to ensure the smooth improvement of policy in TRPO. This is because A_π accounts for the difference between Q-value

function and state value function and can smooth out variations caused by sudden value changes.

(3) TRPO focusing on policy difference instead of policy itself can naturally disregard minor variations and smooth out significant ones. This enables the faster learning process of value function which is not applicable in other DRL methods.

The advantage function A_π can be formulated as

$$A_\pi(\mathbf{s}(t_m), \boldsymbol{\alpha}(t_m)) = Q_\pi(\mathbf{s}(t_m), \boldsymbol{\alpha}(t_m)) - V_\pi(\mathbf{s}(t_m)), \tag{4.22}$$

with

$$\begin{aligned} &Q_\pi\left(\mathbf{s}(t_m), \boldsymbol{\alpha}(t_m)\right) = \\ &\mathbb{E}_{\mathbf{s}(t_m+1), \boldsymbol{\alpha}(t_m+1), \ldots}\left[\sum_{l=0}^{\infty} \gamma^l \zeta(\mathbf{s}(t_m + l))\right], \end{aligned} \tag{4.23}$$

and

$$V_\pi\left(\mathbf{s}(t_m)\right) = \mathbb{E}_{\boldsymbol{\alpha}(t_m), \mathbf{s}(t_m+1), \ldots}\left[\sum_{l=0}^{\infty} \gamma^l \zeta(\mathbf{s}(t_m + l))\right]. \tag{4.24}$$

The advantage of new policy π' over old policy π^{old} can be represented as

$$\xi(\pi') = \xi(\pi^{old}) + \mathbb{E}_{\mathbf{s}_0, \boldsymbol{\alpha}_0, \cdots \sim \pi'}\left[\sum_{t=0}^{\infty} \gamma^t A_{\pi^{old}}\left(\mathbf{s}(t_m), \boldsymbol{\alpha}(t_m)\right)\right], \tag{4.25}$$

with

$$\xi(\pi) = \mathbb{E}_{s_0, a_0, \ldots}\left[\sum_{t_m=0}^{\infty} \gamma^{t_m} \zeta(\mathbf{s}(t_m))\right], \tag{4.26}$$

Accordingly, we can get

$$\begin{aligned} \xi(\pi') = \xi(\pi^{old}) + \sum_{\mathbf{s}} \sum_{t_m=0}^{\infty} \gamma^{t_m} P\left(\mathbf{s}(t_m) = \mathbf{s} \mid \pi'\right) \\ \times \sum_{\boldsymbol{\alpha}} \pi'(\boldsymbol{\alpha} \mid \mathbf{s}) A_{\pi^{old}}(\mathbf{s}, \boldsymbol{\alpha}). \end{aligned} \tag{4.27}$$

Representing the discounted transition probability distribution for policy π at state $\mathbf{s}$ as

$$\eta_\pi(\mathbf{s}) = \sum_{t_m=0}^{\infty} \gamma^{t_m} P\left(\mathbf{s}(t_m) = \mathbf{s} \mid \pi\right) \tag{4.28}$$

and substituting (4.28) into (4.27), we can get

$$\xi(\pi') = \xi(\pi^{old}) + \sum_{\mathbf{s}} \eta_{\pi'}(\mathbf{s}) \sum_{\boldsymbol{\alpha}} \pi'(\boldsymbol{\alpha} \mid \mathbf{s}) A_{\pi^{old}}(\mathbf{s}, \boldsymbol{\alpha}). \tag{4.29}$$

We set the objective function as

$$\begin{aligned} L_{\pi^{old}}(\pi') =& \xi(\pi^{old}) + \\ & \sum_{\mathbf{s}} \eta_{\pi^{old}}(\mathbf{s}) (\sum_{\boldsymbol{\alpha}} \pi'(\boldsymbol{\alpha} \mid \mathbf{s}) A_{\pi^{old}}(\mathbf{s}, \boldsymbol{\alpha}) + \mathscr{E}(\mathbf{s})), \end{aligned} \tag{4.30}$$

where $\mathscr{E}(\mathbf{s})$ represents entropy loss and is formulated as

$$\mathscr{E}(\mathbf{s}) = -\sum_{\boldsymbol{\alpha}} \pi'(\boldsymbol{\alpha} \mid \mathbf{s}) \ln \pi'(\boldsymbol{\alpha} \mid \mathbf{s}). \tag{4.31}$$

It is clear that maximizing (4.31) encourages the DRL agent to explore more in deciding the next timestep action. To enhance the agent's exploration capability and prevent early convergence of TRPO policy network to suboptimal policies, we formulate the objective function of TRPO considering (4.31), where the $\pi'(\boldsymbol{\alpha} \mid \mathbf{s})$ needs to be minimized. Here, in (4.30), $\eta_{\pi^{old}}(\mathbf{s})$ is utilized to approximate $\eta_{\pi'}(\mathbf{s})$. Note that this approximation can only be utilized only if the KL divergence between π^{old} and π' is small. Accordingly, the formulated problem for TRPO can be represented as [91]

$$\begin{aligned} &\underset{\pi'}{\text{maximize}} \;\; L_{\pi^{old}}(\pi') \\ &\text{subject to} \;\; D_{\mathrm{KL}}\left(\pi'(\cdot \mid s) \| \pi^{old}(\cdot \mid s)\right) \leq \delta, \end{aligned} \tag{4.32}$$

where $D_{\mathrm{KL}}\left(\pi'(\cdot \mid s) \| \pi^{old}(\cdot \mid s)\right)$ represents the KL divergence between π' and π^{old}, and δ is a predetermined threshold with a small value. The constraint in (4.32) means that the KL divergence of π^{old} and π' should within δ. It is necessary for the previous approximation and the monotonic improvement guarantee.

4.3.2.3 Lyapunov Optimization (Step 3 in Fig. 4.2)

In this subsection, we present the Lyapunov optimization method utilized in this paper to satisfy the long-term constraint (4.21b). It is integrates into TRPO to mitigate the limitation of TRPO in handling long-term constraint, and perform power allocation (optimal $\mathbf{p}(t_m)$) given

$\boldsymbol{\alpha}(t_m)$ in Step 2. Normally, Lyapunov optimization is implemented in the queuing system to maximize/minimize certain objective function while stabilizing all queues. It formulates a virtual queue to transform the long-term system constraints into per timestep constraints and is widely adaptied to address long-term system performance. Specifically, we formulate the virtual queue for $\mathbf{p}(t_m)$ and $\mathbf{n}(t_m) = [N_1(t_m), N_2(t_m), \cdots, N_U(t_m)]$ as

$$\begin{aligned} M(t_m+1) &= \max\left(M(t_m) + \boldsymbol{\alpha}^T(t_m)\mathbf{n}(t_m) - N, 0\right) \\ E(t_m+1) &= \max\left(E(t_m) + \boldsymbol{\alpha}^T(t_m)\mathbf{p}(t_m) - P, 0\right), \end{aligned} \tag{4.33}$$

Accordingly, the total queue backlog can is $\boldsymbol{\Theta}(t_m) = \{M(t_m), E(t_m)\}$ and the Lyapunov function and Lyapunov drift can be formulated as

$$L(\boldsymbol{\Theta}(t_m)) \triangleq \frac{1}{2}M(t_m)^2 + \frac{1}{2}E(t_m)^2, \tag{4.34}$$

and

$$\Delta(\boldsymbol{\Theta}(t_m)) \triangleq \mathbb{E}\{L(\boldsymbol{\Theta}(t_m+1)) - L(\boldsymbol{\Theta}(t_m)) \mid \boldsymbol{\Theta}(t_m)\}, \tag{4.35}$$

respectively. Based on (4.35), the drift for virtual queue $M(t_m)$ and $E(t_m)$ can be formulated as

$$\Delta(M(t_m)) = \frac{M^2(t_m+1) - M^2(t_m)}{2} \tag{4.36}$$

and

$$\Delta(E(t_m)) = \frac{E^2(t_m+1) - E^2(t_m)}{2}, \tag{4.37}$$

respectively. Considering $\max(a, 0) \leq a^2$, from (4.33) we can get

$$M^2(t_m+1) \leq (M(t_m) + y_1(t_m))^2 \tag{4.38}$$

and

$$E^2(t_m+1) \leq (E(t_m) + y_2(t_m))^2, \tag{4.39}$$

respectively, where

$$\begin{aligned} y_1(t_m) &= \boldsymbol{\alpha}^T(t_m)\mathbf{n}(t_m) - N \\ y_2(t_m) &= \boldsymbol{\alpha}^T(t_m)\mathbf{p}(t_m) - P. \end{aligned} \tag{4.40}$$

Thus, the upper bound for (4.36) and (4.37) can be formulated as

$$\Delta(M(t_m)) \leq \frac{y_1(t_m)^2}{2} + M(t_m)y_1(t_m), \tag{4.41}$$

and

$$\Delta(E(t_m)) \leq \frac{y_2(t_m)^2}{2} + E(t_m)y_2(t_m), \tag{4.42}$$

respectively. Based on the above formulation, we can define the drift-plus-penalty function as

$$\Gamma(\mathbf{\Theta}(t_m)) \triangleq \Delta(\mathbf{\Theta}(t_m)) - V\mathbb{E}\{R(t_m) \mid \mathbf{\Theta}(t_m)\}. \tag{4.43}$$

where V is the importance weight, which determines the importance of $\mathbb{E}\{R(t_m) \mid \mathbf{\Theta}(t_m)\}$. By substituting (4.41) and (4.42) into (4.43) we can get the upper bound of (4.43) as

$$\begin{aligned}\Gamma(\mathbf{\Theta}(t_m)) \leq & B - V\mathbb{E}\{R(t_m) \mid \mathbf{\Theta}(t_m)\} \\ & + M(t_m)\mathbb{E}\left\{y_1(t_m) \mid \mathbf{\Theta}(t_m)\right\} \\ & + E(t_m)\mathbb{E}\left\{y_2(t_m) \mid \mathbf{\Theta}(t_m)\right\},\end{aligned} \tag{4.44}$$

where B is a positive constant that satisfies

$$B \geq \frac{1}{2}\mathbb{E}\left\{y_1(t_m)^2 \mid \mathbf{\Theta}(t_m)\right\} + \frac{1}{2}\mathbb{E}\left\{y_2(t_m)^2 \mid \mathbf{\Theta}(t_m)\right\}. \tag{4.45}$$

We adopt the approach in [92] to minimize the upper bound in (4.44) instead of the original objective function in (4.43). Accordingly, by omitting the constant part of (4.44), the objective function can be formulated as

$$M(t_m)\boldsymbol{\alpha}^T(t_m)\mathbf{n}(t_m) + E(t_m)\boldsymbol{\alpha}^T(t_m)\mathbf{p}(t_m) - VR(t_m). \tag{4.46}$$

Thus, the subproblem of finding optimal $\mathbf{p}(t_m)$ can be formulated as

$$\begin{aligned}\underset{\boldsymbol{\alpha},\mathbf{p}}{\text{maximize}} \ & - M(t_m)\boldsymbol{\alpha}^T(t_m)\mathbf{n}(t_m) - E(t_m)\boldsymbol{\alpha}^T(t_m)\mathbf{p}(t_m) \\ & + VR(t_m) \\ \text{subject to } & \boldsymbol{\alpha}(t_m) \in \{0,1\}^U, \mathbf{p}(t_m) \in [0,P].\end{aligned} \tag{4.47}$$

The optimization problem (4.47) is now transformed into convex optimization given $\boldsymbol{\alpha}(t)$ (from TRPO Step 2) and can be solved by any convex problem solver like interior-point

method with low complexity [92]. Note that the subproblem (4.47) and (4.32) are solved alternately to generate the final frequency/computational resource allocation scheme. In the following, we discuss the solvability of (4.47) in terms of avaliablity of relevant parameters.

(1) Channel state information (CSI): It can be accurately estimated by using pilot-based channel estimation techniques.
(2) Virtual queues: Both $M(t_m)$ and $E(t_m)$ are available once the resource allocation decisions $\boldsymbol{\alpha}$ and $\mathbf{p}$ are available.
(3) Reward: $R(t_m)$ is obtained before solving problem (4.47) and is updated iterative.

Thus, problem (4.47) is solvable, and its formulation will remain unaffected by the rapidly changing environment.

4.3.2.4 Reward Calculation (Step 4 in Fig. 4.2)

After Step 2 and 3, we calculate the reward in Step 4. To facilitate the calculation of reward, we first calculate $N_u(t_m)$ based on (4.8), (4.10) and (4.14) for TO, DS and DTT respectively, given the results from Step 2 $\boldsymbol{\alpha}^*(t_m)$ and Step 3 $\mathbf{p}^*(t_m)$. Then, the reward $R(t_m)$ can be calculated according to (4.18) and the final long-term reward can be formulated based on (4.19).

We also update the state in Step 4, where the state set not only includes previous mentioned elements, but also include virtual queues $M(t_m)$ and $E(t_m)$ constructed in Step 3. It can be represented as $\mathcal{S}(t_m) = \{N(t), \mathbf{r}^*(t_m), r_u^{\text{DTT}}(t+1), M(t_m), E(t_m)\}$. Additionally, the update process of $M(t_m)$ and $E(t_m)$ can be made according to (4.33). The reply buffer stores the tuple $\{\mathcal{S}(t_m), \boldsymbol{\alpha}(t_m), \mathcal{S}(t_m+1), \mathbf{p}(t_m), R(t_m), \mathbf{n}(t_m)\}$ gathered in these four steps for network training.

4.4 Data-Driven Classification and Prediction

In the previous section, we proposed the PW-DRL to solve the resource allocation problem, and a prediction network consisting of two parts is introduced (Step 1) to enhance the learning

process of the TRPO. In this section, we delineate on the structure of this prediction network, which is shown in Fig. 4.3.

4.4.1 Online Prediction Network

The online part takes advantage of the temporal correlation between DTT user requests across different timesteps. With the knowledge of future user requests, the PW-DRL can optimize the current action to ensure relatively sufficient resources for the next timestep.

Specifically, for the online prediction network, we use LSTM, a well-established recurrent neural network (RNN) used for sequential data prediction. As part of the prediction network, the LSTM further improves the performance of PW-DRL, particularly when handling traffic exhibiting strong periodic patterns. Specifically, LSTM predicts the service rate r for the DTT traffic type with predictable traffic patterns. A good example of typical DTT traffic is the one generated by sensors in a factory routinely updating device status. Note that the LSTM cannot predict the rate for non-periodic traffic types (DS and TO). In the prediction network, the input into the LSTM is the DTT user request $\mathbf{r}^{\mathrm{DTT}}(t)$ and the output is the predicted DTT user request $\mathbf{r}^{\mathrm{DTT}}(t+1)$ of the next timestep. At the beginning of the prediction process, the prediction accuracy is low due to insufficient training. Inaccurate prediction results will mislead DRL and lead to poor system performance. To ensure that only accurate prediction is adopted by DRL, we design an activation function given by

$$\gamma_{pred} = Sigmoid\left(\frac{T}{\sum_{t=0}^{T}\left(r_{pred}(t+1) - r_{true}(t+1)\right)^2}\right), \tag{4.48}$$

where

$$Sigmoid(z) = \frac{1}{1+e^{-z}}. \tag{4.49}$$

Then (4.48) is compared with a threshold γ_{th}, and the prediction result can be used only when $\gamma_{pred} \geq \gamma_{th}$. At the beginning of the training process, this prediction network cannot get sufficient training samples to make accurate predictions and thus cannot provide useful information to DRL. The selection process for γ_{th} involves two steps. We first generate a synthetic dataset using the DTT distribution and use this dataset to train the LSTM network.

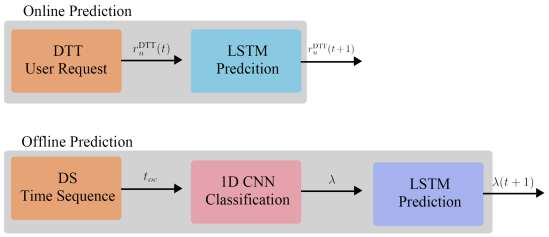


FIGURE 4.3. The structure of the prediction network.

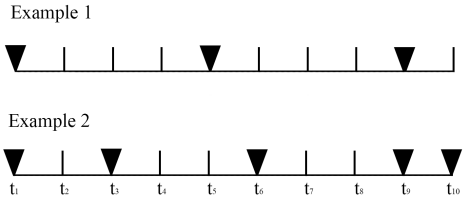


FIGURE 4.4. Two examples of $\mathbf{t}_{oc}$ with the inverse triangle represents the occurrence of DS user request.

This value serves as the initial value for γ_{th}. We then fine-tune the value of γ_{th} by slightly varying it around the initial value and selecting the value that achieves the best system performance.

4.4.2 Offline Prediction Network

In this subsection, we propose a new offline prediction network that encompasses a classification network and a sequential prediction network to aid DRL throughout the training process. As $r_u^{\text{DTT}}(t)$ can only be observed online, we choose to predict λ (state for MMPP). The idea is to use a pre-trained network to first determine the value of λ used to generate the DS user request during a certain period and then use this λ at the current timestep to predict the next timestep's λ. Let $\lambda(t)$ denote the λ at timestep t. In the following subsection, each part will be elaborated on.

4.4.2.1 Classification network

Fig. 4.5 illustrates the network architecture for 1D-CNN, which comprises two 1D convolution layers, two ReLU activation functions, two normalization layers, an average pooling layer, a

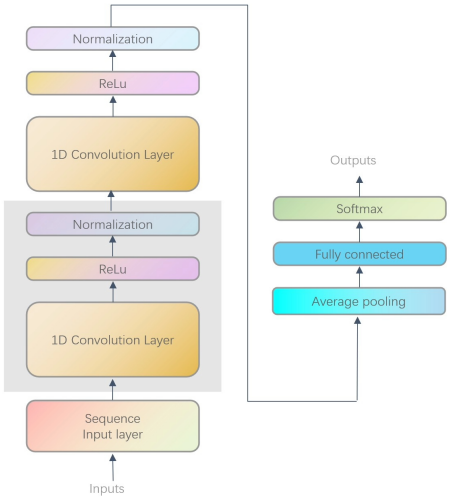


FIGURE 4.5. The Network structure of 1D-CNN.

fully connected layer, and a Softmax output layer. This network is a highly efficient tailored network structure for classification developed by Matlab. Therefore, we directly adopt this structure without the need for fine-tuning hyperparameters. The input of the classification network is the time sequence $\mathbf{t}_{oc}$ that represents the number of DS user requests that occurred during a certain period. Fig. 4.4 shows two examples of $\mathbf{t}_{oc}$ in a period of 10 mini-slots . Specifically, in Example 1, 3 DS user requests occurred at t_1, t_5, and t_9. Therefore, $\mathbf{t}_{oc}$ is a 3×1 vector with $\mathbf{t}_{oc} = [t_1, t_5, t_9]$. Similarly, $\mathbf{t}_{oc}$ is a 5×1 vector in Example 2. Each example has a unique λ and thus resulting in $\mathbf{t}_{oc}$ of different lengths. For a variable length input, we apply a 1D convolutional neural network (1D CNN), which leverages a one-dimensional kernel to generate classification results. This network is trained using a synthetic dataset with each training sample constructed as $[\mathbf{t}_{oc}, \lambda(t)]$. After training, the output of the classification network is the corresponding $\lambda(t)$ used to generate $\mathbf{t}_{oc}$.

4.4.2.2 Sequential prediction network

We leverage LSTM to construct the sequential prediction network, where the input is $\lambda(t)$ obtained from the classification network, and the output is $\lambda(t+1)$. With the information of $\lambda(t+1)$, the cumulative density function $F^{t_s}(t_{\text{next}})$, which represents the time we need to

Algorithm 1 Data-driven Classification and Prediction

for $k = 1$ to K **do**
  while $t \leq T$ **do**
    Given previous DTT user request $r_u^{\text{DTT}}(t)$ and predict future user request $r_u^{\text{DTT}}(t+1)$ using LSTM.
    Given time sequence t_{oc} and predict $\lambda(t)$ using 1D CNN
    Given $\lambda(t)$ from previous Step and predict $\lambda(t+1)$ using LSTM
    Update state variable $\mathcal{S}(t_m)$ for TRPO
  end while
end for

wait for a new user request to occur after the occurrence of request at t_s, can be represented as

$$F^{t_s}(t_{\text{next}}) = 1 - e^{-\lambda(t+1)t_{\text{next}}}. \tag{4.50}$$

The time instance for the next user request to occur can be calculated by letting $F^{t_s}(x) = \frac{1}{2}$ [93]. Accordingly, the state vector $\mathcal{S}(t_m)$ can be formulated as $\mathcal{S}(t_m) = \{N(t), \mathbf{r}^*(t_m), r_u^{\text{DTT}}(t+1), \lambda(t+1), t_{\text{next}}, M(t_m), E(t_m)\}$. We summarize the workflow of the prediction network in Algorithm 1.

4.5 Simulation Results and Comparisons

This section presents simulation results used in the evaluation of the proposed PW-DRL approach. In our simulations, the TTI is set to 1 ms and each mini-slot is set to 0.1 ms. For the CSI, path loss a_u is generated based on the expression $35.3+37.6\log 10(d_u)+X_{\sigma_u}$ [94], where d_u is a random value between 5 to 25 meters, while σ_u is set to 8 dB. Moreover, the single-sided noise spectral density N_0 for AWGN is set as -173 dBm/Hz. Other hyperparameters for online and offline prediction networks ,user request generation and TRPO are shown in Table 4.1, Table 4.2 and Table 4.3, respectively.

4.5.1 Benchmark Methods

Here, the baseline methods adopted for comparison are introduced.

TABLE 4.1. The hyperparameters of the online network.

Name of hyperparameters	Value
LSTM Momentum	0.9000
LSTM Initial Learn Rate	0.0100
LSTM Learn Rate Drop Factor	0.2000
LSTM Learn Rate Drop Period	5
LSTM Max Epochs	100
LSTM Mini Batch Size	2048
LSTM Validation Frequency	50
Threshold γ_{th}	0.8
1D-CNN Optimizer	Adam
1D-CNN Max Epochs	30
1D-CNN Mini Batch Size	4096

TABLE 4.2. Hyperparameters for user requests.

Name of hyperparameters	Value
Exponential distribution mean value for reading time generation	50ms
mean value for file size generation	10
variance value for file size generation	1
upper limit	5 megabytes
packet size k_1	1.2
packet size k_2	20
packet size k_3	250 bytes
reading time k_1	1.1
reading time k_2	2.5
reading time k_3	12.5 bytes

TABLE 4.3. Hyperparameters for the training of TRPO.

Name of hyperparameters	Value
The discount factor	0.95
Experience Horizon	1024
Number of hidden layers	6
Activation function	Sigmoid
size of the layer	20,32,64,128,256,512

(1) *Random*: Under this method, resource allocation decisions are generated based on a random distribution. It is obvious that this is the worst case scenario and can served as the baseline for all other methods.

(2) *TRPO* [91]: An on-policy method that can only generate one type of actions. It has a monotonic policy improvement guarantee compared with others.

(3) *S-PW-DRL* [95]: A simplified version of PW-DRL that does not have the prediction network.

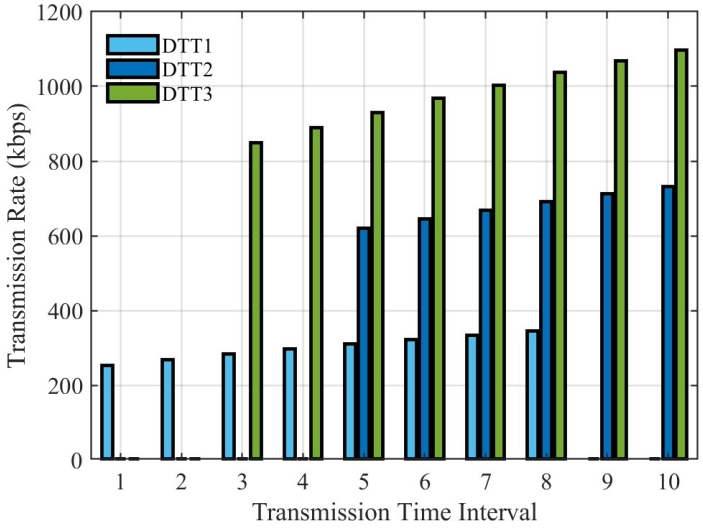
(4) *SAC* [96]: A well-known off-policy method which is featured by regularization. It is trained with the goal of balancing between return and entropy, which can be treated as the balancing between exploration and exploitation in other methods.

(5) *AC* [97]: Conventional actor-critic based DRL method.

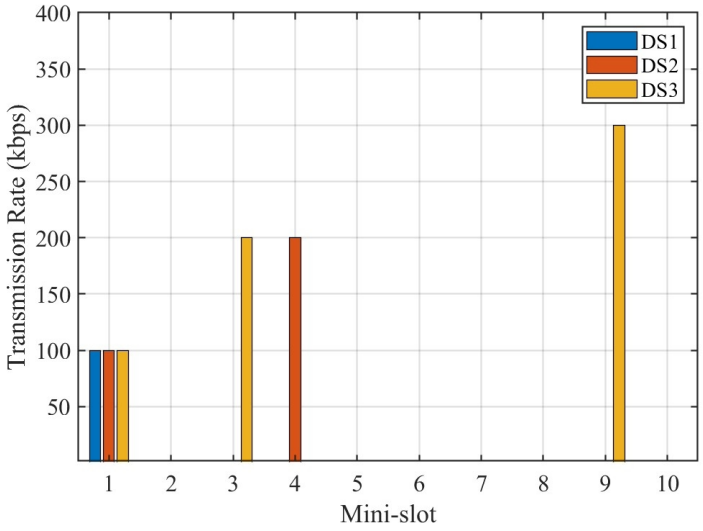
4.5.2 Performance Comparison

Transmission rate variation over time for each user is shown in Figure 4.6. Figure 4.6a generates packet sizes for three kinds of DTT requests using the Beta distribution defined in (4.12). It can be seen that all kinds of DTT requests have an upward tendency in the transmission rate, proving high autocorrelation. That is to say, such requests are predictable in the online prediction network. Figure 4.6b shows the DS requests, for which the transmission rate that each user needs is quite different in each mini-slot because it is randomly chosen out of six possible states. In Figure 4.6c, the packet lengths and reading times for the TO requests are generated by the distributions in (4.3) and (4.4). This indicates that the required transmission rate is significantly higher for TO requests as compared to the other two types of requests; indeed, this would be expected from the nature of this service. Further, it can also be seen from Figure 4.6c that TO user 1 sends requests only during the 1st and 7th TTIs, while in Figure 4.6a, all DTT users send requests during each of the TTIs.

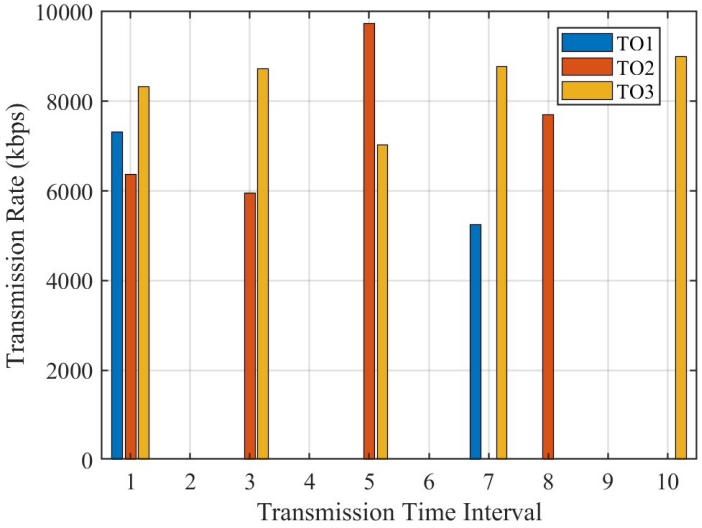
Figure 4.7 depicts how the number of units of the resource available changes over time. From this figure, one can see that the number of units of the resource available varies between TTIs, from which the reader may infer that the action $\boldsymbol{\alpha}(t)$ influences the number of units of the resource available in the next timestep. The above is the second complication from the end



(A) DTT request



(B) DS request



(C) TO request

FIGURE 4.6. The transmission rate comparison between DTT, DS and TO.

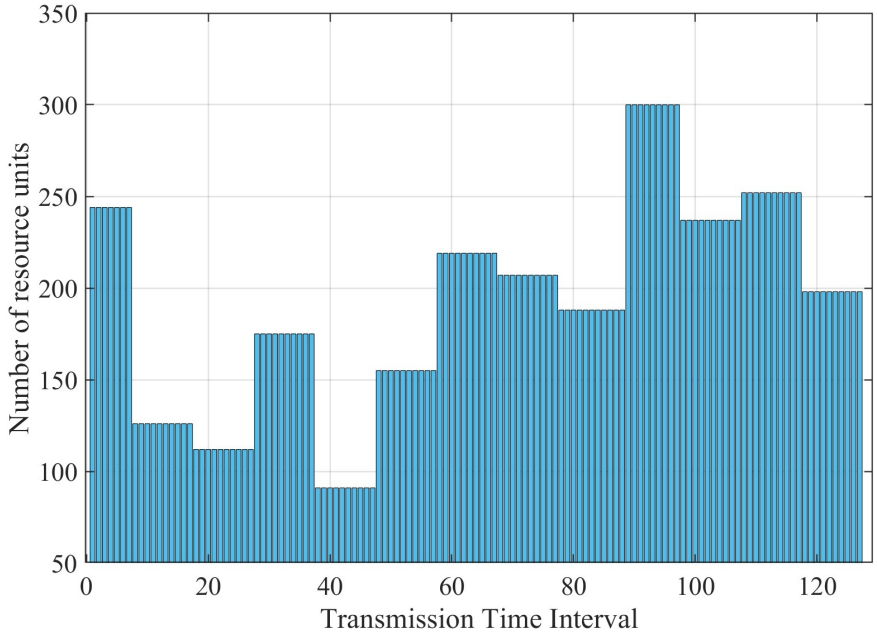


FIGURE 4.7. The variation of the number of resource units.

of Section 2.6. Therefore, the WDP problem formulated in this paper is more complicated compared to the traditional WDP problem. Still another point that may be added to the discussion is the fact that variability in the available resource units is random and hence difficult to predict

In Fig. 4.8, the training reward comparison between PW-DRL and baselines are presented. In this comparison, we set the available resource units to 500 per TTI which simulates the situation where sufficient resources are presented to be able to meet the requirements of almost all users. In this work, the baseline methods considered are: actor-critic (AC), TRPO [98], random action, soft actor-critic (SAC), and S-PW-DRL. Here, S-PW-DRL is a simplified version of PW-DRL which does not have the proposed prediction network. It can be seen from Fig. 4.8 that PW-DRL converges fast and achieves the best reward compared with baselines. This proves the benefits achieved by the proposed prediction network and utilizing Lyapunov to transfer the long-term constraints into per timestep manageble constraints. As expected, the random algorithm which randomly allocate resources to each user requests without taking into account environment information achieves the lowest reward.

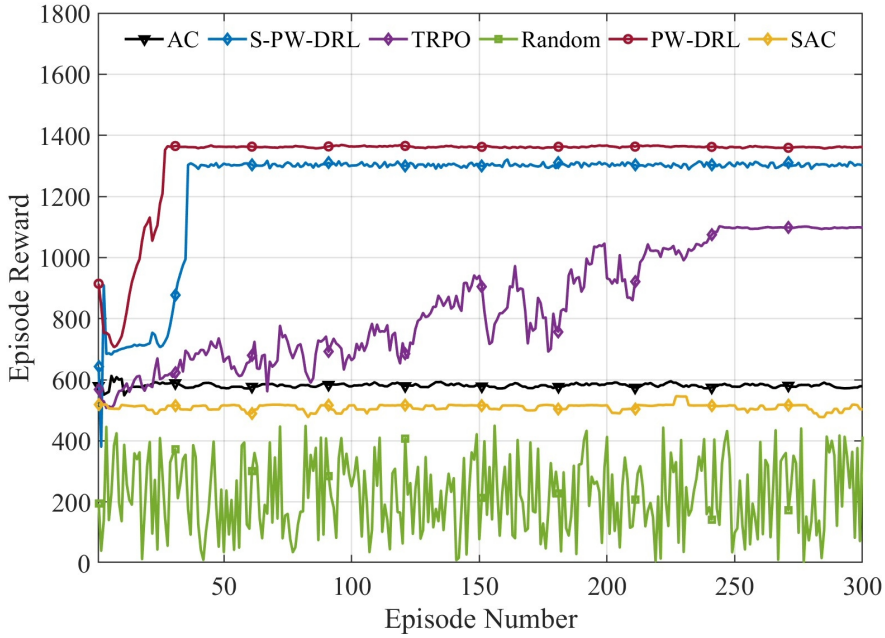


FIGURE 4.8. Training reward for 500 resource units.

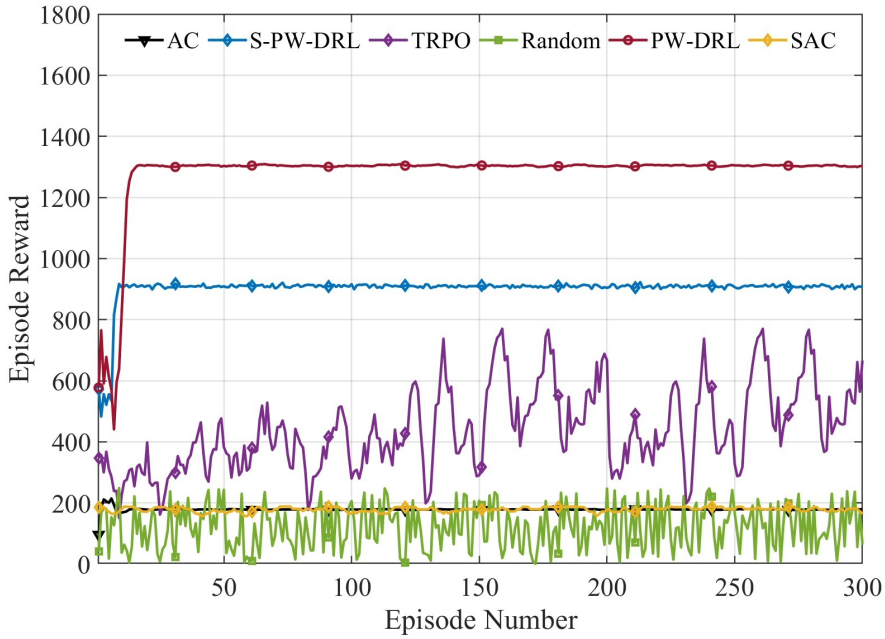
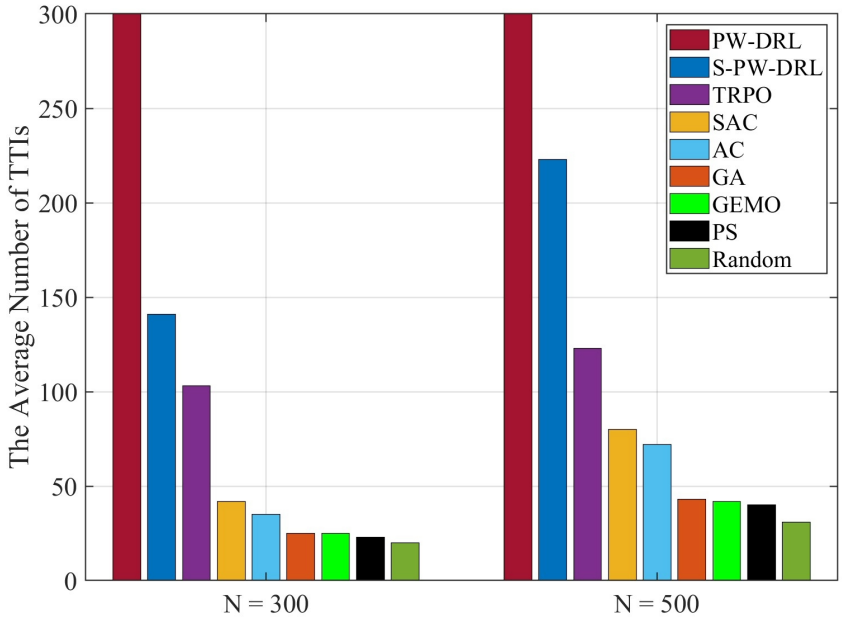
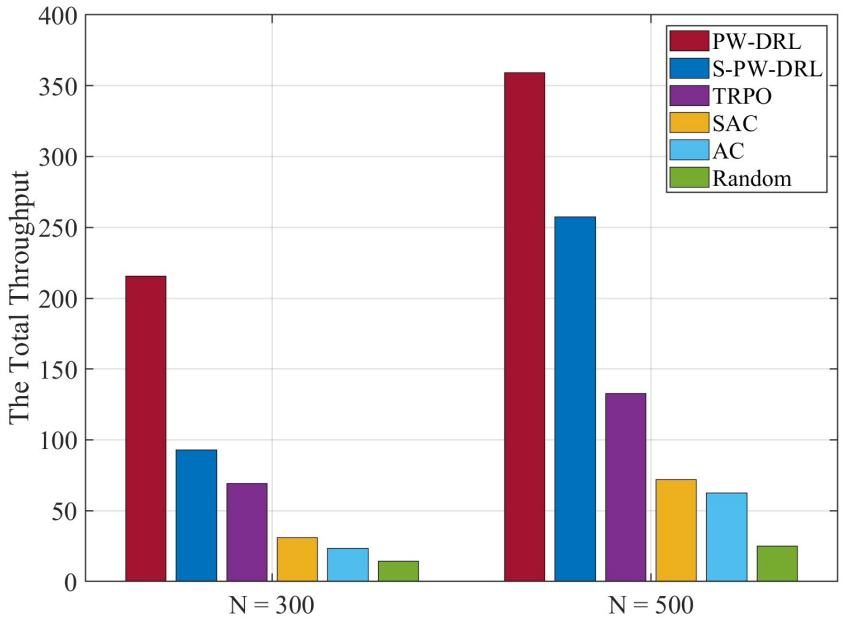


FIGURE 4.9. Training reward for 300 resource units.

Fig. 4.9 compares the performance of PW-DRL with baselines under the scenario where the total available resource units cannot meet requirements of all users. We choose 300 resource units per TTI to simulate this scenario. It can be noticed that PW-DRL still achieves fastest convergence and lowest reward. It is obvious that the reward of PW-DRL in Fig. 4.9 is



(A) The Average Number of TTIs as the metric



(B) The total throughput as the metric

FIGURE 4.10. The TTI and throughput comparison of all approaches.

lower than that in Fig. 4.8 due to the decrease in total available resource units. However, the decrease is only slightly for PW-DRL while that for S-PW-DRL and TRPO is significant. Moreover, TRPO even fails to converge, and the reward difference between S-PW-DRL and PW-DRL becomes more significant compared with that in Fig. 4.8. This is because the impact

of current actions on furter ones becomes stronger when available resources become less. Therefore, the performance boost of utilizing the prediction network becomes significant.

In Fig. 4.10a and Fig. 4.10b, we not only compare PW-DRL with aforementioned baselines but also several non-ML-methods: genetic algorithm (GA), pattern search (PS), and genetic algorithm for multi-object (GAMO). The comparison is carried out for the average number of TTIs $\bar{t}$ and average throughput in Fig. 4.10a and Fig. 4.10b, respectively, for 300 and 500 resource units.

In Fig. 4.10a, the result is obtained by averaging 300 simulations with each simulation contains 300 TTIs at most and is terminated if any constraint ((4.21a), (4.21b), (4.21c)) is violated. Note that in these simulations, we do not update neural network parameters and this process is called inference. As the non-ML-based methods cannot handle the long-term dependencies, we ignore the long-term constraint for these methods.

It is straightforward to see that PW-RL achieves the highest performance for maximizing $\bar{t}$ both for $N = 300$ and $N = 500$. This means PW-DRL obtains the most effective resource allocation strategy that can meet both the long-term and short-term constraints. As expected, the random algorithm is the worst, which agrees with the observations from Figures 4.8 and 4.9. Besides, all the approaches that are free of machine learning behave poorly towards solving problem $\mathcal{P}$.

Figure 4.10b plots the throughput, which is computed from the accepted user requests and is averaged over 300 simulations, as done similarly to Figure 4.10a. The performance trends exhibited are also similar to those in Figure 4.10a. In particular, Figure 4.10a shows that the throughput of our method grows with respect to the number of resource units, while the average number of TTIs does not change for both 300 and 500 resource units. The reason is that the maximum TTI in Figure 4.10a is capped at 300, while there is no upper bound on throughput in Figure 4.10b.

4.6 Conclusion

In this chapter, an efficient resource allocation scheme for the 5G network is proposed, with its performance validated by simulations. As stated at the beginning of this chapter, efficient resource allocation is very important for achieving the goals of satisfying distinct QoS requirements of different services and establishing network slices. Specifically, we proposed a new method called PW-DRL to solve the resource allocation problem in RAN slicing. PW-DRL can be integrated into the medium access control (MAC) layer and the radio resource control (RRC) layer, as it meets the functions description of these layers. MAC layer in 5G is designed specifically for managing and scheduling resources. Integrating PW-DRL into the MAC layer can boost the online resource allocation effectiveness of this layer and cover more service types. RRC layer in 5G is a layer that oversees all resources and is responsible for accepting/declining user requests at a higher level than MAC. Integrating PW-DRL in the RRC layer can boost the speed of RRC in making user acceptance decisions while considering long-term system performance.

At the beginning of the chapter, we also pointed out that several limitations exist in state-of-the-art learning-based approaches. In the following, we conclude how PW-DRL mitigates these limitations.

(1) In our resource allocation framework, different services are generated based on different distributions. Specifically, we consider three types of services: DS, TO, and DTT. In each type of service, three distributions are utilized to generate user requests. We proposed two timescale system models to accommodate different service requirements and set the arrival process for each service according to the realistic 3GPP traffic model.

(2) In this work, the time sequential decision-making problem is formulated taking into account both long-term and short-term system performance. To capture the correlations between future and current environment states, we have used the LSTM method to make predictions of future user requests given previous user requests. Also, we proposed to use TRPO to provide the acceptance decisions that can maximize the

predefined value function. The user request prediction provided by LSTM is treated as part of the state and then fed into TRPO to help TRPO provide more accurate decisions.

(3) In this work, we separate the original MINLP problem into two subproblems where one is solved using TRPO and another using Lyapunov optimization. Specifically, we use the acceptance decisions received from TRPO and formulate an optimization problem using the Lyapunov optimization method. The optimal power allocation scheme can be seen as the solution to this optimization problem.

However, we should point out the following limitations when integrating PW-DRL into the practical 5G system. The first point is that despite the fast online inference of PW-DRL, offline training is time-consuming and of high cost. For practical applications, approaches to reduce training complexity are essential to meet the demands of large-scale networks. The second point is that due to the limitation of DRL-based approaches, the generated resource allocation decisions cannot be proved to be optimal, which means it is possible that we just obtain a suboptimal solution. The performance of PW-DRL can only be proved using simulations, and it still remains an open question of how to prove the optimality of DRL-generated results. The last point is that we assume no large difference in distribution between the training and testing environments, which means the training dataset and the inference dataset follow the same distribution. However, due to the rapidly changing 5G network environment, the violation of this assumption might occur and inevitably cause degraded system performance. Mitigating this problem remains an open challenge at the time of writing.

In the next chapter, we elaborate on the task offloading and resource allocation problem in SAGIN. As stated in Chapter 1, SAGIN is an important part of 6G to achieve global seamless connection. Different from 5G, the SAGIN environment changes more rapidly due to the movement of both air and space segments, and the assumption that no large difference exists between the training and testing environments does not hold anymore. Approaches need to be proposed to solve this problem.

CHAPTER 5

Task Offloading and Resource Allocation in SAGIN

In this chapter, we present the task offloading and resource allocation strategy referred to as GDRL. The main problems lie in how to efficiently offload tasks and allocate resources to satisfy distinct QoS requirements of different services. In the following section, we first give a brief summery of the discussion in Chapter 1 related to SAGIN. Then, we present our system model and formulate the problem. After that, we present our proposed method, GDRL, and give a detailed description of each part. Simulation results are presented to validate the performance. Finally, we conclude this chapter.

SAGIN is the key solution in 6G networks to achieve seamless global connection and satisfy the high availability requirement of telecommunication systems in disasters. For SAGIN, resources are distributed among ground, air, and space segments, and tasks can be processed in each segment. Different from 5G, SAGIN consists of a large number of segments, and complex interconnections exist among them. Thus, tasks need to be first offloaded to one segment from processing, and then corresponding resources can be allocated based on this task offloading decision. The decision to offload the task to which segment significantly influences the achievable latency/data rate and then determines the final achieved QoS. Thus, in SAGIN, task offloading and resource allocation need to be performed together to achieve the performance requirements of 6G. In Chapter 1, we elaborated on the importance of task offloading and resource allocation in SAGIN. The reasons are summarized as follows.

(1) Task offloading/resource allocation is important for energy and computing capability constraint devices to satisfy the QoS requirements of different services. Specifically, devices with low computing capability can offload their computing-intensive tasks to

other segments, reducing the latency induced by waiting and processing of all tasks. Additionally, these devices can reserve resources for services with stringent latency requirements, avoiding the violation of SLA. Furthermore, by moving computing-intensive tasks to other segments, devices can also reduce energy consumption, thus reducing costs and extending the usable time of these devices.

(2) Task offloading/resource allocation is important in achieving the best system performance under limited available resources. As stated in Chapter 1, the main reason for moving from one generation of communication networks to another is the more and more limited available resources caused by the increasing number of users. Task offloading involves transferring computational tasks from resource-constraint nodes (terrestrial network) to resource-rich nodes (airborne platforms and satellites) to enhance processing speed and reduce latency. Moreover, resource allocation involves distributing frequency bands, computational resources, storage, and energy to simultaneously satisfy the QoS requirements of different applications at a low cost. SAGIN is a heterogeneous network consisting of various segments, and each segment has different resource limitations. For example, energy limitation is more important for air segments than for ground segments. Air segments like UAVs have very limited energy available and need to be recharged frequently with high cost. Thus, offloading computing-intensive tasks from UAV to space segments can significantly reduce power usage and thus lower the cost and achieve better system performance. Therefore, efficient task offloading/resource allocation is crucial for utilizing limited resources to achieve the best system performance.

Next, in Chapter 2, we present a detailed literature review on state-of-the-art task offloading/resource allocation schemes for SAGIN. Specifically, these approaches can be categorized into optimization-based and learning-based approaches. the optimization-based approaches suffer from the same problems listed in the previous section. To mitigate these limitations, learning-based approaches are proposed. However, these approaches have the following limitations.

(1) Most of these methods are proposed to solve the one-shot optimization problem. As stated in the previous section, the one-shot optimization problem has many limitations, like ignoring the correlations between task offloading/resource allocation actions. Besides these common limitations, in SAGIN, the static offloading/allocation fails to account for dynamic correlations between tasks, where the execution of current tasks influences the available resources and, in turn, affects the offloading and allocation decisions for future tasks.

(2) These approaches do not consider the difference between the training environment and the testing environment. As stated in the previous section, SAGIN has a more rapidly changing environment compared with 5G due to the movement of both space and air segments. Thus, the difference between the testing and training environment is more significant. As all these DRL methods are built on the assumption of no large difference, they cannot perform efficiently when differences exist.

(3) These approaches can only generate discrete or continuous actions but not both. As task offloading decisions are usually discrete and resource allocation decisions are continuous, the task offloading resource allocation problems usually include both discrete and continuous decisions and are formulated as MINLP problems. To solve MINLP problems, separation and relaxation are needed, which inevitably induce performance loss.

5.1 System model

5.1.1 Network Model

Fig. 5.1 shows the overall channel model for our proposed space-air-ground integrated network (SAGIN) Specifically, three segments are considered in this paper: the space segment (low earth orbit (LEO) network/constellation), the air segment (high altitude platform stations (HAPS) network), and the ground segment (user equipment (UE)). In this paper, we assume that UEs can directly communication with both LEO and HAPS through the mounted global navigation satellite system (GNSS). For simplicity, we denote UE as user in the following.

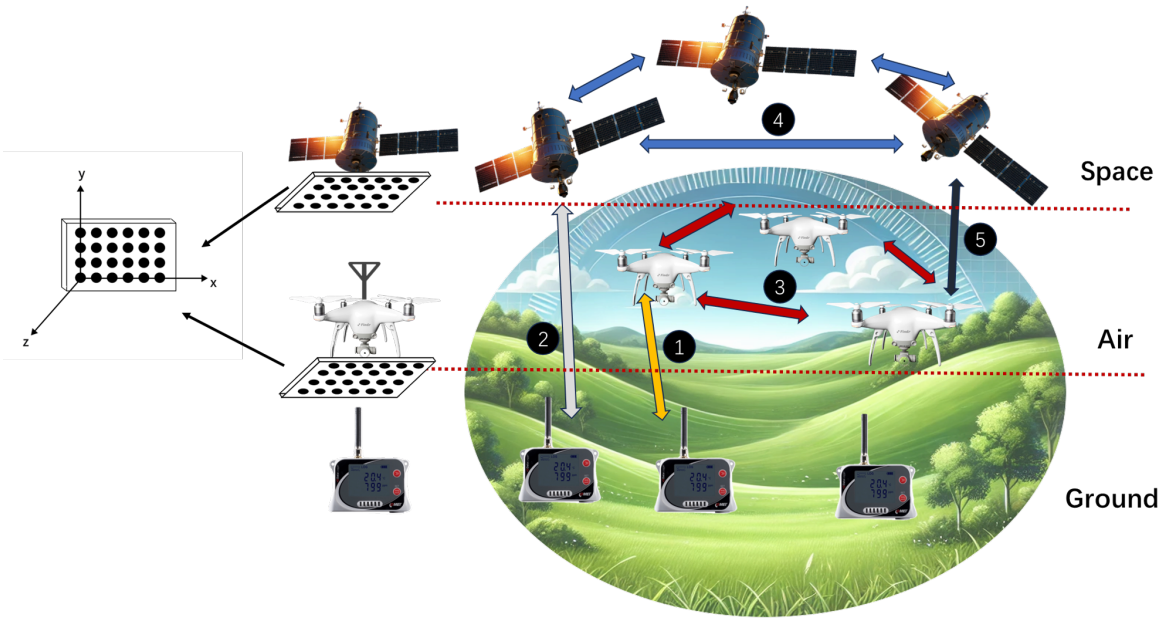


FIGURE 5.1. System architecture of the space-air-ground integrated network (SAGIN).

We denote the set of LEOs, HAPSs, and users as $l \in \mathcal{L} = \{1, \cdots, L\}$, $n \in \mathcal{N} = \{1, \cdots, N\}$, and $\mathcal{U} = \{1, \cdots, U\}$, respectively. Here, two types of task can be generated by user: delay sensitive (DS) task which requires low latency, and delay tolerant (DT) task with relaxed latency requirement. We denote DS task and DT task as μ^D and μ^T, respectively. Accordingly, the DS task generated by user u can be denoted as μ_u^D. We utilize orthogonal frequency division multiple access (OFDMA) to avoid user interference. This work utilizes the time-slotted model which is commonly used in wireless communication. In this model, the timeline is divided into two timescales, namely the long timescale (LT) and the short timescale (ST). We denote the LT as $\mathcal{T} = \{1, 2, \ldots, T\}$, which is further divided into S ST denoted by $\mathcal{S} = \{1, 2, \ldots, S\}$.

In this work, we performe resource allocation for both virtual machines (VMs) and subchannels. Here VMs denote the computational resources and subchannels frequency resources. Specifically, we have total K subchannels and $a_u \in [0, 1]$ is proposed to denote the portion of subchannels assigned to user u.

5.1.2 Channel Model

Five challenges are considered in this paper, namely the ground-to-air channel(❶), ground-to-space channel(❷), air-to-space channel(❸), inter-satellite link (ISL)(❹), and inter-HAPS link (IHL ❺).

For the antenna deployments, HAPS is equipped with a uniform planar array (UPA), which has $M^1 = M_x^1 \times M_y^1$ antennas to communicate with users and a single antenna to communicate with LEO. LEO is also equipped with UPA of $M^2 = M_x^2 \times M_y^2$ antennas with M_x^2 represents x-axis antennas and M_y^2 y-axis antennas.

The channel coefficient for ❶ and ❷ can be calculated as

$$\mathbf{h}_{u,k}^{n/l} = \sqrt{G_u^{n/l}\left(\frac{c}{4\pi d_u^{n/l} f_k^{l/n}}\right)} w_u^{n/l} g_{u,k}^{n/l} \cdot \mathbf{n}_u^{n/l}, \tag{5.1}$$

where $\mathbf{h}_{u,k}^{n}$ is the channel coefficient for ❶, $\mathbf{h}_{u,k}^{l}$ the channel coefficient for ❷, and $G_u^{n/l}$ the total antenna gain, a product of the user's antenna gain and that of HAPS n or LEO l. The term $d_u^{n/l}$ denotes the distance between user u and HAPS n or LEO l, $w_u^{n/l}$ environmental effects-induced attenuation,$g_{u,k}^{n/l}$ the small-scale fading, and $\mathbf{n}_u^{n/l} \in \mathbb{C}^{M\times 1}$ the array response vector written as

$$\mathbf{n}_u^{n/l} = \mathbf{a}_{M_x^{1/2}}\left(\sin\theta_{n/l,u}^{y}\cos\theta_{n/l,u}^{x}\right) \otimes \mathbf{a}_{M_y}\left(\cos\theta_{n/l,u}^{y}\right), \tag{5.2}$$

with

$$\mathbf{a}_{M_*^{1/2}}(x) = \frac{1}{\sqrt{M_*^{1/2}}}\left(1, e^{-j\frac{2\pi c d_*}{f_k^{n/l}}x}, \cdots, e^{-j\frac{2\pi c d_*}{f_{k_l}}\left(M_*^{1/2}-1\right)x}\right), \tag{5.3}$$

where $\theta_{n/l,u} = \left(\theta_{n/l,u}^{x}, \theta_{n/l,u}^{y}\right)$ denotes the angles-of-departure (AoDs) for downlink (DL) transmission from LEO l or HAPS n to user u or the angles-of-arrival (AoAs) for UL transmission, $*$ represents x or y axis, and d_* is the antenna spacing along either x or y axis.

5.1.3 Transmission Rate Model

The data transmission rate for task μ_u^T on channel ❶ and ❷ can be represented by

$$R_{\mu_u^T,k}^{n/l} = \log_2\left(1 + \frac{p_{\mu_u^T}\mathbf{h}_{u,k}^{n/l}(\mathbf{h}_{u,k}^{n/l})^H}{\sigma^2}\right), \tag{5.4}$$

where $p_{\mu_u^T}$ is the transmit power of task μ_T generated by user u, $\mathbf{h}_{u,k}^{n/l}$ represents the channel coefficient in Eq. (5.1), and σ^2 denotes the variance of AWGN.

For task μ^D, the data transmission rate on channel ❶ and ❷ can be expressed as [99]

$$R_{\mu_u^D,k}^{n/l} = \frac{B}{\ln 2}\Bigg\{\ln\left[1 + \frac{p_{\mu_u^D}^{n/l}\mathbf{h}_{u,k}^{n/l}(\mathbf{h}_{u,k}^{n/l})^H}{N_0 B}\right] - \sqrt{\frac{1}{T_f B}} f_Q^{-1}\left(\epsilon_{\mu_u^D}^d\right)\Bigg\}, \tag{5.5}$$

where $\epsilon_{\mu_u^D}^d$ is the decoding error probability, B is bandwidth per subcarrier, T_f is the time-length of a ST, f_Q^{-1} is the inverse Q function, N_0 is the single-side noise spectral density, and $p_{\mu_u^D}$ represents the user u's transmit power for task μ_D.

5.1.4 Computing Model

Here, we use a four tuple $T_u(t) = \{o_u(t), s_u(t), \upsilon_u(t), \iota_u(t)\}$ to represent the task generated by user. Specifically, $o_u(t)$ denotes the computing intensity, $s_u(t)$ the original size of the task, $\upsilon_u(t)$ the processed size of the task, and $\iota_u(t)$ the latency. We assume that at the beginning of each timestep, each user generate a task with probability ϵ_u. We also design an adjacency matrix $\mathbf{E} \in \{0,1\}^{(U+L+N)\times(U+L+N)}$ where $1 \leq$ row $\leq U$ is user nodes, $U+1 \leq$ row $\leq U+L$ LEO nodes and $U+L+1 \leq$ row $\leq U+L+N$ HAPS nodes, to represent the connections among all three segments in SAGIN.

5.1.4.1 Task Offloading Decisions

Task offloading decisions are made based on three indicators.

- Local computing indicator $\alpha_u(t) \in \{0,1\}$ which indicates whether the task is processed locally in UE or not.
- Vertical task offloading indicator $\beta_{u,j}(t) \in \{0,1\}$ with $1 \leq j \leq L$ represents task is offloaded to LEO and $L+1 \leq j \leq L+N$ to HAPS.
- Horizontal task offloading indicator $\eta_{u,j,k}(t) \in \{0,1\}$ which represents whether task is further offloading to the next hop HAPS/LEO or not.

We can get the following constraints based on the definition of the above mentioned indicators.

$$\alpha_u(t) + \sum_{j=1}^{L+N} \beta_{u,j}(t) + \sum_{j=1}^{L+N} \sum_{k=1}^{L+N+1} e_{u,j} e_{j,k} \eta_{u,j,k}(t) \leq 1. \tag{5.6}$$

Specifically, the constraint restricts the offloading decision for each task to one of the following options: dropped, locally processed, one-step offloading, or two-step offloading.

5.1.4.2 Latency

The total latency consists of three parts: propagation, processing, and handover. In the following, we elaborate on the calculation for each part of the total latency.

For propagation latency, it can be calculated based on

$$L_p^u(t) = \begin{cases} 0, & \alpha_u(t) = 1, \\ \frac{s_u}{a_u K R_{u,k}}, & \beta_{u,j}(t) = 1, \\ \frac{s_u}{a_u K R_{u,k}} + \frac{d_{j,k}}{c}, & \eta_{u,j,k}(t) = 1, \end{cases} \tag{5.7}$$

For processing latency, the calculation is achieved by

$$L_c^u(t) = \begin{cases} \frac{o_u(t)}{C_u}, & \alpha_u(t) = 1, \\ \frac{o_u(t)}{c_j^u(t) * C_j * \tau^{(max)}}, & \beta_{u,j}(t)/\eta_{u,j,k}(t) = 1, \end{cases} \tag{5.8}$$

where $c_j^u(t)$ denotes the portion of the total number of VM instances allocated by server j to a task from user u at timestep t, C_j the total number of VM instances in server j, $\tau^{(max)}$ the maximum CPU cycle (Hz/s) of each VM instance, and C_u the computational capability of user u.

For transfer latency, it can be calculated as

$$L_h^u(t) = \begin{cases} 0, & \alpha_u(t) = 1, \\ \frac{v_u}{a_u K R_{u,k}}, & \beta_{u,j}(t) = 1, \\ \frac{v_u}{a_u K R_{u,k}} + \frac{d_{j,k}}{c}, & \eta_{u,j,k}(t) = 1, \end{cases} \tag{5.9}$$

Accordingly, the total latency is $L_o^u(t) = L_p^u(t) + L_c^u(t) + L_h^u(t)$.

5.2 Problem formation

In this section, we formulate an optimization problem that aims to find the offloading decision triple $\{\boldsymbol{\alpha} \in \{0,1\}^{U\times 1}, \boldsymbol{\beta} \in \{0,1\}^{U\times(N+L)}, \boldsymbol{\eta} \in \{0,1\}^{U\times(N+L)\times(N+L)}\}$, and allocate computational resources $\mathbf{C} \in \mathbb{R}^{U\times(N+L)}$ and portion of subchannels allocated $\mathbf{a} \in \mathbb{R}^{U}$. The problem can be formulated as

$$\mathcal{P}: \min_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\eta},\mathbf{C},\mathbf{a}} \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{t=ST} L_o(t) \tag{5.10}$$

$$\sum_{u\in\mathcal{U}} c_j^u(t) \leq C_j, \tag{5.11a}$$

$$\sum_{u} a_{u,k} \leq K, \tag{5.11b}$$

$$L_o^u(t) \leq \iota_u(t), \tag{5.11c}$$

$$Eq.\ (5.6) \tag{5.11d}$$

where C_j is the total VM instances in server j.

This formulated problem $\mathcal{P}$ is difficult to solve due to the following reasons.

(C1) $\mathcal{P}$ is a mixed-integer nonlinear programming problem (MINLP) which is NP-hard. It is also a sequential decision-making problem where previous resource allocation decisions will explicitly affect available resources and implicitly affect future resource

allocation decisions. Conventional methods are capable of solving the resource allocation problem for a single timeslot t but not over the entire duration ST.

(C2) As the SAGIN environment is continuously affected by various factors like the movement of satellites, the underlying features of the environment may change from time to time. This may lead to significant changes between the testing environment and the training environment. Conventional DRL methods cannot solve problems where the testing environment and training environment are subject to dynamic change. Thus, it is hard for conventional DRL methods to solve $\mathcal{P}$ efficiently with low complexity.

(C3) The choice of solution set $\{\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\eta}, \mathbf{C}, \mathbf{A}\}$ at previous timestep will affect that at the next timestep implicitly by directly influencing the total available resources $\{C_j, K\}$. Therefore, $\mathcal{P}$ is also a sequential decision-making problem that is hard to be solved due to its high complexity.

5.3 Proposed Solution

5.3.1 Motivation

To address the aforementioned difficulties, we propose a new method called GDRL. The core of this GDRL is to train a neural network that can directly generate task offloading and resource allocation decisions by interacting with the environment. In the following, we will elaborate on the influence of each difficulty (C1-C3) on the design of this GDRL.

To address (C1), we propose a model-free method to directly generate offloading and resource allocation decisions based on current observations. Specifically, we modify the actor-critic TRPO structure to enable the simultaneous output of both discrete and continuous action. Also, we formulated a reward function to handle constraints (5.11a), (5.11b), (5.11c), (5.11d). To tackle (C2), we propose a novel encoder-decoder-based action mapping network. With the implementation of this network, we do not need to retrain the whole network for different

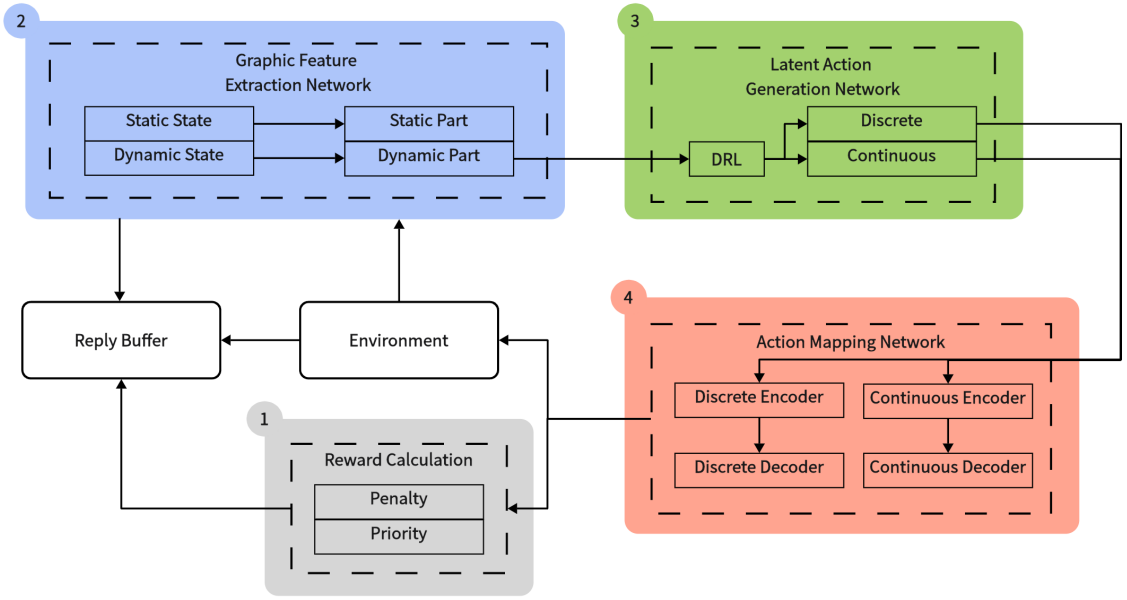


FIGURE 5.2. The architecture of the proposed GDRL.

numbers of users. Moreover, since both networks are pretrained, they offer low implementation complexity and can be used in other networks to address the dimension problems. To address (C3), we separate the environmental states into static and dynamic ones. The dynamic states are directly affected by the solution set at the previous timestep, whereas the static states remain unaffected. Each state is then processed by a neural network specifically tailored to its unique features.

The proposed GDRL is shown in Fig. 5.2. It consists of three main parts (Part 1,2,3), one auxiliary part (Part 4) and a reply buffer that can store the transactions (e.g., state, action, reward, next state). Generally speaking, in Part 1, the environment states are divided into static states and dynamic states based on whether the state is influenced by the previous timestep actions. In Part 2, separated states are inputted into different parts of the actor network to generate latent actions for both discrete and continuous actions. In Part 3, the latent actions are inputted into the action mapping network to generate the exact offloading and resource allocation decisions. In Part 4, the DRL agent interacts with the environment according to the actions generated by the action mapping network, and the final reward is calculated accordingly. In the following subsections, we elaborate on each part of this GDRL.

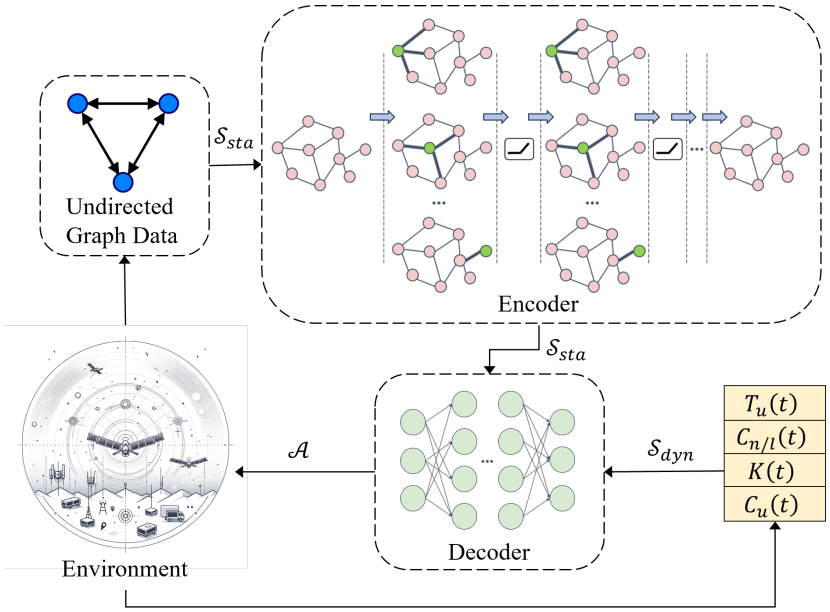


FIGURE 5.3. The structure of the proposed GFEN.

5.3.2 The component of GDRL

5.3.2.1 State separation (Part 1 in Fig. 5.2)

As mentioned in the motivation part, some states are influenced by previous actions and some states are not. To capture the features of different states, we separate the states into static and dynamic ones. Note that the static states only represents the state that is not affected by the actions and do not imply states that remain unchanged over time. The components of static and dynamic states are shown in Fig. 5.3. Specifically, the static states contain the location information of LEO loc_l, HAPS loc_n and user loc_u, the connection status $\mathbf{E}$, the initial computation $C_{n/l}^{\text{init}}$ and frequency $K_{n/l}^{\text{init}}$ resources of LEO and HAPS, and the initial computation capacity C_u^{init} of user u. The dynamic states include the tasks $T_u(t)$, the remaining computation $C_{n/l}$ and frequency $K_{n/l}$ resources of LEO and HAPS, and the remaining computation resources C_u of user u. It is straightforward that the resources are influenced by actions. The reason for categorizing tasks in the dynamic states is that the task is influenced by the offloading decisions. For μ^T, if the task is rejected ($\alpha = 0, \beta = 0, \gamma = 0$), it will be given a new priority value and wait to be processed at the next timestep. If the delay μ^T experiences exceed its delay requirement, the task will be discarded. In contrast, μ^D is

immediately discarded upon rejection due to its stringent latency requirements. Until a task is processed (whether discarded or offloaded), no new task of the same category (μ^D or μ^T) will be generated. Therefore, tasks are influenced by the actions (offloading decisions) and should be categorized into dynamic states. We denote $\mathcal{S}_{\text{sta}}$ and $\mathcal{S}_{\text{dyn}}$ as static states and dynamic states, respectively.

After the separation of states, $\mathcal{S}_{\text{sta}}$ and $\mathcal{S}_{\text{dyn}}$ will be received by different parts of the graph feature extraction network (GFEN) to generate latent actions. Fig. 5.3 shows the structure of the proposed GFEN. It can be seen that the GFEN consists of two parts: the encoder part and the decoder part. Static states $\mathcal{S}_{\text{sta}}$ are forwarded directly to the encoder, and the encoder output, along with dynamic states $\mathcal{S}_{\text{dyn}}$, are then input into the decoder for latent action generation. Here, we leverage the graph convolutional network (GCN) to construct the encoder part. The reason for choosing GCN lies in its high efficiency for extracting features of graph-structured data. Since static states $\mathcal{S}_{\text{sta}}$ are graph-structured data, GCN is the best choice. Specifically, we formulate an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{\mathcal{U}, \mathcal{L}, \mathcal{N}\}$ represents the set of nodes, and $\mathcal{E}$ represents the set of edges. Each user, LEO, and HAPS are treated as a node in the graph $\mathcal{G}$, the connection status $\mathbf{E}$ is equivalent to an adjacency matrix, and other parts of $\mathcal{S}_{\text{sta}}$ can be treated as the node feature $\mathbf{X} = \left\{l_i, C_i^{\text{init}}, K_i^{\text{init}}\right\}_i \in \mathbb{R}^{(U+L+N)\times 3}$ with $\mathbf{x} \in \mathbb{R}^{U+L+N}$ represents each type of feature. Note that for users, the value of K_u^{init} is set to 0, as users do not have subchannels. The degree matrix $\mathbf{D}$ can be derived from the adjacency matrix by

$$d_{i,i} = \sum_j e_{i,j}, \tag{5.12}$$

where $d_{i,i}$ represents the diagonal value of $\mathbf{D}$. Therefore, $\mathbf{D}$ can be represented as

$$\mathbf{D} = \begin{cases} d_{i,i} & i = j \\ 0 & i \neq j, \end{cases} \tag{5.13}$$

where $\mathbf{D}$ only has values at its diagonal and zeros elsewhere. Accordingly, the multiplication of $\mathbf{x}$ with a filter $g_\theta = \text{diag}(\theta) \in \mathbb{R}^{U+L+N}$ in the Fourier domain can be represented as

$$\mathbf{g}_\theta * \mathbf{x} = \mathbf{U}\mathbf{g}_\theta\mathbf{U}^\top\mathbf{x}, \tag{5.14}$$

where $\mathbf{U}$ is the matrix of eigenvectors and can be obtained from the decomposition of the normalized graph Laplacian matrix $\mathbf{L}$, which is

$$\mathbf{L} = \mathbf{I}_{(U+L+N)} - \mathbf{D}^{-\frac{1}{2}}\mathbf{E}\mathbf{D}^{-\frac{1}{2}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\top}, \tag{5.15}$$

where $\mathbf{\Lambda}$ represents the diagonal matrix of eigenvalues. According to the theory of GCN, $\mathbf{g}_\theta$ can be calculated based on eigenvalues of $\mathbf{L}$, which can be further denoted as $\mathbf{g}_\theta(\mathbf{\Lambda})$. As the complexity for performing (5.15) is very high ($\mathcal{O}((U+L+N)^2)$), $\mathbf{g}_\theta(\mathbf{\Lambda})$ is approximated by a truncated expansion of Chebyshev polynomials $T_k(x)$ up to K-th order, which is

$$\mathbf{g}_{\theta'}(\mathbf{\Lambda}) \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{\mathbf{\Lambda}}), \tag{5.16}$$

where θ'_k represents the Chebyshev coefficients, and $\tilde{\mathbf{\Lambda}}$ represents a rescaled version of $\mathbf{\Lambda}$ with

$$\tilde{\mathbf{\Lambda}} = \frac{2}{\text{eigen}_{\text{max}}}\mathbf{\Lambda} - \mathbf{I}_{(U+L+N)}, \tag{5.17}$$

where $\text{eigen}_{\text{max}}$ represents the largest eigenvalue of L. The Chebyshev polynomials are defined as

$$\begin{aligned} T_k(x) &= 2xT_{k-1}(x) - T_{k-2}(x), \\ T_0(x) &= 1, T_1(x) = x. \end{aligned} \tag{5.18}$$

Changing $\mathbf{g}(\theta)$ into (5.14) by $\mathbf{g}(\theta')$ in (5.16), we can derive

$$\mathbf{g}_{\theta'} * \mathbf{x} \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{\mathbf{L}})\mathbf{x}, \tag{5.19}$$

where $\tilde{\mathbf{L}}$ represents the rescaled version of $\mathbf{L}$ with

$$\tilde{\mathbf{L}} = \frac{2}{\text{eigen}_{\text{max}}}\mathbf{L} - \mathbf{I}_{(U+L+N)}. \tag{5.20}$$

Accordingly, (5.19) can be simplified by letting $K = 1$ and $\text{eigen}_{\text{max}} \approx 2$, which can be represented as

$$\mathbf{g}_{\theta'} * \mathbf{x} \approx \theta'_0\mathbf{x} - \theta'_1\mathbf{D}^{-\frac{1}{2}}\mathbf{E}\mathbf{D}^{-\frac{1}{2}}\mathbf{x}. \tag{5.21}$$

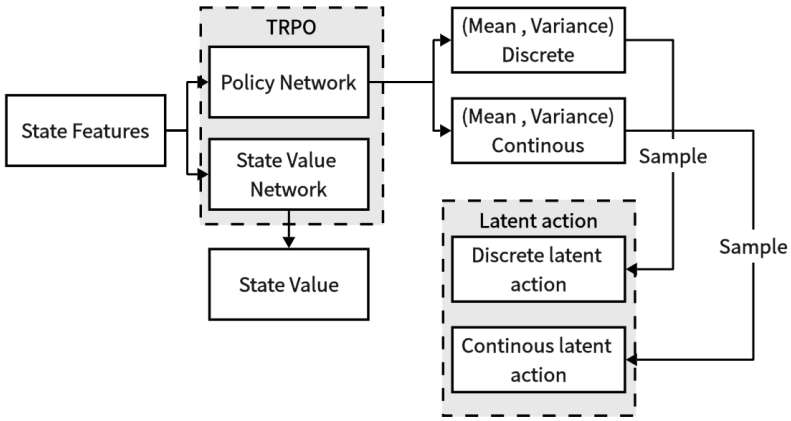


FIGURE 5.4. The structure of the proposed LAGN.

To address the overfitting problem and to minimize the computational complexity, (5.21) can be simplified into

$$\mathbf{g}_\theta * \mathbf{x} \approx \theta \left(\mathbf{I}_{U+L+N} + \mathbf{D}^{-\frac{1}{2}} \mathbf{E} \mathbf{D}^{-\frac{1}{2}} \right), \tag{5.22}$$

where $\theta = \theta'_0 = -\theta'_1$. For $\mathbf{X}$ and GCN with F filters, (5.22) can be generalized into matrix representation as

$$\mathbf{O} = \left(\mathbf{I}_{U+L+N} + \mathbf{D}^{-\frac{1}{2}} \mathbf{E} \mathbf{D}^{-\frac{1}{2}} \right) \mathbf{X} \mathbf{\Theta}, \tag{5.23}$$

where $\mathbf{\Theta} \in \mathbb{R}^{3 \times F}$ represents the matrix of filter parameters, and $\mathbf{O} \in \mathbb{R}^{(U+L+N) \times F}$ represents the graph convolution output. Accordingly, the output of the decoder can be calcalted as

$$\mathbf{H}_{\text{dec}} = \text{ReLu}(\mathbf{X}_{dyn} \mathbf{\Theta}_{\text{dec}}), \tag{5.24}$$

where $\mathbf{\Theta}_{\text{dec}} \in \mathbb{R}^{\dim(X) \times D_h}$ is the matrix of FNN hidden state parameters, and D_h the dimension of the hidden state.

5.3.2.2 Latent action generation network (Part 2 in Fig. 5.2)

After step 1, we can get the output from GFEN and forward it to the latent action generation network (LAGN) for latent action generation. Each component of LAGN is shown in Fig. 5.4. It can be seen that the output of the LAGN consists of two parts: the discrete latent action part and the continuous latent action part. It should be noted that the term “discrete latent action” does not imply that the action itself is discrete. Rather, it refers to the segment

of the latent action that is utilized to generate the final discrete action. The latent action generation problem can be formulated as a Markov decision process (MDP) problem The Markov decision process (MDP) problem for the generation of latent action can be denoted by a tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{M}, \mathcal{V}, g, \rho_0\}$, where $\mathcal{A}$ represents the latenet action set with $\mathcal{A} = \{\mathbf{a}_c, \mathbf{a}_d\}$ where $\mathbf{a}_c$ represents the continuous latent action and $\mathbf{a}_d$ represents the discrete latent action, $\mathcal{M} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ represents the transition probability distribution, $\mathcal{V}$ represents reward function, $g \in (0, 1)$ represents the discount factor, and $\rho_0 : \mathcal{S} \to \mathbb{R}$ represents the distribution of initial states. Note that the set of states $\mathcal{S}$ is the output received from the GFEN. We use Gaussian distribution as the main policy distribution, and the direct output of LAGN is the mean and variance of Gaussian distribution. Then, the latent actions are samples drawn from these distributions.

LAGN is built upon trust region policy optimization (TRPO), and two main reasons exist for utilizing TRPO instead of other DRL algorithms. First, TRPO's policy update has a guarantee of monotonic improvement, a feature absent in other DRL methods. This characteristic of TRPO ensures that each update step yields an improved policy over the last, contributing to faster convergence. Second, TRPO implements the advantage function, A_π, which offers a more stable policy update than other policy gradient methods and can be defined as

$$A_\pi(\mathbf{o}(t), \mathbf{a}_c(t), \mathbf{a}_d(t)) = Q_\pi(\mathbf{o}(t), \mathbf{a}_c(t), \mathbf{a}_d(t)) - V_\pi(\mathbf{o}(t)), \tag{5.25}$$

where $\pi : \mathcal{A} \times \mathcal{S} \to [0, 1]$ represents the policy, $\mathbf{o}(t)$ represents the output obtained from the GFEN, $\mathbf{a}_c(t)$ is the continuous latent action, $\mathbf{a}_d(t)$ denotes the discrete latent action, $Q_\pi(\mathbf{o}(t), \mathbf{a}_c(t), \mathbf{a}_d(t))$ represents the Q-value function, and $V_\pi(\mathbf{o}(t))$ is the state value function. To simplify notations, we replace $\mathbf{a}_c(t)$ and $\mathbf{a}_d(t)$ with $\mathbf{a}(t) = [\mathbf{a}_c(t), \mathbf{a}_d(t)]$. Therefore, the Q-value function $Q_\pi(\mathbf{o}(t), \mathbf{a}(t))$ can be represented as

$$\begin{aligned} &Q_\pi(\mathbf{o}(t), \mathbf{a}(t)) = \\ &\mathbb{E}_{\mathbf{o}(t+1), \mathbf{a}(t+1), \ldots} \left[\sum_{l=0}^{\infty} g^l \zeta(\mathbf{o}(t+l)) \right], \end{aligned} \tag{5.26}$$

where g is the discount factor, l is the time lag. According to the formation of $A_\pi(\mathbf{o}(t), \mathbf{a}(t))$, we can now define the advantage of old policy π_{old} over new policy after an update π_{new} as

$$\xi(\pi_{\text{new}}) = \xi(\pi_{\text{old}}) + \mathbb{E}_{\mathbf{o}_0, \mathbf{a}_0, \cdots \sim \pi_{\text{new}}} \left[\sum_{t=0}^{\infty} g^t A_{\pi_{\text{old}}} \left(\mathbf{o}(t), \mathbf{a}(t) \right) \right], \tag{5.27}$$

with

$$\xi(\pi) = \mathbb{E}_{\mathbf{o}_0, \mathbf{a}_0, \ldots} \left[\sum_{t=0}^{\infty} g^t \zeta(\mathbf{o}(t)) \right]. \tag{5.28}$$

Changing the expectation term in (5.27), we can now derive

$$\xi(\pi_{\text{new}}) = \xi(\pi_{\text{old}}) + \sum_{\mathbf{o}} \eta_{\pi_{\text{new}}}(\mathbf{o}) \sum_{\mathbf{a}} \pi_{\text{new}}(\mathbf{a} \mid \mathbf{o}) A_{\pi_{\text{old}}}(\mathbf{o}, \mathbf{a}), \tag{5.29}$$

with

$$\eta_\pi(\mathbf{o}) = \sum_{t=0}^{\infty} g^t P\left(\mathbf{o}(t) = \mathbf{o} \mid \pi \right), \tag{5.30}$$

where $\eta_\pi(\mathbf{o})$ is the discounted transition probability distribution. According to the advantage of one policy over another, we formulate the objective function as

$$\begin{aligned} L_{\pi_{\text{old}}}(\pi_{\text{new}}) =& \xi(\pi_{\text{old}}) + \\ & \sum_{\mathbf{o}} \eta_{\pi_{\text{old}}}(\mathbf{o}) \Big(\sum_{\mathbf{a}} \pi_{\text{new}}(\mathbf{a} \mid \mathbf{o}) A_{\pi_{\text{old}}}(\mathbf{o}, \mathbf{a}) + \mathscr{E}\left(\mathbf{o}\right) \Big), \end{aligned} \tag{5.31}$$

where $\mathscr{E}\left(\mathbf{o}\right)$ is entropy loss with

$$\begin{aligned} \mathscr{E}\left(\mathbf{o}\right) = & - \sum_{\mathbf{a}_c} \pi_{\text{new}}\left(\mathbf{a}_c \mid \mathbf{o}\right) \ln \pi_{\text{new}}\left(\mathbf{a}_c \mid \mathbf{o}\right) \\ & - \sum_{\mathbf{a}_d} \pi_{\text{new}}\left(\mathbf{a}_d \mid \mathbf{o}\right) \ln \pi_{\text{new}}\left(\mathbf{a}_d \mid \mathbf{o}\right). \end{aligned} \tag{5.32}$$

Accordingly, the problem to be solved by TRPO can be represented as

$$\begin{aligned} & \underset{\pi_{\text{new}}}{\text{maximize}} \;\; L_{\pi_{\text{old}}}(\pi_{\text{new}}) \\ & \text{subject to} \;\; D_{\text{KL}}\left(\pi_{\text{new}}(\cdot \mid \mathbf{o}) \| \pi_{\text{old}}(\cdot \mid \mathbf{o})\right) \leq \delta, \end{aligned} \tag{5.33}$$

where D_{KL} represents the KL divergence, δ the trust region constraint. Using linear and quadratic approximation for the objective function and KL divergence, respectively, we can

simplify the problem (5.33) into

$$\begin{aligned} &\underset{\boldsymbol{\theta}'}{\text{maximize}} \;\; \mathbf{z}^T \left(\boldsymbol{\theta}^p_{\text{new}} - \boldsymbol{\theta}^p_{\text{old}}\right) \\ &\text{subject to} \;\; \frac{1}{2}\left(\boldsymbol{\theta}^p_{\text{new}} - \boldsymbol{\theta}^p_{\text{old}}\right)^T \mathbf{C}_{\theta^p}\left(\boldsymbol{\theta}^p_{\text{new}} - \boldsymbol{\theta}^p_{\text{old}}\right) \leq \delta, \end{aligned} \tag{5.34}$$

where $\boldsymbol{\theta}^p$ represents the parameters of the policy network, $\mathbf{C}_{\theta^p}$ the Hessian matrix of the KL divergence and $\mathbf{z}$ the gradient of $L_{\pi_{\text{old}}}(\pi_{\text{new}})$ Problem (6.22) can be solved using Lagrangian duality, resulting in the policy network parameter update rule as

$$\boldsymbol{\theta}^p_{\text{new}} = \boldsymbol{\theta}^p_{\text{old}} + \kappa\sqrt{\frac{2\delta}{\mathbf{z}^T\mathbf{C}^{-1}_{\theta^p}\mathbf{z}}}\mathbf{C}^{-1}_{\theta^p}\mathbf{z}, \tag{5.35}$$

where κ can be calculated utilizing line search. For value network, the update rule is

$$\phi_{new} = \arg\min_{\phi} \frac{1}{ST}\sum_{t=0}^{ST}\left(V_\phi\left(\mathbf{o}(t)\right) - \zeta(\mathbf{o}(t))\right)^2, \tag{5.36}$$

where ϕ is value network parameter, and $\zeta(\mathbf{o}(t))$ the reward.

5.3.2.3 Action mapping network (Part 3 in Fig. 5.2)

After Part 2, we then move to Part 3, where the output of LAGN is forwarded into the action mapping network (AMN) for final resource allocation action generation. Fig. 5.5 shows the network architecture of AMN. It can be seen that the AMP leverages the long-short-term memory (LSTM) network to handle input with different dimensions. Specifically, LSTM can generate a fixed-dimension output with varying-dimension input. It can store long-term dependencies by introducing memory cells and three types of gates. Namely, the input gate, forget gate, and output gate. Specifically, gates are presented as

$$\begin{aligned} \boldsymbol{\psi}_i(u) &= \delta(\boldsymbol{\Theta}^l_i\mathbf{h}(u-1) + \boldsymbol{\theta}^l_i\mathbf{x}(u) + \mathbf{b}_i) \\ \boldsymbol{\psi}_f(u) &= \delta(\boldsymbol{\Theta}^l_f\mathbf{h}(u-1) + \boldsymbol{\theta}^l_f\mathbf{x}(u) + \mathbf{b}_f) \\ \boldsymbol{\psi}_o(u) &= \delta(\boldsymbol{\Theta}^l_o\mathbf{h}(u-1) + \boldsymbol{\theta}^l_o\mathbf{x}(u) + \mathbf{b}_o), \end{aligned} \tag{5.37}$$

where $\boldsymbol{\psi}_i(u), \boldsymbol{\psi}_f(u), \boldsymbol{\psi}_o(u)$ represent the input gate, the forget gate, and the output gate, respectively; σ represents the sigmoid function, $\{\boldsymbol{\Theta}_i^l, \boldsymbol{\theta}_i^l, \mathbf{b}_i\}, \{\boldsymbol{\Theta}_f^l, \boldsymbol{\theta}_f^l \mathbf{b}_f\}, \{\boldsymbol{\Theta}_o^l, \boldsymbol{\theta}_o^l, \mathbf{b}_o\}$ represents the learnable parameters for LSTM input gate, forget gate and output gate, respectively; $\mathbf{h}$ represents the hidden state. Furthermore, memory cells are represented as

$$\begin{aligned} \boldsymbol{\psi}_i(u) &= \tanh(\boldsymbol{\Theta}_c^l \mathbf{h}(u-1) + \boldsymbol{\theta}_c^l \mathbf{x}(u) + \mathbf{b}_c) \\ \mathbf{c}(u)) &= \boldsymbol{\psi}_f(t) \odot \mathbf{c}(u-1) + \boldsymbol{\psi}_i(u) \odot \boldsymbol{\psi}_c(u), \end{aligned} \tag{5.38}$$

where $\boldsymbol{\psi}_c(u)$ is the input cell, $\mathbf{c}(u)$ the output cell, $\{\boldsymbol{\Theta}_c^l, \boldsymbol{\theta}_c^l, \mathbf{b}_c\}$ are the learnable parameters for LSTM input cell. According to (5.37) and (5.38), the hidden state $\mathbf{h}(u)$ can be calculated as

$$\mathbf{h}(u) = \boldsymbol{\psi}_o(u) \odot \mathbf{c}(u). \tag{5.39}$$

It's important to note that the index in question is the user index u, rather than the time index t. The input $\mathbf{x}(u)$ represents the actions of a single user, and these user actions are inputted into the network sequentially. Figure illustrates the process. The last hidden state $\mathbf{h}(u_{\text{act}})$ is then treated as the input of the next FNN network.

It is clear that LSTM is the core component of the AMP network in both the encoder and the decoder part. In the encoder part, the LSTM receives true discrete action and generates latent

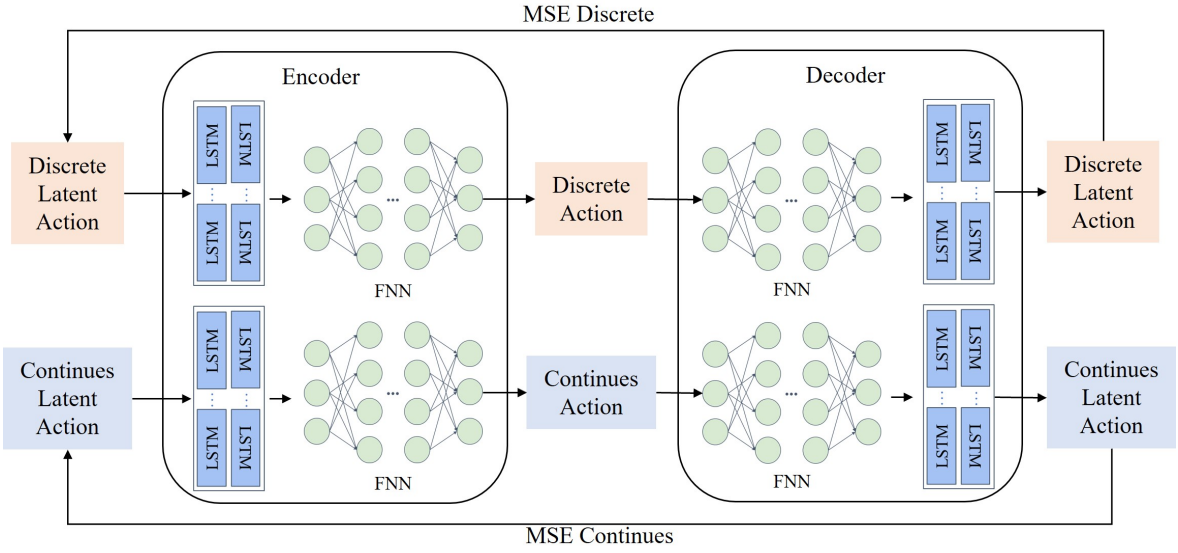


FIGURE 5.5. The structure of the proposed AMN.

vectors. The latent vector is then forwarded to the FNN which generates means and variances for various Gaussian processes. The latent actions are samples of these Gaussian processes. In the decoder part, the latent actions are forwarded into the FNN network and then into the LSTM network for true actions generation. For discrete actions, the decoder LSTM generates the probability of each discrete action and the discrete action with the highest probability is chosen as the true action. For continuous actions, the decoder LSTM directly generates the portion of total resources allocated to each user. Note that the number of users is set to the maximum number of users, and resources allocated to non-active users are set to zero. Here, we use $\mathbf{a}_{\text{true}}$ to represent the true action, whose dimension varies with the number of active users, $\mathbf{a}_{\text{latent}}$ to represent the latent action, whose dimension does not change and is a hyperparameter to be set in the TRPO policy network, $\mathbf{a}_{\text{max}}$ to represent the maximum number of users that can be accepted into the network. After training AMN with a predefined dataset, the decoder part of AMN is implemented into the DRL architecture directly to generate true actions.

5.3.2.4 Reward calculation (Part 4 in Fig. 5.2)

In this part, we formulate the reward function that can address the priorities of different tasks, delay requirements, and total resource constraints. Specifically, the reward function for each user is shown in (5.40),

$$R(t) = \begin{cases} \lambda_1 \sum_j (C_j - \sum_u c_j^u(t)) + \lambda_2 (K - \sum_u a_{u,k}) & \\ \quad + \lambda_3 \left(\sum_u \kappa(\iota_u(t) - L_o^u(t)) \right), & \text{if } L_o^u(t) \leq \iota_u(t), \\ \lambda_1 \sum_j (C_j - \sum_u c_j^u(t)) + \lambda_2 (K - \sum_u a_{u,k}) & \\ \quad - \sum_u b_p, & \text{if } L_o^u(t) > \iota_u(t). \end{cases} \tag{5.40}$$

where b_p represents a large value penalty term that is tuned during training, and $(\lambda_1, \lambda_2, \lambda_3)$ are hyperparameters for each constraint terms.

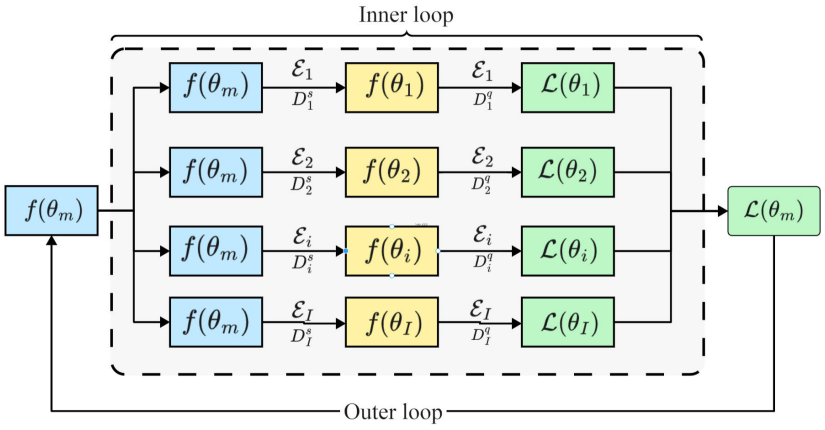


FIGURE 5.6. The learning process of MAML.

5.4 The Proposed Meta-GDRL

5.4.1 Motivation

In the previous section, we give a detailed description of each part of GDRL and explain why GDRL can efficiently perform task offloading and resource allocation with low complexity. However, in practice, the situation is more complex, where key parameters like available resources, location of LEO/HAPS, and task-related parameters may change dramatically due to the rapid movement of space and air segments. Even though DRL has a stronger generalization ability compared with optimization-based approaches, it has the limitation of being unable to maintain inference performance when a mismatch between the training and testing environment exists [100].

To address this problem, the Model-Agnostic Meta-Learning (MAML) approach is integrated into our proposed GDRL. Specifically, MAML enables the quick adaptation of the DRL method to a new environment with only a few training examples obtained from the new environment by leveraging the idea of meta-learning [101] .

5.4.2 Definition

We first define the phrase 'environmental tasks' as environments with different key parameters and 'tasks' as tasks generated by users. In this paper, key parameters include the parameters for user request generation and the total initial resources. These parameters are set to follow the distribution $p(\mathcal{E})$ with $\mathcal{E}$ denoting the environmental task. The core idea behind MAML is the assumption that its internal network representation (θ_m) of the distribution $p(\mathcal{E})$ has some shared features, and these features can be learned through training. These features can boost the learning process and make the DRL model adapt quickly to the new environmental tasks generated by the same distribution with only a few steps of training required. Generally, the learning process of MAML is constructed by two loops: inner and outer loop. The inner loop takes the responsibility of adapting θ_m to one environmental task, and the outer loop updates the globe model (representing common features) based on the loss obtained from the training under all environmental tasks.

The learning process of MAML is shown in Fig. 5.6. It can be seen that θ_i is calculated for each environmental task with the initial value equal to θ_m. It does not modify θ_m. Here, θ_i is the adjustable parameters for the environment task $\mathcal{E}_i$, and each $\mathcal{E}_i$ has its own θ_i. Another important notation is the supporting dataset for $\mathcal{E}_i$, we use $\mathcal{D}_i^s$ to represent this dataset.

The one-step inner loop update can be formulated as

$$\theta_i = \theta_m - \lambda_{\text{in}} \nabla_{\theta_m} \mathcal{L}_i \left(\theta_m; \mathcal{D}_i^s\right), \tag{5.41}$$

with λ_{in} representing the learning rate, and $\mathcal{L}_i$ denoting the loss function for $\mathcal{E}_i$.

After the inner loop, we can obtain the updated model $f(\theta_i)$ for each $\mathcal{E}_i$. Note that the initial model is denoted as $f(\theta_m)$. Utilizing a set of $f(\theta_i)$, we can generate a query dataset and can then perform the outer loop. The outer loop starts to update the $f(\theta_m)$ based on the loss obtained from all $\mathcal{E}_i$ using this dataset. This update process of θ_m is performed with the goal of minimizing the loss

$$\min_{\theta} \sum_{\mathcal{E}_i \sim p(\mathcal{E})} \mathcal{L}_i(\theta_i; \mathcal{D}_i^{\text{q}}), \tag{5.42}$$

where $\mathcal{D}_i^{\mathrm{q}}$ is the query dataset. Hence, the θ_m is updated as

$$\theta_m = \theta_m - \lambda_{\text{out}} \nabla_{\theta_m} \sum_{\mathcal{E}_i \sim p(\mathcal{E})} \mathcal{L}_i \left(\theta_i; \mathcal{D}_i^q\right), \tag{5.43}$$

with λ_{out} denoting the learning rate.

5.4.3 Implementation

To incorporate MAML into TRPO, we must analyze the TRPO loss functions and their parameter gradients. Specifically, the loss function in Eq. (4.30) for the TRPO policy network measures performance differences between the new policy π_{new} and the old policy π_{old}. This concept can be generalized as follows.

$$\mathcal{L}_{\pi_{\text{old}}}(\pi_{\text{new}}) = \mathop{\mathbb{E}}_{\mathbf{o},\mathbf{a}\sim\pi_{\text{old}}} \left[\frac{\pi_{\text{new}}}{\pi_{\text{old}}} A_{\pi_{\text{old}}}(\mathbf{o}, \mathbf{a}) \right]. \tag{5.44}$$

Here, the advantage function $A_{\pi_{\text{old}}}(\mathbf{o}, \mathbf{a})$ is approximated by its GAE-based estimate $A^{\text{GAE}}(\mathbf{o}, \mathbf{a})$ using the generalized advantage estimation method, as follows.

$$A^{\text{GAE}}(\mathbf{o}(t), \mathbf{a}(t)) = \sum_{l=0}^{\infty} (gn)^l \delta(t+l), \tag{5.45}$$

and

$$\delta(t+l) = \mathcal{V}(\mathbf{o}(t)) + gV_\pi(\mathbf{o}(t+1)) - V_\pi(\mathbf{o}(t)), \tag{5.46}$$

where $\mathcal{V}(\mathbf{o}(t))$ represents the reward function, and $n \in [0, 1]$ is the variance reduction parameter controlling the bias-variance trade-off. Following [102], the MAML loss $\mathcal{L}_i$ can be expressed as

$$\mathcal{L}_i = -\mathcal{L}_{\pi_{\text{old}}^i}(\pi_{\text{new}}^i), \tag{5.47}$$

with π^i representing the policy obtained from $\mathcal{E}_i$. Combining Eq. (5.41) and Eq. (5.47), the one-step inner loop parameter update is given by

$$\boldsymbol{\theta}_{\text{new}}^i = \boldsymbol{\theta}_m + \lambda_{\text{in}} \nabla_{\boldsymbol{\theta}_m} \mathcal{L}_{\pi_{\text{old}}^i}(\pi_{\text{new}}^i). \tag{5.48}$$

Note that the difference between $f(\boldsymbol{\theta}_{\text{new}})$ and $f(\boldsymbol{\theta}_m)$ must remain within the trust region to ensure a monotonic improvement guarantee. Thus, considering Eq. (6.23), the final update is

$$\boldsymbol{\theta}^i_{\text{new}} = \boldsymbol{\theta}_m + \kappa \sqrt{\frac{2\delta}{\mathbf{z}^T \mathbf{C}^{-1}_{\theta_m} \mathbf{z}}} \mathbf{C}^{-1}_{\theta_m} \mathbf{z}, \tag{5.49}$$

where the gradient $\mathbf{z}$ and can be calculated based on

$$\mathbf{z} = \frac{1}{ST} \sum_{t=0}^{ST} \hat{A}\left(\mathbf{o}(t), \mathbf{a}(t)\right) \nabla_{\boldsymbol{\theta_i}} \log \pi\left(\mathbf{a}(t) \mid \mathbf{o}(t)\right). \tag{5.50}$$

The outer loop for the update of θ_m should be

$$\theta_m = \theta_m + \lambda_{\text{out}} \nabla_{\theta_m} \sum_{\mathcal{E}_i \sim p(\mathcal{E})} \mathcal{L}_{\pi^i_{\text{old}}}(\pi^i_{\text{new}}). \tag{5.51}$$

Note that for the outer loop, we can choose a value of λ_{out} and do not need to satisfy the trust region constraint. For the value network of TRPO, the loss function is

$$\mathcal{L}_{\text{policy}} = \sum_{t=0}^{ST} \left(V_\phi\left(\mathbf{o}(t)\right) - \zeta(\mathbf{o}(t))\right)^2, \tag{5.52}$$

and thus the inner loop one-step update for the value network can be formulated as

$$\phi_i = \phi_m - \lambda_{\text{in}} \nabla_{\phi_m} \mathcal{L}_{\text{policy}}, \tag{5.53}$$

and the outer loop for the value network is

$$\phi_m = \phi_m - \lambda_{\text{out}} \sum_{\mathcal{E}_i \sim p(\mathcal{E})} \nabla_{\phi_m} \mathcal{L}_{\text{policy}}. \tag{5.54}$$

The algorithm for the integration of MAML and TRPO (MAML-TRPO) is shown in 2.

5.5 Simulation Results

In this section, simulation results are elaborated on to evaluate the performance of our proposed GDRL. We first validate that our proposed GDRL performs the best compared with other benchmark methods to solve the SAGIN task offloading and resource allocation problem $\mathcal{P}$. Then, we demonstrate the performance enhancements achieved through the

Algorithm 2 MAML-TRPO

I_{in}: number of inner loop steps
I_{out}: number of outer loop steps
E: number of environmental tasks
for I_{out} iterations **do**
  Sample E environmental tasks: $\mathcal{E}_i \sim p(\mathcal{E})$
  for each $\mathcal{E}_i$ **do**
    Copy the parameter θ_m to θ_i of $\mathcal{E}_i$
    for I_{in} iterations **do**
      Interact with the environment
      Compute $A^{\text{GAE}}(\mathbf{o}(t), \mathbf{a}(t))$ using Eq. (5.45)
      Compute $\mathbf{z}$ using Eq. (5.50)
      Update $\boldsymbol{\theta}^i_{\text{new}}$ using Eq. (5.49)
      Update ϕ_i using Eq. (5.53)
    end for
    Calculate $\mathcal{L}_i$ using Eq. (5.47)
    Calculate $\mathcal{L}_{\text{policy}}$ using Eq. (5.52)
    Update θ_m using Eq. (5.51)
    Update ϕ_m using Eq. (5.54)
  end for
end for

implementation of GFEN and AMN. We need to highlight a key point for all these following figures. All the baseline methods (except random) use our proposed encoding scheme and the trained AMN to directly generate actions. Without the usage of the encoding scheme and AMN, these methods cannot work properly due to the reasons mentioned in previous sections.

To begin with, we first list the relevant parameters for the simulation. For task generation, we use the following parameters:

$$\text{DT } s_u(t) \in [8000, 10000] \text{ bits}, \quad \text{DS } s_u(t) \in [200, 500] \text{ bits},$$

$$\text{DT } v_u(t) \in [5000, 6000] \text{ bits}, \quad \text{DS } v_u(t) \in [50, 200] \text{ bits},$$

$$\text{DT and DS } o_u(t) \in [1000, 3000] \text{ bits},$$

$$C_j \in [100, 130] \text{ for LEO}, \quad C_j \in [50, 80] \text{ for HAPS},$$

$$\tau^{(\max)} = 10^6 \text{ Hz}.$$

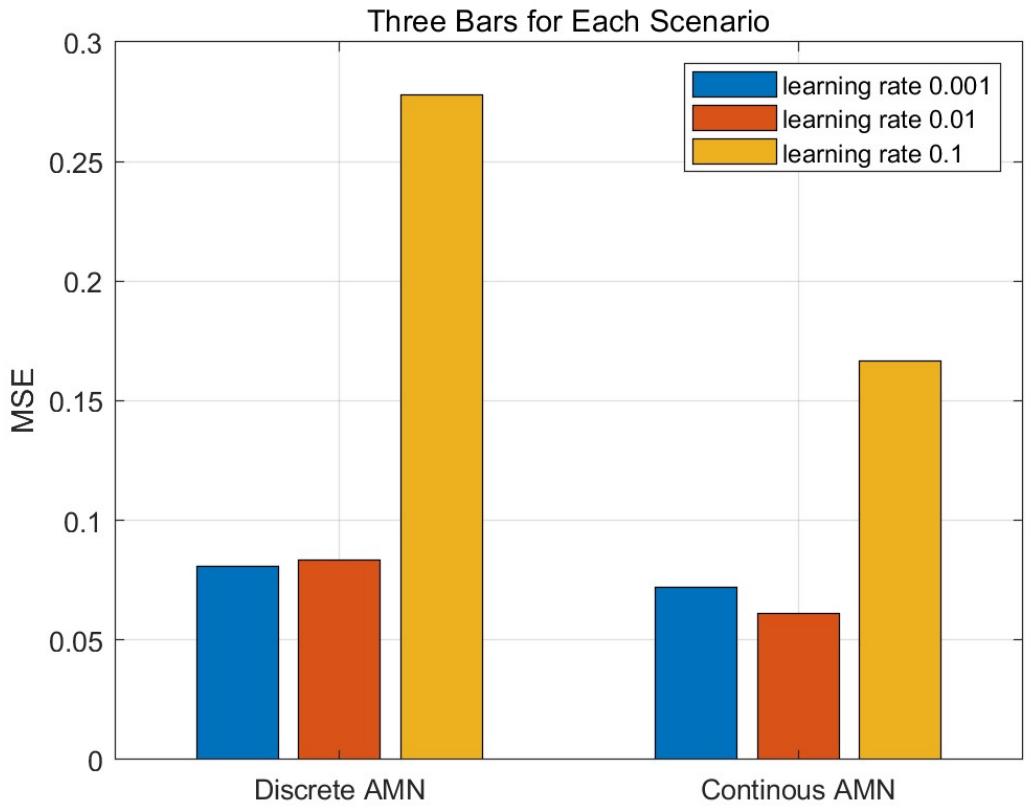


FIGURE 5.7. MSE between latent actions from TRPO and AMN.

For GDRL training, we set the batch size to 128, the experience horizon to 1024, the activation function to *Sigmoid*, and the optimizer to *Adam*. Additionally,

$$B = 15 \text{ kHz}, \quad \epsilon^{d}_{\mu^{D}_{u}} = 10^{-3}, \quad T_f = 10^{-3} \text{ s}, \quad K = 100,$$

$$N_0 = -173 \text{ dBm/Hz}, \quad G^{l/n}_{u} = 10, \quad M^{l/n}_{x/y} = 6.$$

5.5.1 Training Loss

To investigate the selection of learning rate and its influence of the reconstruction error produced by AMN, we conduct the test in Fig. 5.7. It shows the mean square error (MSE) between the latenct action generated by LAGN and that obtained from the output of AMN during the training process of GDRL. On the left hand side shows the MSE for discrete AMN, and on the right hand side shows that for continuous AMN. In this simulation, we select three

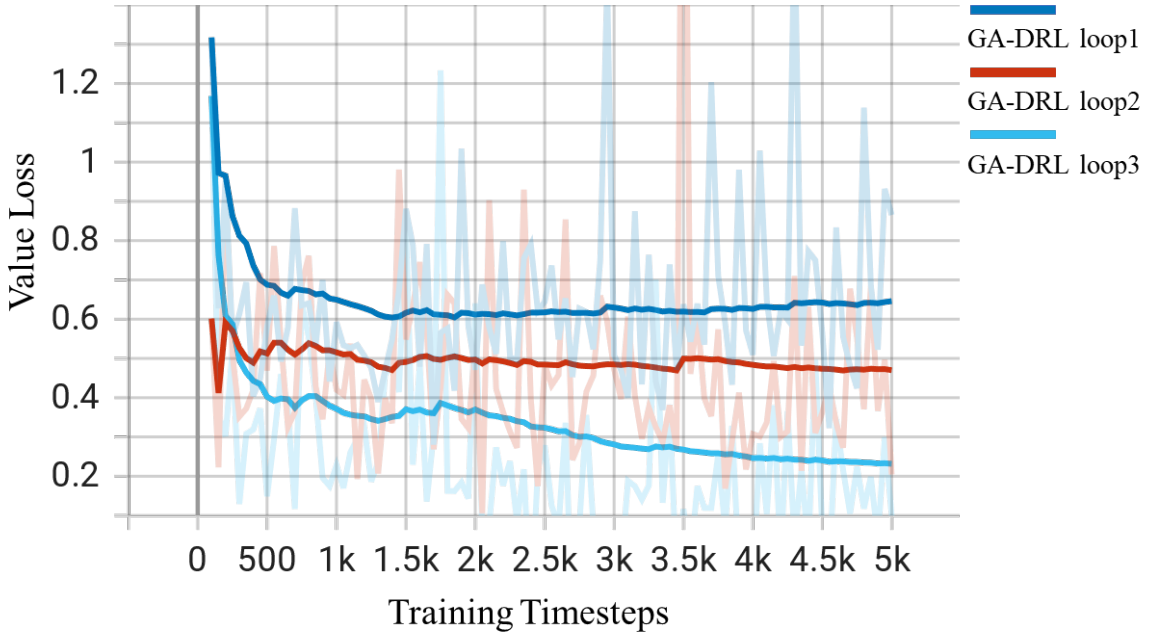


FIGURE 5.8. Value function loss for GDRL.

learning rates: 0.1, 0,01, and 0,001 and gives the MSE value. It can be noticed that learning rates of 0.01 and 0.001 yield lower variance than 0.1, indicating that both achieve similarly improved reconstruction performance relative to 0.1.

To reveal the impact of the number of training loops on value function loss of GDRL, we have carried out simulations for three training loops, and the results are presented in Fig. 5.8. Here, the value function loss is the difference between the GDRL's expectation of a state's value and the empirically observed value of that state. Note that, to collect latent actions and perform alternative training between AMN and TRPO, multiple training loops must be conducted. The solid lines represent the smoothed loss values, while the background near transparent lines indicate the actual loss at each timestep. It is clearly shown that, for each iteration, the general trend is that the smoothed value function loss reduces as learning progresses. It also reveals that the smoothed value function loss is lowered with the increase of the loop index, indicating the convergence of the alternative training between AMN and TRPO. When it comes to the actual value function loss, it is observed that, as the training index increases, the loss fluctuation range significantly narrows, indicating the improvement of the system stability.

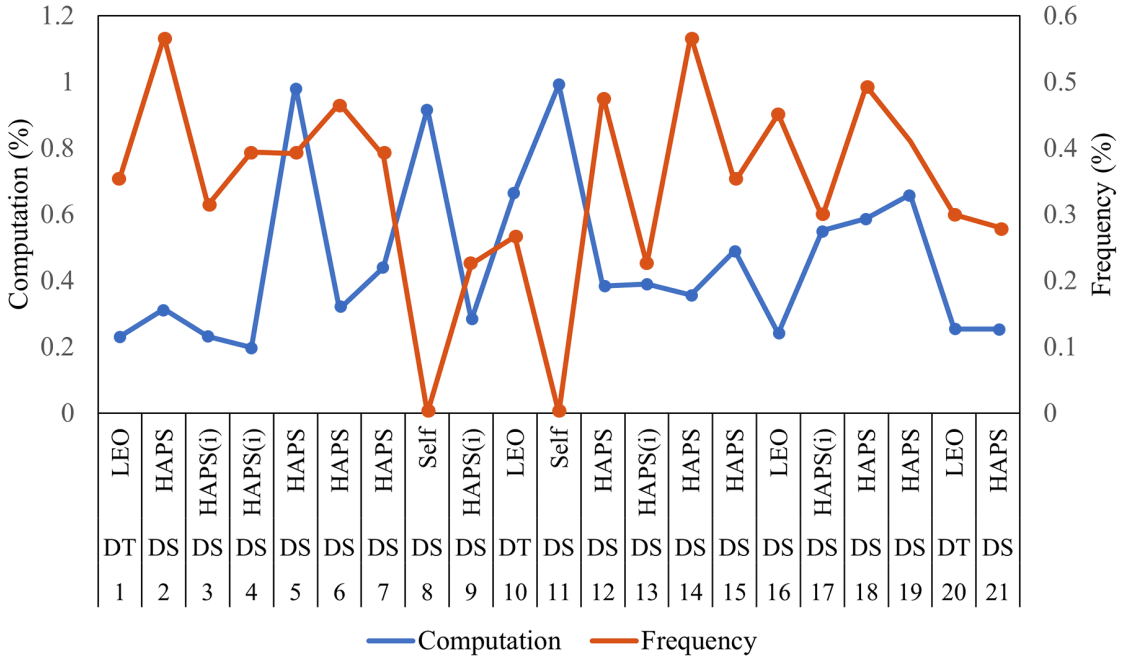


FIGURE 5.9. The task offloading and resource allocation actions for different service types.

One example of the task offloading and resource allocation decisions made by GDRL is shown in Fig. 5.9. Note that these decisions are for only one user, and the parameters of GDRL stay unchanged during this process (Inference) to give an insight of GDRL's decision-making procedure. It can be noticed that the x-axis is divided into three components: task type (DS/DT), timestep (1-21), and server (HAPS/LEO/HAPS(I)). Here, we denote the task was initially offloaded to one HAPS before being transferred to another as HAPS(I), and show a randomly selected 20 consecutive timesteps. The figure reveals that most DS tasks are directed to HAPS, while DT tasks are routed to LEO, likely due to LEO's closer proximity and lower latency. After several timesteps (2–7) of assigning tasks to HAPS, GDRL switches to local processing of DS tasks at steps 8 and 11, presumably to prevent exceeding resource constraints. This decision may also be influenced by the frequency resource allocation observed in the previous timesteps (6–7), where a large portion of the available frequency was already utilized.

5.5.2 Baseline Methods for Comparison

Here, the baseline methods adopted for comparison are introduced.

(1) *Random*: Under this method, task offloading and resource allocation decisions are generated based on a random distribution. It is obvious that this is the worst case scenario and can served as the baseline for all other methods.
(2) *TRPO* [91]: An on-policy method that can only generate one type of actions. It has a monotonic policy improvement guarantee compared with others.
(3) *PPO* [95]: Proximal policy optimization (PPO) is another on-policy method which is a simplified version of TRPO. Compared with TRPO, it has lower training complexity.
(4) *SAC* [96]: A well-known off-policy method which is featured by regularization. It is trained with the goal of balancing between return and entropy, which can be treated as the balancing between exploration and exploitation in other methods.
(5) *DDPG* [97]: Another off-policy method that closely resembles Q-learning. It utilizes off-policy data and Belmore equation to learn a Q-function.

5.5.3 Performance Comparison

Fig. 5.10 compares the return between our proposed GDRL and the baseline methods. It can be seen that GDRL achieves the highest return among all methods. For off-policy methods DDPG and SAC, each of them converges to its own sub-optimal solution which is worse than on-policy methods TRPO and PPO. Note that, compared with on-policy methods, off-policy methods introduce more randomness into the environment and thus are less robust in solving problem $\mathcal{P}$. It is also seen that for GDRL, the case with GFEN significantly outperforms the case without GFEN. This clearly verifies the benefit that GFEN brings to GDRL. It is also observed that the initial training result for our proposed GDRL with GFEN is significantly higher than all other methods. This is because, as stated in Section VI A, GDRL has been trained twice beforehand, while all the others have not been trained at all beforehand. Overall, Fig. 5.10 substantiates the superiority of our proposed GDRL over the baseline methods.

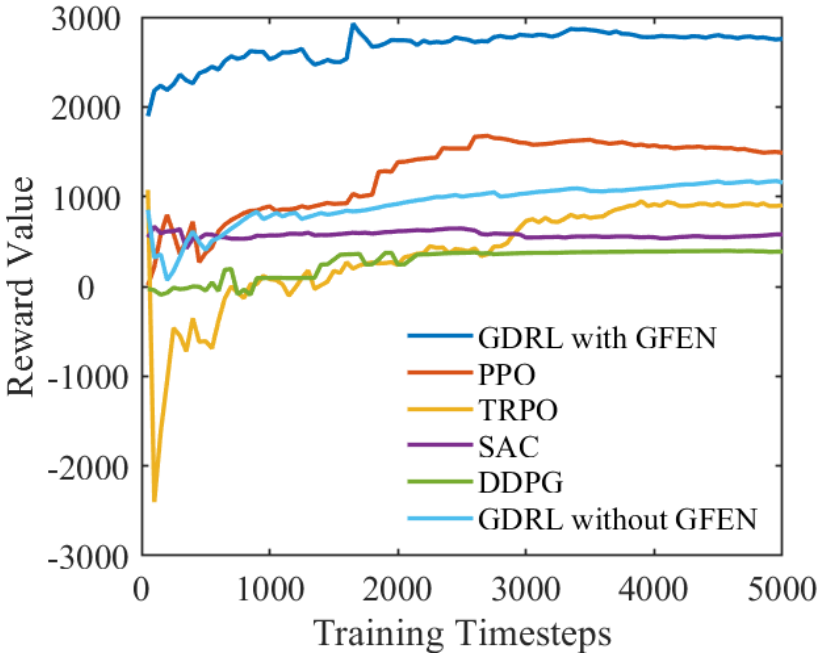


FIGURE 5.10. Training reward comparison.

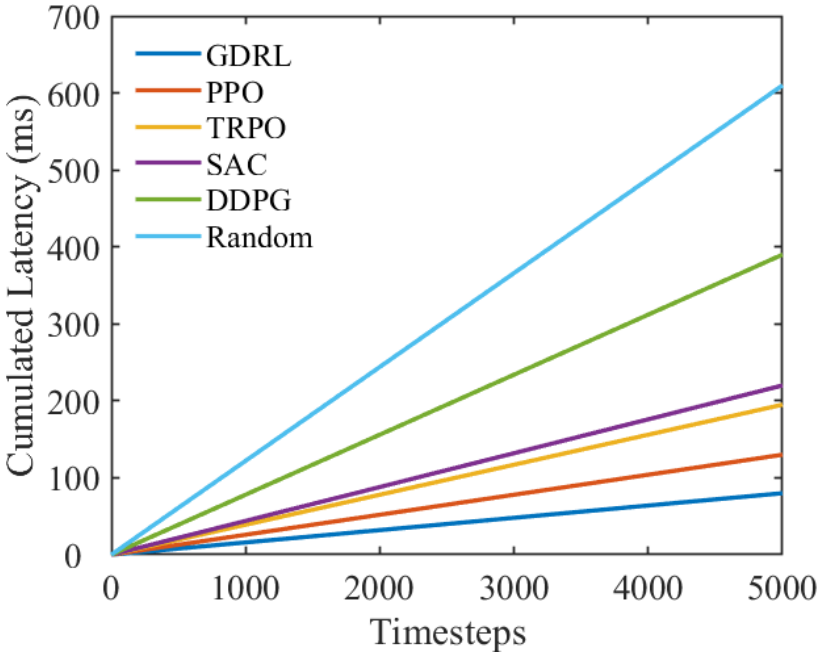


FIGURE 5.11. Cumulative latency comparison.

After training, a comparison is carried out in Fig. 5.11 in terms of cumulative latency. It can be seen that GDRL achieves the lowest cumulative latency consistently across all timesteps. In contrast, as expected, the random method is inferior to all the other methods. It can also

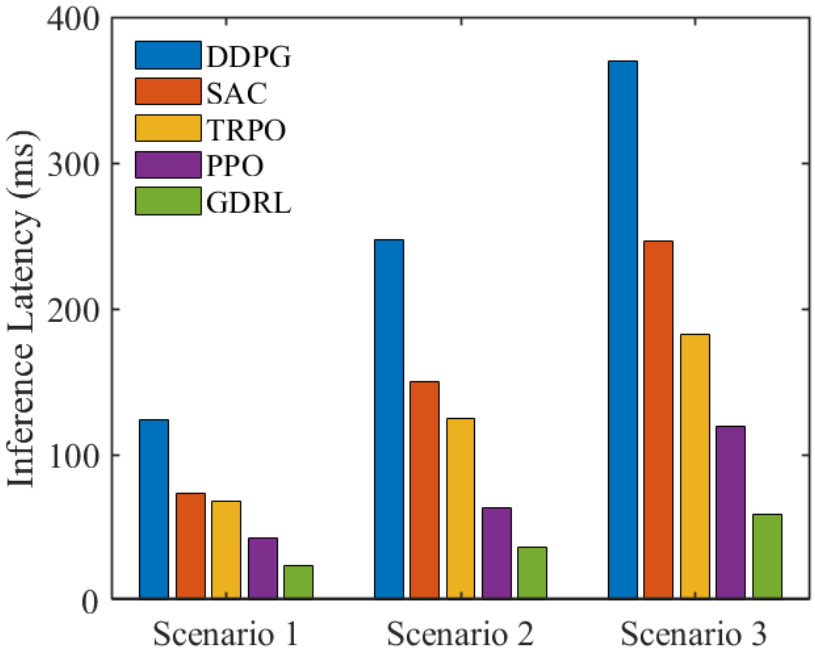


FIGURE 5.12. Inference latency comparison.

be seen that all methods exhibit an increase in cumulative latency over time. This trend is to be expected, as there is an inherent accumulation of processed tasks over time. Furthermore, GDRL has the least slope, indicating that it has the lowest rate of latency accumulation as time progresses. This verifies that GDRL can efficiently minimize latency and maintain long-term consistent performance. It can also be noticed that the slopes of DDPG and SAC are steeper compared to TRPO and PPO, which is consistent with other results. In summary, GDRL's performance in latency management outperforms the baseline methods, evidencing its superior efficiency in solving $\mathcal{P}$.

Note that, in Fig. 5.12, the total available resource is assumed to be fixed. To evaluate these methods in a more practical environment, we compare the inference latency in Fig. 5.12 for three scenarios where the total available resources of the LEO/HAPS vary with time. For Scenarios 1-3, LEO and HAPS resources are set to be between 150-200 and 100-150, between 100-150 and 50-80, and between 50-80 and 30-60, respectively. Specifically, Scenarios 1-3, in ascending order, represent abundant, moderate, and stringent resource availability, respectively. For each of these five methods, it is shown in Fig. 5.12 that, consistent with expectation, the inference latency increases with the reduction in available resource levels. In

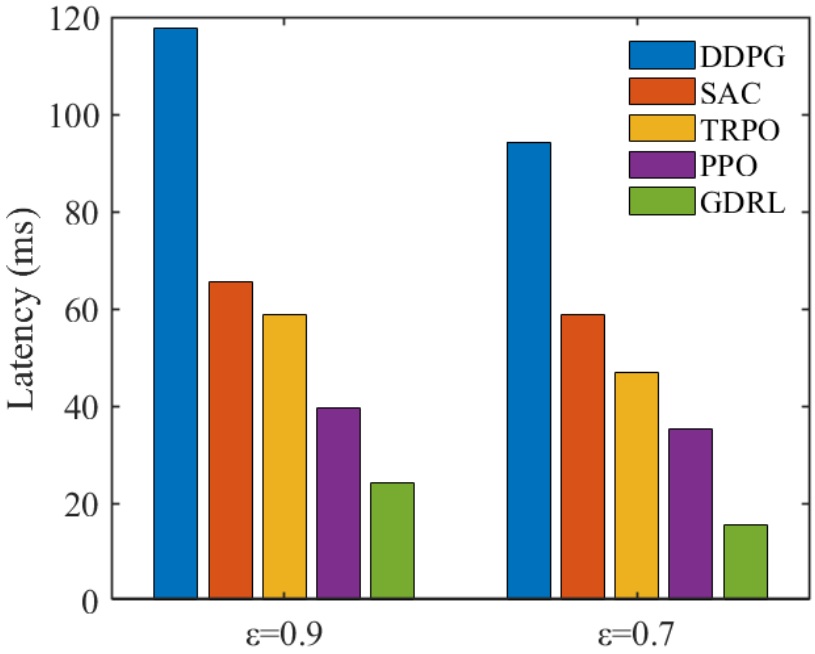


FIGURE 5.13. Latency comparison under different ϵ.

addition, in all three scenarios, GDRL achieves significantly lower inference latency than all the other four methods, verifying the advantage of GDRL regardless of the available resource level. It is also observed that, in all resource levels, on-policy methods TRPO and PPO result in lower latency than off-policy methods SAC and DDPG, which is consistent with the observation in Figs. 5.10 and 5.11. Overall, Fig. 5.12 clearly proves that GDRL performs best in resource allocation to each task to reduce latency for a broad range of available resource levels.

A similar comparison is carried out in Fig. 5.13 for two different task generation frequencies, where ϵ=0.9 embodies a high task generation probability scenario and ϵ=0.7 a moderate one. It can be seen that, for each method, latency increases when ϵ increases from 0.7 to 0.9, as fewer tasks are generated for ϵ=0.7. In addition, the relative supremacy of these methods in terms of latency is the same as that in Fig. 5.12.

5.6 Computational Complexity and Scalability

In this section, we discuss the computational complexity and the scalability of the proposed method. specifically, the computational complexity analysis of the DL-based approaches consists of two parts: the training and inference complexity. Here, the training complexity refers to the complexity that experienced during the training process which is mainly caused by updating parameters and backpropogation. The inference complexity refers to the complexity occurred during the inference process, where the parameters stay unchanged. This complexity originates from the action generation based on current environment states.

5.6.1 Training Complexity

The GFEN consists of two layers of GCN and four layers of multi-layer perception (MLP). The computational complexity can be represented as [103]

The training complexity for GDRL can be separated in two three parts according to its structure, namely the GFEN, the LAGN, and the AMN part. For GFEN, the computational complexity can be derived from [103]

$$\begin{aligned} C_{\text{MLP}} &= n_i n_1 + \sum_{l=2}^{4} n_{l-1} n_l + n_4 n_o, \\ C_{\text{GCN}} &= \sum_i F_i^2 |\mathcal{E}| + F_i |\mathcal{V}|, \\ C_{\text{GFEN}} &= C_{\text{GCN}} + C_{\text{MLP}}, \end{aligned} \tag{5.55}$$

for a 4-layer MLP, a 2-layer GCN, and the GFEN, respectively. In this notation, n_i is the input dimension, n_1 and n_2 are the numbers of neurons in the first and second MLP layers, F_i denotes the node features for the i-th GCN layer, $|\mathcal{E}|$ is the number of edges, and $|\mathcal{V}|$ is the number of nodes.

The LAGN consists of a total of eight layers MLP with four layers critic network and four layers actor network. Accordingly, its computational complexity can be represented as [104]

$$C_{\text{LAGN}} = 2\left(n_i n_1 + \left(\sum_{l=1}^{3} n_l n_{l+1}\right) + n_3 n_o\right). \tag{5.56}$$

Finally, the AMN is constructed by four layers MLP where the complexity of each layer can be calculated based on Eq. (6.38), and two layers of LSTM with the each layers' complexity of

$$C_{\text{LSTM}} = n_i n_h (4n_i + 4n_h + 3 + n_o), \tag{5.57}$$

where n_h is the number of hidden units. Accordingly, AMN's complexity is

$$C_{\text{AMN}} = 2C_{\text{LSTM}} + C_{\text{MLP}}. \tag{5.58}$$

Thus, the training complexity of GDRL can be represented as

$$C_{\text{GDRL}} = C_{\text{GFEN}} + C_{\text{LAGN}} + C_{\text{AMN}}. \tag{5.59}$$

5.6.2 Inference Complexity

As the GDLR model is trained offline, the online inference complexity remains very low, which can be expressed as $O(n_{\text{all}})$ [105], where n_{all} denotes the overall number of neurons.

5.6.3 Complexity Comparison

To ensure a fair computational complexity comparison, the methods in [80, 81] as well as the baseline algorithms (TRPO, SAC, PPO, DDPG) must also integrate AMN. Without AMN, the discrete action space has size $2 + 2^{(N+L)} + 2^{(N+L)^2}$, which grows exponentially with the number of LEOs/HAPS and prevents these schemes from converging. By incorporating AMN, the action space size is reduced to $2^{3+\log_2(\max(N,L))}$, making the problem tractable. When all methods include AMN, GDRL's training complexity is higher due to the GFEN. However, since only two node features are used, the additional complexity of GFEN remains low and

scales linearly with the number of neurons. Simulations confirm that this minor increase in complexity significantly boosts performance. The key factor impacting system scalability is computational complexity. GDRL employs AMN and an efficient encoding scheme to substantially reduce the offline training burden. Moreover, its DRL-based nature allows for low-complexity online deployment after training. As a result, GDRL is highly scalable and can support large numbers of UEs, LEOs, and HAPS in practical settings.

5.6.4 Scalability

The primary barrier to system scalability is computational complexity. GDRL integrates the AMN and an efficient encoding scheme to greatly reduce the complexity of offline training. Moreover, as a DRL-based method, GDRL can be deployed online with minimal overhead once trained. This leads to high scalability, enabling GDRL to support large numbers of UEs, LEOs, and HAPS for practical applications.

5.7 Conclusion

In this chapter, we presented an efficient online task offloading and resource allocation approach in SAGIN, with its performance validated by simulation. As stated at the beginning of this chapter, efficient task offloading/resource allocation is crucial for the integration of three segments in SAGIN and achieving the best system performance with limited available resources. Specifically, we proposed GDRL to generate both task offloading and resource allocation decisions in an end-to-end manner. The key feature of GDRL is the utilization of GNN to capture the features of topologically structured SAGIN environment states. In GDRL, a state feature extraction network is proposed to extract features from different types of environment states, a latent action generation network to generate latent actions from the extracted state features and an action mapping network that leverages the encoding scheme to generate resource allocation and offloading decisions directly. Each part features a modified encoder-decoder structure and can be easily adapted to other networks to leverage their advantages.

Additionally, at the beginning of this chapter, we listed the limitations of current state-of-the-art approaches. In the following, we conclude how GDRL mitigates these limitations.

(1) In this work, we proposed a time-sequential decision-making problem that takes into account the correlations between actions. We formulate a SAGIN model that encompasses two timescales, diverse user priorities, and different antenna configurations, which can more accurately reflect the real environment compared with other SAGIN models.
(2) In this work, the meta-learning approach is integrated into GDRL to mitigate the differences between the training and testing environments. Specifically, MAML enables the quick adaptation of the DRL method to a new environment with only a few training examples obtained from the new environment by leveraging the idea of meta-learning.
(3) In this work, an encoding scheme is proposed to replace the conventional one-hop representation. This encoding scheme can significantly reduce the size of the action space by eliminating invalid action combinations. Additionally, we proposed a separate network referred to as AMN to directly generate both discrete offloading and continuous allocation actions. AMN is of autoencoder structure and can be trained with low complexity. It can also be easily implemented in other networks to mitigate the limitations of DRL-based approaches.

The work proposed in this chapter focuses on inter-task features, as stated in Chapter 1. Thus, the intra-task features like task dependency is not considered in this work. In the next chapter, intra-task features are emphasized, which also bring new challenges like the changing number of air/space segments, the feature extraction of topologically structured tasks, and the handling of two graphs. These challenges will be described in detail in the next chapter.

CHAPTER 6

Task Offloading and Resource Allocation in SAGIN with Task Dependencies

---

In this chapter, we present the task offloading and resource allocation strategy referred to as GF-DRL to utilize the resources from all three segments efficiently. Different from Chapter 4, this work focuses on intra-task features as discussed in Chapter 1 (introduction). In the following, we first give an introduction of intra-task features. Then, we present our system model and formulate the problem based on this model. After that, we present our solution, GF-DRL, and give a detailed description of each part. Then, we introduce the reincarnating technique and implement this into GF-DRL to achieve low retraining complexity. Simulation results validate the performance of GF-DRL.

Intra-task features are seldom explored in the field of task offloading resource allocation in SAGIN. Of all the works listed in Chapter 2, only 2 works consider task dependencies. Additionally, for both works, no modification of the task offloading/resource allocation approaches is made to cater to these dependencies. Despite the little attention paid to it, intra-task features are very important in real-world SAGIN implementation. The reasons are summarized as follows.

(1) Most tasks generated by one application/user are dependent, which means that the processing of one task requires the results of another. Ignoring the task dependencies results in the exclusion of a wide range of service types.
(2) Simplifying the relationship among tasks leads to the degrading performance of task offloading/resource allocation schemes when implemented in the real world. As all these methods do not consider task dependency in the first place, all dependencies

between tasks will be lost, which inevitably leads to slower convergence, suboptimal offloading/allocation decisions, and higher costs compared with ones that take into account these dependencies.

In the previous chapter, we summarized the importance of efficient task offloading/resource allocation in the successful implementation of SAGIN. As little work focuses on intra-task features, this motivates us to investigate the task offloading and resource allocation scheme in SAGIN that focuses on intra-task correlations.

Next, in Chapter 2, we present a detailed literature review on task offloading/resource allocation schemes for SAGIN. From the perspective of intra-task features, these works have the following limitations.

(1) Most of these works do not take into account the correlation between tasks, which limits the type of services covered. For works that consider task dependency, this dependency is not considered as graphical data and only task offloading decisions are provided. Additionally, no modification of the task offloading/resource allocation approaches is made to cater to these dependencies.
(2) None of these methods take into account the inherent topological structure of the SAGIN states and tasks. Since graph neural networks perform better with graph-structured data compared to multilayer perceptrons (MLPs), these methods are inefficient at capturing features in the SAGIN environment due to MLP-based DRL. This results in slow convergence and performance degradation in terms of minimizing latency.
(3) All these methods assume that the total number of segments in SAGIN remains constant. This assumption is essential for DRL-based approaches since the input and output dimensions of DRL must stay fixed. However, due to the movement of LEO/HAPS, the number of segments can change dynamically, necessitating frequent retraining of the DRL-based methods.

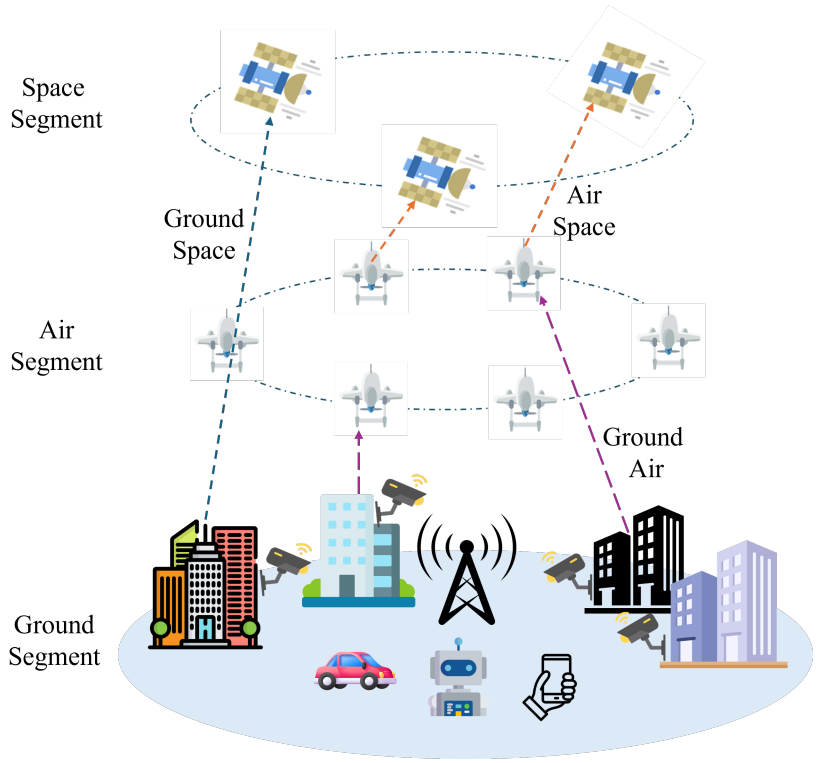


FIGURE 6.1. System architecture of the space-air-ground integrated network (SAGIN).

6.1 System Model and Problem Formulation

6.1.1 Network Graph Model

We consider a SAGIN shown in Fig. 6.1, which consists of three segments: the space segment (LEO), the air segment (HAPS), and the ground segment (UE). Note that, unlike the system in the previous chapter, LEO, HAPS, and UE are all equipped with single-input, single-output (SISO) antennas. In this network, there are L LEOs, N HAPSs, and U UEs. Both LEOs and HAPSs are equipped with computing servers capable of processing tasks generated by UE. We further divide the time horizon into T equal slots with $t \in \{1, \cdots, T\}$ denoting each timeslot. This approach is widely adopted in scheduling and resource allocation problems, as it simplifies analysis by discretizing continuous time into manageable intervals. To better represent the typology of this network, in this paper, we introduce an undirected communication graph to model the interaction among these segments, i.e., $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, f_v, f_e\}$,

where $\mathcal{V}$ represents nodes, including LEOs, UAVs, and UEs, $\mathcal{E}$ represents edge connections between nodes, f_v maps each node to its features $v \to \mathbb{R}^{d_v}$ with d_v represents the feature dimension of node v, and f_e maps each edge to its features $e \to \mathbb{R}^{d_e}$ with d_e represents the feature dimension of edge e. We define node sets for LEOs, HAPSs, and UEs as $l \in \mathcal{L} = \{1, \cdots, L\}$, $n \in \mathcal{N} = \{1, \cdots, N\}$, and $u \in \mathcal{U} = \{1, \cdots, U\}$, respectively. We further categorize the LEO/HAPS into two groups: access and non-access. Access LEO/HAPS can exchange data with UE, while non-access ones can only exchange data through the relay via access LEO or HAPS. Thus, the edge exists only between each UE and its corresponding access LEO/HAPS. The node sets for access LEOs and non-access LEOs are defined as $l^A \in \mathcal{L}^A$ and $l^{NA} \in \mathcal{L}^{NA}$, respectively, with $\mathcal{L}^A \cup \mathcal{L}^{NA} = \mathcal{L}$. Also, the node sets for access HAPSs and non-access HAPSs are defined as $n^A \in \mathcal{N}^A$ and $n^{NA} \in \mathcal{N}^{NA}$, respectively, with $\mathcal{N}^A \cup \mathcal{N}^{NA} = \mathcal{N}$. The total number of nodes is $|\mathcal{V}| = L + N + U$.

6.1.2 Task Graph Model

We assume that at each timeslot, each user generates one task [1], containing J $(j = 1, \cdots, J)$ subtasks with inherent dependency [43]. Thus, there are a total of U $(i = 1, \cdots, U)$ tasks. Each task generated by a UE requires several timeslots to be finished. The dependencies between subtasks are modeled as parent-child relationships. For example, in Fig. 6.2, subtask 1 and subtask 9 are referred to as the entry subtask and exit subtask, respectively. Subtask 2 (subtask 4) is the parent (child) of subtask 4 (subtask 2), and subtask 4 needs the results from both subtask 2 and subtask 3. To better represent task i generated at timeslot t, we define it as a four-item tuple $(o_i(t), s_i(t), \upsilon_i(t), \iota_i(t))$ [87], where $o_i(t)$ is the computational workload in CPU cycles per bit, $s_i(t)$ the original task size in bits, $\upsilon_i(t)$ the size of the task after processing, and $\iota_i(t)$ the maximum delay allowed for task i in number of timeslots. We can then formulate a three-item tuple as $(o_i^j(t), s_i^j(t), \upsilon_i^j(t))$ to represent subtask j within task

[1] This assumption is made to test the system's performance under extreme circumstances. Masking can be applied to exclude users who do not generate tasks.

i. The relationship between a task and its associated subtasks can be shown as

$$o_i(t) = \sum_{j=1}^{J} o_i^j(t) \tag{6.1a}$$

$$s_i(t) = \sum_{j=1}^{J} s_i^j(t) \tag{6.1b}$$

$$\upsilon_i(t) = \sum_{j=1}^{J} \upsilon_i^j(t). \tag{6.1c}$$

To better capture the subtask dependency within a task, we further model each task as a graph $\mathcal{D}(t) = \{\mathcal{M}(t), \mathcal{Q}(t), f_m(t), f_q(t)\}$, where $\mathcal{M}(t)$ represents nodes (subtask), $\mathcal{Q}(t)$ represents edges (dependencies between subtasks), $f_m(t)$ maps each node to its feature $m \to \mathbb{R}^{d_m}$ with d_m being the feature dimension of node m, and $f_q(t)$ maps each edge to its feature $q \to \mathbb{R}^{d_q}$, where d_q denotes the feature dimension of edge q. If the computation of one subtask requires the result of another one, an edge is established between the two subtasks.

6.1.3 Channel Model

After defining the network and task model, we then present the channel model for three types of links: ground-space, ground-air, and air-space, as shown in Fig. 6.1. The air-space transmission experiences free space propagation, and the channel gain can be calculated as [85]

$$I_{n,l} = \frac{G_n G_l \lambda_c^2}{(4\pi d_{n,l})^2}, \tag{6.2}$$

where G_n and G_l are the antenna gains for HAPS n and LEO l, respectively, $d_{n,l}$ is the distance between them, and λ_c denotes the carrier wavelength. The corresponding transmission rate is

$$R_{n,l} = B_{n,l} \log_2 \left(1 + \frac{p_{n,l} I_{n,l}}{\sigma_{n,l}^2}\right), \tag{6.3}$$

where $B_{n,l}$ denotes the single subcarrier bandwidth between HAPS n and LEO l, $p_{n,l}$ is the transmission power between them, and $\sigma_{n,l}$ represents the power of the noise at the receiver.

The ground-air channel gain can be represented as [76]

$$I_{n,u} = \frac{G_n G_u \lambda_n^2 w_{n,u}}{(4\pi d_{n,u})^2}, \tag{6.4}$$

where G_n and G_u are the antenna gains of HAPS n and UE u, respectively, $d_{n,u}$ represents the distance between HAPS n and UE u, and $w_{n,u}$ represents the environmental attenuation covering rain attenuation and atmospheric loss. In this paper, we utilize the value provided by ITU-R in [106]. The corresponding transmission rate is

$$R_{n,u} = B_{n,u} \log_2 \left(1 + \frac{p_{n,u} I_{n,u}}{\sigma_{n,u}^2}\right), \tag{6.5}$$

where $B_{n,u}$ represents the single subcarrier bandwidth between HAPS n and UE u, $p_{n,u}$ is the transmission power between HAPS n and UE u, and $\sigma_{n,u}$ denotes the power of the noise at the receiver.

The ground-space channel shares the same model as the ground-air channel, and its channel gain can be written as

$$I_{l,u} = \frac{G_l G_u \lambda_l^2 w_{l,u}}{(4\pi d_{l,u})^2}. \tag{6.6}$$

Here, the environmental effect $w_{l,u}$ consists of two components [107]: tropospheric effect and ionospheric effect. In this paper, we utilize the attenuation value recommended by ITU-R in [108] to represent these two effects. Then, the corresponding transmission rate is

$$R_{l,u} = B_{l,u} \log_2 \left(1 + \frac{p_{l,u} I_{l,u}}{\sigma_{l,u}^2}\right). \tag{6.7}$$

In this paper, we use R to represent the transmission rate for all three channels.

6.1.4 Computing Model

Each subtask can be processed by locally by a UE, or offloaded to LEO, or HAPS. The following constraints prevent offloading loops.

(1) Subtasks received by non-access LEO/HAPS can not be further offloaded.

(2) No offloading is allowed between access LEO/HAPS.

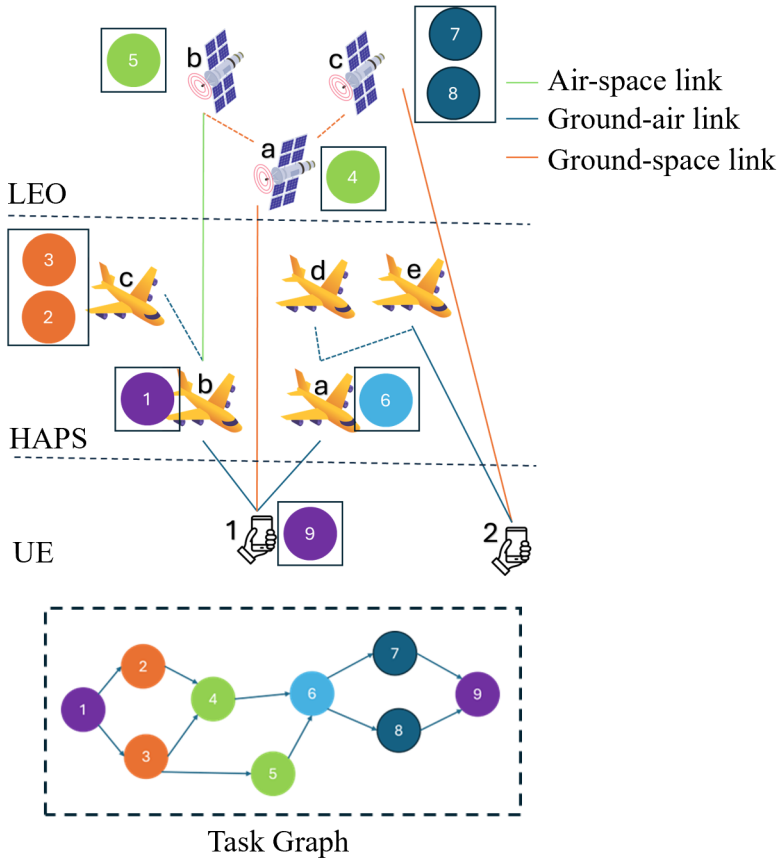


FIGURE 6.2. One example for task offloading.

Fig. 6.2 shows an example of task offloading in a SAGIN network, with the numbered circles corresponding to the subtasks shown in the task graph. In this example, subtask 9 is locally processed by UE 1. Subtask 6 is offloaded to access HAPS a, and it can be further offloaded to non-access HAPS d. Subtask 1 is offloaded to access HAPS b and can be further offloaded to non-access LEO b. Subtask 4 is offloaded directly to access LEO a. We also define the offloading from UE to access LEO/HAPS as one-step offloading and from UE to access followed by non-access LEO/HAPS as two-step offloading.

Let $a \in \{n, l\}$ denote either LEO l or HAPS n, a^A access LEO/HAPS, and a^{NA} non-access LEO/HAPS. Then, we can define the following three indicators to represent different offloading decisions.

(1) Local processing indicator $\alpha_i^j(t)$, where $\alpha_i^j(t) = 1$ indicates that subtask j of task i generated at timeslot t is processed locally at UE.

(2) One-step offloading indicator $\beta_{i,a}^{j}(t)$, where $\beta_{i,a}^{j}(t) = 1$ indicates that subtask j of task i generated at timeslot t is offloaded to LEO l or HAPS n.

(3) Two-step offloading indicator $\eta_{i,a^A \to a^{NA}}^{j}(t)$. Here, $\eta_{i,a^A \to a^{NA}}^{j}(t) = 1$ indicates that subtask j of task i generated at timeslot t is offloaded to the access LEO/HAPS, followed by non-access LEO/HAPS.

Clearly, based on the above three indicators, we have the following constraint

$$\alpha_i^j(t) + \sum_a \beta_{i,a}^j(t) + \sum_{a^A} \sum_{a^{NA}} \eta_{i,a^A \to a^{NA}}^j(t) \leq 1. \tag{6.8}$$

Specifically, the constraint restricts the offloading decision for each subtask to one of the following options: dropped, locally processed, one-step offloading, or two-step offloading.

The latency for task offloading includes three components: transmission latency, processing latency, and waiting latency. Here, waiting latency refers to the time a subtask must wait until all the results from its parent subtasks are available. For example, in Fig 6.2, this refers to the time that subtask 3 must waits for the results of subtask 1 and subtask 2. Transmission latency counts for the transmission time in uplink (UE to LEO/HAPS) and downlink (LEO/HAPS to UE). The uplink transmission latency can be written as

$$L_i^{j,up}(t) = \begin{cases} 0 & \text{local processing} \\ \frac{s_i^j(t)}{K_i^j(t) R_{a,u}(t)} & \text{one-step offloading} \\ \frac{s_i^j(t)}{K_i^j(t) R_{a,u}(t)} + \frac{s_i^j(t)}{K_i^j(t) R_{n,l}(t)} & \text{two-step offloading,} \end{cases} \tag{6.9}$$

where $K_i^j(t)$ represents the number of subcarriers allocated to subtask j of task i, $s_i^j(t)$ denotes the j-th subtask size in i-th task. The downlink transmission latency can be formulated as

$$L_i^{j,down}(t) = \begin{cases} \frac{v_i^j(t)}{K_i^j(t) R_{a,u}(t)} & \text{if the last subtask} \\ 0 & \text{otherwise,} \end{cases} \tag{6.10}$$

where $v_i^j(t)$ represents the j-th subtask size after processing. Then, the total transmission latency for subtask j can be calculated as

$$L_i^{j,tr}(t) = L_i^{j,up}(t) + L_i^{j,down}(t). \tag{6.11}$$

The processing latency for subtask j can be calculated as

$$L_i^{j,pr}(t) = \begin{cases} \frac{s_i^j(t)o_i^j(t)}{C_u} & \text{local processing} \\ \frac{s_i^j(t)o_i^j(t)}{C_{i,a}^j(t)} & \text{otherwise,} \end{cases} \tag{6.12}$$

where C_u represents the computational resources UE u have, $C_{i,a}^j(t)$ represents the computational resources LEO l or HAPS n allocated to subtask j, and $o_i^j(t)$ represents the computational workload in CPU cycles per bit for subtask j.

For waiting latency, we denote the set of subtasks at the g-th layer of task i generated at timeslot t as $\mathcal{O}_g(t)$. Subtasks are considered to be in the same layer if they have a common parent subtask. For example, in Fig. 6.2, subtask 2 and subtask 3 are in the same layer as they have the same parent subtask 1. We also denote the parent subtask for subtask j as $j' \in \mathcal{P}^j$. Assuming that subtask j is in layer g, the waiting latency can be calculated as

$$\begin{aligned} L_b^{wa}(t) = &\max_{j' \in \mathcal{P}^j \cap \mathcal{O}_{g-1}(t)} (T_i^{j,j'}(t)) \\ &+ \max_{j' \in \mathcal{P}^j \cap \mathcal{O}_{g-1}(t)} (L_i^{j',tr}(t) + L_i^{j',pr}(t)) \\ &- \min_{j' \in \mathcal{P}^j \cap \mathcal{O}_{g-1}(t)} (L_i^{j',tr}(t) + L_i^{j',pr}(t)), \end{aligned} \tag{6.13}$$

where $\max_{j' \in \mathcal{P}^j \cap \mathcal{O}_{g-1}(t)}(T_i^{j,j'}(t))$ denotes the maximum transmission time for any parent subtask j' to j. Here, $T_i^{j,j'}(t)$ can be calculated as

$$T_i^{j,j'}(t) = \begin{cases} 0 & \text{if } j \text{ and } j' \text{ are at same } u/l/n \\ \frac{v_i^{j'}(t)}{K_i^{j'}(t)R_{a,u}(t)} & \text{one subtask at } u \text{ and one at } l/n \\ \frac{v_i^{j'}(t)}{K_i^{j'}(t)R_{n,l}(t)} & \text{if } j \text{ and } j' \text{ are at different } l/n. \end{cases} \tag{6.14}$$

Additionally, $\max_{j' \in \mathcal{P}_j \cap \mathcal{O}_{g-1}(t)}(L_i^{j',tr}(t) + L_i^{j',pr}(t))$ and $\min_{j' \in \mathcal{P}_j \cap \mathcal{O}_{g-1}(t)}(L_i^{j',tr}(t) + L_i^{j',pr}(t))$ denote the maximum and minimum of the combined (transmission + processing) latency experienced by all parent subtasks j' of j in the layer $g-1$, respectively.

Then, the total latency of task i can be calculated as

$$L_i(t) = \sum_g \max_{j \in \mathcal{O}_g(t)} (L_i^{j,tr}(t) + L_i^{j,pr}(t) + L_i^{j,wa}(t)). \tag{6.15}$$

With this formation, we can now define our problem in the following subsection.

6.1.5 Problem Formulation

In this subsection, we present the task offloading and resource allocation problem in SAGIN. It takes into account the topological structures of both UE and SAGIN environment and short-term constraints. Specifically, this problem is formulated as an MINLP problem and is NP-hard with large action space. To begin with, we define an indicator for the completion of task i generated at timeslot t as

$$\Delta_i(t) = \begin{cases} 1 & \text{if completed within task delay constraint } \iota_i(t) \\ 0 & \text{else.} \end{cases} \tag{6.16}$$

Then, the task offloading and resource allocation process can be formulated as an optimization problem in the form of

$$\mathcal{P}: \max_{\boldsymbol{\alpha}(t),\boldsymbol{\beta}(t),\boldsymbol{\eta}(t),\mathbf{C}(t),\mathbf{K}(t)} \frac{1}{TU} \sum_{t=1}^{T} \sum_{i=1}^{U} \Delta_i(t) \tag{6.17}$$

$$\sum_{t=1}^{T} \sum_{i=1}^{U} \sum_{j=1}^{J} C_{i,a}^j(t) \leq C_a, \tag{6.18a}$$

$$\sum_{t=1}^{T} \sum_{i=1}^{U} \sum_{j=1}^{J} K_i^j(t) \leq K, \tag{6.18b}$$

$$(6.8), \tag{6.18c}$$

where $\boldsymbol{\alpha}(t)$ represents the set of local processing indicators $\alpha_i^j(t)$, $\boldsymbol{\beta}(t)$ the set of one-step offloading indicators $\beta_{i,a}^j(t)$, $\boldsymbol{\eta}(t)$ the set of two-step offloading indicators $\eta_{i,a^A\to a^{NA}}^j(t)$, $\mathbf{C}(t)$ the set of computational resource allocation decisions $C_{i,a}^j(t)$, and $\mathbf{K}(t)$ the set of frequency resource allocation decisions $K_i^j(t)$. Constraint (6.18a) ensures that, at any time, the combined allocated computational resources do not exceed the total available resource C_a. Constraint (6.18b) ensures that the total allocated frequency resources stay within the limit K.

However, Problem $\mathcal{P}$ is very challenging to solve due to the following reasons.

(C1) $\mathcal{P}$ is a non-convex NP-hard mixed-integer nonlinear programming (MINLP) dynamical optimization problem with both discrete and continuous variables. It involves sequential decision-making, where previous timeslot decisions $\{\boldsymbol{\alpha}(t-1), \boldsymbol{\beta}(t-1), \boldsymbol{\eta}(t-1), \mathbf{C}(t-1), \mathbf{K}(t-1)\}$ affect current decisions $\{\boldsymbol{\alpha}(t), \boldsymbol{\beta}(t), \boldsymbol{\eta}(t), \mathbf{C}(t), \mathbf{K}(t)\}$. Due to this interdependency, the conventional static optimization-based approaches, such as BnB and generic algorithm (GA), cannot solve this optimization problem.

(C2) DRL offers a potential solution to sequential decision-making problems by directly mapping the SAGIN states (tasks and SAGIN segments) to task offloading and resource allocation decisions. However, traditional DRL is struggling to solve $\mathcal{P}$ for three main reasons.

1) It utilizes multi-layer perceptron (MLP) to extract state features, which cannot effectively capture the topological structure of both the SAGIN segment and tasks, as MLP treats different tasks and SAGIN segments independently.
2) The dimensions of all decision variables $\{\boldsymbol{\alpha}(t), \boldsymbol{\beta}(t), \boldsymbol{\eta}(t), \mathbf{C}(t), \mathbf{K}(t)\}$ vary with U, L, and N. DRL is trained under fixed $\{U, L, N\}$ settings and requires retaining if these parameters change. This retraining process is time-consuming and computationally complex.
3) Typically, DRL output actions are either discrete or continuous; however, the decisions in $\mathcal{P}$ contain both discrete $\{\boldsymbol{\alpha}(t), \boldsymbol{\beta}(t), \boldsymbol{\eta}(t)\}$ and continuous $\mathbf{C}(t), \mathbf{K}(t)$ variables.

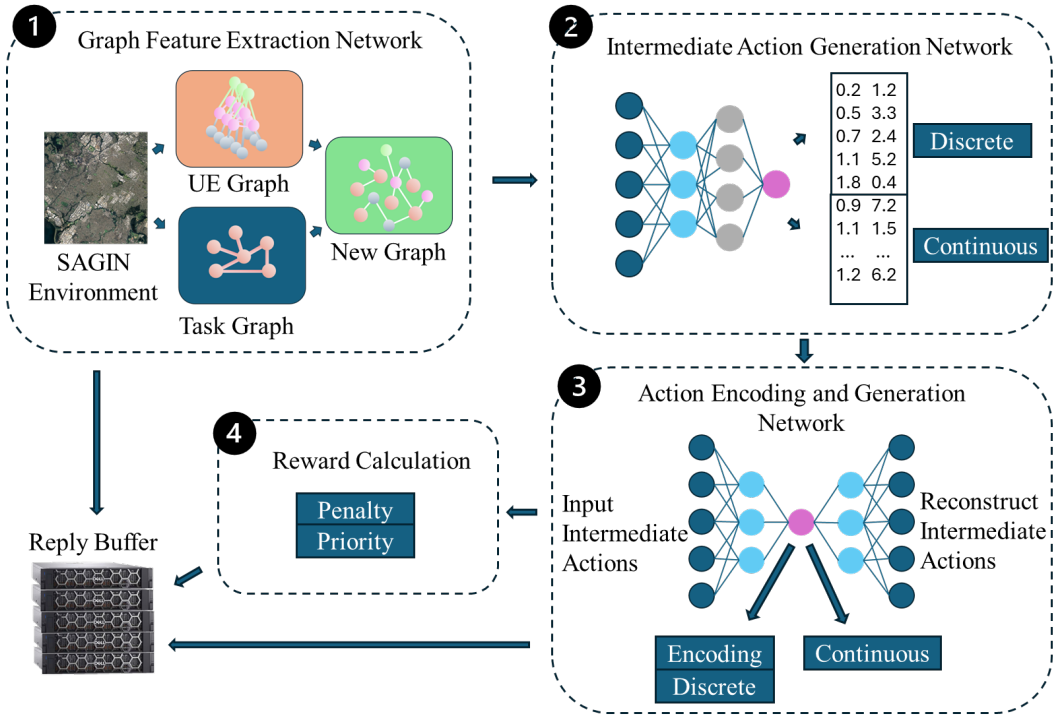


FIGURE 6.3. The architecture of the proposed GF-DRL.

6.2 Proposed GF-DRL Framework

In this section, we present a detailed description of each component within GF-DRL. The proposed GF-DRL architecture, shown in Fig. 6.3, consists of three main components (Part 1, Part 2, and Part 3), an auxiliary component (Part 4), and a reply buffer for storing all state-action pairs and rewards.

6.2.1 Graph Feature Extraction Network (GFEN)

The network structure of GFEN is shown in Fig. 6.4. GFEN aims to efficiently extract features from topological structured tasks and SAGIN environment states. GFEN consists of three steps: initial feature extraction (IFE), edge construction (EC), and feature merging (FM). Each step is detailed below.

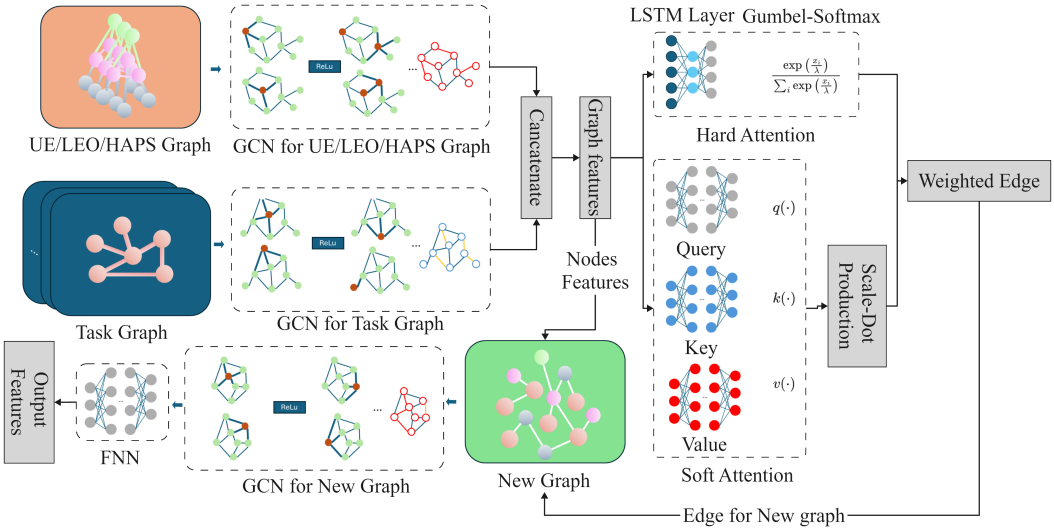


FIGURE 6.4. System structure for GFEN.

6.2.1.1 Initial Feature Extraction

The first step in GFEN is to extract and concatenate features from $\mathcal{G}$ and $\mathcal{D}$ to construct a new graph. Task graph $\mathcal{D}$ consists of the distinct QoS requirements of each subtask, and $\mathcal{G}$ reflects current available resources in SAGIN. In the IFE step, we utilize GCN [109] to extract features from graph $\mathcal{G}$ and $\mathcal{D}$. Denoting a GCN layer as g_c, the IFE process can be represented as

$$\begin{aligned} \mathbf{H}_{\mathcal{G}} &= g_c(\mathbf{I}_{\mathcal{G}}, \mathbf{E}_{\mathcal{G}}) \\ \mathbf{H}_{\mathcal{D}} &= g_c(\mathbf{I}_{\mathcal{D}}, \mathbf{E}_{\mathcal{D}}) \\ \mathbf{H}_{\text{IFE}} &= \text{CON}(\mathbf{H}_{\mathcal{G}}, \mathbf{H}_{\mathcal{D}}), \end{aligned} \tag{6.19}$$

where $\mathbf{H}_{\mathcal{G}} \in \mathbb{R}^{(U+N+L)\times F}$ and $\mathbf{H}_{\mathcal{D}} \in \mathbb{R}^{IU\times F}$ represents the output feature for graph $\mathcal{G}$ and graph $\mathcal{D}$, respectively, $\mathbf{I}_{\mathcal{G}} \in \mathbb{R}^{(U+N+L)\times 2}$ and $\mathbf{I}_{\mathcal{D}} \in \mathbb{R}^{IU\times 3}$ the features for all nodes in graph $\mathcal{G}$ and graph $\mathcal{D}$, respectively, $\mathbf{E}_{\mathcal{G}} \in \mathbb{R}^{2\times(U+L+N)}$, and $\mathbf{E}_{\mathcal{D}} \in \mathbb{R}^{2\times IU}$ the edge index for graph $\mathcal{G}$ and graph $\mathcal{D}$, respectively, CON concatenate operation, and $\mathbf{H}_{\text{IFE}} \in \mathbb{R}^{(IU+U+N+L)\times F}$ the final output of IFE.

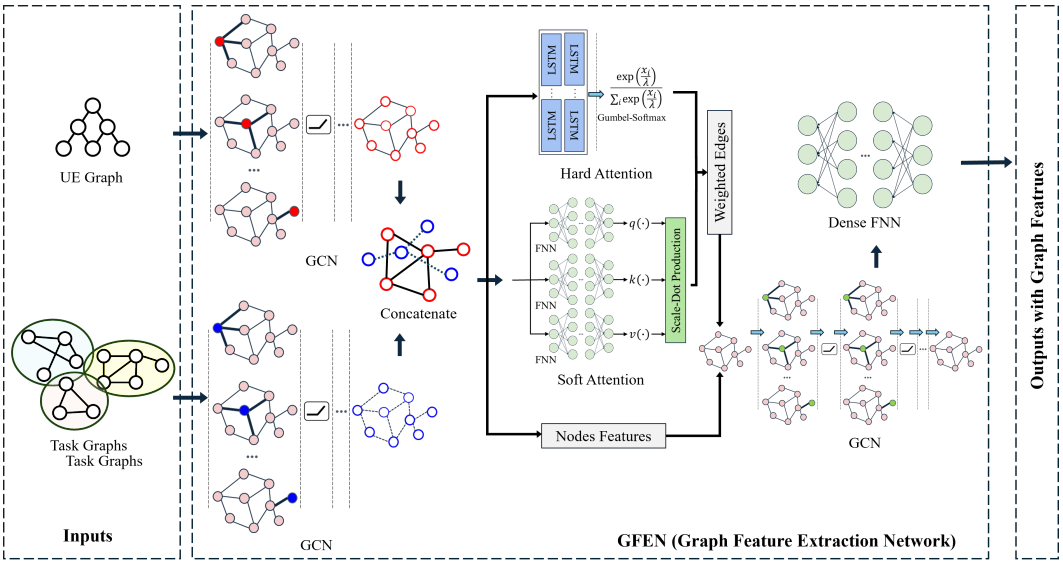


FIGURE 6.5. Graph merging process for GFEN.

6.2.1.2 Edge Construction

The EC step consists of two parts: hard attention and soft attention. Hard attention generates the edges between nodes, while soft attention assigns weights to each edge. As long-term dependency is crucial for the sequential decision-making problem $\mathcal{P}$, a long short-term memory (LSTM) network is used in the hard attention mechanism. The final output of the LSTM network can be represented as $\mathbf{C}_{\text{LSTM}} \in \mathbb{R}^{(IU+U+N+L)\times F_l}$. This output is then passed to Gumbel softmax, which uses the Gumbel-max trick for efficient sampling from a categorical distribution with class probability p_c, where c represents different classes.

The soft attention part is constructed by the scaled dot-production attention. Specifically, key $\mathbf{K}$, query $\mathbf{Q}$, and value $\mathbf{V}$ are generated from the fully connected neural network (FNN) based on $\mathbf{C}_{\text{LSTM}}$. According to [110], the output from the soft attention part is

$$\mathbf{H}_{\text{SA}} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{\sigma}}\right)\mathbf{V}, \tag{6.20}$$

where $\frac{1}{\sqrt{\sigma}}$ is the scaling factor.

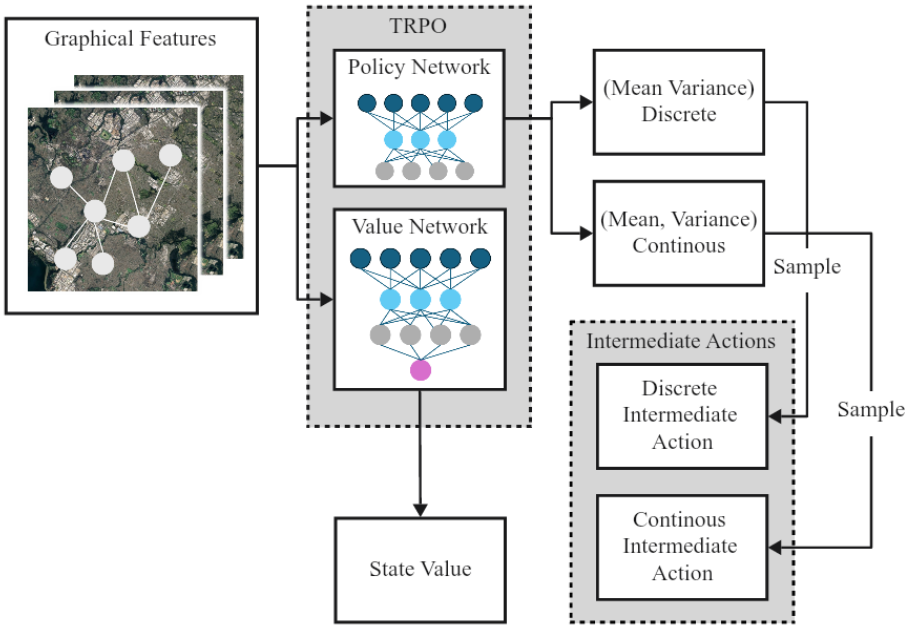


FIGURE 6.6. System structure for IAGN.

6.2.1.3 Feature Merging

In the FM step, the weighted edge $\mathbf{A}_w$ and the extracted features $\mathbf{H}_{\text{IFE}}$ are forwarded to the GCN followed by FNN to generate extracted graph features for other parts of GF-DRL. Specifically, the output from FM $\mathbf{H}_{\text{FM}}$ can be calculated as

$$\begin{aligned} \mathbf{H}_1 &= g(\mathbf{H}_{\text{IFE}}, \mathbf{A}_w) \\ \mathbf{H}_{\text{FM}} &= \mathbf{W}_1\mathbf{H}_1 + \mathbf{b}_1, \end{aligned} \tag{6.21}$$

where $\{\mathbf{W}_1, \mathbf{b}_1\}$ are the learnable parameters for FNN. The full process is illustrated in Figure. 6.5

6.2.2 Intermediate Action Generation Network (IAGN)

The extracted feature $\mathbf{H}_{FM}$ is forwarded into the IAGN to generate intermediate action. Fig. 6.6 shows the IAGN architecture. It can be seen that IAGN generates two outputs: intermediate continuous action and discrete action, sampled from the corresponding Gaussian distribution. In this paper, we incorporate a modified version of TRPO to construct IAGN.

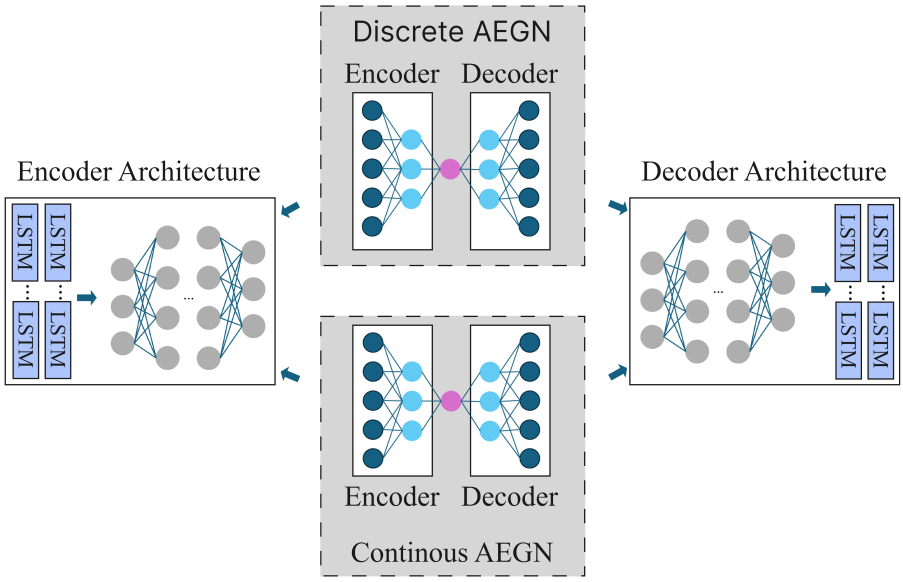


FIGURE 6.7. System structure for AEGN.

The intermediate action generation problem can be represented as

$$\begin{aligned} &\underset{\boldsymbol{\theta}^p}{\text{maximize}} \;\; \mathrm{g}^T \left(\boldsymbol{\theta}^p_{\text{new}} - \boldsymbol{\theta}^p_{\text{old}}\right) \\ &\text{subject to} \;\; \frac{1}{2}\left(\boldsymbol{\theta}^p_{\text{new}} - \boldsymbol{\theta}^p_{\text{old}}\right)^T \mathbf{H}_{\theta^p} \left(\boldsymbol{\theta}^p_{\text{new}} - \boldsymbol{\theta}^p_{\text{old}}\right) \leq \delta, \end{aligned} \tag{6.22}$$

where $\boldsymbol{\theta}^p$ represents the parameter for the policy network, g the gradient, and $\mathbf{H}_{\theta^p}$ the Hessian matrix. Lagrangian duality is utilized to solve (6.22), and the rule for updating the policy network parameters is

$$\boldsymbol{\theta}^p_{\text{new}} = \boldsymbol{\theta}^p_{\text{old}} + \kappa \sqrt{\frac{2\delta}{\mathbf{g}^T \mathbf{H}^{-1}_{\theta^p} \mathbf{g}}} \mathbf{H}^{-1}_{\theta^p} \mathbf{g}, \tag{6.23}$$

where κ represents the step size. The update rule for the value network can be formulated as

$$\boldsymbol{\theta}^v_{\text{new}} = \arg\min_{\boldsymbol{\theta}^v} \frac{1}{T} \sum_{t=0}^{T} \left(V_\pi(\mathbf{H}_{\text{FM}}) - \zeta(\mathbf{H}_{\text{FM}})\right)^2, \tag{6.24}$$

where $\boldsymbol{\theta}^v$ represents the parameters for the value network.

6.2.3 Action Encoding and Generation Network (AEGN)

$$R(t) = \begin{cases} \mu_1 \sum_u \sum_i \gamma_i^u - \mu_2(\sum_{l/n} C^{l/n} - \sum_u \sum_i \sum_j C_{(i,j)}^{(l/n,u)}) - \mu_3(K - \sum_u \sum_i \sum_j K_{(i,j)}^u) - b, & i \in N_i < J \\ \mu_1 \sum_u \sum_i \gamma_i^u - \mu_2(\sum_{l/n} C^{l/n} - \sum_u \sum_i \sum_j C_{(i,j)}^{(l/n,u)}) - \mu_3(K - \sum_u \sum_i \sum_j K_{(i,j)}^u), & i \in N_i = J. \end{cases} \tag{6.25}$$

To address the large action space problem, we propose an encoding scheme that reduces the size of the discrete action space to $2^{3+\log_2(\max(N,L))}$ by eliminating invalid actions. The encoding scheme consists of two segments: the basic offloading decision (BOD) and the LEO/HAPS Index (LHI). The BOD represents the set of potential offloading decisions and the LHI represents which specific LEO or HAPS will receive the task. Based on this encoding scheme, we propose an online real-time low-complexity action generation network referred to as AEGN. Fig. 6.7 shows the architecture of the proposed AEGN. AEGN consists of two parts: The continuous AEGN and the discrete AEGN. Both continuous and discrete AEGN are based on autoencoder structure, where the encoder receives $(\mathbf{a}_c, \mathbf{a}_d)$ from the IAGN and reconstruct $(\mathbf{a}_c, \mathbf{a}_d)$ in the decoder. Specifically, the continuous AEGN encoder receives $\mathbf{a}_c$ from IAGN, and its decoder reconstructs $\mathbf{a}_c$. The output of the continuous encoder represents the resource allocation decisions for all subtasks. Similarly, the discrete AEGN encoder receives $\mathbf{a}_d$, and its decoder reconstructs $\mathbf{a}_d$, where the output of the discrete encoder represents the offloading decisions for all subtasks.

6.2.4 Reward Calculation

We first define an indicator for the completion of subtask j of task i as

$$\delta_i^j = \begin{cases} 1 & \text{if completed within task delay constraint } \iota_i \\ 0 & \text{otherwise.} \end{cases} \tag{6.26}$$

Accordingly, the total number of completed subtasks for task i is $\sum_{j=1}^{J} \delta_i^j$. We define a priority value for task i as

$$\gamma_i = \lambda_1 \tilde{o}_i \tilde{s}_i + \lambda_2 \tilde{\iota}_i + \lambda_3 \sum_{j=1}^{J} \delta_i^j + \lambda_4 \Delta_i, \tag{6.27}$$

where λ_1, λ_2, λ_3, and λ_4 are hyperparameters, $\tilde{o}_i\tilde{s}_i$ the normalized required CPU cycles, and $\tilde{\iota}_i$ the normalized latency requirement. Accordingly, we formulate the reward as (5.40), where (μ_1, μ_2, μ_3) represents the scaling hyperparameters.

6.3 Performance Improvement and Validation

6.3.1 Reincarnating GF-DRL

In this subsection, we propose a reincarnating GF-DRL (RGF-DRL) to solve the high re-training complexity problem of GF-DRL. Specifically, instead of training GF-DRL from the

Algorithm 3 Reincarnating Two-layer Graphic Deep Reinforcement Learning

1: Perform the initial training with a specific set of UE/LEO/HAPS configurations and a defined number of subtasks in the SAGIN network ($\mathcal{E}_{init}$).
2: Update the parameters for policy network $\boldsymbol{\theta}^p_{init}$ and value network $\boldsymbol{\theta}^v_{init}$.
3: **for** I iterations **do**
4: Change the configurations of UE/LEO/HAPS and the number of subtasks in the SAGIN network ($\mathcal{E}_i$).
5: Map partial of $(\boldsymbol{\theta}^p_{init}, \boldsymbol{\theta}^v_{init})$ obtained from $\mathcal{E}_{init}$ to the corresponding $(\boldsymbol{\theta}^p_i, \boldsymbol{\theta}^v_i)$ for $\mathcal{E}_i$.
6: **for** T training step **do**
7: Interact with the environment and obtain the extracted features from GFEN
8: Take actions according to the output from AEGN
9: Collect state-action-next state transactions in the replay buffer for training.
10: Compute the cumulative reward $R(t)$
11: Update the remaining parameters for the policy network and the value network.
12: **end for**
13: **end for**

beginning, we leverage the previously trained policy and value network parameters as the starting point for new ones. The algorithm for the proposed R-GF-DRL is shown in Algorithm 3. To simplify notations, we denote (p_n, p_o) as new and old policy network, (v_n, v_o) as new and old value network, $(\boldsymbol{\theta}^i_{p_n}, \boldsymbol{\theta}^i_{p_o})$ as i-th layer parameters for the new and old policy network, and $(\boldsymbol{\theta}^i_{v_n}, \boldsymbol{\theta}^i_{v_o})$ as i-th layer parameters for the new and old value network. Here, we define the mapping from $(\boldsymbol{\theta}^i_{p_o}, \boldsymbol{\theta}^i_{v_o})$ to $(\boldsymbol{\theta}^i_{p_n}, \boldsymbol{\theta}^i_{v_n})$ as

$$\begin{aligned} \boldsymbol{\theta}^{i}_{p_n} &= \boldsymbol{\theta}^{i}_{p_o} + \mathbf{b}_p \quad i \in \{D(i_n) = D(i_o)\} \\ \boldsymbol{\theta}^{i}_{v_n} &= \boldsymbol{\theta}^{i}_{v_o} + \mathbf{b}_v \quad i \in \{D(i_n) = D(i_o)\}, \end{aligned} \tag{6.28}$$

where $D(i_n)$ and $D(i_o)$ represent the dimension of i-th layer parameters in the old and new network, respectively, and $\mathbf{b}_p$ and $\mathbf{b}_v$ represent the bias for policy and value network, respectively. The condition $i \in \{D(i_n) = D(i_o)\}$ means that only parameters with the same dimension in the same layer can be mapped from the old network to the new network. The new parameters obtained from the old parameters are excluded from the update process, and the gradient for these parameters is not calculated.

In this paper, most of the parameters in GFEN are not affected by the number of UE/HAPS/LEO due to the implementation of GCN, which can naturally handle graphs with different numbers of nodes. The layers with different dimensions for the new and old network are: the input layer of LSTM, the the soft attention FNN input/output layer, and the input layer of the FNN after GCN. IAGN is not affected by the number of UE/HAPS/LEO as it only generates intermediate actions. For AEGN, as the number of UE/HAPS/LEO affects the fundamental formation of the encoding scheme, it needs to be retrained, and information from previous training cannot be leveraged. It should be noted that AEGN has an autoencoder structure, and the retraining complexity is very low. Therefore, RGF-DRL can greatly reduce the retraining complexity of GF-DRL.

6.3.2 Performance Validation of GF-DRL over DRL

In this subsection, we mathematically demonstrate the superior performance of GF-DRL over DRL in solving $\mathcal{P}$. We begin by introducing lemmas, definitions, and conclusions that establish GNN's superiority over MLP in solving graph optimization problems. Then, we prove our problem is a graph optimization problem, and that GF-DRL outperforms other DRL-based approaches. Finally, we prove that merging two graphs $\mathcal{D}$ and $\mathcal{G}$ together achieves lower computational complexity than treating them separately.

6.3.2.1 GNN outperforms MLP

We define a family of algorithms called distributed message passing (DMP) algorithms. DMP is a classic set of iterative algorithms to solve graph optimization problems. In each iteration, every node sends messages, receives messages from its neighbors, and updates its state based on the received messages. According to the theorem in [111], this powerful algorithm family can solve any graph optimization problem.

Lemma 1: For any graph optimization problem $\mathcal{P}_1$, a DMP algorithm exists that can solve it.

Proof: The proof is available in [112]. Specifically, a DMP step can implement any permutation-invariant function, and the graph optimization problem is permutation invariant. Therefore, DMP can solve any graph optimization problem. With the definition of DMP, we can now prove that GNN is a special case of DMP. The relationship between GNN and DMN is listed in the following Lemma.

Lemma 2: Let GNN(T) be a GNN with T layers and DMP(T) be a DMP with T iterations. There exists an algorithm in DMP(T) that solves the same problems as GNN(T). Furthermore, for any DMP(T), there is an algorithm in GNN(T) that solves the same problems as DMP(T).

Proof: This Lemma is proved by Theorem 2 in [111], Theorem 7 in [112], and Theorem 3.1 in [113].

Conventional DMP algorithms are optimization-based with predefined parameters for message encoding, aggregation, and update functions. In contrast, GNN is a DL method, where the parameters for each function are learned from data. It has been shown in [112] that GNN can significantly reduce the iterations required by optimization-based DMP. By combining Lemma 1 and Lemma 2, we reach the following conclusion.

Conclusion 1: There exists a corresponding GNN(T) that can solve each graph optimization problem.

Conclusion 1 shows the capability of GCN to address graph optimization problems. However, as MLPs are universal approximators, they can also approximate DMP algorithms. In the following section, we will directly compare the performance of GCN and MLP.

To measure the performance of each system, we adopt probably approximately correct learning (PAC-learning) [114], a standardized method for quantifying generalization errors with limited training samples. This allows us to theoretically evaluate and compare the generalization performance between two DL-based systems. Specifically, PAC-learning is defined as follows.

Definition 1 ([114]): Denote the error parameter as $\epsilon > 0$, the failure probability $\delta \in (0, 1)$, and the N i.i.d samples draw from distribution $\mathcal{D}$ as $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{N}$ with $\mathbf{y}_i = g(\mathbf{x}_i)$. We define g as (N, ϵ, δ)-learnable with algorithm $\mathcal{A}$ if $\mathbb{P}_{\mathbf{x}\ \mathcal{D}}[||f(\mathbf{x}) - g(\mathbf{x})|| \leq \epsilon] \geq 1 - \delta$. Here, f is a function generated by the learning algorithm $\mathcal{A}$. We also denote the sample complexity as $S_{\mathcal{A}}(g, \epsilon, \delta)$ which represents the minimum number of N that guarantees g is (N, ϵ, δ)-learnable with algorithm $\mathcal{A}$.

In the field of DL, g is the oracle algorithm that can output the optimal solution for the graph optimization problem, $\mathcal{A}$ is the gradient descent algorithm to train the neural network, and f is the neural network. With Definition 1, we can derive that f with lower sample complexity (smaller number of N) is closer to g with high probability. Note that in order to implement PAC-learning, there must be a f that can approximate g, which is guaranteed by the universal approximation property of DL-based methods. To perform tractable comparisons between DL-based methods, we incorporate an algorithmic alignment (AA) framework. Specifically, we have the following definition for AA.

Definition 2: Denote the set of neural networks as $\mathcal{Q} = \{\mathcal{Q}_1, \cdots, \mathcal{Q}_I\}$, and a group of functions $(f_1, \cdots, f_I)$ that generates g for $\mathcal{Q}$. The neural network $\mathcal{Q}$ is (N, ϵ, δ)-algorithmic align with the oracle function g if it satisfies:

(a) $(f_1, \cdots, f_I)$ generate g.

(b) $\mathcal{A}_i$ exists that satisfies $n \cdot \max_i S_{\mathcal{A}_i}(f_i, \epsilon, \delta)$.

Definition 2 indicates that directly learning g in neural networks is computationally prohibitive. Instead, the complex task of learning g can be divided into server subproblems by learning $(f_1, \cdots, f_I)$. Once the agreement between the algorithm and the neural network is established using Definition 2, we can analyze the sample complexity and generalization error. With the AA and PAC-learning framework, we can drive Lemma 3.

Lemma 3: In the commonly used GNN architecture from [111], GNNs align better with DMP algorithms than MLPs.

Proof: The proof can be found in [111]. By applying lemma 3 with the generalization bound of neural networks from [115], we reach the following conclusion.

Conclusion 2: Given Lemma 3, we have the following results

(a) *For a given node number $|\mathcal{V}|$, MLP requires $\mathcal{O}(|\mathcal{V}|)$ more samples than GNN.*
(b) *If GNN achieves an error of ϵ and $\delta > \Omega(\exp -\epsilon)$, the MLP architecture achieves an error of $\mathcal{O}(|\mathcal{V}|\epsilon)$*

Conclusion 2 suggests that GNNs outperform MLPs in terms of sample complexity and generalization error for solving graph optimization problems, with performance improving as the number of nodes increases.

6.3.2.2 GF-DRL outperforms other DRL-based approaches

We first define the graph optimization problem as

$$\begin{aligned} \mathcal{P}_1 : &\min_{\mathbf{X}} f(\mathbf{X}, \mathbf{N}, \mathbf{A}) \\ &\text{subject to } C(\mathbf{X}, \mathbf{N}, \mathbf{A}) \leq 0, \end{aligned} \tag{6.29}$$

where $\mathbf{N} \in \mathbb{R}^{|\mathcal{V}| \times d_1}$ represents node features, $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times n}$ the optimization variable on all nodes with $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_{|\mathcal{V}|}]$ with $\mathbf{x}_v$ denoting the optimization variable on the v-th node, and $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}| \times d_2}$ represents the adjacency feature tensor which is

$$A_{(i,j,:)} = \begin{cases} 0, & \text{if } \{i,j\} \notin \mathcal{E} \\ f(e_{(i,j)}) & \text{otherwise,} \end{cases} \tag{6.30}$$

where $e_{(i,j)}$ represents the edge between nodes i and j, $\mathcal{E}$ represents the collection of edges, and $f(e_{(i,j)})$ represents the edge feature of $e_{(i,j)}$. Accordingly, we can have the following theorem,

Theorem 1: After the discrete action encoding process, problem $\mathcal{P}$ becomes a graph optimization problem $\mathcal{P}_1$.

Proof: The optimization variables for $\mathcal{P}$ can be divided into two parts, the discrete offloading decisions ($\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\eta}$) and the continuous resource allocation decisions ($\mathbf{C}$, $\mathbf{K}$). After the discrete action encoding process, $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\eta})$ can be represented by a vector with the dimensions equal to the sum of BOD length and LHI length, denoted by l_e. We denote by $\mathbf{a}_d \in \mathbb{R}^{|\mathcal{V}| \times l_e}$ the discrete action after encoding. Accordingly, the optimization variables for $\mathcal{P}$ can be formulated as

$$\begin{aligned} &\mathbf{X} = \{\mathbf{a}_d, \mathbf{C}, \mathbf{K}\} \in \mathbb{R}^{\mathcal{V} \times (l_e+2)}, \\ &\text{With}: \mathbf{C} \in \mathbb{R}^{\mathcal{V} \times 1} \\ &\mathbf{K} \in \mathbb{R}^{\mathcal{V} \times 1} \end{aligned} \tag{6.31}$$

According to (6.27), γ_i is the function of $(\mathbf{X}, \mathbf{N}, \mathbf{A})$, therefore, problem $\mathcal{P}$ can be transferred into

$$\mathcal{P}: \max_{\boldsymbol{X}} \sum_i f(\mathbf{X}, \mathbf{N}, \mathbf{A}) \tag{6.32}$$

$$Q_1(\mathbf{X}) - C^{l/n} \leq 0, \tag{6.33a}$$

$$Q_2(\mathbf{X}) - K \leq 0, \tag{6.33b}$$

where Q_1 maps the $\mathbf{X}$ to $\sum_i \sum_j C^{l/n}_{(i,j)}$, Q_2 maps $\mathbf{X}$ to $\sum_i \sum_j K^{l/n}_{(i,j)}$. Comparing (6.33) with (6.29), it is obvious that $\mathcal{P} \in \mathcal{P}_1$. With *Lemma 1* and *Conclusion 2*, we can derive Conclusion 3 as

Conclusion 3: GF-DRL outperforms other DRL-based approaches in solving $\mathcal{P}$.

GF-DRL incorporates GNN, while other DRL-based approaches use only MLP. As GNN is superior to MLP in solving graph optimization problems like P_1, GF-DRL achieves better performance.

6.3.2.3 Merging graphs

Merging $\mathcal{D}$ and $\mathcal{G}$ can further explore both inter- and intra-correlations of the two graphs, outperforming separate graph processing in capturing features, as shown in simulations. Additionally, we demonstrate that merging $\mathcal{D}$ and $\mathcal{G}$ reduces computational complexity. Specifically, we present the following theorem.

Theorem 2: Denote the merged graph by $\mathcal{J} = \{\mathcal{W}, \mathcal{Z}, f_w, f_z\}$ where $\mathcal{W}$ represents nodes, $\mathcal{Z}$ represents edges, f_w maps each node to its feature $w \to \mathbb{R}^{d_w}$ with d_m being the feature dimension of node w, and f_z maps each edge to its feature $z \to {}_F$ with d_z denoting the feature dimension of edge z. Merging $\mathcal{D}$ and $\mathcal{G}$ achieves a lower computational complexity than treating graphs separately if $\mathcal{Z} < \mathcal{Q} + \mathcal{E}$.

Proof: We first define the computational complexity of GCN as

$$C_{\text{GCN}} = \sum_i F_i^2|\mathcal{E}| + F_i|\mathcal{V}|, \tag{6.34}$$

where F_i represents the node features for the i-th layer of GCN, $|\mathcal{E}|$ represents the number of edges, and $|\mathcal{V}|$ represents the number of nodes. For simplicity, we only consider GCN with two layers: input and output. Accordingly, to process graph $\mathcal{G}$, the computational complexity is

$$C_{\text{GCN}}(\mathcal{G}) = d_v^2|\mathcal{E}| + d_v|\mathcal{V}| + d_w^2|\mathcal{E}| + d_w|\mathcal{V}|. \tag{6.35}$$

To process graph $\mathcal{D}$, the computational complexity is

$$C_{\text{GCN}}(\mathcal{D}) = d_m^2|\mathcal{Q}| + d_m|\mathcal{M}| + d_w^2|\mathcal{Q}| + d_w|\mathcal{M}|. \tag{6.36}$$

Accordingly, the dimension for extracted features is $\mathbb{R}^{d_w \times (|\mathcal{V}|+|\mathcal{M}|)}$, and computational complexity for processing $\mathcal{D}$ and $\mathcal{G}$ separately is $C_{\text{GCN}} + C_{\text{GCN}}$. The computational complexity of

processing $\mathcal{J}$ is

$$C_{\text{GCN}}(\mathcal{J}) = d_v^2|\mathcal{E}| + d_v|\mathcal{V}| + d_m^2|\mathcal{Q}| + d_m|\mathcal{M}| + d_w^2|\mathcal{Z}| + d_w|\mathcal{W}|, \tag{6.37}$$

Thus, if $\mathcal{Z} < \mathcal{Q} + \mathcal{E}$, the achieved computational complexity of merging the two graphs is less than that of processing them separately.

6.4 Simulation Results and Analysis

In this section, we first evaluate the performance of our proposed AEGN in terms of reconstruction error. Note that both our proposed GF-DRL and baseline methods need to incorporate AEGN to generate discrete and continuous actions simultaneously. Next, we demonstrate that GF-DRL significantly outperforms widely used DRL-based baseline methods in terms of task completion ratio and return during the inference stage. Finally, we show that reincarnating GF-DRL brings significant benefits in terms of convergence speed relative to simply retraining GF-DRL.

6.4.1 Simulation Setup

In this simulation, training was conducted on a computer equipped with an NVIDIA GeForce RTX 3060 GPU, an AMD Ryzen 7 5800X 8-Core Processor, and 32 GB of RAM. The Python 3.8 environment was managed using Anaconda, and all code was developed and executed within the PyCharm IDE. The relevant parameters in the simulation are listed as follows. For the task generation, we have $s_i(t) \in [8000, 10000]$ bits, $v_i(t) \in [5000, 6000]$ bits, $o_i(t) \in [1000, 3000]$, HAPS $C_n \in [50, 80]$, the number of UEs $U = 10$, the number of LEOs $L = 10$, the number of HAPSs $N = 10$, and the number of subtasks per task $J = 30$ unless stated otherwise. Tasks are generated using a uniform distribution. For GF-DRL training, the batch size is 128, the experience horizon is 1024, activation function is *Sigmoid*, and the optimizer is *Adam*. We also have $B = 15$ kHz, $K = 100$, $N_0 = -173$ dBm/Hz, and $G_u^{l/n} = 10$. Other parameters are listed in table 6.1.

Component	Dimension
GCN-Task	(5, 16), (16, 4)
GCN-UE	(2, 16), (16, 4)
FNN-Key	(4, 1)
FNN-Query	(4, 1)
FNN-Value	(4, 1)
LSTM	$(4, U + N + L + U * (J + 2))$
GCN-Out	(4, 1)
FNN-Out	$(U + N + L + U * (J + 2), 64), (64, 32)$

TABLE 6.1. Crucial network components and corresponding dimensions

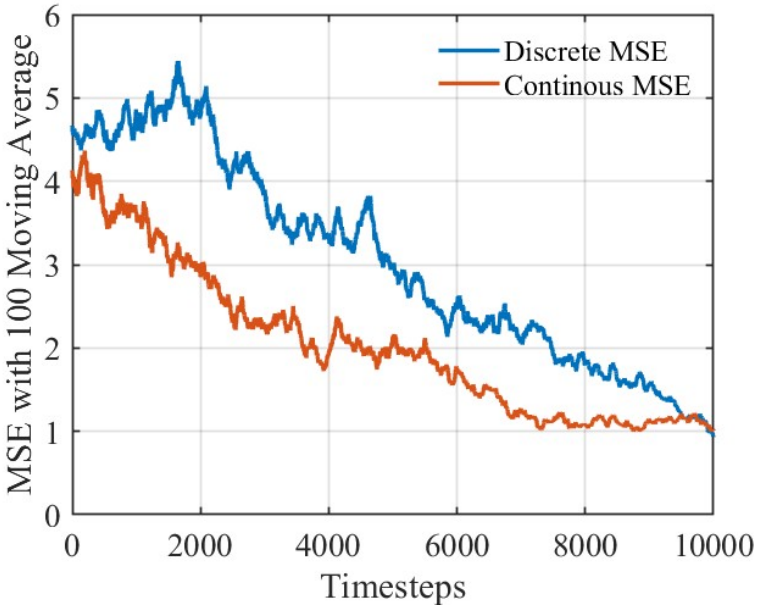


FIGURE 6.8. MSE between the original intermediate actions (IAGN) and reconstructed (AEGN).

6.4.2 Simulation Result

Fig. 6.8 illustrates the MSE performance of discrete and continuous AEGN in reconstructing intermediate actions from the original IAGN. Note that lower MSE indicates better feature capture of intermediate actions. It can be seen that both discrete and continuous AEGN exhibit a significant decrease in MSE as timesteps increase. This clearly indicates that, through the training of AEGN, underlying action patterns can be precisely captured, which facilitates the accurate modeling of intermediate actions. It can also be seen that continuous AEGN consistently outperforms its discrete counterpart except at the end of the training, where their

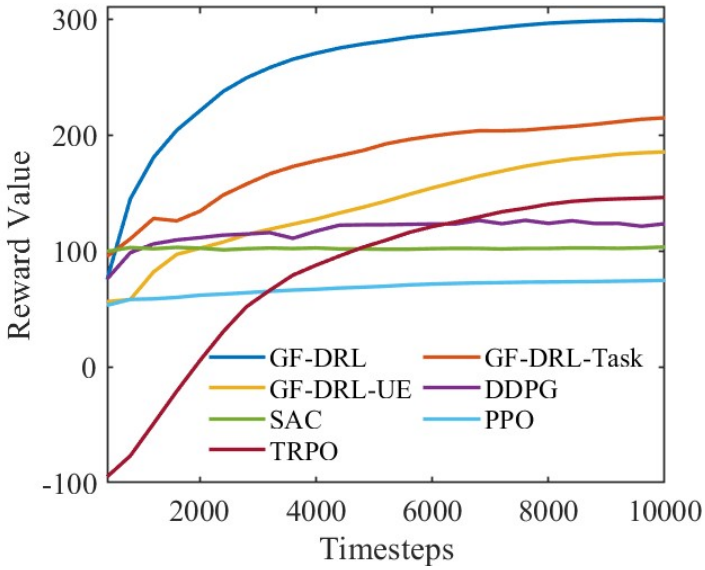
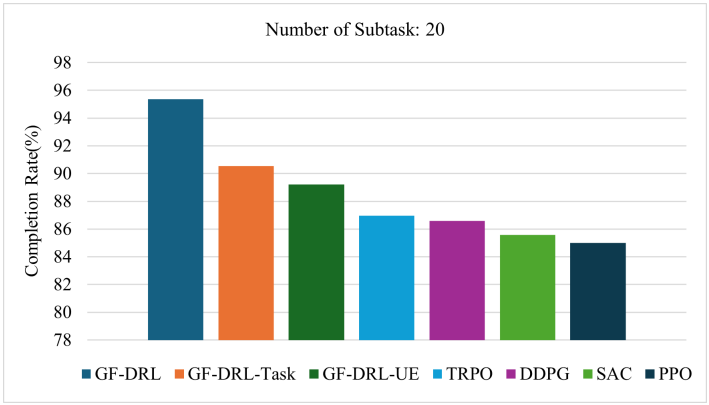


FIGURE 6.9. Training return comparison for GF-DRLs and the baseline methods.

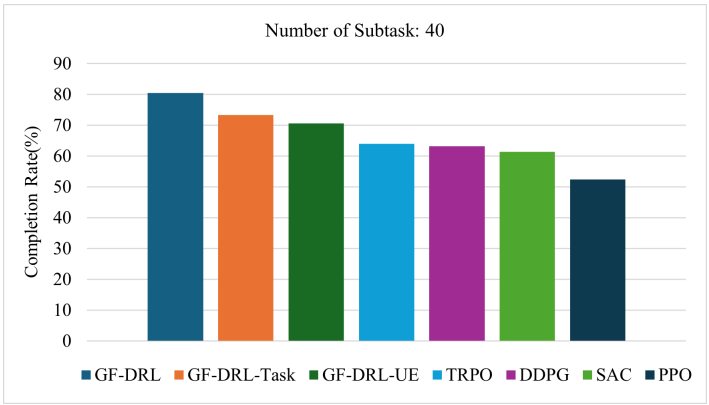
performances converge. This is because the higher-dimensional hidden layers in discrete AEGN require more samples to achieve similar performance levels as continuous AEGN.

Figs. 6.9 to 6.12 compare the performance of GF-DRL with six baseline methods during both training (parameter learning) and inference (fixed parameters). Before delineating key findings, we briefly describe each baseline method:

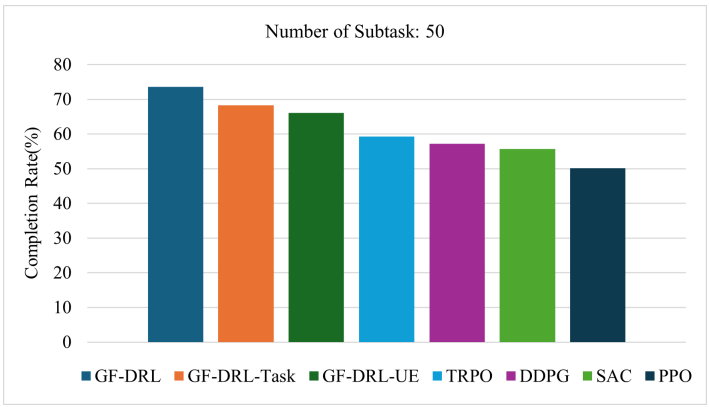
(1) TRPO [91]: An on-policy method featuring a policy network for action generation and a value network for action evaluation. TRPO uniquely guarantees a monotonic improvement of policy.
(2) PPO [95]: A simplified version of TRPO with reduced computational complexity achieved by clipping its objective function.
(3) SAC [96]: An off-policy DRL method that uses soft regularization to balance return and entropy, enabling a broader exploration of actions.
(4) DDPG [97]: An off-policy DRL method that learns a Q-function using off-policy data and the Bellman equation. It optimizes policies directly in continuous domains, making it effective in environments with continuous action spaces.



(A) Task completion rate for $J = 20$.



(B) Task completion rate for $J = 40$.



(C) Task completion rate for $J = 50$.

FIGURE 6.10. Task completion rate comparison for GF-DRLs and the baseline methods for different subtask numbers.

(5) GF-DRL-UE: A simplified version of GF-DRL we designed that considers extracting graphical features from $\mathcal{G}$ only.

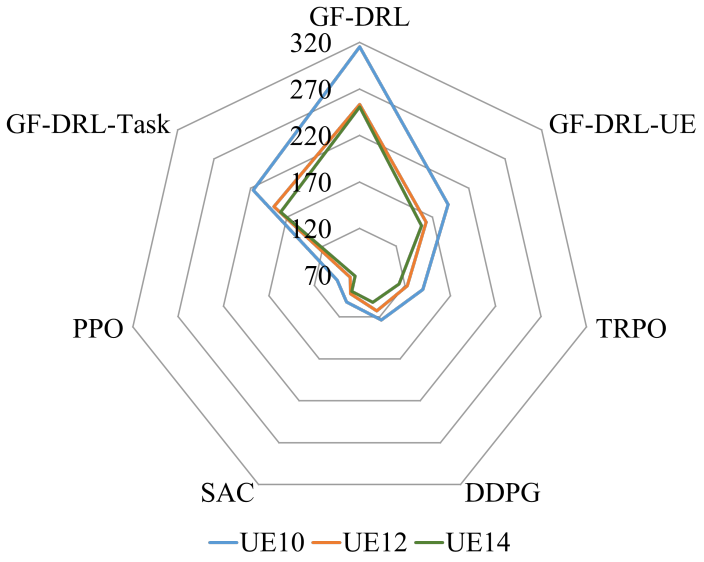


FIGURE 6.11. Inference return comparison for GF-DRLs and the baseline methods.

(6) GF-DRL-Task: A simplified version of GF-DRL we designed that considers extracting graphical features from $\mathcal{D}$ only.

Fig. 6.9 shows the training return comparison between GF-DRL and baseline methods. It can be seen that GF-DRL achieves the highest return among all methods. The return superiority of GF-DRL originates from its attention mechanism (through the merge of $\mathcal{G}$ and $\mathcal{D}$) in capturing the intra-dependency and inter-dependency of both graphs. It can also be noticed that GF-DRL-Task consistently achieves higher return than GF-DRL-UE. This suggests that graph $\mathcal{D}$ (with task features) plays a more significant role than graph $\mathcal{G}$ (with UE-LEO-HAPS features) in the decision-making process. As expected, DDPG, PPO, and SAC all perform significantly worse than GF-DRL. In particular, PPO achieves the lowest return of all the methods, due to the clipping of its objective function. It can also be seen that TRPO is significantly inferior to GF-DRL, and this is due to the lack of GFEN in TRPO.

Fig. 6.10 carries out the comparison between GF-DRL and baseline methods in terms of task completion rate for $J = 20$, $J = 40$, and $J = 50$ subtasks per task. It can be seen that, for both cases, GF-DRL achieves the best task completion rate. Specifically, for 40 subtasks per task, GF-DRL performs 16% better than the best baseline method (GF-DRL-Task) and 23%

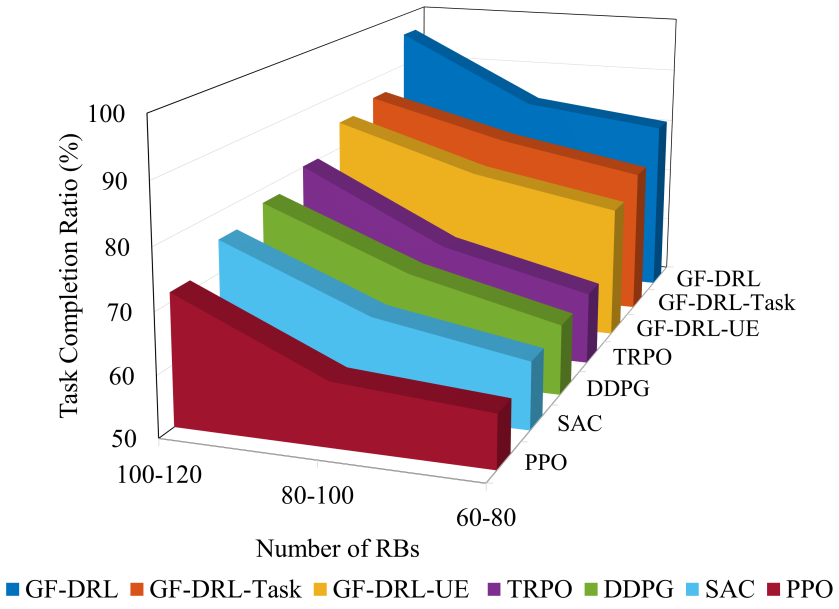


FIGURE 6.12. Task completion rate comparison for GF-DRLs and the baseline methods across varying RBs.

better than the worst baseline method (PPO). Note that all these methods need to be retrained whenever the number of UEs or subtasks per task changes.

Fig. 6.11 shows the comparison of online inference return between GF-DRL and baseline methods under three different numbers of UEs (10, 12, and 14). It is a polygon plot with concentric layers, where the return values increase from the center outwards. It can be seen that, in all three cases, our proposed GF-DRL achieves a significantly higher return than all the six baseline methods. It can also be observed that as the number of UEs increases, the achieved return for all methods decreases. With fixed available resources, more tasks are not able to be completed with the increase in user numbers. This inevitably increases the associated penalty, resulting in a lower return.

Fig. 6.12 presents the comparison of task completion rate between GF-DRL and the baseline methods, all without retraining. In specific, three uniformly distributed ranges of resource blocks (RB) are considered: stringent (60-80), limited (80-100), and moderate (100-120) initial resources, and 10 simulations are conducted for each range. The final task completion

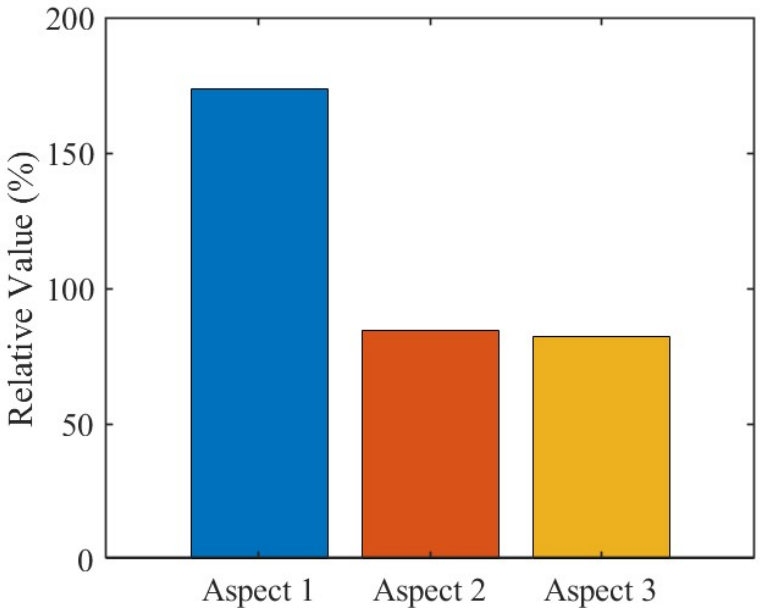


FIGURE 6.13. Performance comparison between RGF-DRL and GF-DRL in terms of convergence speed and task completion ratio.

rate for each case is calculated as the average of these simulations. It can be seen that GF-DRL significantly outperforms all the baseline methods across the whole simulated range. Specifically, it achieves a 25% and 16% improvement over the best (GF-DRL-Task) and worst (PPO) baseline method, respectively. This clearly demonstrates GF-DRL's excellent generalization performance.

Fig. 6.13 carries out the performance comparison between RGF-DRL and GF-DRL from three different aspects: convergence speed (Aspect 1), task completion rate when number of RB changes (Aspect 2), and task completion rate when number of subtasks per task changes (Aspect 3). The purpose is to validate the ability of RGF-DRL in reducing retraining complexity. Note that, originally, Aspect 1 is measured in timesteps while Aspect 2/3 in percentage. For the convenience of comparison, without loss of generality, we set the value of RGF-DRL for each of these three aspects as 100%, and only present the results for GF-DRL. Furthermore, for Aspect 1, convergence is considered to be achieved if the variation between two adjacent timesteps is small. It can be seen that the value of Aspect 1 for GF-DRL is much larger than that of R-GF-DRL (slower convergence), and the value of Aspect 2/3 for

GF-DRL much smaller than that of RGF-DRL (lower task completion rate). Therefore, the performance boost of implementing the reincarnating algorithm in GF-DRL is significant.

6.5 Computational Complexity

The computational complexity for GF-DRL originates from two different processes: training and inference. The former involves parameter update and backpropagation and the latter involves action generation based on current states.

6.5.1 Training Complexity

The training complexity for GF-DRL consists of three parts: GFEN, IAGN, and AEGN. According to [103], the training complexity of GFEN can be calculated as

$$\begin{aligned} C_{\text{GFEN}} = n_{in}n_1 + \sum_{l=2}^{7} n_{l-1}n_l + n_7 n_{out} \\ + \sum_{j=1}^{5} F_j|\mathcal{E}| + F_j|\mathcal{V}|, \end{aligned} \tag{6.38}$$

where n_{in} represents the input size, n_l the number of neurons in the l-th layer of the MLP, F_j the number of node features for the j-th layer of GCN, $|\mathcal{E}|$ the number of edges, and $|\mathcal{V}|$ the number of nodes.

The computational complexity for IAGN can be calculated as [104]

$$C_{\text{IAGN}} = 2\left(n_{in}n_1 + \sum_{l=1}^{3} n_l n_{l+1} + n_3 n_{out}\right). \tag{6.39}$$

The AEGN consists of four layers of MLP and two layers of LSTM. Its computational complexity is

$$\begin{aligned} C_{\text{AEGN}} = n_{in}n_1 + \sum_{l=2}^{3} n_{l-1}n_l + n_3 n_{out} \\ + 2n_{in}n_h(4n_{in} + 4n_h + 3 + n_{out}), \end{aligned} \tag{6.40}$$

Then, the training complexity for GF-DRL can be calculated as

$$C_{\text{GF-DRL}} = C_{\text{GFEN}} + C_{\text{IAGN}} + C_{\text{AEGN}}. \tag{6.41}$$

6.5.2 Inference Complexity

According to [103], the inference complexity can be represented as $O(n_{\text{all}})$, where n_{all} denotes the total number of neurons. In practice, n_{all} is typically small. Consequently, the inference complexity is low and can be ignored for the online deployment of GF-DRL.

6.6 Conclusion

In this chapter, we presented an efficient online task offloading and resource allocation scheme in SAGIN focusing on intra-task features. At the beginning of this chapter, we stated that intra-task features are very important in SAGIN, and little work has focused on this aspect. Specifically, we proposed GF-DRL, where a DRL agent interacts with the SAGIN environment and learns the mapping between offloading/allocation decisions and SAGIN states by maximizing a reward function. The reward function considers communication/computational resource constraints, long-term system performance, offloading limits, and task priorities. The unique characteristics of GF-DRL lie in exploring the topological structures of both task dependencies and SAGIN segments.

Additionally, at the beginning of this chapter, we listed the limitations of current state-of-the-art approaches. In the following, we conclude how GF-DRL mitigates these limitations.

(1) We leverage graph-based representations (network graph and task graph) to capture the complex interactions within SAGIN segments and the dependencies within subtasks in each task, systematically modeling the relationship between graph nodes (tasks/SAGIN segments). Additionally, we address a task offloading and resource allocation problem in SAGIN by formulating it as a time-sequential decision-making problem.

(2) We merge the two graphs into a unified new graph that can explore both inter-dependencies (between two graphs) and intra-dependencies (within each graph). Specifically, we propose the GFEN that utilizes GCN to extract features from both UE and task graphs and two attention mechanisms (hard and soft) to merge the two graphs by generating a new weighted graph based on extracted features. Compared with conventional multilayer perceptron (MLP)-only feature extraction layers, GFEN performs message propagation and graph convolution directly on graph-structured data, eliminating the possible loss of spatial information.

(3) We introduce the reincarnating technique and propose reincarnating GF-DRL, which can be retrained with low complexity by reusing some parameters from previous training rounds, allowing it to adapt to fluctuating numbers of UE/HAPS/LEO. Specifically, instead of training GF-DRL from the beginning, we leverage the previously trained policy and value network parameters as the starting point for new ones.

In conclusion, the work proposed in this chapter focuses on intra-task features, as stated in Chapter 1. Thus, the inter-task features such as timescales and distributions for different services are not considered. In the next chapter, we conclude all three works presented in this thesis and give a brief overview of possible future works.

CHAPTER 7

Conclusion and Future Work

In this chapter, we summarize the contributions and findings of this thesis. Furthermore, we provide an overview of future works.

7.1 Summary of Results and Contributions

This thesis gives a comprehensive analysis of state-of-the-art task offloading/resource allocation approaches. In the first and second chapters, we first present a detailed description of each generation of communication networks and highlight their advancements over the previous generation. We also highlight the importance of resource allocation for each generation and investigate the features of resource allocation for each generation. In the third chapter, we give a detailed literature review for resource allocation in 5G and task offloading/resource allocation in SAGIN. The limitations of state-of-the-art approaches are listed, and the motivations for proposing the three works are presented. In the third, fourth, and fifth chapters, our proposed approaches are presented in detail. Specifically, PW-DRL is proposed in Chapter 4 to solve the resource allocation problem in 5G. GDRL is proposed in Chapter 5 to solve the task offloading/resource allocation problem in SAGIN, focusing on inter-task features, and GF-DRL is proposed in Chapter 6, focusing on intra-task features. In this final chapter, we present a summary of this thesis and highlight possible future works. The goal of this thesis is to be a base for future research in the field of task offloading/resource allocation.

7.2 Future Works

In the previous section, we conclude the main points of this thesis, including the development of communication networks, the features of resource allocation strategies for each generation, and our proposed resource allocation approaches. From the previous description, we can infer that learning-based approaches like DL and DRL are starting to replace conventional heuristic methods and optimization-based approaches. This is because optimization-based approaches cannot handle the rapidly changing network environment, and the decision-making process of these approaches is of high computational complexity, which cannot satisfy the strict latency requirements of current service types. These challenges are caused by the complex interconnection between communication network elements, the increasing number of users, and emerging service types. The key advantage of learning-based approaches is that they can directly generate allocation decisions (task offloading/resource allocation) based on features of the network environments. Thus, the efficiency of extracting features is crucial for learning-based approaches to have good performance.

Filters are the key comportment that performs feature extraction from data. In the field of signal processing and learning-based approaches, filters are the most crucial part, as they directly affect usable information for further processing. Specifically, filters represent the information processing frameworks that highlight important data content (input data) related to the task to be handled. The importance of filters can be inferred from the success of attention mechanisms and ChatGPT. However, conventional filters are designed only for data on Euclidean domains and inevitably perform worse for other irregular data structures. Communication networks have an inherent topological structure, which is described in Section 5, and the graph is the best way to preserve spectral information and complex interconnection between network elements. Thus, conventional filters cannot be applied to communication networks directly without loss of information and feature extraction efficiency.

To mitigate this limitation of conventional filters, a new type of filer referred to as graph filter is proposed for extracting features from topologically structured data. Graph filters naturally incorporate features of conventional filters like linearity, invariance to shift, and

parametric functions of input data. By utilizing spectral graph theory, graph filters take advantage of spectral interpretation. Graph filters also have their own advantages when considering topologically structured data. Firstly, graph filters are equivalent to permutations. This feature is inherent in the nature of the graph, where the graph is naturally equivariant to permutations. Secondly, graph filters can be implemented in a distributed manner. This is because graph filters can capture the spectral correlations directly. Finally, graph filters have strong generalization ability, where they can have multiple generalized forms like node varying and edge varying. Nowadays, it has been widely used in the fields of signal processing, industrial 4.0, IoT, and robotics.

Despite its success in different fields, the implementation of graph filters in task-offloading resource allocation of communication networks is still an open book. In the following, we discuss the possibility of implementing graph filters in this field and list the possible future works for our work presented in Chapters 3, 4, and 5. In Chapter 3, each network element in 5G can be treated as a node, and the interconnection between network elements (BS and UE) can be treated as an edge. The channel status can be treated as edge features, and the formulated graphs change dynamically over time. Graphs can be used to capture spectral information, and graph filters can be designed to assist the feature extraction process of DRL approaches. In Chapter 4, states are divided into static and dynamic states, and the static states are forwarded to GCN for feature extraction. Graph filters can be implemented in GCN to aid the feature extraction process. Additionally, as graph filters can be implemented in a distributed manner, multiagent DRL can be implemented in the network, with each having a specially designed graph filter. Each agent only needs local information for the graph filters to function normally. This can significantly reduce the size of the action space and enhance the system's scalability. In Chapter 5, two graphs are formulated and merged together to form a new graph. Three graph filters can be designed to suit the needs of each graph separately. Specifically, two graph filters can be implemented together with GCN to extract features from the UE graph and network graph, respectively. One graph filter-enhanced GCN can be implemented to extract features from the merged graph. As graphs change dynamically over time, parameters for each graph filter can change accordingly.

Bibliography

[1] Haijun Zhang et al. 'Fronthauling for 5G LTE-U Ultra Dense Cloud Small Cell Networks'. In: *IEEE Wireless Communications* 23.6 (2016), pp. 48–53. DOI: 10.1109/MWC.2016.1600066WC.

[2] Haijun Zhang et al. 'Network Slicing Based 5G and Future Mobile Networks: Mobility, Resource Management, and Challenges'. In: *IEEE Communications Magazine* 55.8 (2017), pp. 138–145. DOI: 10.1109/MCOM.2017.1600940.

[3] Mingjin Gao et al. 'Task Partitioning and Offloading in DNN-Task Enabled Mobile Edge Computing Networks'. In: *IEEE Transactions on Mobile Computing* 22.4 (2023), pp. 2435–2445. DOI: 10.1109/TMC.2021.3114193.

[4] Yaser Azimi et al. 'Energy-Efficient Deep Reinforcement Learning Assisted Resource Allocation for 5G-RAN Slicing'. In: *IEEE Transactions on Vehicular Technology* 71.1 (2022), pp. 856–871. DOI: 10.1109/TVT.2021.3128513.

[5] Cheng-Xiang Wang et al. 'On the Road to 6G: Visions, Requirements, Key Technologies, and Testbeds'. In: *IEEE Communications Surveys & Tutorials* 25.2 (2023), pp. 905–974. DOI: 10.1109/COMST.2023.3249835.

[6] Oltjon Kodheli et al. 'Satellite Communications in the New Space Era: A Survey and Future Challenges'. In: *IEEE Communications Surveys & Tutorials* 23.1 (2021), pp. 70–109. DOI: 10.1109/COMST.2020.3028247.

[7] Jorge Navarro-Ortiz et al. 'A Survey on 5G Usage Scenarios and Traffic Models'. In: *IEEE Communications Surveys & Tutorials* 22.2 (2020), pp. 905–929. DOI: 10.1109/COMST.2020.2971781.

[8] Cheng-Xiang Wang et al. 'A Survey of 5G Channel Measurements and Models'. In: *IEEE Communications Surveys & Tutorials* 20.4 (2018), pp. 3142–3168. DOI: 10.1109/COMST.2018.2862141.

[9] Cutifa Safitri et al. ‘Comprehensive Survey: Quality of Service in Railway Communication Using Information-Centric Networking and Light Fidelity’. In: *IEEE Transactions on Intelligent Transportation Systems* 25.12 (2024), pp. 19218–19251. DOI: 10.1109/TITS.2024.3472698.

[10] Mohammed Chahbar et al. ‘A Comprehensive Survey on the E2E 5G Network Slicing Model’. In: *IEEE Transactions on Network and Service Management* 18.1 (2021), pp. 49–62. DOI: 10.1109/TNSM.2020.3044626.

[11] Xuemin Shen et al. ‘AI-Assisted Network-Slicing Based Next-Generation Wireless Networks’. In: *IEEE Open Journal of Vehicular Technology* 1 (2020), pp. 45–66. DOI: 10.1109/OJVT.2020.2965100.

[12] Lubna Nadeem et al. ‘Integration of D2D, Network Slicing, and MEC in 5G Cellular Networks: Survey and Challenges’. In: *IEEE Access* 9 (2021), pp. 37590–37612. DOI: 10.1109/ACCESS.2021.3063104.

[13] Yulei Wu et al. ‘A Survey of Intelligent Network Slicing Management for Industrial IoT: Integrated Approaches for Smart Transportation, Smart Energy, and Smart Factory’. In: *IEEE Communications Surveys & Tutorials* 24.2 (2022), pp. 1175–1211. DOI: 10.1109/COMST.2022.3158270.

[14] Ibrahim Afolabi et al. ‘Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions’. In: *IEEE Communications Surveys & Tutorials* 20.3 (2018), pp. 2429–2453. DOI: 10.1109/COMST.2018.2815638.

[15] Shalitha Wijethilaka and Madhusanka Liyanage. ‘Survey on Network Slicing for Internet of Things Realization in 5G Networks’. In: *IEEE Communications Surveys & Tutorials* 23.2 (2021), pp. 957–994. DOI: 10.1109/COMST.2021.3067807.

[16] Hsu-Tung Chien et al. ‘End-to-End Slicing With Optimized Communication and Computing Resource Allocation in Multi-Tenant 5G Systems’. In: *IEEE Transactions on Vehicular Technology* 69.2 (2020), pp. 2079–2091. DOI: 10.1109/TVT.2019.2959193.

[17] Alcardo Alex Barakabitze et al. ‘QoE Management of Multimedia Streaming Services in Future Networks: A Tutorial and Survey’. In: *IEEE Communications Surveys & Tutorials* 22.1 (2020), pp. 526–565. DOI: 10.1109/COMST.2019.2958784.

[18] Daniel Ayepah Mensah et al. 'Federated Policy Distillation for Digital Twin-Enabled Intelligent Resource Trading in 5G Network Slicing'. In: *IEEE Transactions on Network and Service Management* (2024), pp. 1–1. DOI: 10.1109/TNSM.2024.3476480.

[19] Mahdieh Ahmadi et al. 'Generalizable 5G RAN/MEC Slicing and Admission Control for Reliable Network Operation'. In: *IEEE Transactions on Network and Service Management* 21.5 (2024), pp. 5384–5399. DOI: 10.1109/TNSM.2024.3437217.

[20] Latif U. Khan et al. 'Network Slicing: Recent Advances, Taxonomy, Requirements, and Open Research Challenges'. In: *IEEE Access* 8 (2020), pp. 36009–36028. DOI: 10.1109/ACCESS.2020.2975072.

[21] Ali Chouman, Dimitrios Michael Manias and Abdallah Shami. 'A Modular, End-to-End Next-Generation Network Testbed: Toward a Fully Automated Network Management Platform'. In: *IEEE Transactions on Network and Service Management* 21.5 (2024), pp. 5445–5463. DOI: 10.1109/TNSM.2024.3416031.

[22] Qiao Ren et al. 'Resource Allocation and Slicing Strategy for Multiple Services Co-Existence in Wireless Train Communication Network'. In: *IEEE Transactions on Wireless Communications* (2024), pp. 1–1. DOI: 10.1109/TWC.2024.3493240.

[23] Felipe Arnhold et al. 'Network Slicing Support by Fronthaul Interface in Disaggregated Radio Access Networks: A Survey'. In: *IEEE Transactions on Network and Service Management* 21.4 (2024), pp. 4510–4530. DOI: 10.1109/TNSM.2024.3400019.

[24] Suvidha Mhatre et al. 'Intelligent QoS-aware Slice Resource Allocation with User Association Parameterization for Beyond 5G O-RAN-based Architecture Using DRL'. In: *IEEE Transactions on Vehicular Technology* (2024), pp. 1–14. DOI: 10.1109/TVT.2024.3483288.

[25] Luis A. Garrido et al. 'Resource Demand Prediction for Network Slices in 5G Using ML Enhanced With Network Models'. In: *IEEE Transactions on Vehicular Technology* 73.8 (2024), pp. 11848–11861. DOI: 10.1109/TVT.2024.3373490.

[26] Xinggang Fan et al. 'Optimizing Task Offloading and Resource Allocation in Vehicular Edge Computing Based on Heterogeneous Cellular Networks'. In: *IEEE Transactions*

on Vehicular Technology 73.5 (2024), pp. 7175–7187. DOI: 10.1109/TVT.2023.3345364.

[27] Hongyu Xiang, Shi Yan and Mugen Peng. 'A Realization of Fog-RAN Slicing via Deep Reinforcement Learning'. In: *IEEE Transactions on Wireless Communications* 19.4 (2020), pp. 2515–2527. DOI: 10.1109/TWC.2020.2965927.

[28] Jiajia Liu et al. 'Space-Air-Ground Integrated Network: A Survey'. In: *IEEE Communications Surveys & Tutorials* 20.4 (2018), pp. 2714–2741. DOI: 10.1109/COMST.2018.2841996.

[29] Federica Rinaldi et al. 'Non-Terrestrial Networks in 5G & Beyond: A Survey'. In: *IEEE Access* 8 (2020), pp. 165178–165200. DOI: 10.1109/ACCESS.2020.3022981.

[30] Renchao Xie et al. 'Satellite-Terrestrial Integrated Edge Computing Networks: Architecture, Challenges, and Open Issues'. In: *IEEE Network* 34.3 (2020), pp. 224–231. DOI: 10.1109/MNET.011.1900369.

[31] Zhaoji Zhang et al. 'User Activity Detection and Channel Estimation for Grant-Free Random Access in LEO Satellite-Enabled Internet of Things'. In: *IEEE Internet of Things Journal* 7.9 (2020), pp. 8811–8825. DOI: 10.1109/JIOT.2020.2997336.

[32] Yuyi Mao et al. 'A Survey on Mobile Edge Computing: The Communication Perspective'. In: *IEEE Communications Surveys & Tutorials* 19.4 (2017), pp. 2322–2358. DOI: 10.1109/COMST.2017.2745201.

[33] Dinh C. Nguyen et al. 'Enabling AI in Future Wireless Networks: A Data Life Cycle Perspective'. In: *IEEE Communications Surveys & Tutorials* 23.1 (2021), pp. 553–595. DOI: 10.1109/COMST.2020.3024783.

[34] Peng Cheng et al. 'Deep Reinforcement Learning for Online Resource Allocation in IoT Networks: Technology, Development, and Future Challenges'. In: *IEEE Communications Magazine* 61.6 (2023), pp. 111–117. DOI: 10.1109/MCOM.001.2200526.

[35] Qian Chen et al. 'Multi-Tier Hybrid Offloading for Computation-Aware IoT Applications in Civil Aircraft-Augmented SAGIN'. In: *IEEE Journal on Selected Areas*

in Communications 41.2 (2023), pp. 399–417. DOI: 10.1109/JSAC.2022.3227031.

[36] Pavel Mach and Zdenek Becvar. 'Mobile Edge Computing: A Survey on Architecture and Computation Offloading'. In: *IEEE Communications Surveys & Tutorials* 19.3 (2017), pp. 1628–1656. DOI: 10.1109/COMST.2017.2682318.

[37] Weihao Mao et al. 'UAV-Assisted Communications in SAGIN-ISAC: Mobile User Tracking and Robust Beamforming'. In: *IEEE Journal on Selected Areas in Communications* (2024), pp. 1–1. DOI: 10.1109/JSAC.2024.3460065.

[38] Yulan Gao, Ziqiang Ye and Han Yu. 'Cost-Efficient Computation Offloading in SAGIN: A Deep Reinforcement Learning and Perception-Aided Approach'. In: *IEEE Journal on Selected Areas in Communications* 42.12 (2024), pp. 3462–3476. DOI: 10.1109/JSAC.2024.3459073.

[39] Minh Dat Nguyen, Long Bao Le and André Girard. 'Joint Computation Offloading, UAV Trajectory, User Scheduling, and Resource Allocation in SAGIN'. In: *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*. 2022, pp. 5099–5104. DOI: 10.1109/GLOBECOM48099.2022.10001422.

[40] Yue Cai et al. 'Deep Reinforcement Learning for Online Resource Allocation in Network Slicing'. In: *IEEE Transactions on Mobile Computing* 23.6 (2024), pp. 7099–7116. DOI: 10.1109/TMC.2023.3328950.

[41] Min Wu et al. 'Joint Optimization Design of RIS-Assisted Hybrid FSO SAGINs Using Deep Reinforcement Learning'. In: *IEEE Transactions on Vehicular Technology* 73.3 (2024), pp. 3025–3040. DOI: 10.1109/TVT.2023.3324970.

[42] Zheyi Chen et al. 'Traffic-Aware Lightweight Hierarchical Offloading Toward Adaptive Slicing-Enabled SAGIN'. In: *IEEE Journal on Selected Areas in Communications* 42.12 (2024), pp. 3536–3550. DOI: 10.1109/JSAC.2024.3459020.

[43] Chong Huang et al. 'Joint Offloading and Resource Allocation for Hybrid Cloud and Edge Computing in SAGINs: A Decision Assisted Hybrid Action Space Deep Reinforcement Learning Approach'. In: *IEEE Journal on Selected Areas in Communications* 42.5 (2024), pp. 1029–1043. DOI: 10.1109/JSAC.2024.3365899.

[44] Yunfeng Wang et al. ‘E2E Network Slicing Optimization for Control- and User-Plane Separation-Based SAGINs With DRL’. In: *IEEE Transactions on Vehicular Technology* 73.11 (2024), pp. 16680–16696. DOI: 10.1109/TVT.2024.3415964.

[45] Haotong Cao et al. ‘Efficient Resource Allocation of Slicing Services in Softwarized Space-Aerial-Ground Integrated Networks for Seamless and Open Access Services’. In: *IEEE Transactions on Vehicular Technology* 73.7 (2024), pp. 9284–9295. DOI: 10.1109/TVT.2023.3328500.

[46] Yuejiao Xie et al. ‘Online Scheduling for Multi-Access in Space-Air-Ground Integrated Networks using iGNNSeq-Enhanced Reinforcement Learning’. In: *IEEE Transactions on Vehicular Technology* (2024), pp. 1–14. DOI: 10.1109/TVT.2024.3489216.

[47] Yue Xiao et al. ‘Space-Air-Ground Integrated Wireless Networks for 6G: Basics, Key Technologies, and Future Trends’. In: *IEEE Journal on Selected Areas in Communications* 42.12 (2024), pp. 3327–3354. DOI: 10.1109/JSAC.2024.3492720.

[48] Peiying Zhang et al. ‘Energy Aware Space-Air-Ground Integrated Network Resource Orchestration Algorithm’. In: *IEEE Transactions on Vehicular Technology* (2024), pp. 1–11. DOI: 10.1109/TVT.2024.3440385.

[49] Salah Eddine Elayoubi et al. ‘5G RAN Slicing for Verticals: Enablers and Challenges’. In: *IEEE Communications Magazine* 57.1 (2019), pp. 28–34. DOI: 10.1109/MCOM.2018.1701319.

[50] Danish Sattar and Ashraf Matrawy. ‘Optimal Slice Allocation in 5G Core Networks’. In: *IEEE Networking Letters* 1.2 (2019), pp. 48–51. DOI: 10.1109/LNET.2019.2908351.

[51] Weiqi Liu et al. ‘Reinforcement Learning-Based Network Slicing Scheme for Optimized UE-QoS in Future Networks’. In: *IEEE Transactions on Network and Service Management* 21.3 (2024), pp. 3454–3464. DOI: 10.1109/TNSM.2024.3368294.

[52] Haonan Bai et al. ‘Latency Equalization Policy of End-to-End Network Slicing Based on Reinforcement Learning’. In: *IEEE Transactions on Network and Service Management* 20.1 (2023), pp. 88–103. DOI: 10.1109/TNSM.2022.3210012.

[53] Danish Sattar and Ashraf Matrawy. 'Towards Secure Slicing: Using Slice Isolation to Mitigate DDoS Attacks on 5G Core Network Slices'. In: *2019 IEEE Conference on Communications and Network Security (CNS)*. 2019, pp. 82–90. DOI: 10.1109/CNS.2019.8802852.

[54] Chia-Wei Liao, Fuchun Joseph Lin and Yoichi Sato. 'Evaluating NFV-enabled Network Slicing for 5G Core'. In: *2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS)*. 2020, pp. 401–404. DOI: 10.23919/APNOMS50412.2020.9236978.

[55] Meysam Masoudi et al. 'Energy-Optimal End-to-End Network Slicing in Cloud-Based Architecture'. In: *IEEE Open Journal of the Communications Society* 3 (2022), pp. 574–592. DOI: 10.1109/OJCOMS.2022.3162116.

[56] Mojdeh Karbalaee Motalleb et al. 'Resource Allocation in an Open RAN System Using Network Slicing'. In: *IEEE Transactions on Network and Service Management* 20.1 (2023), pp. 471–485. DOI: 10.1109/TNSM.2022.3205415.

[57] Lei Wang et al. 'Resource Allocation for Multi-Traffic in Cross-Modal Communications'. In: *IEEE Transactions on Network and Service Management* 20.1 (2023), pp. 60–72. DOI: 10.1109/TNSM.2022.3207776.

[58] Mengqiu Tian et al. 'Optimized Sparrow Search-based Multiplexing of eMBB and URLLC in 5G/B5G Networks'. In: *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*. 2022, pp. 3658–3663. DOI: 10.1109/GLOBECOM48099.2022.10001001.

[59] Ammara Anjum Khan et al. 'An End-to-End (E2E) Network Slicing Framework for 5G Vehicular Ad-Hoc Networks'. In: *IEEE Transactions on Vehicular Technology* 70.7 (2021), pp. 7103–7112. DOI: 10.1109/TVT.2021.3084735.

[60] Arjun Anand, Gustavo de Veciana and Sanjay Shakkottai. 'Joint Scheduling of URLLC and eMBB Traffic in 5G Wireless Networks'. In: *IEEE/ACM Transactions on Networking* 28.2 (2020), pp. 477–490. DOI: 10.1109/TNET.2020.2968373.

[61] Yuxiu Hua et al. 'GAN-Powered Deep Distributional Reinforcement Learning for Resource Management in Network Slicing'. In: *IEEE Journal on Selected Areas*

in Communications 38.2 (2020), pp. 334–349. DOI: 10.1109/JSAC.2019.2959185.

[62] Rongpeng Li et al. 'The LSTM-Based Advantage Actor-Critic Learning for Resource Management in Network Slicing With User Mobility'. In: *IEEE Communications Letters* 24.9 (2020), pp. 2005–2009. DOI: 10.1109/LCOMM.2020.3001227.

[63] Jie Mei et al. 'Intelligent Radio Access Network Slicing for Service Provisioning in 6G: A Hierarchical Deep Reinforcement Learning Approach'. In: *IEEE Transactions on Communications* 69.9 (2021), pp. 6063–6078. DOI: 10.1109/TCOMM.2021.3090423.

[64] Zhaoying Wang et al. 'Utility Optimization for Resource Allocation in Multi-Access Edge Network Slicing: A Twin-Actor Deep Deterministic Policy Gradient Approach'. In: *IEEE Transactions on Wireless Communications* 21.8 (2022), pp. 5842–5856. DOI: 10.1109/TWC.2022.3143949.

[65] Wenjun Wu et al. 'Heterogeneous Markov Decision Process Model for Joint Resource Allocation and Task Scheduling in Network Slicing Enabled Internet of Vehicles'. In: *IEEE Wireless Communications Letters* 11.6 (2022), pp. 1118–1122. DOI: 10.1109/LWC.2022.3152177.

[66] Amir Gharehgoli et al. 'AI-based Resource Allocation in End-to-End Network Slicing under Demand and CSI Uncertainties'. In: *IEEE Transactions on Network and Service Management* (2023), pp. 1–1. DOI: 10.1109/TNSM.2023.3243837.

[67] Sachula Meng et al. 'RAN Slice Strategy Based on Deep Reinforcement Learning for Smart Grid'. In: *2019 Computing, Communications and IoT Applications (ComComAp)*. 2019, pp. 6–11. DOI: 10.1109/ComComAp46287.2019.9018826.

[68] Tianjian Dong et al. 'Intelligent Joint Network Slicing and Routing via GCN-Powered Multi-Task Deep Reinforcement Learning'. In: *IEEE Transactions on Cognitive Communications and Networking* 8.2 (2022), pp. 1269–1286. DOI: 10.1109/TCCN.2021.3136221.

[69] Xinyuan Zhang et al. 'Energy-Efficient Computation Peer Offloading in Satellite Edge Computing Networks'. In: *IEEE Transactions on Mobile Computing* (2023), pp. 1–15. DOI: 10.1109/TMC.2023.3269801.

[70] Zhengyu Song et al. 'Energy-Efficient Multiaccess Edge Computing for Terrestrial-Satellite Internet of Things'. In: *IEEE Internet of Things Journal* 8.18 (2021), pp. 14202–14218. DOI: 10.1109/JIOT.2021.3068141.

[71] Xuelin Cao et al. 'Edge-Assisted Multi-Layer Offloading Optimization of LEO Satellite-Terrestrial Integrated Networks'. In: *IEEE Journal on Selected Areas in Communications* 41.2 (2023), pp. 381–398. DOI: 10.1109/JSAC.2022.3227032.

[72] Yan Kyaw Tun et al. 'Collaborative Computing Services at Ground, Air, and Space: An Optimization Approach'. In: *IEEE Transactions on Vehicular Technology* 73.1 (2024), pp. 1491–1496. DOI: 10.1109/TVT.2023.3304713.

[73] Sun Mao, Shunfan He and Jinsong Wu. 'Joint UAV Position Optimization and Resource Scheduling in Space-Air-Ground Integrated Networks With Mixed Cloud-Edge Computing'. In: *IEEE Systems Journal* 15.3 (2021), pp. 3992–4002. DOI: 10.1109/JSYST.2020.3041706.

[74] Yan Kyaw Tun et al. 'Energy-Efficient Resource Management in UAV-Assisted Mobile Edge Computing'. In: *IEEE Communications Letters* 25.1 (2021), pp. 249–253. DOI: 10.1109/LCOMM.2020.3026033.

[75] Yi Liu et al. 'Online Computation Offloading for Collaborative Space/Aerial-Aided Edge Computing Toward 6G System'. In: *IEEE Transactions on Vehicular Technology* 73.2 (2024), pp. 2495–2505. DOI: 10.1109/TVT.2023.3312676.

[76] Lijun He et al. 'Balancing Total Energy Consumption and Mean Makespan in Data Offloading for Space-Air-Ground Integrated Networks'. In: *IEEE Transactions on Mobile Computing* 23.1 (2024), pp. 209–222. DOI: 10.1109/TMC.2022.3222848.

[77] Minh Dat Nguyen, Long Bao Le and André Girard. 'Integrated Computation Offloading, UAV Trajectory Control, Edge-Cloud and Radio Resource Allocation in SAGIN'. In: *IEEE Transactions on Cloud Computing* 12.1 (2024), pp. 100–115. DOI: 10.1109/TCC.2023.3339394.

[78] Seonghoon Yoo et al. 'Cache-Assisted Mobile-Edge Computing Over Space–Air–Ground Integrated Networks for Extended Reality Applications'. In: *IEEE Internet of Things Journal* 11.10 (2024), pp. 18306–18319. DOI: 10.1109/JIOT.2024.3361907.

[79] Peng Qin et al. ‘Energy-Efficient Resource Allocation for Space–Air–Ground Integrated Industrial Power Internet of Things Network’. In: *IEEE Transactions on Industrial Informatics* 20.4 (2024), pp. 5274–5284. DOI: 10.1109/TII.2023.3331127.

[80] Tiago Koketsu Rodrigues and Nei Kato. ‘Hybrid Centralized and Distributed Learning for MEC-Equipped Satellite 6G Networks’. In: *IEEE Journal on Selected Areas in Communications* 41.4 (2023), pp. 1201–1211. DOI: 10.1109/JSAC.2023.3242700.

[81] Nan Cheng et al. ‘Space/Aerial-Assisted Computing Offloading for IoT Applications: A Learning-Based Approach’. In: *IEEE Journal on Selected Areas in Communications* 37.5 (2019), pp. 1117–1129. DOI: 10.1109/JSAC.2019.2906789.

[82] Furong Chai et al. ‘Joint Multi-Task Offloading and Resource Allocation for Mobile Edge Computing Systems in Satellite IoT’. In: *IEEE Transactions on Vehicular Technology* 72.6 (2023), pp. 7783–7795. DOI: 10.1109/TVT.2023.3238771.

[83] Sheikh Salman Hassan et al. ‘Satellite-Based ITS Data Offloading & Computation in 6G Networks: A Cooperative Multi-Agent Proximal Policy Optimization DRL With Attention Approach’. In: *IEEE Transactions on Mobile Computing* (2023), pp. 1–18. DOI: 10.1109/TMC.2023.3300314.

[84] Hang Shen et al. ‘Slicing-Based Task Offloading in Space-Air-Ground Integrated Vehicular Networks’. In: *IEEE Transactions on Mobile Computing* (2023), pp. 1–15. DOI: 10.1109/TMC.2023.3283852.

[85] Yeguang Qin et al. ‘Differentiated Federated Reinforcement Learning Based Traffic Offloading on Space-Air-Ground Integrated Networks’. In: *IEEE Transactions on Mobile Computing* (2024), pp. 1–14. DOI: 10.1109/TMC.2024.3389011.

[86] Anal Paul et al. ‘Digital Twin-Assisted Space-Air-Ground Integrated Networks for Vehicular Edge Computing’. In: *IEEE Journal of Selected Topics in Signal Processing* 18.1 (2024), pp. 66–82. DOI: 10.1109/JSTSP.2023.3340107.

[87] Yidong Liu et al. ‘Joint Delay and Completion Rate Optimization for Dependent Task Offloading in Space-Air-Ground Integrated Network’. In: *2023 9th International*

Conference on Computer and Communications (ICCC). 2023, pp. 327–331. DOI: 10.1109/ICCC59590.2023.10507613.

[88] Wenwu Zhu et al. 'Cost-Efficient 6G Space-Air-Ground Integrated Mobile Edge Computing for Smart City: A PPO-Based Offloading Decision and Resource Allocation Algorithm'. In: *2023 IEEE International Conference on High Performance Computing & Communications, Data Science & Systems, Smart City & Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*. 2023, pp. 241–248. DOI: 10.1109/HPCC-DSS-SmartCity-DependSys60770.2023.00041.

[89] Kexin Fan et al. 'Demand-Driven Task Scheduling and Resource Allocation in Space-Air-Ground Integrated Network: A Deep Reinforcement Learning Approach'. In: *IEEE Transactions on Wireless Communications* (2024), pp. 1–1. DOI: 10.1109/TWC.2024.3398199.

[90] Chen-Shang Chang and J.A. Thomas. 'Effective bandwidth in high-speed digital networks'. In: *IEEE Journal on Selected Areas in Communications* 13.6 (1995), pp. 1091–1100. DOI: 10.1109/49.400664.

[91] John Schulman et al. 'Trust region policy optimization'. In: *International conference on machine learning*. PMLR. 2015, pp. 1889–1897.

[92] Suzhi Bi et al. 'Lyapunov-Guided Deep Reinforcement Learning for Stable Online Computation Offloading in Mobile-Edge Computing Networks'. In: *IEEE Transactions on Wireless Communications* 20.11 (2021), pp. 7519–7537. DOI: 10.1109/TWC.2021.3085319.

[93] James Goulding, Simon Preston and Gavin Smith. 'Event Series Prediction via Non-Homogeneous Poisson Process Modelling'. In: *2016 IEEE 16th International Conference on Data Mining (ICDM)*. 2016, pp. 161–170. DOI: 10.1109/ICDM.2016.0027.

[94] Rui Dong et al. 'Deep Learning for Radio Resource Allocation With Diverse Quality-of-Service Requirements in 5G'. In: *IEEE Transactions on Wireless Communications* 20.4 (2021), pp. 2309–2324. DOI: 10.1109/TWC.2020.3041319.

[95] John Schulman et al. ‘Proximal policy optimization algorithms’. In: *arXiv preprint arXiv:1707.06347* (2017).

[96] Tuomas Haarnoja et al. ‘Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor’. In: *International conference on machine learning*. PMLR. 2018, pp. 1861–1870.

[97] Timothy P Lillicrap et al. ‘Continuous control with deep reinforcement learning’. In: *arXiv preprint arXiv:1509.02971* (2015).

[98] John Schulman et al. ‘Trust Region Policy Optimization’. In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. ICML’15. Lille, France: JMLR.org, 2015, pp. 1889–1897.

[99] Changyang She, Chenyang Yang and Tony Q. S. Quek. ‘Cross-Layer Optimization for Ultra-Reliable and Low-Latency Radio Access Networks’. In: *IEEE Transactions on Wireless Communications* 17.1 (2018), pp. 127–141. DOI: 10.1109/TWC.2017.2762684.

[100] Ye Hu, Xiaodong Wang and Walid Saad. ‘Distributed and Distribution-Robust Meta Reinforcement Learning (D^2-RMRL) for Data Pre-Storage and Routing in Cube Satellite Networks’. In: *IEEE Journal of Selected Topics in Signal Processing* 17.1 (2023), pp. 128–141. DOI: 10.1109/JSTSP.2022.3232944.

[101] Guanjin Qu et al. ‘DMRO: A Deep Meta Reinforcement Learning-Based Task Offloading Framework for Edge-Cloud Computing’. In: *IEEE Transactions on Network and Service Management* 18.3 (2021), pp. 3448–3459. DOI: 10.1109/TNSM.2021.3087258.

[102] Ziyang Lu and M Cenk Gursoy. ‘Dynamic channel access via meta-reinforcement learning’. In: *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2021, pp. 01–06.

[103] Zonghan Wu et al. ‘A comprehensive survey on graph neural networks’. In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24.

[104] Pedro J Freire et al. ‘Performance versus complexity study of neural network equalizers in coherent optical systems’. In: *Journal of Lightwave Technology* 39.19 (2021), pp. 6085–6096.

[105] Yizhen Xu et al. 'Constrained Reinforcement Learning for Resource Allocation in Network Slicing'. In: *IEEE Communications Letters* 25.5 (2021), pp. 1554–1558. DOI: 10.1109/LCOMM.2021.3053612.

[106] ITU Radiocommunication Assembly. 'Preferred characteristics of systems in the fixed service using high altitude platforms operating in the bands 47.2-47.5 GHz and 47.9-48.2 GHz'. In: *ITU-R F* 1500 ().

[107] I Recommendation. 'Propagation data required for the design of earth-space land mobile telecommunication systems'. In: *International Telecommunication Union: Geneva, Switzerland* (2017), pp. 681–710.

[108] *Propagation data required for the design systems in the land mobile-satellite service*. Tech. rep. Aug. 2019. Recommendation ITU-R P.681-11, Aug. 2019.

[109] Thomas N Kipf and Max Welling. 'Semi-supervised classification with graph convolutional networks'. In: *arXiv preprint arXiv:1609.02907* (2016).

[110] Ashish Vaswani et al. 'Attention is all you need'. In: *Advances in neural information processing systems* 30 (2017).

[111] Yifei Shen et al. 'Graph neural networks for wireless communications: From theory to practice'. In: *IEEE Transactions on Wireless Communications* 22.5 (2022), pp. 3554–3569.

[112] Yifei Shen et al. 'Graph neural networks for scalable radio resource management: Architecture design and theoretical analysis'. In: *IEEE Journal on Selected Areas in Communications* 39.1 (2020), pp. 101–115.

[113] Andreas Loukas. 'What graph neural networks cannot learn: depth vs width'. In: *International Conference on Learning Representations*. 2020.

[114] Leslie G Valiant. 'A theory of the learnable'. In: *Communications of the ACM* 27.11 (1984), pp. 1134–1142.

[115] Sanjeev Arora et al. 'Fine-grained analysis of optimization and generalization for over-parameterized two-layer neural networks'. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 322–332.