RESOURCE ALLOCATION IN THE WIRELESS INTERNET-OF-THINGS

By

Guangchen Wang

SUBMITTED IN FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
AT
CENTRE OF EXCELLENCE IN TELECOMMUNICATIONS
SCHOOL OF ELECTRICAL AND INFORMATION ENGINEERING
THE UNIVERSITY OF SYDNEY

AUGUST 2023

To my loving girlfriend Lujing Zhang

and

my beloved parents Qinglong Wang and Jixian Song

Acknowledgements

First and foremost, I am tremendously grateful to my supervisor, Prof. Yonghui Li, whose wisdom and accomplishments as a scholar are beyond measure. It was Yonghui who unveiled the complex yet intriguing world of research to me, carefully fostering my innate curiosity. With insightful guidance and constant support, he directed me along the winding path of academic exploration, illuminating the way with his own vast knowledge and experience.

I would like to express my profound gratitude to Dr. Peng Cheng, whose guidance and support have been pivotal not only in the completion of this thesis but also in furthering my academic growth. I am also grateful to other committee members who contributed their unique insights and wisdom to this thesis, notably including Dr. Zhuo Chen and Prof. Branka Vucetic. Their contributions enriched the quality of my research, and I am sincerely appreciative of their involvement.

I appreciate the financial support that has enabled my research journey. This includes the Department of Education, Australian Government for awarding me the Research Training Program Scholarship, and the University of Sydney for granting the Paulette Isabel Jones Completion Scholarship. In addition, I want to express my thanks to Prof. Yonghui Li and recognize the University of Sydney's Postgraduate Research Support Scheme for their generous financial backing of my participation in international conferences.

Last but not least, I must extend my deepest and most heartfelt gratitude to my parents, whose unwavering support has been my guiding light. I would also like to take this opportunity to express my sincere appreciation to my girlfriend, Lujing Zhang, for her love and support in my journey.

Statement of Originality

The work presented in this thesis is the result of original research carried out by myself, in collaboration with my supervisors, while enrolled in the School of Electrical and Information Engineering at the University of Sydney as a Ph.D. candidate.

These studies were conducted under the supervision of Prof. Branka Vucetic, Dr. Peng Cheng, and Prof. Yonghui Li. It has not been submitted for any other degree or award in any other university or educational institution.

Guangchen Wang
School of Electrical and Information Engineering
The University of Sydney
August 2023

Authorship Attribution Statement

This thesis makes its primary contributions within Chapters 3, 4, and 5, with each chapter corresponding to a respective paper: [J1], [J2], and [J3]. In each case:

- Chapter 3 includes the content from paper [J1] that is under review. I designed the study, performed simulations, implemented the prototype, collected and analyzed the data, and wrote the manuscript drafts.
- Chapter 4 includes the content from paper [J2] that is under review. I designed the study, performed simulations, collected and analyzed the data, and wrote the manuscript drafts.
- Chapter 5 includes the content from paper [J3] that is under review. I designed the study, performed simulations, collected and analyzed the data, and wrote the manuscript drafts.

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 by the corresponding author.

Guangchen Wang, August 2023

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

Yonghui Li, August 2023

Abstract

The Internet-of-Things (IoT) is widely regarded as a transformative and promising paradigm, marking a revolutionary shift in the way that technology interacts with the world around us. Comprising a network of interconnected devices and sensors, IoT seeks to foster seamless communication and automation across various platforms. Despite the rising popularity and extensive integration of IoT across diverse domains, such as smart homes and industrial automation, the development and deployment of these interconnected systems are met with considerable challenges. One notable challenge is the scarcity of spectrum resources, which poses a significant obstacle in accommodating the massive data transmission expected among the myriad devices that will function under the current resource allocation scheme. This problem is further exacerbated by the existing technology's inability to manage such colossal volumes of data. In addition, small-sized mobile terminals are constrained by limited computation and energy resources. These limitations make the traditional standalone operation of devices increasingly unfeasible. The recent literature has offered hope in this context, highlighting several emerging concepts such as mobile edge computing (MEC), multiple-input multiple-output (MIMO), and non-orthogonal multiple access (NOMA). These innovations have been identified as promising solutions for addressing the resource-oriented challenges prevalent in IoT.

In this thesis, we delve into comprehensive task offloading and resource allocation strategies with the goal of enhancing the efficient utilization of limited resources in wireless IoT systems, encompassing spectrum, computation, and energy resources. In the first part, we incorporate device-to-device (D2D) communication into the multi-layer computing network and propose a full-dimensional task offloading scheme. On this basis, we formulate an mixed-integer nonlinear programming (MINLP) problem with discrete task offloading decisions and continuous computation/communication resource allocation. To solve this problem, we develop an inverse reinforcement learning (IRL) based algorithm. This offers a unified solution to a broad class of MINLP

problems in wireless IoT networks by accelerating the optimal branch-and-bound (B&B) algorithm with significantly reduced complexity but without sacrificing the global optimality. In the second part, we delve into strategies to optimize energy efficiency within the downlink cell-free massive MIMO systems. We develop a green energy scheme by simultaneously optimizing power allocation and access point (AP) selection and formulate it as a non-convex MINLP problem. To solve this challenging problem, we propose a novel optimization-embedded deep reinforcement learning (DRL) algorithm, which enjoys the benefits of directly inferring solutions for the formulated problem. To enable the algorithm to adapt to the variations of the system parameters without learning from scratch, we further develop graph transformer networks (GTN). In the last part, we develop a NOMA-based task offloading scheme in a multi-layer computing network to minimize the latency across all mobile devices. On this basis, we formulate the task offloading scheme as a non-convex mixed-integer optimization problem and propose a reincarnating DRL algorithm, where accumulated apriori information is incorporated for fast retraining. Besides, we design a reward function and an evaluation phase to ensure the communication/computation constraints to be satisfied with a high probability.

Contents

List of Figures

List of Acronyms

2-D	two-dimensional
5G	fifth-generation
AC	actor critic
ADMM	alternating direction method of multipliers
AP	access point
AR	augmented reality
B&B	branch-and-bound
BS	base station
CB	conjugate beamforming
CDF	cumulative distribution functions
CPU	central processing unit
CS	cloud server
CSI	channel state information
D2D	device-to-device
DDPG	deep deterministic policy gradients
DNN	deep neural network
DRL	deep reinforcement learning
FD	full-dimensional
FSB	full strong branching
GNN	graph neural network
GTN	graph transformer network

i.i.d.	independent and identically distributed
IoT	Internet of Things
IRL	inverse reinforcement learning
LB	lower bound
MDP	Markov decision process
MCC	mobile cloud computing
MD	mobile device
MEC	mobile edge computing
MS	mobile edge computing server
MILP	mixed-integer linear programming
MINLP	mixed-integer nonlinear programming
MIMO	multiple-input multiple-output
MMSE	minimum mean square error
NOMA	non-orthogonal multiple access
NLP	nonlinear programming
OMA	orthogonal multiple access
QoS	quality of service
RL	reinforcement learning
RPB	pseudo-cost branching policy
SAC	soft actor critic
SGD	soft stochastic gradient descent
SIC	successive interference cancellation
SINR	signal-to-interference-plus-noise ratio
TDD	time-division duplex
UB	upper bound
UE	user equipment
ZF	zero-forcing

List of Symbols and Notations

$\mathcal{N}(\cdot,\cdot)$	the Gaussian distribution
$\mathcal{CN}(\cdot,\cdot)$	the complex Gaussian distribution
$\lceil\cdot\rceil$	the ceiling function
$\lfloor\cdot\rfloor$	the floor function
$(\cdot)^*$	the conjugate operation
$(\cdot)^T$	the transpose operation
$\odot$	the Schur product operation
$(\cdot)^H$	the conjugate-transpose operation
$\mathbf{I}_N$	an $N \times N$ identity matrix
$\mathbb{E}[\cdot]$	the expectation operator
$\lvert\cdot\rvert^2$	the absolute value operation
$\lVert\cdot\rVert$	the Euclidean norm
N_0	the average power of the additive white Gaussian noise

List of Publications

The following is a list of publications in refereed journals and conference proceedings produced during my Ph.D. candidature. In some cases, the journal papers contain materials overlapping with the conference publications.

Journal Papers

[J1] **G. Wang**, P. Cheng, Z. Chen, B. Vucetic, and Y. Li, "Inverse Reinforcement Learning with Graph Neural Networks for Full-Dimensional Task Offloading in Edge Computing," *Transactions on Mobile Computing*, pp. 1–18, 2023.

[J2] **G. Wang**, P. Cheng, Z. Chen, B. Vucetic, and Y. Li, "Green Cell-free Massive MIMO: An Optimization Embedded Deep Reinforcement Learning Approach," *IEEE Transactions on Signal Processing*, major revision, Jul. 2023.

[J3] **G. Wang**, P. Cheng, Z. Chen, B. Vucetic, and Y. Li, "Partial NOMA-based Online Task Offloading in Multi-layer Mobile Computing Networks," submitted to *IEEE Communications Letters*.

[J4] **G. Wang**, P. Cheng, Z. Chen, B. Vucetic, and Y. Li, "Deep Inverse Reinforcement Learning for Large-Scale Resource Allocation," to be submitted to *IEEE Journal on Selected Areas in Communications*.

Conference Papers

[C1] **G. Wang**, P. Cheng, Z. Chen, W. Xiang, B. Vucetic, and Y. Li, "Inverse reinforcement learning with graph neural networks for IoT resource allocation," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Rhodes Island, Greece, 2023, pp. 1-5.

[C2] **G. Wang**, P. Cheng, Z. Chen, B. Vucetic, and Y. Li, "Learning-based Energy Efficiency Optimization in Cell-free Massive MIMO," in *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*, Kuala Lumpur, Malaysia, accepted, Aug. 2023.

Chapter 1

Introduction

In this chapter, we first introduce the background of our research. Subsequently, we detail the impetus behind our chosen research direction, concurrently accentuating the multifaceted challenges associated with resolving the problems we have decided to tackle. Finally, we summarize the fundamental contributions made by this thesis, reinforcing its value to its field of study.

1.1 Background

The wireless Internet of Things (IoT) network marks a significant milestone in the communication and information technology field [1]. It aims to create interconnected and intelligent environments that can accommodate a broad spectrum of applications [2]. The ubiquitous wireless IoT is projected to establish connections with billions of devices worldwide. These encompass mobile sensors, manufacturing machinery, smart devices, and industrial utilities, thus shaping an intricate web of interconnections [3]. The idea behind the wireless IoT is to integrate the physical and digital worlds, promoting efficiency, improving accuracy, and enabling things to respond in real time. It has the potential to dramatically enhance the way we interact with the world around

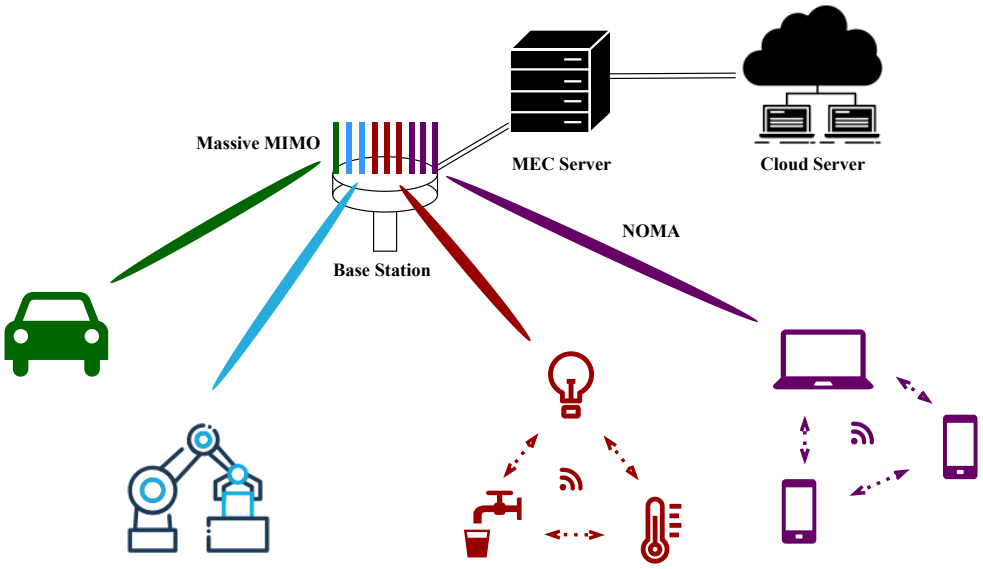


Figure 1.1: A diagram of IoT network.

us, impacting a wide array of sectors including healthcare, transportation, agriculture, and manufacturing [4–7].

The wireless IoT is underpinned by several key technologies, including mobile edge computing (MEC) [8], multiple-input multiple-output (MIMO) [9], non-orthogonal multiple access (NOMA) [10], and machine learning [11]. Together, these technologies facilitate a smooth interaction between 'things' on the wireless IoT network, making it possible to create systems that are smart, responsive, and capable of self-regulation. While the wireless IoT holds great promise, it also introduces a set of challenges. The cumulative challenges considerably strain the existing approaches and systems of resource allocation, configuration, and deployment [12]. Next, we summarize challenges from three perspectives: spectrum, computing, and energy.

Spectrum perspective: The frequent information interactions and data transmissions among the vast number of IoT devices will generate an unprecedented volume

of data to be transmitted via wireless communication [13], which needs an abundant spectrum resource supply. However, there has been a persistent scarcity in the availability of the radio frequency (RF) spectrum [14]. There are only limited frequency bands available for use, and many of them are already taken up by existing services [15]. As more and more devices and technologies look to use the RF spectrum in IoT networks, the demand has greatly increased. Furthermore, these frequencies cannot just be used by anyone at any time. Different parts of the spectrum are allocated to different uses and users, often through auctions, where companies can bid to secure exclusive rights to use certain frequencies. As such, it is clear that the demand for space on this spectrum has been consistently exceeding the supply, leading to issues of congestion, competition, and potential interference, especially as our reliance on wireless technologies continues to grow [16]. Therefore, determining how to integrate the considerable number of IoT devices effectively given the limited spectrum resources, has become a pivotal aspect of developing future IoT networks.

Computation perspective: From a computational perspective, a myriad of emerging applications powered by the IoT, such as facial recognition, participatory sensing, and autonomous driving are producing an unprecedented volume of data [17]. These computation-intensive applications play a substantial role in the explosive growth of data production, thereby exerting significant strain on the data transmission capacities of wireless IoT networks. Each of the devices in the IoT networks, however, has limitations in terms of computation resources [18]. They may be highly effective in their specific roles—sensing, actuating, and collecting data—but they lack the capacity to fully process and interpret it on their own. This is where offloading computational tasks to more powerful systems comes into play. Many IoT devices

rely on larger, more capable systems to process their data. This offloading process frees up resources for IoT devices to perform their core functions more efficiently.

In the case of connected cars, for example, the data they generate could be sent to a cloud-based system for detailed analysis, while the car itself focuses on sensing and actuating tasks. Once this data is offloaded, the cloud-based system usually with extensive computation resources could process tasks much faster than the car itself [19]. Though the task offloading process may introduce some transmission latency, the overall latency is notably reduced thanks to rapid computational solving. Besides, offloading tasks to a powerful remote server can enhance safety by enabling advanced processing capabilities, such as real-time analysis of data from multiple cars to detect and respond to potential hazards or providing access to more accurate and up-to-date mapping and traffic information. Traditional data processing techniques often fall short in handling such large quantities of data, especially in real-time. At this point, advanced computational methods, including machine learning, data analytics, and artificial intelligence, come into play [20]. Machine learning algorithms can identify patterns and learn from the data without explicit programming, making them highly effective at handling large data sets. By incorporating machine learning, the system can significantly enhance its real-time decision-making capabilities for task offloading, leading to improved efficiency and extended functionality.

Energy perspective: Most wireless IoT devices allow for maximum flexibility in their placement and usage [21]. This flexibility has facilitated the deployment of IoT devices in all sorts of environments, from homes and offices to factories. However, the wireless nature of these devices often means they rely on batteries for power [22]. As many IoT devices are small and portable [23], these batteries also need to be

compact, making the balance between battery size and lifespan a critical issue. This is where low-energy consumption becomes a vital characteristic of IoT devices. The low energy of these devices also contributes to the scalability of IoT systems. As the number of devices increases, the collective energy consumption could, however, become significant. Therefore, individual devices need to be as energy-efficient as possible to make large-scale deployments viable. To achieve this energy efficiency, IoT devices often use advanced, low-power computing components and communication protocols [24]. In summary, the combination of wireless connectivity, battery power, and low-energy operation is a critical trifecta in the design and deployment of IoT devices. These characteristics enable the flexibility, versatility, and scalability of IoT applications, making them a significant force in driving digital transformation across various sectors of society and the economy.

1.2 Research Problems

In the previous section, we presented an overview of the wireless IoT network, examining it from the perspectives of spectrum, computation, and energy. The potential and utilization of wireless IoT networks remain somewhat constrained by the effectiveness of various resource utilization strategies. In addition, advanced learning methodologies, including artificial intelligence and machine learning algorithms, have been implemented to revolutionize traditional communication networks. A pertinent question emerges when we consider the integration of IoT networks and learning techniques: how can we implement the latter within IoT networks to enhance their performance in the aspect of resource allocation? This question prompts us to delve

into various resource allocation strategies. In this thesis, we focus on resource allocation in wireless IoT systems, with a particular emphasis on addressing scarcity in three key resources. We elaborate on these research challenges below.

In Chapter 3, we investigate a task offloading scheme in a multi-layer mobile computing network including mobile devices (MDs), MEC servers (MSs), and a cloud server (CS). Most existing works [25–27] only consider vertical task offloading schemes where the task of each MD could be processed locally or offloaded to the MS, or further offloaded to the CS from the MS. Recently, [28] proposed a two-dimensional task offloading that included horizontal computation cooperation among the MSs. However, none of them take into consideration device-to-device (D2D) [29], which enables direct transmission between proximate devices without relaying information through a base station (BS). D2D communication plays an integral role in enhancing low-latency communications, primarily due to its high transmission efficiency. This efficiency becomes particularly beneficial when dealing with small-packet task exchanges, which are often encountered in massive machine-type communications. Consequently, the D2D cooperation in the multi-layer computing network proposed in Chapter 3 allows the scheme to adapt to various task sizes, harvesting the benefits of both edge/cloud servers for computation-intensive tasks and D2D cooperation for small-packet ones.

In Chapter 4, we explore a green cell-free massive MIMO in the IoT network. Cell-free massive MIMO can address degraded cell-edge performance by distributing a number of antennas over a large geographic area and fully eliminating inter-cell interference. Meanwhile, the widely distributed access points (APs) required more fronthaul links and more hardware-related power consumption, resulting in significantly

degraded energy efficiency. Most related works, such as [30–32] consider strategies to increase energy efficiency, primarily by optimizing the power allocation or AP selection. It is observed that some APs are potentially located far from user equipments (UEs), so it might not be necessary to activate all APs all the time. However, none of them simultaneously optimizes the power allocation and AP selection to maximize the energy efficiency in cell-free Massive MIMO.

In Chapter 5, we investigate a low-latency online task offloading scheme in a mobile computing network. NOMA allows multiple MDs to share the same sub-channel by multiplexing their tasks in the power domain, thereby increasing throughput compared to orthogonal multiple access (OMA). Therefore, NOMA in the mobile computing network has prompted a large number of studies in the past decade (see [33–35] and references therein). However, the cost is the introduction of interference between MDs, resulting in larger transmission latency. Only a few papers have considered fully utilizing the sub-channels while reducing transmission latency. From the network perspective, most related works, such as [36–38], only focused on the two-layer NOMA-based mobile computing network where each MD can process its task locally or offload it to its associated MS. Currently, a multi-layer mobile computing network has been proposed [39], and the tasks could be further offloaded to a computation-intensive CS. This multi-layer collaboration enables fast task processing for different computing requirements, thereby significantly reducing the computation delay. To the best of our knowledge, no one has ever considered the collaboration between an effective NOMA scheme and a multi-layer mobile computing network to achieve a low-latency online task offloading scheme.

1.3 Contributions

In Chapter 3 (Publication [J1]), we consider a multi-layer cooperative computing network by leveraging both D2D cooperation and horizontal edge cooperation. In this framework, a task can be either executed locally or offloaded to proximate MDs, collaborative MSs, or the CS. The joint task offloading and resource allocation can be formulated as an mixed-integer nonlinear programming (MINLP) problem. To obtain a global optimal solution, we develop a learning-based policy with much lower complexity. Our main contributions can be summarized as follows.

- We propose a full-dimensional task offloading scheme by jointly optimizing the task offloading and communication/computation resources and formulate it as an MINLP problem.

- We resort to imitation learning and propose a GIRL variable selection policy. This approach solves the challenge that the reward function cannot be appropriately designed as the branching order of variables cannot be known in advance. Without sacrificing the global optimality, our approach could achieve much lower complexity than the full strong branching (FSB).

- We develop a graph neural network (GNN) as a parameterized reward function in the GIRL, which directly handles the graphic features in the enumeration tree of the B&B algorithm. We further design an attention mechanism in the GNN and incorporate self-imitation into the GIRL. This results in a strong model generalization capability, empowering the GIRL to easily adapt to the variations of the key network parameters without learning from scratch.

- Simulations verify that the proposed GIRL can significantly accelerate the original B&B algorithm. It achieves a much lower computational complexity compared to the existing variable selection policies. Furthermore, our proposed full-dimensional task offloading scheme achieves better performance than the existing schemes in terms of average delay for all MDs.

In Chapter 4 (Publication [J2]), we develop a green energy scheme in downlink cell-free massive MIMO where a subset of APs can be deactivated to sleep mode, leading to reduced energy consumption without significantly impacting the quality of service. We propose a learning-based online power allocation and AP selection to minimize the total energy consumption. Our main contributions can be summarized as follows.

- We develop a green energy scheme in downlink cell-free massive MIMO by simultaneously optimizing power allocation and AP selection based on zero-forcing (ZF) beamforming.

- We propose a new optimization algorithm, OSAC-G, which embeds non-convex optimization into contemporary deep reinforcement learning (DRL) methods. OSAC-G enables fast online power allocation and AP selection with significantly lower computational complexity.

- We develop graph transformer networks (GTN) to empower OSAC-G to easily adapt to the variations of the class II parameters without learning from scratch.

- Simulation results confirm that the green energy scheme outperforms the existing ones in terms of power consumption.

In Chapter 5 (Publication [J3]), we aim to minimize the average delay across all MDs by jointly optimizing task offloading and resource allocation, and we formulate a non-convex optimization problem involving a number of discrete and continuous variables. We develop a learning-based algorithm to solve the formulated problem with very low complexity, and the constraints could be satisfied with a high probability. The main contributions of this chapter are summarized as follows:

- We develop a flexible approach, partial NOMA-based (PNOMA), by leveraging the advantages of high throughput of NOMA and low interference of OMA, resulting in low-latency transmission across various data sizes.

- We propose a PNOMA task offloading scheme in a multi-layer computing network, which enables fast task processing for different computing requirements, thereby significantly reducing the computation delay. We optimize the average delay across all MDs by jointly optimizing task offloading and resource allocation.

- We propose an RPPO algorithm taking advantage of the DRL algorithm and the reincarnating technique. RPPO can infer online the optimal task offloading and resource allocation with a very low complexity. It is adaptive to system parameter variations in the PNOMA-based multi-layer computing network.

- Simulation results indicate that the proposed scheme achieves a much lower average delay than the existing ones, and it is shown that RPPO could significantly outperform the conventional DRL.

1.4 Thesis Outline

The rest of this thesis is structured as follows: Chapter 2 provides an introduction to the theoretical concepts employed in the proposed schemes examined within this thesis. The main contributions of this thesis are comprehensively detailed across Chapters 3 to 5. In Chapter 3, we focus on inverse reinforcement learning based on full-dimensional task offloading in the computing network. In Chapter 4, we develop a green cell-free massive MIMO with an optimization-embedded deep reinforcement learning approach. In Chapter 5, we propose a partial NOMA-based online task offloading scheme in a multi-layer mobile edge computing network. Finally, Chapter 6 summarises this thesis and the major findings, followed by some concluding remarks and future research directions.

Chapter 2

Literature Review

2.1 Mobile Computing Network

The advent of the IoT has profoundly expanded the scope of mobile computing, creating a resilient network encompassing a diverse spectrum of devices—from everyday smartphones to high-capacity CSs [40]. This multifaceted mesh of interconnected systems facilitates seamless integration and communication between various devices, thereby enhancing efficiency and refining user experiences [41]. However, such a multifarious network is not devoid of challenges. Many tasks delegated to the mobile computing network struggle with the intrinsic constraints of MDs, such as limited computational capabilities, finite battery life, and inadequate memory storage [42]. In this section, we will conduct a comprehensive review of various ground-breaking network paradigms pertinent to task offloading. A subsequent analysis will shed light on their individual merits and shortcomings, providing a balanced perspective on this critical aspect of contemporary mobile computing.

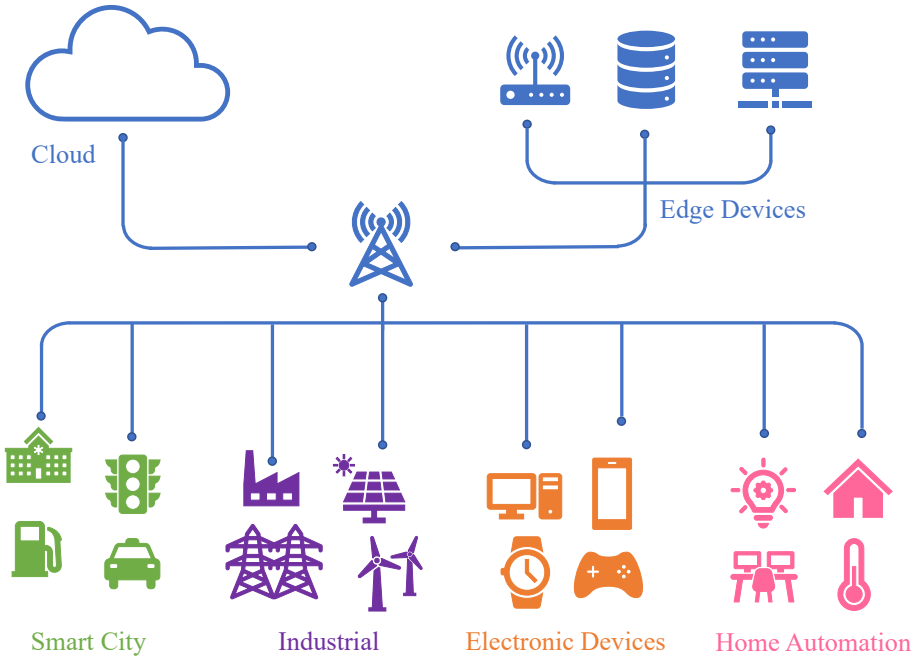


Figure 2.1: A diagram of the mobile computing network.

2.1.1 Mobile Cloud Computing

Mobile cloud computing (MCC) merges cloud computing with the ubiquity of MDs, enabling a powerful paradigm for modern network computing [43]. It leverages the immense processing power of CSs to augment the capabilities of MDs, especially those with inherent computational limitations [44]. Several cloud entities have been successful in transforming into viable commercial services. Notable companies such as Microsoft Azure, Google Cloud, and Amazon Web Services have pioneered offerings that allow mobile users to rent combined storage and computational platforms [45]. These platforms can run custom mobile applications, broadening the accessibility to high-powered computing resources. As a result, even devices with basic specifications are capable of running complex applications that demand substantial

computational power and storage. This is possible because the bulk of the processing tasks are offloaded to the cloud, allowing for more efficient utilization of mobile device resources. MCC offers numerous advantages over local task computing, including extended battery lifetime, enhanced computational capability, and improved data storage. Nevertheless, it is important to recognize the challenges, particularly in the context of task offloading. The remote location of the CS can introduce high transmission latency, potentially affecting the real-time responsiveness of applications and services [46].

2.1.2 Mobile Edge Computing

MEC has emerged as a transformative technique designed to enable delay-sensitive applications. It addresses critical latency and energy-efficiency demands in the modern network ecosystem by deploying MSs in close proximity to MDs. In a typical MEC architecture, MSs act as immediate computational resources for MDs, thereby facilitating task offloading. This proximate placement of MSs significantly reduces transmission delay and energy consumption, which are two paramount concerns in network efficiency. Consequently, this reduction not only alleviates traffic congestion but also enhances the quality of service (QoS) across the entire network. The concept of task offloading is not novel and has been the subject of numerous research efforts, as evidenced by the literature [47–49]. However, earlier work in this field has primarily concentrated on a method known as vertical task offloading. In this approach, tasks originating from an MD could only be offloaded to its directly associated MS or a CS. This focus on vertical relationships underscores the specialization of earlier techniques, highlighting a key aspect of MEC ongoing evolution.

In [50], the authors focus on a D2D-assisted vertical MEC system. They aimed to minimize the average response cost by developing a method that jointly optimizes the processes of offloading, transmission scheduling, and computation allocation. The integration of these aspects results in an efficient mechanism for handling computational tasks within the network. Similarly, the authors of [51] worked on maximizing the aggregate offloading benefits in vertical MEC systems, specifically exploring the trade-offs between delay and energy consumption. Their methodology is grounded on the optimization of D2D pairing within the system, presenting a solution that balances efficiency and performance. Another important contribution is found in [52], where the work is centered on minimizing computation latency in a three-layer vertical MEC network. The authors jointly optimize different layers with D2D cooperation, a strategy that results in a more responsive and streamlined computing network. Despite the significant advancements in these studies, a common limitation is that they primarily focus on the synergy between D2D communication and edge computing. They tend to overlook the potential computational capacity of collaborative MSs, an aspect that might be pivotal in further enhancing the efficiency of such systems.

2.2 Massive Multiple-input Multiple-output

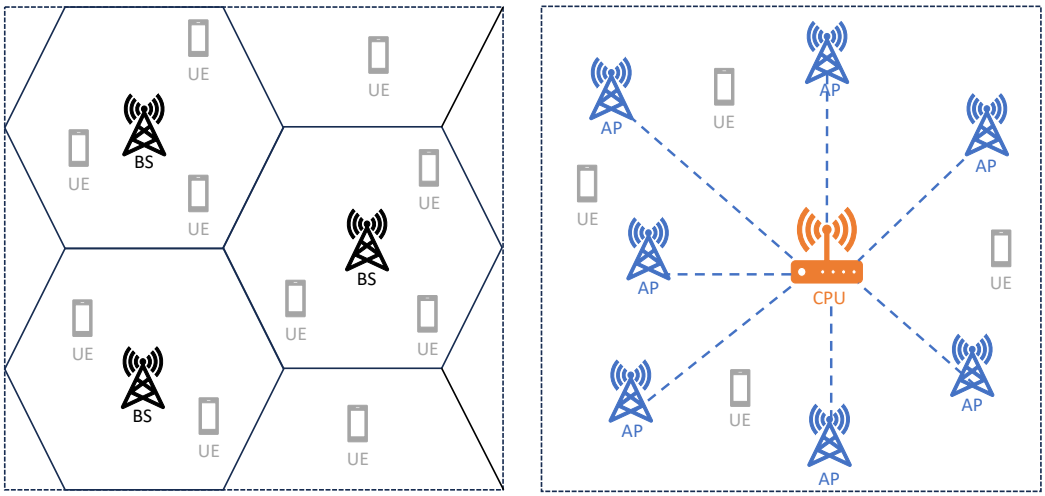
Massive MIMO technology represents a forefront innovation in the field of wireless communications, serving as an essential component in IoT networks [53]. These networks comprise a diverse array of devices that vary in size and function, including everything from miniature sensors to substantial appliances. All these devices are interlinked, enabling seamless communication and data exchange. In an era marked by a ceaseless rise in the demand for higher data rates, superior coverage, and minimal

latency, the integration of massive MIMO technology within wireless IoT networks has become not just relevant but increasingly vital. The deployment of this cutting-edge technology promises to enhance connectivity and promote efficiency, thereby playing a pivotal role in shaping the future of wireless communication.

Massive MIMO utilizes a large number of antennas at the BS to support a multitude of user devices. It allows the system to simultaneously communicate with multiple devices, greatly improving the spectral efficiency, data rate, and network capacity [54]. This makes it an excellent fit for IoT networks, where a myriad of devices need to be connected, often with varying data transmission requirements. Another key feature of massive MIMO in the context of IoT is its ability to enhance overall network reliability without requiring additional bandwidth or power. This makes massive MIMO a highly energy-efficient and cost-effective solution for improving wireless IoT communications, a significant advantage in IoT networks where devices often run on limited power supply. Massive MIMO technology comes in various types such as multi-cell and cell-free.

2.2.1 Multi-cell Massive MIMO

Multi-cell massive MIMO is a typical MIMO technology where BSs are equipped with a large number of antenna elements that are used to serve multiple users simultaneously [55]. The key premise of multi-cell massive MIMO is that by leveraging the spatial domain, the system can deliver better signal quality and more efficient use of the spectrum. In a multi-cell environment, this approach enables spatial multiplexing, allowing for significantly higher spectral and energy efficiency compared to traditional MIMO networks. Multi-cell Massive MIMO utilizes the full potential of



(a) (b)

Figure 2.2: A comparison between (a) Multi-cell massive MIMO and (b) Cell-free massive MIMO.

spatial multiplexing by coordinating the transmission and reception of signals among multiple cells. Moreover, this technology allows for beamforming [56], the process of focusing a wireless signal toward a specific device, which improves connection quality and reduces interference. By dynamically adjusting the phases and amplitudes of the transmitted signals, multi-cell massive MIMO enables precise beamforming, which results in superior wireless coverage and higher data rates.

While multi-cell massive MIMO offers substantial benefits, there are some drawbacks associated with its implementation in wireless IoT networks. The first challenge is inter-cell interference, a consequence of simultaneous communications across different cells. It can degrade the network performance by reducing signal quality, and hence the overall efficiency of the system. Second, the implementation of multi-cell massive MIMO involves complex signal processing algorithms to handle large-scale data and interference management. This adds to the computational burden and necessitates powerful and costly hardware. Finally, although multi-cell massive MIMO

is known to enhance energy efficiency, the energy consumption associated with the use of numerous antennas and the complex processing they require cannot be overlooked [57]. This becomes especially crucial in IoT networks, where devices often run on limited power resources. Despite these challenges, multi-cell massive MIMO continues to play a significant role in wireless IoT networks, with ongoing research aimed at overcoming these issues to further improve the potential of this transformative technology.

2.2.2 Cell-free Massive MIMO

Cell-free Massive MIMO represents a transformative evolution in wireless communication [58]. Conventional multi-cell massive MIMO segregates the network into different cell areas, each served by a BS. On this basis, a multi-cell MIMO system is limited by cell-centric design with inherent inter-cell interference. In contrast, cell-free Massive MIMO involves a large-scale deployment of APs scattered throughout the service area and serving the UE in a cooperative, cell-less fashion. This decentralized structure leverages the power of Massive MIMO, which uses a high number of antennas at each access point to enhance the network spatial multiplexing and beamforming capabilities.

In this case, the concepts of "cell" and "cell edges" are no longer relevant, and it mitigates issues like inter-cell interference and handover complexities [59]. Consequently, significantly increased spectral efficiency and system capacity can be achieved. Moreover, the distributed nature of APs in a cell-free system can provide a consistent and high-quality signal strength, improving the reliability and robustness of wireless

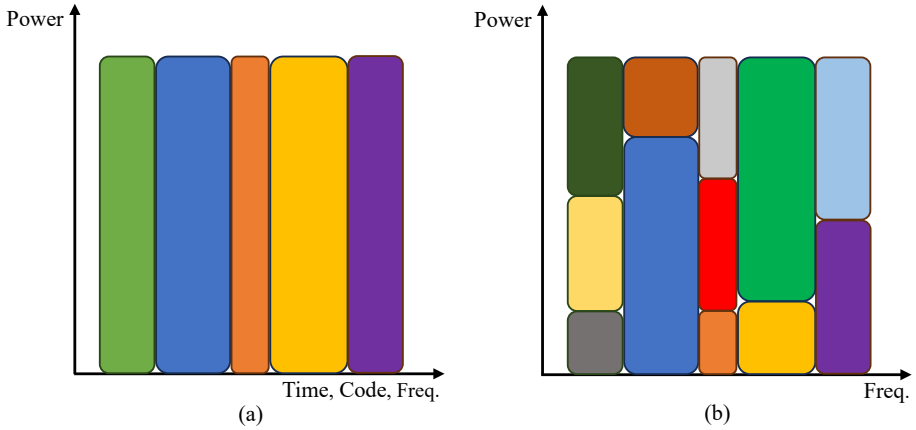


Figure 2.3: A comparison between (a) OMA and (b) NOMA.

networks [60]. Besides, some antennas are potentially closer to the UEs in the cell-free massive MIMO configuration, which yields a higher degree of macro-diversity and lower path losses.

For IoT networks, which typically feature numerous low-power devices spread over a large area, cell-free Massive MIMO offers a unique set of advantages. It allows the system to serve multiple devices simultaneously, increasing network capacity and energy efficiency – key factors in IoT applications. It also provides a more uniform service experience, an aspect that is crucial in scenarios where IoT devices need to communicate reliably.

2.3 Multiple Access

Multiple Access refers to the ability of multiple users to simultaneously share the same communication channel [61]. It is a fundamental concept in telecommunications that enables the efficient use of the available spectrum, making it possible for several users to communicate over the same frequency band.

2.3.1 Orthogonal Multiple Access

In the first generation (1G) mobile communication system, frequency-division multiple access (FDMA) was the predominant method, which allocated individual frequency bands to different users [62]. The second generation (2G) introduced time-division multiple access (TDMA) [63], where users were given alternating time slots on the same frequency channel, thereby allowing a more efficient utilization of bandwidth. Code-division multiple access (CDMA) [64] became a significant advancement in 2G and continued into the third generation (3G), as it used unique codes to separate users' communications on the same frequency, further improving capacity and reducing interference. The fourth generation (4G) adopted orthogonal frequency-division multiple access (OFDMA) [65], a method that subdivides a channel into multiple orthogonal sub-carriers, offering even more efficient frequency utilization and enabling higher data rates. Each of these access schemes has played a vital role in enhancing the capacity, efficiency, and functionality of their respective generations of cellular networks.

All these conventional multiple access schemes fall under the category of OMA technologies. In these systems, different users are assigned orthogonal resources in time, frequency, or code domains to reduce multiple access interference [66]. This helps maintain a stable connection for each user. However, OMA schemes have limitations, particularly when supporting massive connectivity with diverse QoS requirements. Due to system constraints, users with superior channel quality are given higher priority, while those with inferior channel quality must wait their turn. This disparity leads to significant unfairness and increased latency in the system. Furthermore, the allocation of a degree of freedom exclusively to users with poor channel quality is

an inefficient use of system resources. The situation can exacerbate inequalities in the network, resulting in sub-optimal performance and dissatisfaction among users. Strategies to enhance system efficiency and fairness should be explored to overcome these challenges.

2.3.2 Non-orthogonal Multiple Access

NOMA system is regarded as a crucial technology for fifth-generation (5G) and beyond [67] due to its high spectral efficiency, massive connectivity, low latency, and high user fairness [68]. NOMA improves spectrum utilization and user connectivity, pivotal in a world of growing data demands and increased device connectivity. As shown in Fig. 2.3, this innovative access method marks a significant departure from traditional OMA techniques that face scalability limitations as the number of devices in a network increases. Unlike conventional OMA schemes where users are separated in time, frequency, or code domain, NOMA serves multiple users in the same resource block. According to the domain of multiplexing, existing NOMA could be classified into two categories, i.e., code domain multiplexing and power domain multiplexing [69].

NOMA techniques within the code domain, including approaches like low-density spreading (LDS), sparse code multiple access (SCMA), and pattern division multiple access (PDMA) [70–72], use coding and spreading methods to introduce redundancy. This redundancy aids in the separation of users at the receiver, making the process more robust and efficient. In the code domain NOMA, superposition coding is used in the transmitter to send signals to multiple users over the same time and frequency

resource. This method employs superposition coding at the transmitter, which capitalizes on the different power levels to transmit signals concurrently. At the receiving end, successive interference cancellation (SIC) [73] is applied to mitigate inter-user interference, thus enhancing the overall performance of the system. Implementing code domain NOMA also has challenges. The complexity of the receiver increases due to the need for advanced signal processing techniques. Furthermore, error propagation in the signal decoding process can degrade the performance of weaker users. Power domain NOMA represents a more streamlined approach when compared to code domain NOMA, requiring only minor modifications in the physical layer operations at the transmitter side [74]. This simplicity is enhanced by the power domain NOMA's ability to facilitate more flexible resource allocation. Relaxing the orthogonality requirement can lead to significant improvements in various aspects of NOMA performance, including spectral efficiency, energy efficiency, and user fairness across the network.

In the realm of IoT, where countless devices demand simultaneous connectivity, NOMA can substantially improve both the system's capacity and spectral efficiency. This technology presents an intriguing solution for catering to the immense connectivity requirements of Machine-Type Communications (MTC) [75] and D2D communication [76] within the complex IoT landscape. Implementing NOMA in IoT networks is not without its hurdles. Challenges such as intricate user pairing [77], sophisticated power allocation [78], and the prerequisite for flawless channel state information (CSI) [79] must be carefully addressed. Despite these obstacles, the promise of NOMA is undiminished. It stands poised to transform the wireless IoT framework, paving the way for the comprehensive establishment of next-generation networks, including but

not limited to 5G technologies and beyond [80]. This innovation could mark a significant milestone in achieving the seamless integration and efficiency that the future of communication demands.

Chapter 3

Inverse Reinforcement Learning with Graph Neural Networks for Full-Dimensional Task Offloading in Edge Computing

In this chapter, we consider an efficient task offloading strategy to meet the growing demand for scarce communication and network resources, a need generated by ubiquitous IoT applications. To tackle this problem, we incorporate D2D communication into the multi-layer computing network and propose a full-dimensional task offloading scheme by jointly optimizing task offloading decisions and computation/communication resource allocation. We formulate it as an MINLP problem, where the optimal B&B algorithm features an extremely high complexity. To address this challenge, we propose GIRL to generate a new variable selection policy. Without sacrificing the global optimality, the GIRL can directly infer the variable selection with a much lower complexity, significantly accelerating the original B&B algorithm. Simulation results show that the GIRL achieves a lower complexity without sacrificing global optimality. Furthermore, our proposed full-dimensional task offloading scheme achieves better performance than the existing schemes.

3.1 Introduction

3.1.1 Multi-layer Mobile Computing Network

The ubiquitous IoT is envisioned to connect billions of mobile sensors, manufacturing machines, smart devices, industrial utilities, etc., entailing a high demand for scarce communication and network resources [81]. Furthermore, many emerging IoT-enabled computation-intensive applications, such as face recognition, participatory sensing, and autonomous driving [82] [83], generate an unprecedented volume of data for pattern recognition. However, MDs are usually limited by their computation resources and battery life. Featured by abundant computation and storage resources, MCC [84] facilitates the handling of computing-intensive tasks via task offloading from MDs. However, due to the remote location of the CS, the task offloading will inevitably introduce high transmission latency [46].

As a transformative technique, MEC [8] is proposed to enable delay-sensitive applications. Typically, MSs are deployed much closer to MDs for task offloading, which significantly reduces transmission delay and energy consumption. This helps decrease traffic congestion and improve the QoS of the whole network. Many task offloading schemes have been proposed in the literature. Earlier work focuses on vertical task offloading, where tasks from an MD can only be offloaded to its associated MS or CS. The authors in [85] consider a multi-user multi-layer mobile computing network (MD-MS-CS), and aim to minimize the energy cost, computation, and delay for all MDs. In [27], the authors strive to minimize the energy consumption for delay-constrained applications, where the task offloading and resource allocation are jointly optimized in a hierarchical network architecture. The authors in [86] propose a

greedy optimization approach to minimize communication costs in an MEC network where low computing but high communication capabilities are needed. Recently, some works [28] [87] have focused on a two-dimensional task offloading scheme where horizontal cooperation between MSs is considered in a multi-layer network. In [28], the authors propose an alternating direction method of multipliers (ADMM) method to minimize the average task delay in a multi-layer MEC network with vertical and horizontal edge computing cooperation. The authors in [87] propose a Gaussian process imitation learning (GPIL) method to minimize the average task delay subject to communication/computation resources and energy consumption constraints.

The rapid emergence of new IoT applications such as augmented reality (AR) and mobile healthcare [88] introduce a large number of tasks with a relatively small packet size. In this case, MDs need to frequently offload their tasks to MSs, resulting in increased latency and degraded energy efficiency due to significant wireless communication overhead between MDs and MSs. Recently, D2D communication has been proposed as a promising technology in 5G and beyond networks [76], where the core idea is to enable direct transmission between proximate devices, without relaying information through the BS. Therefore, D2D can enable low-latency communications with a significantly increased transmission efficiency, especially suitable for small-packet task exchanges in massive machine-type communications [89].

In this chapter, we incorporate D2D communication into the multi-layer computing network and propose a new full-dimensional task offloading scheme by jointly optimizing task offloading decisions and communication/computation resources. In the presence of both D2D cooperation and horizontal edge cooperation, a task can be either executed locally or offloaded to proximate MDs, collaborative MSs, or the

cloud server. In this case, this scheme can be adaptive to various task sizes, harvesting the benefits of both edge/cloud servers for computation-intensive tasks and D2D cooperation for small-packet ones. Mathematically, we formulate the full-dimensional task offloading into an MINLP problem, which involves mutually coupled discrete (task offloading decisions) and continuous variables (communication and computation resource allocation). However, an MINLP problem is usually NP-hard and no optimal solution can be found by a polynomial-time algorithm. This calls for efficient strategies that scale favorably with the MINLP problem size.

3.1.2 Solutions to MINLP Problems

Several contemporary methods are available to potentially solve the MINLP problems. The branch-and-bound (B&B) algorithm [90] is a global exhaustive search method that achieves a global optimal solution and provides the performance upper bound, but its computational complexity is prohibitively high with a large number of discrete variables. Consequently, some heuristic optimization algorithms with a low complexity were proposed in the literature. The typical ones in [91] include genetic algorithm (GA), particle swarm optimization algorithm (PSO), and relaxation-based algorithm. These algorithms are fast in solving MINLP problems. However, they are sub-optimal due to their heuristic nature, and the performance gap with the optimal B&B algorithm is difficult to quantify.

In addition, many related works are focused on optimization-based methods to solve optimization problems in multi-layer computing networks. In [92], authors jointly optimize task partitioning and user association in a multi-layer computing network. The formulated problem is divided into two subproblems, where the first one

is solved directly under a given user association. The second one is solved by a dual decomposition-based method. In [93], a long-term problem is formulated in a two-layer computing network. It is solved by a regularization and rounding-based method including two sequential phases, where the long-term problem is decomposed series of one-shot fractional problems in the first phase, and the fractional solution is rounded to an integer solution by a dependent rounding scheme in the second phase. In [94], an enders-decomposition-based method is proposed to solve the resource allocation problem, and a significant number of iterations are involved rendering the method of significant complexity. However, these three optimization-based decomposition methods are developed for specific formulated problems in a multi-layer computing network, and cannot be directly applied to solve our MINLP problem. Besides, these methods involve a large number of iterations and suffer significant complexity.

Reinforcement learning (RL) [95], which involves an agent making observations and taking actions within an environment to receive rewards, offers a potential avenue to the MINLP problem. However, the MINLP problem combines discrete and continuous variables, which makes it difficult for RL to directly handle a large mixed action space. Furthermore, the MINLP problem involves a number of constraints, and RL cannot guarantee the strict satisfaction of these constraints. Besides, many related works are focused on ML-based methods in multi-layer computing networks. The authors in [96] propose a deep neural network (DNN) based method to minimize the energy consumption of all the user equipment in the edge computing network. In [97], authors propose a DRL-based online offloading algorithm to maximize the sum computation rate in a two-layer computing network. The authors in [98] utilize a DRL method to optimize resource scheduling aiming to minimize the task latency

of the tasks. However, the above methods decompose the original MINLP problem into a sub-problem with discrete variables addressed by the ML-based method, and the other sub-problem with continuous variables addressed by classical optimization algorithms. The optimality might not be guaranteed due to the decomposition (i.e. not joint optimization).

As the B&B algorithm can offer a global optimal solution to the MINLP problem but at the cost of usually unacceptable complexity, in this chapter, we develop a new approach to significantly accelerate the B&B algorithm. Basically, the B&B has three fundamental policies, namely, node pruning, node selection, and variable selection policies [90]. Node selection and pruning policies determine which node is pruned or preserved in the enumeration tree. Consequently, the major efforts in accelerating the B&B algorithm are placed on improving the efficiency of these two policies by various heuristic algorithms. On the other hand, it is worth noting that the variable selection policy determines the order of the variables for branching, also having a significant impact on the size of the enumeration tree. The FSB policy [99], as the optimal variable selection policy of the original B&B algorithm, can achieve a minimum size of an enumeration tree. However, the associated variable selection process is prohibitively tedious, especially for a large number of variables. This hinders its implementation for our multi-layer full-dimensional task offloading scheme. Under the umbrella of the B&B algorithm, we leverage the learning mechanism to generate a new variable selection policy referred to as inverse reinforcement learning with graph neural networks (GIRL), aiming to achieve a comparable performance but with a significantly lower complexity relative to the FSB.

3.1.3 Organization

The rest of this chapter is organized as follows. We first provide system model and formulate an MINLP problem in Section 3.2. Then, we introduce the B&B algorithm in Section 3.3. In Section 3.4, we elaborate on the design of the proposed GIRL variable selection policy for accelerating the B&B algorithm. Furthermore, we develop an attention GNN as the parameterized reward function in Section 3.5. On this basis, in Section 3.6, we propose the self-imitation to enhance the model generalization capability, which is followed by performance optimization and complexity analysis. We provide the simulation results in Section 3.7. Finally, we give the conclusions in Section 3.8.

3.2 System Model

3.2.1 System Model

As shown in Fig. 3.1, we consider a multi-layer mobile computing network including N MDs indexed by the set $\mathcal{N} = \{1, 2, ..., N\}$, M MSs indexed by the set $\mathcal{M} = \{1, 2, ..., M\}$ and a CS. The D2D communication between MDs and the communication between MD and its associated MS are through a wireless network. The inter-connected communication between MSs is through a wired network, and each MS is connected to the CS via a backhaul wired network.

In our three-level full-dimensional cooperation architecture, MD $i \in \mathcal{N}$ has a computation task $Q_i = \{D_i, C_i\}$, where D_i is the data size and C_i is the required computation resources (in CPU cycles). Specifically, Q_i can be processed locally (Level 0 local computing) or by an adjacent peer MD $j \in \hat{\mathcal{N}}_i = \{\mathcal{N} \setminus i\}$ (Level 0 D2D

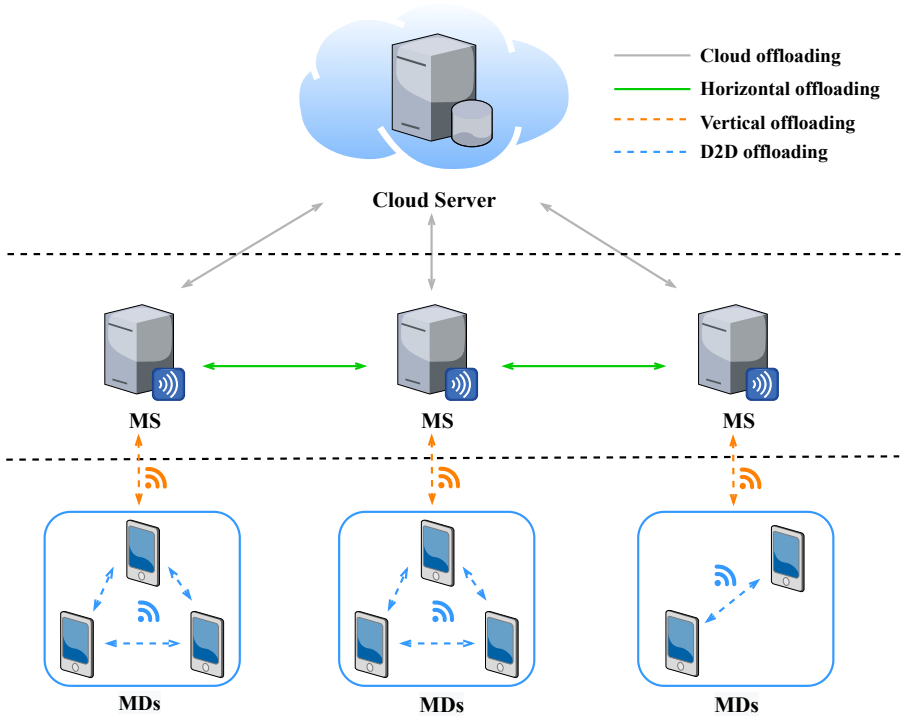


Figure 3.1: Multi-layer mobile computing network with full-dimensional task offloading scheme.

cooperation computing) when MD j has extra computing resources. Alternatively, the task Q_i can be offloaded to the associated MS $l_i \in \mathcal{M}$. From here, the task can be either processed locally (Level 1 vertical edge computing), or forwarded to the other free MSs $k \in \hat{\mathcal{M}}_i = \{\mathcal{M} \setminus l_i\}$ (Level 1 horizontal edge computing). In addition, the task could be offloaded to the CS for processing (Level 2 cloud computing). In summary, the task offloading binary decision could be classified as the following five indicators

- Level 0 local computing indicator $x_{i,i} \in \{0, 1\}$: task Q_i is processed by the MD i when $x_{i,i} = 1$.

- Level 0 D2D cooperation computing indicator $x_{i,j} \in \{0, 1\}$: task Q_i is processed

by the peer MD j when $x_{i,j} = 1$.

- Level 1 vertical edge computing indicator $y_{i,l_i} \in \{0, 1\}$: task Q_i is vertically offloaded to the associate MS l_i for processing when $y_{i,l_i} = 1$.
- Level 1 horizontal edge computing indicator $y_{i,k} \in \{0, 1\}$: task Q_i is forwarded to the MS k by MS l_i horizontally for processing when $y_{i,k} = 1$.
- Level 2 cloud computing indicator $z_i \in \{0, 1\}$: task Q_i is offloaded to the CS for processing when $z_i = 1$.

Overall, the offloading decision for the task Q_i could be constrained by

$$x_{i,i} + \sum_{j \in \hat{\mathcal{N}}_i} x_{i,j} + \sum_{k \in \mathcal{M}} y_{i,k} + z_i = 1. \tag{3.2.1}$$

3.2.2 Communication Model

In this chapter, the communication model consists of the wireless link and the wired link. The communication of level 0 D2D cooperation computing and level 1 vertical edge computing is through the wireless link. For level 0 D2D cooperation, the data rate between MD i and the peer MD j could be calculated as

$$r_{i,j} = b_i \log \left(1 + \frac{p_i g_{i,j}}{\sum_{i' \in \hat{\mathcal{N}}_i} p_{i'} g_{i',j} + \sigma_2} \right), \tag{3.2.2}$$

where b_i is the allocated bandwidth of MD i, p_i is the transmission power, $g_{i,j}$ is the channel gain between MD i and the peer MD j, and σ^2 is the power of the additive white Gaussian noise. Here, $\sum_{i' \in \hat{\mathcal{N}}_i} p_{i'} g_{i',j}$ represents the interference caused by D2D transmission.

Similarly, for the level 1 vertical transmission, the uplink data rate between the

MD i and associated MS l_i could be calculated as

$$r_{i,l_i} = b_i \log\left(1 + \frac{p_i g_{i,l_i}}{\sum_{i' \in \mathcal{U}_{l_i}, i' \neq i} p_{i'} g_{i',l_i} + \sigma_2}\right), \tag{3.2.3}$$

where $\mathcal{U}_{l_i}$ is the set of MDs associated to MS l_i, and $\sum_{i' \in \mathcal{U}_{l_i}, i' \neq i} p_{i'} g_{i',l_i}$ represents the interference from MDs that offload resources to MS l_i. The total bandwidth for MS k should not exceed its maximum bandwidth capacity $B_k^{\max}$, so we have the constraint

$$\sum_{i \in \mathcal{U}_k} b_i \leq B_k^{\max}, \tag{3.2.4}$$

where $\mathcal{U}_k$ is the set of MDs associated to MS k. With the data rate, the transmission delay could be calculated as

$$d_{i,(\cdot)}^{\mathrm{tr}} = \frac{D_i}{r_{i,(\cdot)}},\ (\cdot) \in \{j, l_i\}, \tag{3.2.5}$$

and the corresponding energy consumption could be obtained as

$$e_{i,(\cdot)}^{\mathrm{tr}} = d_{i,(\cdot)}^{\mathrm{tr}} p_i^{\mathrm{tr}},\ (\cdot) \in \{j, l_i\}. \tag{3.2.6}$$

For level 1 horizontal edge computing, task Q_i is transmitted from MS l_i to MS $k \in \hat{\mathcal{M}}_i$ via the wired link, commonly established through an optical fiber connection with a notably high link capacity. This round-trip delay is denoted as $d_{i,k}^{\mathrm{rrt}}$ which could be assumed to be fixed and estimated with the historical data beforehand [100]. For level 2 cloud computing, task Q_i would be offloaded from MS l_i to the CS via the backhaul wired link, and this transmission delay is denoted as d^{bc}.

3.2.3 Computation Model

Level 0 Local Computation with D2D Cooperation

In Level 0, task Q_i could be processed locally or by a peer MD with D2D cooperation. For local computing, the required computation delay for Q_i could be obtained

as

$$d_i^{\mathrm{L}} = d_i^{\mathrm{comp}} = \frac{C_i}{f_i^{\mathrm{L}}}, \tag{3.2.7}$$

where f_i^{L} is the computation capability of the local MD i (in CPU frequency). The corresponding energy consumption for processing Q_i could be obtained as

$$e_i^{\mathrm{L}} = d_i^{\mathrm{L}} p_i^{\mathrm{L}}, \tag{3.2.8}$$

where p_i^{L} is the computation power of the local MD i (in watt).

For the D2D cooperation computing, task Q_i is transferred from MD i to the peer MD $j \in \hat{\mathcal{N}}_i$ for processing, and the total delay d_i^{D} is composed of the computation delay d_j^{comp} and the transmission delay $d_{i,j}^{\mathrm{tr}}$, and we have

$$d_i^{\mathrm{D}} = d_{i,j}^{\mathrm{tr}} + d_j^{\mathrm{comp}}. \tag{3.2.9}$$

Also, the energy consumption for this D2D cooperation could be calculated as

$$e_i^{\mathrm{D}} = d_{i,j}^{\mathrm{tr}} p_i^{\mathrm{tr}} + d_j^{\mathrm{comp}} p_j^{\mathrm{L}}. \tag{3.2.10}$$

Level 1 Edge Computation with Horizontal Edge Cooperation

In Level 1, task Q_i is vertically offloaded to its associated MS l_i, and then it can be processed by MS l_i itself or other MS $k \in \hat{\mathcal{M}}_i$ with horizontal edge cooperation. For vertical edge computing, the required computation delay for Q_i in MS l_i could be calculated as

$$d_{i,l_i}^{\mathrm{comp}} = \frac{C_i}{f_{i,l_i}}, \tag{3.2.11}$$

where f_{i,l_i} is the computation capability of MS l_i for task Q_i. And the total delay d_{i,l_i}^{V} for vertical edge computing consists of the transmission delay d_{i,l_i}^{tr} between MD i and MS l_i and the computation delay $d_{i,l_i}^{\mathrm{comp}}$. Combining (3.2.11) and Section 3.2.2, we can obtain

$$d_{i,l_i}^{\mathrm{V}} = d_{i,l_i}^{\mathrm{tr}} + d_{i,l_i}^{\mathrm{comp}}. \tag{3.2.12}$$

Similarly, for the horizontal edge computing between MS l_i and the MS K, the total delay $d_{i,k}^{\mathrm{H}}$ is composed of the computation delay for task Q_i in MS k ($d_{i,k}^{\mathrm{comp}} = \frac{C_i}{f_{i,k}}$), the transmission delay d_{i,l_i}^{tr} between from MD i to MS l_i, and the round-trip delay $d_{l_i,k}^{\mathrm{rrt}}$ between MSs l_i and k, and we have

$$d_{i,k}^{\mathrm{H}} = d_{i,l_i}^{\mathrm{tr}} + d_{l_i,k}^{\mathrm{rrt}} + d_{i,k}^{\mathrm{comp}}. \tag{3.2.13}$$

Besides, for MS k, the sum of the computation resource allocated to all MDs in $\mathcal{N}$ should not be more than its total computational capacity $f_k^{\max}$, and we have

$$\sum_{i \in \mathcal{N}} f_{i,k} \leq f_k^{\max}, \quad k \in \mathcal{M}. \tag{3.2.14}$$

Level 2 Cloud Computation

In Level 2, task Q_i is first transmitted to the associated MS l_i, and then it is transmitted to the CS. The required computation delay by the CS could be obtained as

$$d_{i,c}^{\mathrm{comp}} = \frac{C_i}{f_{i,c}}, \tag{3.2.15}$$

where $f_{i,c}$ is the computation capability of the CS, which is a predetermined value according to the cloud computing service [84]. And the CS has a sufficiently large capacity than those of MSs, so we have $f_{i,c} \gg f_{i,k}$. Here, the total delay for the cloud computation is composed of the transmission delay d_{i,l_i}^{tr}, the backhaul delay d^{BC}, and the computation delay $d_{i,c}^{\mathrm{comp}}$, and we have

$$d_{i,c}^{\mathrm{C}} = d_{i,l_i}^{\mathrm{tr}} + d^{\mathrm{BC}} + d_{i,c}^{\mathrm{comp}}. \tag{3.2.16}$$

3.2.4 Problem Formulation

Here, we formulate the joint task offloading and resource allocation problem[1] to minimize the average delay for all MDs subject to communication, computation, and energy constraints. Accordingly, the full-dimensional task offloading optimization problem is formulated as

$$\begin{aligned}
\boldsymbol{P}: \min_{\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{f},\mathbf{b}} & \sum_{i\in\mathcal{N}}\Big(x_{i,i}d_i^{\text{comp}}+\sum_{j\in\hat{\mathcal{N}}_i}x_{i,j}(d_{i,j}^{\text{tr}}+d_j^{\text{comp}})+y_{i,l_i}(d_{i,l_i}^{rmtr}+d_{i,l_i}^{\text{comp}}) \\
& +\sum_{k\in\hat{\mathcal{M}}_i}y_{i,k}(d_{i,l_i}^{\text{tr}}+d_{l_i,k}^{\text{rrt}}+d_{i,k}^{\text{comp}})+z_i(d_{i,l_i}^{\text{tr}}+d^{\text{BC}}+d_{i,c}^{\text{comp}})\Big), && (3.2.17)\\
\text{s.t.} \quad & x_{i,i}d_i^{\text{comp}}+\sum_{j\in\hat{\mathcal{N}}_i}x_{i,j}d_{i,j}^{\text{tr}}p_i^{\text{tr}}+\Big(1-\sum_{j\in\mathcal{N}}x_{i,j}\Big)d_{i,l_i}^{\text{tr}}p_i^{\text{tr}}\le e_i^{\max}, && (3.2.17a)\\
& x_{i,i}+\sum_{j\in\hat{\mathcal{N}}_i}x_{i,j}+\sum_{k\in\mathcal{M}}y_{i,k}+z_i=1, && (3.2.17b)\\
& \sum_{i\in\mathcal{U}_k}b_i\le B_k^{\max},\ b_i>0, && (3.2.17c)\\
& \sum_{i\in\mathcal{N}}f_{i,k}\le f_k^{\max},\ f_{i,k}>0,\ k\in\mathcal{M}, && (3.2.17d)\\
& \mathbf{x},\mathbf{y},\mathbf{z}\in\{0,1\}, && (3.2.17e)
\end{aligned}$$

where $\mathbf{x}=[x_{1,1},...,x_{1,N},...,x_{N,1},...,x_{N,N}]^T$, $\mathbf{y}=[y_{1,1},...,y_{1,M},...,y_{N,1},...,y_{N,M}]^T$, $\mathbf{z}=[z_1,...,z_N]^T$, $\mathbf{f}=[f_{1,1},...,f_{1,M},...,f_{N,1},...,f_{N,M}]^T$, and $\mathbf{b}=[b_1,...,b_N]^T$. For convenience, we denote the discrete task offloading as $\boldsymbol{\alpha}=[\mathbf{x}^T,\mathbf{y}^T,\mathbf{z}^T]^T$, and resource allocation as $\boldsymbol{\beta}=[\mathbf{f}^T,\mathbf{b}^T]^T$. The objective function $\boldsymbol{O}(\boldsymbol{\alpha},\boldsymbol{\beta})$ and $\boldsymbol{C}(\boldsymbol{\alpha},\boldsymbol{\beta})$ in (3.2.17) represents the total delay for all MDs subject to constraints $\boldsymbol{C}(\boldsymbol{\alpha},\boldsymbol{\beta})$ in (3.2.17a)-(3.2.17e). In practice, a centralized controller executes the optimization algorithm to solve $\boldsymbol{P}$. The controller is much closer to MSs than to the CS benefiting from low

[1]Note that a single objective (average latency) is considered in this chapter. Our algorithm could also be adopted to solve the multi-objective problem by transforming it into a single objective [101].

latency transmission via high-speed wired links. The controller requires two types of information: system information (e.g., local computation capability f_i^{L}, local energy capability $e_i^{\max}$, MS computation capability $B_k^{\max}$, and CS computation capability $f_{i,k}$) and task-related information (e.g., data size D_i, channel gain between MD and the associated MS g_{i,l_i}, and channel gains between MDs $\{g_{i,j} \mid j = 1, \cdots, N\}$). The controller has pre-stored the system information, so it only needs to collect task-related information, typically in kilobytes (KB), which leads to a very low latency of probing. Therefore, the latency of probing is not considered in $\boldsymbol{P}$.

The problem $\boldsymbol{P}$ is an MINLP with discrete variables $\boldsymbol{\alpha}$ and continuous variables $\boldsymbol{\beta}$. In fact, $\boldsymbol{P}$ consists of $2^{N(N+M+1)}$ sub-problems. In other words, its computational complexity increases exponentially with the dimension of $\boldsymbol{\alpha}$. As one of the state-of-the-art algorithms, B&B can find a global optimal solution to $\boldsymbol{P}$, and it can fathom sub-problems with relaxations to control the exponential nature of the search. Besides, due to constraint (3.2.17b), the complexity of the original B&B can be further reduced. For example, if $x_{i,j} = 1$, naturally we have $y_{i,j} = 0$ and $z_i = 0$. Therefore, in the worst case, the complexity has been reduced to solving $2^{N(M+1)}$ sub-problems. Despite the complexity reduction, it is still of exponential nature. Consequently, no optimal solution could be found by a polynomial-time algorithm. Therefore, our optimization problem $\boldsymbol{P}$ is NP-hard. To address this challenge, we translate our formulated $\boldsymbol{P}$ to the optimization problem $\boldsymbol{P}'$ which is suitable to be solved with the B&B algorithm. Specifically, we can translate $\boldsymbol{P}$ into

$$\boldsymbol{P'}: \min_{\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{f},\mathbf{b}} \sum_{i\in\mathcal{N}} \Big(\sum_{j\in\mathcal{N}} x_{i,j}^2 \Big(\frac{D_i}{r_{i,j}} + \frac{C_i}{f_i^{\mathrm{L}}} \Big) + y_{i,l_i}^2 \Big(\frac{D_i}{r_{i,l_i}} + \frac{C_i}{f_{i,l_i}} \Big) + \sum_{k\in\hat{\mathcal{M}}_i} \Big(y_{i,k}^2 \Big(\frac{D_i}{r_{i,l_i}} + \frac{C_i}{f_{i,k}} \Big) + y_{i,k} d_{l_i,k}^{\mathrm{rrt}} \Big) + z_i^2 \Big(\frac{D_i}{r_{i,l_i}} + \frac{C_i}{f_{i,c}} \Big) + z_i d^{\mathrm{BC}} \Big), \quad (3.2.18)$$

$$\text{s.t.} \quad x_{i,i}^2 \frac{C_i p_i^{\mathrm{L}}}{f_i^{\mathrm{L}}} + \sum_{j\in\mathcal{N}} x_{i,j}^2 \frac{D_i p_i^{\mathrm{tr}}}{r_{i,j}} + \Big(y_{i,l_i}^2 + \sum_{k\in\hat{\mathcal{M}}_i} y_{i,k}^2 + z_i^2 \Big) \frac{D_i p_i^{\mathrm{tr}}}{r_{i,l_i}} \le e_i^{\max}, \quad (3.2.18a)$$

$$\sum_{j\in\mathcal{N}} x_{i,j} + \sum_{k\in\mathcal{M}} y_{i,k} + z_i = 1, \quad (3.2.18b)$$

$$\sum_{i\in\mathcal{U}_k} b_i \le B_k^{\max},\ b_i > 0, \quad (3.2.18c)$$

$$\sum_{i\in\mathcal{N}} f_{i,k} \le f_k^{\max},\ f_{i,k} > 0,\ k \in \mathcal{M}, \quad (3.2.18d)$$

$$\mathbf{x}, \mathbf{y}, \mathbf{z} \in \{0, 1\}, \quad (3.2.18e)$$

where we assume that $r_{i,i} = \infty$ ($d_{i,i}^{\mathrm{tr}} = 0$), and the local computing could be a special D2D cooperation. For the binary variables $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$, we implement the quadratic expression $\mathbf{x}^2$, $\mathbf{y}^2$ and $\mathbf{z}^2$ to replace them. As a result, the form like $x_{i,j}\frac{C_i}{f_i^{\mathrm{L}}}$ is translated to a convex function $x_{i,j}^2\frac{C_i}{f_i^{\mathrm{L}}}$. When binary variables $\boldsymbol{\alpha}$ are relaxed, $\boldsymbol{P'}$ is a convex optimization problem. Though we can obtain the optimal solution of $\boldsymbol{P'}$ with the B&B algorithm, there are $2^{N(M+1)}$ sub-problems to be solved in the worst case and the computational complexity is extremely high. This calls for efficient strategies that scale favorably with the problem size.

3.3 Branch-And-Bound Algorithm

In this section, we introduce the B&B algorithm, an optimal approach to solving combinatorial optimization MINLP problems, and provide an example to illustrate the process of the B&B algorithm. Then, we introduce the variable selection policy

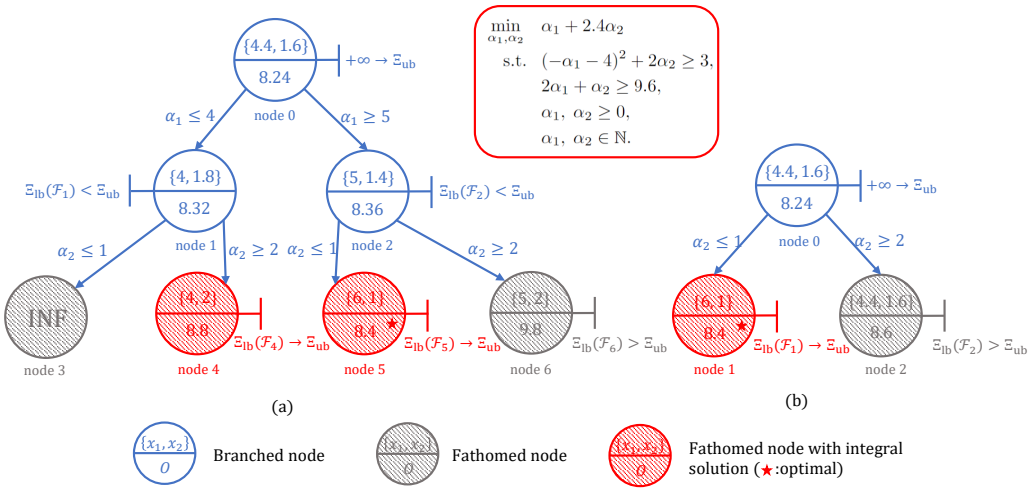


Figure 3.2: An example of the B&B algorithm. (a) The variable α_1 is first selected to branch. (b) The variable α_2 is first selected to branch.

of the B&B algorithm.

3.3.1 Workflow of the Branch-and-Bound Algorithm

The B&B algorithm, combining a diverse mixture of heuristics, is capable of solving reasonably sized MINLP problems within an acceptable time duration while guaranteeing optimality. The key idea is to sequentially partition the original MINLP problem into smaller problems with relaxations to control the exponential nature of the search. The B&B algorithm determines the discrete variables via a recursive method, The B&B algorithm could be represented as an enumeration tree composed of multiple edges and nodes (e.g., the enumeration tree shown in Fig. 3.2(a)). The nodes in the enumeration tree consist of a root node (e.g., node 0) and multiple subtree nodes (e.g., nodes 1-6). The root node comes from the relaxed original MINLP problem by relaxing all the discrete variables to continuous ones, and it corresponds to a convex nonlinear programming problem (NLP), which can be solved by many

convex optimization algorithms such as the interior-point method [102]. The enumeration tree starts at the root node, and then sub-tree nodes are added sequentially until the optimal solution is found. The construction complies with three policies: node selection policy, node pruning policy, and variable selection policy all to be elaborated later.

Node selection policy determines which node to process, and is commonly based on search methods, such as depth-first search and best-first search [90]. Assume that the l-th node first pops out from the unbranched node list, denoted by $\mathscr{L}_{\mathrm{n}}$, based on a node selection policy. Here, the general MINLP problem can be expressed by

$$\min_{\boldsymbol{\alpha},\boldsymbol{\beta}} \boldsymbol{O}(\boldsymbol{\alpha},\boldsymbol{\beta}) \quad \text{s.t. } (\boldsymbol{\alpha},\boldsymbol{\beta}) \in \mathcal{F}_l, \tag{3.3.1}$$

where $\mathcal{F}_l$ represents the feasible set of the problem at l-th node. The optimal solution of (3.3.1) is denoted as $\{\boldsymbol{\alpha}^*(l), \boldsymbol{\beta}^*(l)\}$, and its corresponding optimal objective value is denoted as $\Xi^*(\mathcal{F}_l)$. We further denote the lower bound of $\Xi^*(\mathcal{F}_l)$ as $\Xi_{\mathrm{lb}}(\mathcal{F}_l)$. Meanwhile, we denote the upper bound of $\Xi^*(\mathcal{F}_l)$ as Ξ_{ub} that is updated iteratively. Clearly, we have

$$\Xi_{\mathrm{lb}}(\mathcal{F}_l) \leq \Xi^*(\mathcal{F}_l) \leq \Xi_{\mathrm{ub}}. \tag{3.3.2}$$

Next, we elaborate on the node selection, node pruning, as well as the rule to update Ξ_{ub}. Initially, Ξ_{ub} is assumed to be infinity. Then, the pruning policy would determine whether this node will be fathomed if one of the following three conditions is met. (1) (3.3.1) is infeasible. (2) $\Xi_{\mathrm{lb}}(\mathcal{F}_l) > \Xi_{\mathrm{ub}}$, which means that all solutions of child nodes below cannot be better than Ξ_{ub} in this branch. (3) $\Xi_{\mathrm{lb}}(\mathcal{F}_l) < \Xi_{\mathrm{ub}}$ and all variables in $\boldsymbol{\alpha}^*(l)$ are integers, which means that $\boldsymbol{\alpha}^*(l)$ is the optimal solution in this enumeration tree. In the meantime, the upper bound is updated as $\Xi_{\mathrm{lb}}(\mathcal{F}_l) \to \Xi_{\mathrm{ub}}$. If the l-th node is reserved, there will be some variables that need to be branched,

and they constitute an unbranched variable list $\mathscr{L}_{\mathrm{v}}$.

The next step in the B&B algorithm is the variable selection process. In this process, we need to determine which fractional variable in $\mathscr{L}_{\mathrm{v}}$ should be selected for further branching. The variable α_n is chosen based on the variable selection policy to be elaborated in Section 3.3.2. The optimal value of α_n, denoted as $\alpha_n^{(l)}$, can be found in $\boldsymbol{\alpha}^*(l)$. Given $\alpha_n^{(l)}$, the feasible set $\mathcal{F}_l$ is partition into two subsets

$$\begin{cases} \mathcal{F}_l^- &= \mathcal{F}_l \cap \{(\boldsymbol{\alpha}, \boldsymbol{\beta}) \mid \alpha_n \leq \lfloor \alpha_n^{(l)} \rfloor\} \\ \mathcal{F}_l^+ &= \mathcal{F}_l \cap \{(\boldsymbol{\alpha}, \boldsymbol{\beta}) \mid \alpha_n \geq \lceil \alpha_n^{(l)} \rceil\}. \end{cases} \tag{3.3.3}$$

Accordingly, (3.3.1) is divided into two sub-problems

$$\begin{aligned} &\min_{\boldsymbol{\alpha},\boldsymbol{\beta}} \boldsymbol{O}(\boldsymbol{\alpha}, \boldsymbol{\beta}) \quad \text{s.t.} (\boldsymbol{\alpha}, \boldsymbol{\beta}) \in \mathcal{F}_l^-, \\ &\min_{\boldsymbol{\alpha},\boldsymbol{\beta}} \boldsymbol{O}(\boldsymbol{\alpha}, \boldsymbol{\beta}) \quad \text{s.t.} (\boldsymbol{\alpha}, \boldsymbol{\beta}) \in \mathcal{F}_l^+. \end{aligned} \tag{3.3.4}$$

These two sub-problems are assigned to the next two child nodes which then will be added to the list $\mathscr{L}_{\mathrm{n}}$. The enumeration tree grows after every iteration until there are no nodes left in the list $\mathscr{L}_{\mathrm{n}}$.

Next, we provide an example in Fig. 3.2(a) to illustrate the process aforementioned.

- We start from the root node 0. The relaxed problem corresponds to $\mathcal{F}_0$ where α_1 and α_2 are relaxed to the continuous variables. We have $\alpha_1^{(0)} = 4.4$, $\alpha_2^{(0)} = 1.6$, and $\boldsymbol{\Xi}^*(\mathcal{F}_0) = 8.24$. Assuming that α_1 is selected for further branching, we obtain two child nodes 1 and 2. For node 1, we have $\alpha_1^{(1)} = 4$, $\alpha_2^{(1)} = 1.8$, and $\boldsymbol{\Xi}^*(\mathcal{F}_1) = 8.32$. As $\boldsymbol{\Xi}_{\mathrm{lb}}(\mathcal{F}_1) = \boldsymbol{\Xi}^*(\mathcal{F}_1) < \boldsymbol{\Xi}_{\mathrm{ub}}$, and $\alpha_2^{(1)}$ is fractional, node 1 will be branched. Similarly, node 2 will also be branched.

- For node 1, as only α_2 is left in $\mathscr{L}_{\mathrm{v}}$, we branch it, resulting in nodes 3 and 4. For node 3, the corresponding problem is infeasible, so this node is fathomed.

For node 4, the relaxed solutions are $\alpha_1^{(4)} = 4$, $\alpha_2^{(4)} = 2$, and $\boldsymbol{\Xi}^*(\mathcal{F}_4) = 8.8$. As $\alpha_1^{(4)}$ and $\alpha_2^{(4)}$ are integers while $\boldsymbol{\Xi}_{\rm lb}(\mathcal{F}_4) = \boldsymbol{\Xi}^*(\mathcal{F}_4) < \boldsymbol{\Xi}_{\rm ub}$, we set $\boldsymbol{\Xi}_{\rm ub} = 8.8$. Then, node 4 is fathomed.

- For node 2, node 5 is first selected to branch. Clearly, $\alpha_1^{(5)}$ and $\alpha_2^{(5)}$ are integers and $\boldsymbol{\Xi}_{\rm lb}(\mathcal{F}_5) = \boldsymbol{\Xi}^*(\mathcal{F}_5) = 8.4 < \boldsymbol{\Xi}_{\rm ub} = 8.8$, thus optimal node shifts from node 4 to 5. Then only node 6 is left in $\mathscr{L}_{\rm n}$, where $\boldsymbol{\Xi}_{\rm lb}(\mathcal{F}_6) > \boldsymbol{\Xi}_{\rm ub}$, meaning that the last node has been fathomed, and the B&B process has finished. Thereby we reach the conclusion that the global optimal node is node 5 and the optimal solution is 8.4.

Remark 3.3.1. *For a relaxed MINLP problem $\boldsymbol{P}$, the B&B algorithm can not only handle convex $\boldsymbol{P}$, but also non-convex $\boldsymbol{P}$. The reason why we use a convex $\boldsymbol{P}$ in this chapter is that we can easily resort to highly efficient convex optimization algorithms to solve each convex node (sub-problem) in the enumeration tree of the B&B algorithm with the global optimality. For a non-convex $\boldsymbol{P}$, we need to resort to some sub-optimal algorithms to solve each non-convex node. For example, successive convex approximation (SCA) [103] is an efficient algorithm to handle a non-convex node. Accordingly, the bound to need to be modified by tightening bounds. However, the global optimality might not be guaranteed due to the nature of non-convex problems.*

3.3.2 Variable Selection Policy

To improve the basic B&B routine, efforts in the literature have been placed on more efficient node selection and pruning. However, the variable selection has been less considered, although it significantly impacts the computational complexity of the B&B algorithm. For example, in Fig. 3.2(a), if α_1 is selected first, there will be 7

nodes attached in the tree $(0, ..., 6)$, in stark contrast with only 3 nodes in Fig. 3.2(b) if α_2 is selected first. This illustrates how important the order of selection is, not to mention the case with more variables.

In essence, the variable selection policy has a direct impact on the size of the enumeration tree. The FSB known as the optimal variable selection policy has the highest node creation efficiency. Compared to any other policy, it is able to find the minimum size of an enumeration tree. The FSB works as follows. In node l, the variable α_n with the best $\mathbf{score}_n^{(l)*}$ is selected to branch, where $\mathbf{score}_n^{(l)*} = \max\{\mathbf{score}_n^{(l)} \mid a_n \in \mathscr{L}_{\mathrm{v}}\}$. Here, the score function of the FSB is given by

$$\mathbf{score}_n^{(l)} = (1-\mu) \cdot \min\{\Delta_n^{(l)-}, \Delta_n^{(l)+}\} + \mu \cdot \max\{\Delta_n^{(l)-}, \Delta_n^{(l)+}\}, \tag{3.3.5}$$

where $0 \leq \mu \leq 1$ is the score factor that can be adjusted dynamically. In addition, we have $\Delta_n^{(l)-} = \boldsymbol{\Xi}^*(\mathcal{F}_{l+1}) - \boldsymbol{\Xi}^*(\mathcal{F}_l)$ and $\Delta_n^{(l)+} = \boldsymbol{\Xi}^*(\mathcal{F}_{l+2}) - \boldsymbol{\Xi}^*(\mathcal{F}_l)$. Clearly, $\boldsymbol{\Xi}^*(\mathcal{F}_{l+1})$ and $\boldsymbol{\Xi}^*(\mathcal{F}_{l+2})$ could be obtained from two NLPs with new constraints $\alpha_n \leq \lfloor \alpha_n^{(l)} \rfloor$ and $\alpha_n \geq \lceil \alpha_n^{(l)} \rceil$, respectively. For example, in Fig. 3.2(a), child nodes of node 0 are nodes 1 and 2, where $\boldsymbol{\Xi}^*(\mathcal{F}_0) = 8.24$, $\boldsymbol{\Xi}^*(\mathcal{F}_1) = 8.32$ and $\boldsymbol{\Xi}^*(\mathcal{F}_2) = 8.36$. If we choose $\mu = 1/6$, we have the score $\mathbf{score}_1^{(1)} = 0.11$ for α_1. In Fig. 3.2(b), we can calculate the score $\mathbf{score}_2^{(1)} = 0.33$ for α_2, therefore selecting α_2 for branching in this problem can achieve a smaller enumeration tree and a lower computation complexity compared with selecting α_1 for branching.

Reliability pseudo-cost branching policy (RPB)[99] is a representative low-complexity variable selection method compared to FSB. Essentially, it is a hybrid branching method that uses strong branching for variable selection at the initialization and pseudo-cost branching for the rest variables.

Remark 3.3.2. *Note that the variable selection policy determines the order of the*

variables for branching, and does not involve the optimal node pruning. Therefore, the B&B algorithm with various variable selection policies (including FSB) all can get the global optimal solution (with different complexity).

3.4 The Proposed GIRL Variable Selection Policy

In this section, we first provide our motivation to accelerate the optimal B&B algorithm with significantly reduced complexity but without sacrificing the global optimality. Then, we build the variable selection policy as an Markov Decision Process (MDP)[104]. Finally, we introduce the framework of the GIRL variable selection policy, including Q-learning with modeling learning and the maximum entropy (MaxEnt) gradients.

3.4.1 Motivation

It is clear that, although the FSB can achieve the minimum size of the enumeration tree and the lowest computational complexity of the node solving process (NLPs solving), it needs to solve two NLPs for each candidate in $\mathscr{L}_{\mathrm{v}}$ of one node. Therefore, the variable selection process of the FSB is extremely costly, especially for a large number of variables. This hinders the implementation of the FSB for our multi-layer full-dimensional task offloading scheme. To address this problem, we propose a novel data-driven variable selection policy referred to as the GIRL. The essence of our approach lies in the exploitation of the learning mechanism to generate a new variable selection policy that closely matches the FSB variable selection, but without involving the tedious computations in the FSB.

Table 3.1: TABLE OF SYMBOLS

Variable	Description
$\boldsymbol{\alpha}$, $\boldsymbol{\beta}$	The discrete and the continuous variable sets
$\boldsymbol{C}(\boldsymbol{\alpha}, \boldsymbol{\beta})$	The constraints with $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$
$\boldsymbol{O}(\boldsymbol{\alpha}, \boldsymbol{\beta})$	The objective function with $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$
$\mathcal{F}_l$	The feasible set of the problem at l-th node
$\boldsymbol{\Xi}^*(\mathcal{F}_l)$	The optimal objective value at l-th node
$\boldsymbol{\Xi}_{\text{lb}}(\mathcal{F}_l)$	The lower bound of $\boldsymbol{\Xi}^*(\mathcal{F}_l)$
$\boldsymbol{\Xi}_{\text{ub}}$	The upper bound of $\boldsymbol{\Xi}^*(\mathcal{F}_l)$
$\mathscr{L}_{\text{n}}$, $\mathscr{L}_{\text{v}}$	Unbranched node and variable lists
α_n	The n-th variable in $\mathscr{L}_{\text{v}}$
$\alpha_n^{(l)}$	The optimal solution of α_n at l-th node
ς	The root node of the enumeration tree
τ_ς	The trajectory starting from the root node ς
Ω_ς	The trajectory space of τ_ς
$\mathcal{D}$	The expert demonstration
$\mathcal{D}_\varsigma$	The expert demonstration with ς
$\mathcal{D}'$	The self-imitation demonstration
γ, λ	The discount rate and the learning rate
π_t, π_E	The policy of the GIRL and the expert policy

3.4.2 Markov Decision Process

The B&B algorithm is a sequential decision process with an episodic variable selection. In this case, we formulate this process as an MDP from a model-free perspective. Here, we take the variable selector and the sub-problem solver as the agent and the environment, respectively. The major components of the MDP are summarized as follows.

State and Action: The state $\mathbf{s}_t$ is related to the entire already solved enumeration tree at t-th iteration, and the state space is $\mathcal{S} = \{\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, ...\}$ consisting of variable, constraint and edge features to be specified in Section 3.5.1. Based on $\mathbf{s}_t$, the action a_t selects a variable from $\mathscr{L}_{\mathrm{v}}$, and the action space could be represented as $\mathcal{A} = \{a_1, a_2, a_3...\}$. After a_t is taken, the current problem would be branched, and $\mathbf{s}_t$ transits to $\mathbf{s}_{t+1}$ with the state transition probability $\mathbb{P}(\mathbf{s}_{t+1} \mid \mathbf{s}_t, a_t)$.

Trajectory: The trajectory from the root node to the final fathomed node is represented as $\tau = \{(\mathbf{s}_t, a_t) \mid t = 1, 2, 3, \ldots\}$, capturing the sequence of state-action pairs. When referring to the trajectory starting from the root node ς, it is denoted as τ_ς. The trajectory space of τ_ς is denoted by $\Omega_\varsigma = \{\tau_\varsigma^1, \tau_\varsigma^2, \tau_\varsigma^3...\}$.

Reward and Policy: We denote the immediate reward as $R(\mathbf{s}_t, a_t)$ for the state-action pair $(\mathbf{s}_t, a_t)$, which tells the agent whether the action a_t is good or not. Given the state $\mathbf{s}_t$, policy $\pi : (\mathcal{S} \to \mathcal{A})$ maps the state to the action.

3.4.3 The Framework of the GIRL

It is well known that the RL can handle a general MDP in an online manner with a low computational complexity. Typically, given a reward function, the RL attempts to learn an optimal policy to map the state to the action. However, the RL

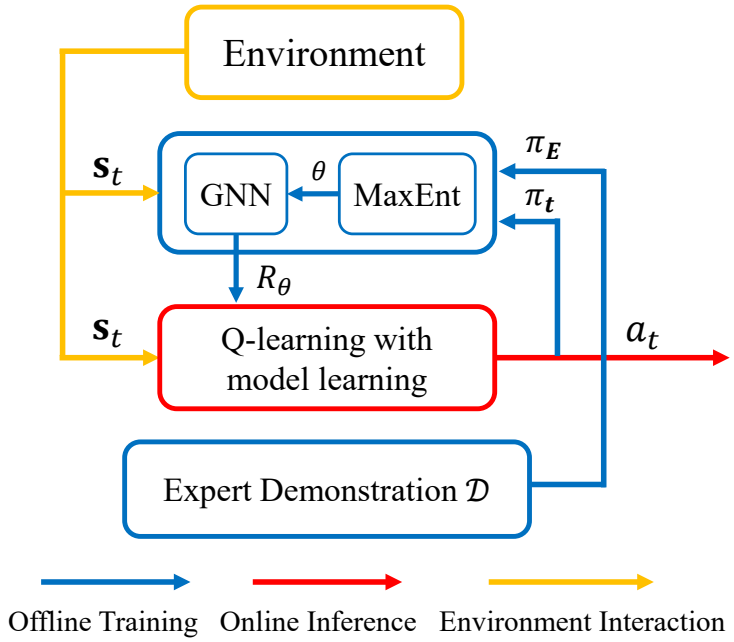


Figure 3.3: The framework of the GIRL.

is inapplicable to the formulated MDP, because in our problem it is very challenging to design an appropriate reward function. Before obtaining the optimal solution, the branching order of variables cannot be known in advance. That is, we cannot know that branching which variable can generate a smaller enumeration tree. For example, in Fig. 3.2, before we get the optimal solution, we can not decide to branch variable α_1 or α_2 in advance.

To address this challenge, we resort to imitation learning[2] and design an IRL-based framework as shown in Fig. 3.3. Instead of trying to manually specify a reward function, we design a GNN $\mathcal{G}$ with neural network parameters $\boldsymbol{\theta}$ as the parameterized

[2]Imitation learning [105] can be categorized into behavior cloning and inverse reinforcement learning (IRL). Behavior cloning involves learning a mapping from inputs (states) to outputs (actions). Clearly, it does not work for our full-dimensional task offloading scheme. Instead, our GIRL algorithm, based on IRL, learns a reward function that leads to a good policy aligned with the expert demonstration.

reward function, which is denoted as $R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)$. Subsequently, $R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)$ is integrated into the formulated MDP described in Section 3.4.2, ultimately yielding a policy that aligns with that of the expert demonstration $\mathcal{D}$ to be elaborated in Section 3.6.2.

The workflow of the proposed GIRL is summarized in **Algorithm 1**. Specifically, after initializing the network parameters $\boldsymbol{\theta}$, each iteration can be divided into three phases: 1) updating reward function $R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)$ via the GNN $\mathcal{G}$ (Line 3); 2) updating the policy (Lines 4-7); 3) updating $\boldsymbol{\theta}$ based on the expert demonstration $\mathcal{D}$ (Lines 8-15). In the first phase, we first update the GNN $\mathcal{G}$ based on $\boldsymbol{\theta}$, and then the GNN $\mathcal{G}$ will generate a new $R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)$. In the second phase, we update the policy π_t and obtain $\mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t)$ via the Q-learning with the model learning, and π_t is implemented to calculate the expected state-action visitation counts $\mathbb{E}[\mu_\varsigma(\mathbf{s}_t, a_t)]$ (Lines 5-7). In the final phase, $\mu_{\mathcal{D}_\varsigma}(\mathbf{s}_t, a_t)$ is calculated based on the expert demonstration $\mathcal{D}_\varsigma$, and then MaxEnt gradients $\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)}$ are obtained. We can obtain $\frac{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)}{\partial \boldsymbol{\theta}}$ by backward propagating the GNN $\mathcal{G}$. Next, $\boldsymbol{\theta}$ is updated based on the network gradients $\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$. When $\boldsymbol{\theta}$ converges to the optimal value $\boldsymbol{\theta}^*$, the GNN $\mathcal{G}$ could obtain the optimal reward function $R_{\boldsymbol{\theta}^*}(\mathbf{s}_t, a_t)$, and the optimal policy could be trained based on $R_{\boldsymbol{\theta}^*}(\mathbf{s}_t, a_t)$.

Q-Learning with Modeling Learning

Q-learning with model learning not only infers the action a_t given the state $\mathbf{s}_t$, but also learns a model by generating the transition probability $\mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t)$. In the proposed GIRL, $\mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t)$ plays a crucial role as it serves as input to the MaxEnt for updating the parameterized reward function (GNN $\mathcal{G}$).

Next, we elaborate on Q-Learning with modeling learning. According to [106],

Algorithm 1 Inverse Reinforcement Learning with Graph Neural Networks.

Input: Expert policy $\mu_{\mathcal{D}_\varsigma}$

Initialize $\boldsymbol{\theta}$.
while $\boldsymbol{\theta}$ not Converge **do**
    Update reward function $R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)$ via the GNN $\mathcal{G}$.
    Update policy π_t and obtain $\mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t)$ via Q-learning with model learning.
    for ς in $\mathcal{S}$ **do**
        Calculate the expected state-action visitation counts $\mathbb{E}[\mu_\varsigma(\mathbf{s}_t, a_t)]$ based on the policy π_t.
    end for
    Determine MaxEnt gradients
    $\dfrac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)} \leftarrow \sum_{\varsigma \in \mathcal{S}} \Big(\mu_{\mathcal{D}_\varsigma}(\mathbf{s}_t, a_t) - \mathbb{E}[\mu_\varsigma(\mathbf{s}_t, a_t)] \Big)$.
    Obtain $\dfrac{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)}{\partial \boldsymbol{\theta}}$ backward propagating neural-network.
    Compute network gradients
    $\dfrac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \leftarrow \sum_{\mathbf{s}_t \in \mathcal{S}} \sum_{a_t \in \mathcal{A}} \dfrac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)} \cdot \dfrac{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)}{\partial \boldsymbol{\theta}}$.
    Update neural-network parameters $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$.
end while
Return the optimal $\boldsymbol{\theta}^*$.
Obtain the optimal reward function $R_{\boldsymbol{\theta}^*}(\mathbf{s}_t, a_t)$ via the GNN $\mathcal{G}$.
Obtain policy π_t^* via Q-learning with model learning.

Output: Optimal policy π_t^*

following policy π, the Bellman equation could be expressed as

$$V^\pi(\mathbf{s}_t) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k} \mid \mathbf{s}_t, \pi\right] = \mathbb{E}\left[R_t + \gamma V^\pi(\mathbf{s}_{t+1}) \mid \mathbf{s}_t, \pi\right], \tag{3.4.1}$$

where

$$R_t = R(\mathbf{s}_t, a_t) = \sum_{\mathbf{s}_{t+1} \in \mathcal{S}} \mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t) R(\mathbf{s}_t, a_t, \mathbf{s}_{t+1}), \tag{3.4.2}$$

and $\gamma \in [0, 1)$ is the discount rate. Here, $V^\pi(\mathbf{s}_t)$ equivalently represents the expected discounted cumulative reward starting at state $\mathbf{s}_t$. Next, following policy π, we formulate the state–action value as

$$Q^\pi(\mathbf{s}_t, a_t) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k} \mid \mathbf{s}_t, a_t, \pi\right] = \mathbb{E}\left[R_t + \gamma Q^\pi(\mathbf{s}_{t+1}, a_{t+1}) \mid \mathbf{s}_t, a_t, \pi\right]. \tag{3.4.3}$$

Compared with $V^{\pi}(\mathbf{s}_t)$, $Q^{\pi}(\mathbf{s}_t, a_t)$ equivalently represents the expected discounted cumulative reward starting at state $\mathbf{s}_t$ after taking action a_t. Substituting (3.4.3) into (3.4.1), the Bellman equation could be written as

$$V^{\pi}(\mathbf{s}_t) = \sum_{a_t \in \mathcal{A}} \pi(\mathbf{s}_t, a_t) Q^{\pi}(\mathbf{s}_t, a_t). \tag{3.4.4}$$

The objective of the Q-learning with model learning is to learn the optimal policy π^*. In our approach, π^* could be obtained when $V^{\pi}(\mathbf{s}_t)$ converges to maximum, or, equivalently, when $Q^{\pi}(\mathbf{s}_t, a_t)$ converges to maximum. Mathematically, we have

$$\pi^* = \arg\max_{\pi} Q^{\pi}(\mathbf{s}_t, a_t). \tag{3.4.5}$$

Clearly, π^* corresponds to the optimal state–action value $Q^*(\mathbf{s}_t, a_t)$, which satisfies the Bellman optimality equation [107] as follows

$$Q^*(\mathbf{s}_t, a_t) = \mathbb{E}\left[R_t + \gamma \max_{a_{t+1} \in \mathcal{A}} Q^*(\mathbf{s}_{t+1}, a_{t+1}) \mid \mathbf{s}_t, a_t, \pi \right]. \tag{3.4.6}$$

To obtain the optimal state–action value $Q^*(\mathbf{s}_t, a_t)$, we should follow the iterative process based on the updated law

$$Q^{\pi}(\mathbf{s}_t, a_t) \leftarrow Q^{\pi}(\mathbf{s}_t, a_t) + \lambda\big(R_t + \gamma \max_{a_{t+1} \in \mathcal{A}} Q^{\pi}(\mathbf{s}_{t+1}, a_{t+1}) - Q^{\pi}(\mathbf{s}_t, a_t)\big), \tag{3.4.7}$$

where λ is the learning rate. By iteratively updating the Q-values using (3.4.7), the Q-learning progressively approaches $Q^*(\mathbf{s}_t, a_t)$, which maximizes the expected cumulative reward for every state-action pair. Consequently, the expected state-action visitation counts $\mathbb{E}[\mu_{\varsigma}(\mathbf{s}_t, a_t)]$ could be generated based on $Q^*(\mathbf{s}_t, a_t)$ according to [108]. Besides, the transition probability $\mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t)$ could be estimated by the following equation

$$\mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t) = \frac{\upsilon(\mathbf{s}_t, a_t, \mathbf{s}_{t+1})}{\sum_{\mathbf{s}_{t+1} \in \mathcal{S}} \upsilon(\mathbf{s}_t, a_t, \mathbf{s}_{t+1})}, \tag{3.4.8}$$

where $\upsilon(\mathbf{s}_t, a_t, \mathbf{s}_{t+1})$ is the number of state transition from $\mathbf{s}_t$ to $\mathbf{s}_{t+1}$.

MaxEnt Gradients

In this step, we present our approach to update $\boldsymbol{\theta}$ in the GNN $\mathcal{G}$ by leveraging MaxEnt gradients [108]. We first introduce some definitions as follows. The expert demonstration $\mathcal{D}$ is grouped into $\mathcal{D}_\varsigma$, and we have $\mathcal{D} = \{\mathcal{D}_\varsigma \mid \varsigma \in \mathcal{S}\}$. Specifically, $\mathcal{D}_\varsigma = \{\tau_\varsigma^i \mid i = 1, 2, 3, \ldots, N_\varsigma\}$ consists of all trajectories starting from the root node ς and ending at the optimal node, where N_ς is the number of trajectories starting from ς. With these definitions, then we will maximize the entropy of $\boldsymbol{\theta}$ subject to the expert demonstration.

The reward function $R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)$ indicates whether a_t follows the branching strategy of the FSB. It quantifies the extent to which the actions in the expert demonstration align with the prescribed FSB strategy. On this basis, $R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)$ is correlated to $\mathbb{P}(a_t|\boldsymbol{\theta})$, the probability of the action a_t in the expert demonstration, thus, we have $R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t) \propto \mathbb{P}(a_t|\boldsymbol{\theta})$. In addition, $\mathbb{P}(a_t|\boldsymbol{\theta})$ is equivalent to the probability of the trajectory τ with a_t in the expert demonstration, and we have $\mathbb{P}(a_t|\boldsymbol{\theta}) \propto \sum_{\tau:a_t\in\tau} \mathbb{P}(\tau|\boldsymbol{\theta})$. Under the principle of the MaxEnt, $\mathbb{P}(\tau|\boldsymbol{\theta})$ could be approximated as

$$\mathbb{P}(\tau|\boldsymbol{\theta}) \approx \frac{1}{Z_\varsigma(\boldsymbol{\theta})} e^{\sum_\tau R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)} \prod_{(\mathbf{s}_t, a_t, \mathbf{s}_{t+1})\in\tau} \mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t), \tag{3.4.9}$$

where $Z_\varsigma(\boldsymbol{\theta}) = \sum_{\tau\in\Omega_\varsigma} e^{\sum_{(\mathbf{s}_t,a_t)\in\tau} R_{\boldsymbol{\theta}}(\mathbf{s}_t,a_t)}$ is the partition distribution function. Based on $\mathbb{P}(\tau|\boldsymbol{\theta})$, the average log-likelihood of $\boldsymbol{\theta}$ over $\mathcal{D}_\varsigma$ could be represented as

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= \sum_{\varsigma\in\mathcal{S}} \frac{1}{N_\varsigma} \sum_{\tau\in\mathcal{D}_\varsigma} \log \mathbb{P}(\tau|\boldsymbol{\theta}) \\ &= \sum_{\varsigma\in\mathcal{S}} \Big(\frac{1}{N_\varsigma} \Big(\sum_{\tau\in\mathcal{D}_\varsigma} \Big(\sum_{(\mathbf{s}_t,a_t)\in\tau} R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t) + \sum_{(\mathbf{s}_t,a_t,\mathbf{s}_{t+1})\in\tau} \mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t) \Big) \Big) - \log Z_\varsigma(\boldsymbol{\theta}) \Big).\end{aligned} \tag{3.4.10}$$

Note that in this context $-\mathcal{L}(\boldsymbol{\theta})$ represents cross-entropy. The minimum cross-entropy could be referred to as the MaxEnt [109]. Under the MaxEnt, we can obtain the

optimal parameters $\boldsymbol{\theta}$ by maximizing $\mathcal{L}(\boldsymbol{\theta})$, given by

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}). \tag{3.4.11}$$

To achieve the maximum $\mathcal{L}(\boldsymbol{\theta})$, we take the partial derivative of (3.4.10) with respect to $\boldsymbol{\theta}$, and we have

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \sum_{\mathbf{s}_t \in \mathcal{S}} \sum_{a_t \in \mathcal{A}} \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)} \cdot \frac{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)}{\partial \boldsymbol{\theta}}. \tag{3.4.12}$$

Here, $\frac{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)}{\partial \boldsymbol{\theta}}$ could be obtained via the back propagating the GNN $\mathcal{G}$. In this case, we need to derive $\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)}$, which can be written as

$$\begin{aligned}
\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)} &= \sum_{\varsigma \in \mathcal{S}} \frac{1}{N_\varsigma} \Big(\frac{\partial \sum_{\tau \in \mathcal{D}_\varsigma} \sum_{(\mathbf{s}_t, a_t) \in \tau} R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)}{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)} \\
&\quad - \frac{1}{Z_\varsigma(\boldsymbol{\theta})} \sum_{\tau \in \Omega_\varsigma} e^{\sum_{(\mathbf{s}_t, a_t) \in \tau} R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)} \prod_{(\mathbf{s}_t, a_t, \mathbf{s}_{t+1}) \in \tau} \mathbb{P}(\mathbf{s}_{t+1} | \mathbf{s}_t, a_t) \times \frac{\partial \sum_{(\mathbf{s}_t, a_t) \in \tau} R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)}{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)} \Big) \\
&= \sum_{\varsigma \in \mathcal{S}} \frac{1}{N_\varsigma} \Big(\frac{\partial \sum_{\tau \in \mathcal{D}_\varsigma} \sum_{(\mathbf{s}\mathbf{s}_t, a_t) \in \tau} R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)}{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)} - \sum_{\tau \in \Omega_\varsigma} \mathbb{P}(\tau | \boldsymbol{\theta}) \frac{\partial \sum_{(\mathbf{s}_t, a_t) \in \tau} R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)}{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)} \Big) \\
&= \sum_{\varsigma \in \mathcal{S}} \Big(\mu_{\mathcal{D}_\varsigma}(\mathbf{s}_t, a_t) - \mathbb{E}[\mu_\varsigma(\mathbf{s}_t, a_t)] \Big) \triangleq \pi_E - \pi_t.
\end{aligned} \tag{3.4.13}$$

Furthermore, we define $\pi_E \triangleq \mu_{\mathcal{D}_\varsigma} \triangleq \sum_{\varsigma \in \mathcal{S}} (\mu_{\mathcal{D}_\varsigma}(\mathbf{s}_t, a_t))$ as the expert policy which has the same behavior as the demonstration $\mathcal{D}$, and define $\pi_t \triangleq \sum_{\varsigma \in \mathcal{S}} (\mathbb{E}[\mu_\varsigma(\mathbf{s}_t, a_t)])$ as the GIRL policy. Then parameters $\boldsymbol{\theta}$ could be updated based on $\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$. The update process of $\boldsymbol{\theta}$ is simplified as

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta \boldsymbol{\theta}, \tag{3.4.14}$$

which is elaborated later in Section 3.6.2. At the end of the iterations, the optimal $\boldsymbol{\theta}^*$ would be finally obtained. Taking the optimal $\boldsymbol{\theta}^*$ back into the GNN $\mathcal{G}$, the optimal reward function $R_{\boldsymbol{\theta}^*}(\mathbf{s}_t, a_t)$ can be generated. Finally, the optimal policy could be trained based on $R_{\boldsymbol{\theta}^*}(\mathbf{s}_t, a_t)$. After the offline training, the learned GIRL could directly infer the variable selection and achieve a global optimal solution (c.f.

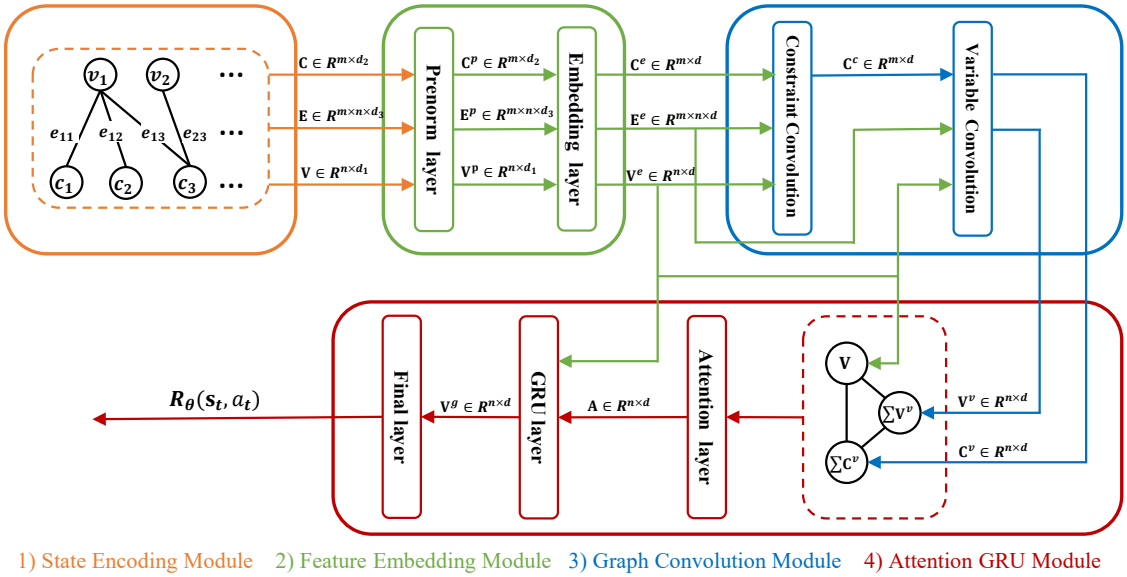


Figure 3.4: The functional structure of the proposed GNN.

Remark 1) to the formulated MINLP problem with significantly low complexity.

3.5 The Proposed Graph Neural Network

In this chapter, we propose GIRL by generating a new variable selection policy to accelerate the B&B algorithm as shown in Fig. 3.3. The GNN $\mathcal{G}$, as the core component of the GIRL, is proposed as a parameterized function to recover the unknown reward. The conventional DNN can also be used to achieve this goal. However, the process of the B&B could be represented as an enumeration tree composed of multiple edges and nodes (e.g., the enumeration tree shown in Fig. 3.2). Compared to DNN, the GNN $\mathcal{G}$ is able to aggregate enough information from graphic structures of the B&B [110], and empower GIRL to easily adapt to dynamic parameters without learning from scratch.

Next, we elaborate on our proposed GNN $\mathcal{G}$ as shown in Fig. 3.4. In the offline

training, the neural network parameters $\boldsymbol{\theta}$ are updated by MaxEnt gradients as shown in Fig. 3.3. In the online inference, the GIRL takes action a_t based on the given state $\mathbf{s}_t$. To indicate whether the GIRL follows the branching strategy of the FSB, the GNN $\mathcal{G}$ generates a reward $R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)$ based on the input state action pair $(\mathbf{s}_t, a_t)$. We draw on the algorithmic structure of the B&B algorithm and take advantage of its strong graphic structure represented in Fig. 3.2, and the structure consists of four modules, namely, state encoding, feature embedding, graph convolution, and attention gated recurrent unit (GRU).

3.5.1 State Encoding Module

In the state encoding module, the graphic features are extracted from the enumeration tree and encoded as the state $\mathbf{s}_t = \{\mathbf{V}, \mathbf{C}, \mathbf{E}\}$, including variable features, constraint features, and edge features.

Variable Features: The variable feature $\mathbf{v} \in \mathbb{R}^{d_v}$ captures the variable information of the branching nodes and the node information except for the constraints $\boldsymbol{C}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ in $\boldsymbol{P}$. There are n fractional variables left in the list $\mathscr{L}_{\mathrm{v}}$, and the variable feature matrix is defined as $\mathbf{V} = [\mathbf{v}_1, \cdots, \mathbf{v}_n]^T \in \mathbb{R}^{n \times d_v}$, where $(\cdot)^T$ denotes the transpose matrix operation. The objective value $\Xi^*(\mathcal{F}_t)$ is also included in $\mathbf{v}$. In addition, $\mathbf{v}$ consists of the lower bound $\Xi_{\mathrm{lb}}(\mathcal{F}_t)$ and upper bound Ξ_{ub} of $\Xi^*(\mathcal{F}_t)$ which provide the potential range $\Xi^*(\mathcal{F}_{t+1})$ for the next state $\mathbf{s}_{t+1}$. Also, we add the average value of all of the previously observed feasible objective values to the distribution of the objective values. Furthermore, some condition features are included in $\mathbf{v}$, such as whether $\Xi^*(\mathcal{F}_t)$ equals to lower bound, i.e., $(\Xi^*(\mathcal{F}_t) = \Xi_{\mathrm{lb}}(\mathcal{F}_t)) \rightarrow 1; (\Xi^*(\mathcal{F}_t) \neq \Xi_{\mathrm{lb}}(\mathcal{F}_t)) \rightarrow 0$, whether $\Xi^*(\mathcal{F}_t)$ equals to upper

bound, i.e., $(\Xi^*(\mathcal{F}_t) = \Xi_{\text{ub}}) \to 1; (\Xi^*(\mathcal{F}_t) \neq \Xi_{\text{ub}}) \to 0$, and whether $\Xi^*(\mathcal{F}_t)$ is fractional. In practice, we normalize the depth by the total number of discrete variables into the variable features.

Constraint Features: The constraint feature $\mathbf{c} \in \mathbb{R}^{d_c}$ captures the constraints $\boldsymbol{C}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ in $\boldsymbol{P}$. The number of the constraints is m, and the constraint feature matrix is defined as $\mathbf{C} = [\mathbf{c}_1, \cdots, \mathbf{c}_m]^T \in \mathbb{R}^{m \times d_c}$. Clearly, $\mathbf{c}$ consists of the constraint normalized by the Euclidean norm of the coefficients, and the number of iterations since the last time the constraint is set. The angle between the objective coefficient vector and constraint vector is taken into account. Moreover, whether the constraints are at their bounds is an important constraint factor.

Edge Features: The edge feature $\mathbf{e} \in \mathbb{R}^{d_e}$ indicates the connection between $\mathbf{v}$ and $\mathbf{c}$. For example, the variable feature $\mathbf{v}_j$ and one of its constraint features $\mathbf{c}_i$ has the edge feature $\mathbf{e}_{i,j}$. In our design, the edge feature $\mathbf{e}_{i,j}$ consists of its index (i, j), the dimension of which is $\mathbb{R}^1$. There are n variable features and m constraint features in total, so the number of edge features should be $n \times m$, The edge feature matrix $\mathbf{E}$ consists of all the edge features, so we have $\mathbf{E} \in \mathbb{R}^{n \times m}$.

In summary, we encode the state $\mathbf{s}_t$ as $\{\mathbf{V}, \mathbf{C}, \mathbf{E}\}$. In fact, $\{\mathbf{V}, \mathbf{C}, \mathbf{E}\}$ can readily extract the graphic characteristics from the enumeration tree of the B&B algorithm. Therefore, the encoding method using $\{\mathbf{V}, \mathbf{C}, \mathbf{E}\}$ is a proper tool to handle the graph-structured data in the B&B algorithm.

3.5.2 Feature Embedding Module

The feature embedding module takes $\mathbf{s}_t$ as its input and normalizes it to a fixed dimension. The purpose of the normalization is to streamline the graph convolution

module, which consists of two successive sub-layers, i.e., the constraint convolution layer and the joint convolution layer. In the feature embedding module, the encoded state $\mathbf{s}_t = \{\mathbf{V}, \mathbf{C}, \mathbf{E}\}$ is initialized and normalized by the prenorm layer and the embedding layer, respectively. This module is a pre-training procedure to improve the model generalization capability of the proposed GNN $\mathcal{G}$, which facilitates handling a large-scale problem.

In standard convolutional neural networks, a weight initiation [111] is usually adopted to normalize the input features $\{\mathbf{V}, \mathbf{C}, \mathbf{E}\}$. The initial weight depends on the number of input features, a parameter unknown prior to the training process in our proposed GNN $\mathcal{G}$. Therefore, instead of weight initiation, we adopt a prenorm layer. In the prenorm layer, the affine transformation $p(\cdot)$ is employed to normalize $\{\mathbf{V}, \mathbf{C}, \mathbf{E}\}$. Mathematically, we have $\mathbf{x}^p = p(\mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu})/\sigma$, where $\mathbf{x} \in \{\mathbf{V}, \mathbf{C}, \mathbf{E}\}$, $\boldsymbol{\mu}$ is the vector of the empirical mean, and σ is the sample variance. Then the initialized features are introduced to the embedding layer to normalize their dimensions and decrease the computational complexity. Specifically, the embedding layer consists of three neural networks with *relu* activation functions. The embedding module for $\mathbf{v}$ and $\mathbf{c}$ can be expressed as

$$\mathbf{v}^e = g^e\left(p\Big(\mathbf{v}\Big)\right),\ \mathbf{c}^e = g^e\left(p\Big(\mathbf{c}\Big)\right),\ \mathbf{e}^e = f^e\left(p\Big(\mathbf{e}\Big)\right), \tag{3.5.1}$$

where g^e and f^e are 2-layer and 1-layer perceptrons (neural networks) in the feature embedding module, respectively. Here, $\mathbf{e}$ is the edge feature, whose linear data structure is simple, so a 1-layer perceptron suffices to extract the features from $\mathbf{e}$. Furthermore, the 1-layer perception is easy to train and provides higher training efficiency compared to the multi-layer perceptron. In contrast, the constraint feature $\mathbf{c}$ is non-linear and far more complex than $\mathbf{e}$. In this case, the 1-layer structure is

no longer effective in feature extraction. Therefore, we resort to a more complex but more effective 2-layer perceptron to fully extract the features from the complex data structure of $\mathbf{c}$.

After the feature embedding, the dimensions d_v, d_c, and d_e are normalized to be a fixed d. Therefore, in the GNN $\mathcal{G}$, regardless of the size of the enumeration trees, the same topology can be processed by the graph convolution module. The embedding module could reduce the computational complexity with negligible information loss. This significantly streamlines the processing at the graph convolution module and leads to improved overall performance.

3.5.3 Graph Convolution Module

The graph convolution module aggregates features of $\mathbf{s}_t$ and captures the spatial and temporal graphic information. It plays a key role in our proposed GNN $\mathcal{G}$. It has been observed that the deep GNN $\mathcal{G}$, even with just three layers, might suffer from over-fitting and over-smoothing, rendering it difficult to gather the required information [112]. Specifically, the graph convolution module [113] is decomposed into two successive sub-layers, the constraint convolution layer and the variable convolution layer.

In the constraint convolution layer, the embedded features in $\{\mathbf{V}^e, \mathbf{C}^e, \mathbf{E}^e\}$ serve as the input. When MINLP problems are solved, constraints are used to determine the value of the variables. Due to the strong correlation between constraint features and variable features, we use a 2-layer perceptron g^c with $relu$ activation functions to form a joint feature $\mathbf{j}^c$ with the same dimension as the embedded constraint feature

$\mathbf{c}^e$. Mathematically, we have

$$\mathbf{j}_j^c = \sum_{i=1}^{n} g^c\left(\mathbf{v}_i^e, \mathbf{c}_j^e, \mathbf{e}_{i,j}^e\right) \in \mathbb{R}^d, \quad \mathbf{J}^c = [\,\mathbf{j}_1^c, \mathbf{j}_2^c, \cdots, \mathbf{j}_m^c]^T \in \mathbb{R}^{m\times d}. \tag{3.5.2}$$

Next, we combine $\mathbf{c}^e$ and $\mathbf{j}^c$, and apply a 2-layer perceptron g^c with $relu$ activation functions. We have

$$\mathbf{c}_j^c = g^c\left(\mathbf{c}_j^e, \mathbf{j}_j^c\right) \in \mathbb{R}^d, \quad \mathbf{C}^c = [\mathbf{c}_1^c, \mathbf{c}_2^c, \cdots, \mathbf{c}_m^c]^T \in \mathbb{R}^{m\times d}, \tag{3.5.3}$$

where $\mathbf{C}^c$ is the new constraint feature matrix. In the variable convolution layer, the input is the new constraint features in $\mathbf{C}^c$ and the embedded features in $\{\mathbf{V}^e, \mathbf{E}^e\}$. To capture the potential node features, we combine features in $\{\mathbf{V}^e, \mathbf{C}^e, \mathbf{E}^c\}$, and rearrange them based on the order of variables and then transfer them to the neural network. Then, we have

$$\mathbf{v}_i^v = \sum_{j=1}^{m} g^v\left(\mathbf{v}_i^e, \mathbf{c}_j^e, \mathbf{e}_{i,j}^e\right) \in \mathbb{R}^d, \quad \mathbf{V}^v = [\mathbf{v}_1^v, \mathbf{v}_2^v, \cdots, \mathbf{v}_n^v]^T \in \mathbb{R}^{n\times d}, \tag{3.5.4}$$

where g^v is a 2-layer perceptron in the variable convolution layer. Regardless of their features, $\mathbf{C}^e$ and $\mathbf{C}^c$ are arranged following the order of the original constraint features in $\mathbf{C}$. In order to strengthen the link between variable features and constraint features, we rearrange the constraint features following the order of the variable features, and the new constraint feature $\mathbf{c}_i^v$ is related to all the constraint features connecting to $\mathbf{v}_i$. We have

$$\mathbf{c}_i^v = \sum_{j=j_a^i}^{j_b^i} f^v\left(\mathbf{c}_j^e\right) \in \mathbb{R}^d, \quad \mathbf{C}^v = [\mathbf{c}_1^v, \mathbf{c}_2^v, \cdots, \mathbf{c}_n^v]^T \in \mathbb{R}^{n\times d}, \tag{3.5.5}$$

where j_a^i and j_b^i is the index of the constraint features connecting to $\mathbf{v}_i$. For example, in the state encoding of Fig. 3.4, $\mathbf{v}_1$ connects to $\mathbf{c}_1$, $\mathbf{c}_2$ and $\mathbf{c}_3$, so j_a^i and j_b^i are 1 and 3, respectively. The outputs of the variable convolution layer are $\mathbf{V}^v$ and $\mathbf{C}^v$, which have the potential node features, and each feature consists of information for its neighbors.

3.5.4 Attention GRU Module

The attention GRU module applies an offset to the convolution module and converts its output features to the temporal-spatial-frequency domain, and obtains the output reward function $R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)$. Specifically, the convolution module is able to integrate different types of relevant information (e.g., node information from $\mathbf{v}$ and $\mathbf{c}$ and edge information from $\mathbf{e}$) via the node-edge-node relationship in the state encoding module of Fig. 3.4. The three different inputs, $\mathbf{V}$, $\mathbf{C}$ and $\mathbf{E}$, are integrated with equal weights before being processed by g^c and g^v. Note that the condition feature in $\mathbf{v}$ directly determines the variable selection in the conventional B&B algorithm. Consequently, $\mathbf{V}$ plays a more significant role, and a more appropriate weight should be allocated. Therefore, there should be a more sensible way to train a set of proper weights for various types of input features. Accordingly, a new module, referred to as the attention GRU module is incorporated into our GNN $\mathcal{G}$ design. This module consists of four major components, namely, mix state, attention layer, GRU layer, and final layer.

Mix State: The mix state is based on some matrix operation on $\mathbf{V}^e$, $\mathbf{V}^v$, and $\mathbf{C}^v$. Clearly, $\mathbf{V}^e$ represents the embedded original information of $\mathbf{V}$. In addition, $\mathbf{V}^v$ includes the combined information of $\{\mathbf{V}, \mathbf{C}, \mathbf{E}\}$, and $\mathbf{C}^v$ consists of the constraint information of each variable node. We have the three matrices $\mathbf{V}^e$ $\mathbf{V}^v$ and $\mathbf{C}^v$, where the order of vectors is consistent with the order of variable features in the bipartite graph of the state encoding module as shown in Fig. 3.4. Specifically, we extract the vectors of these three matrices with the same index k to form a mix state matrix

$$\mathbf{M}_k = [\mathbf{v}_k^e, \mathbf{v}_k^v, \mathbf{c}_k^v]^T. \tag{3.5.6}$$

Here, $\mathbf{v}_k^e$, $\mathbf{v}_k^v$, and $\mathbf{c}_k^v$ can be regarded as vectors of three new nodes, so we express

the t-th row vector in $\mathbf{M}_k$ as a node vector $\mathbf{n}_{k,t} = \mathbf{M}_k[t] \in \mathbb{R}^d$. This mix state matrix can be expressed as a new node-edge graph shown in the attention GRU module of Fig. 3.4. In comparison to the bipartite graph in the state encoding module, this graph has a fixed number of nodes and edges which facilitates the implementation of the ensuing attention mechanism.

Attention Layer: The attentioned information of the node in the new graph is the sum of edge-weighted information of its neighbor node vectors, which is expressed as an attentional vector An attentioned vector, aggregating edge-weighted information of one node and its neighbor nodes, is the output of the attention layer and can be expressed as

$$\mathbf{A}_k = \sum_{\mathbf{n}_{k,j} \in \mathbf{M}_k} w_{\mathbf{n}_{k,j} \to \mathbf{n}_{k,i}} \cdot \mathbf{n}_{k,j} \tag{3.5.7}$$

where $\mathbf{n}_{k,i}$ is the target node vector to update, while $\mathbf{n}_{k,j}$ is one of its neighbor node vectors (we write this neighboring relationship as $\mathbf{n}_{k,j} \to \mathbf{n}_{k,i}$). Furthermore, $w_{\mathbf{n}_{k,i} \to \mathbf{n}_{k,j}}$ is the edge weight from $\mathbf{n}_{k,j}$ to $\mathbf{n}_{k,i}$, and it is calculated as

$$w_{\mathbf{n}_{k,j} \to \mathbf{n}_{k,i}} = \sigma\left(g^a\left([\mathbf{n}_{k,i}; \mathbf{n}_{k,j}]^T \cdot \mathbf{W}_w\right)\right) = \frac{e^{g^a\left([\mathbf{n}_{k,i}; \mathbf{n}_{k,j}]^T \cdot \mathbf{W}_w\right)}}{\sum_{l=1}^{3} e^{g^a\left([\mathbf{n}_{k,i}; \mathbf{n}_{k,l}]^T \cdot \mathbf{W}_w\right)}}, \tag{3.5.8}$$

where ; represents the operation of concatenation, and $\mathbf{W}_w \in \mathbb{R}^{2d}$ is the weight matrix. Besides, a 2-layer perceptron with *LeakyRelu* activation function g^a and softmax function $\sigma(\cdot)$ is applied to realize a rational weights distribution by comparing one neighbor node to all its neighbor nodes.

GRU Layer: In the GRU layer, the attentioned vector $\mathbf{a}_k$ will be updated by a 1-layer GRU layer, which is based on the previous work [114]. Two inputs of the GRU layer include a target node vector $\mathbf{v}_i^e$ and a vector $\mathbf{a}_i$ containing aggregated neighbor information. Thus, the output of the GRU layer, known as the gated variable feature

vector, can be shown as $\mathbf{v}_i^g = \mathrm{GRU}(\mathbf{v}_i^e, \mathbf{A}_i)$. The detailed update process of the GRU layer obeys the following formulas

$$\begin{aligned} \mathbf{z}_i^g &= S(\mathbf{W}_z \mathbf{a}_i + \mathbf{U}_z \mathbf{v}_i^e + \mathbf{b}_z), \\ \mathbf{r}_i^g &= S(\mathbf{W}_r \mathbf{a}_i + \mathbf{U}_r \mathbf{v}_i^e + \mathbf{b}_r), \\ \bar{\mathbf{v}}_i^g &= \tanh(\mathbf{W}_v \mathbf{a}_i + \mathbf{U}_v(\mathbf{r}_i^g \odot \mathbf{v}_i^e) + \mathbf{b}_v), \\ \mathbf{v}_i^g &= \mathbf{v}_i^e \odot (1 - \mathbf{z}_i^g) + \bar{\mathbf{v}}_i^g \odot \mathbf{z}_i^g, \end{aligned} \tag{3.5.9}$$

where $\mathbf{W}_z$, $\mathbf{W}_r$ and $\mathbf{W}_v$ are the GRU weights, $\mathbf{b}_z$, $\mathbf{b}_r$ and $\mathbf{b}_v$ are the GRU bias, and $\mathbf{z}_i^g$ and $\mathbf{r}_i^g$ are updated and reset gates. Furthermore, $S(\cdot)$ is the logistic sigmoid function, and $\odot$ is the Schur product operation.

Final Layer: Finally, we apply a 2-layer perception with $relu$ activation functions to the features in $\mathbf{V}^g$, and we obtain $\mathbf{V}^f$. A softmax function is employed to obtain the reward

$$R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_i) = \sigma\Big(v_i^f\Big) = \frac{e^{v_i^f}}{\sum_{i=1}^{n} e^{v_i^f}}, \ i \in [1, n], \tag{3.5.10}$$

where a_i represents that variable α_i is selected to branch. Note that a_t is to select a variable to branch from $\mathscr{L}_{\mathrm{v}}$ that includes n variables, thus we have $a_t \in \{a_1, ...a_i, ...a_n\}$. Accordingly, with the input $(\mathbf{s}_t, a_t)$, the GNN $\mathcal{G}$ would generate a reward $R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t) \in [0, 1]$. Note that the higher value of $R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)$, the larger probability that the GIRL and the B&B with FSB can select the same variable.

In summary, regardless of the input graph size, enumeration trees are encoded into the graph-structured features $\{\mathbf{V}, \mathbf{C}, \mathbf{E}\}$ in a unified way. It is clear that the operations (3.5.1)-(3.5.10) do not rely on the size of the input graph. Consequently, the output of the GNN $\mathcal{G}$ exhibits permuted equivariance, which empowers the GIRL to easily adapt to the variations of the system parameters without learning from scratch. Besides, the computational complexity is only related to the graph density,

which makes the GNN $\mathcal{G}$ as an ideal choice for processing graph-structured data of the B&B algorithm.

3.6 Performance Analysis

3.6.1 Model Generalization

In practical multi-layer computing networks, parameters such as the number of MDs and required CPU cycles can change dynamically over time. As a key advantage, our GIRL, although trained offline in a specific parameter setting, can be directly applied to a new scenario with dynamic parameters without learning from scratch due to the designed GNN $\mathcal{G}$ and incorporated self-imitation.

Under the GNN $\mathcal{G}$, various enumeration trees with different network parameters are encoded into the graph-structured features $\{\mathbf{V}, \mathbf{C}, \mathbf{E}\}$ in a unified way. Then, the features could be rearranged to a fixed dimension d, resulting in the same topology to be processed by the GNN $\mathcal{G}$. Consequently, the output of the GNN $\mathcal{G}$ is guaranteed to exhibit permuted equivariance and is independent of different parameters.

Self-imitation in the offline training of the GIRL collects some additional problems as the self-imitation demonstration $\mathcal{D}'$, which are not in the original expert demonstration $\mathcal{D} = \{\mathcal{D}_\varsigma \mid \varsigma \in \mathcal{S}\}$. Because $\mathcal{D}'$ has not been learned by the previous policy π_{old}, a better policy π_{new} can be obtained based on the new demonstration $\mathcal{D} \cup \mathcal{D}'$. In other words, the generalization performance of the GIRL is improved iteratively by repeating this process. For our GIRL, extensive simulations have been conducted, revealing that a mere three self-imitation iterations yield remarkable model generalization.

3.6.2 Performance Optimization

To further optimize the GIRL variable selection policy and accelerate the training process, we aim to evaluate the impact of several dominant factors on performance. It is known from Section 3.4.3 that, when $\boldsymbol{\theta}$ converges to $\boldsymbol{\theta}^*$, the optimal reward function $R^*(\mathbf{s}_t, a_t)$ is obtained. Then the optimal policy π_t^* could be derived from $R^*(\mathbf{s}_t, a_t)$ via Q-learning with Modeling Learning. The training performance of the GIRL depends on the speed of convergence, or the variation $\Delta\boldsymbol{\theta}$ in (3.4.14). The speed of convergence is affected by two major factors, the learning rate, and the expert demonstration.

Learning Rate Analysis

In the update process of $\boldsymbol{\theta}$ in (3.4.14), the learning rate λ_θ determines how much the newly acquired information $\Delta\boldsymbol{\theta}$ covers the current state of $\boldsymbol{\theta}$. For a large λ_θ, $\boldsymbol{\theta}$ is likely to converge without a performance guarantee. On the contrary, a small λ_θ would extend the training process and increase the computational complexity. Instead of a fixed learning rate setting, we adopt an adaptive learning rate approach referred to as adaptive moment estimation (ADAM) [115] in our GIRL. ADAM aims to optimize the first-order gradient-based stochastic objective functions, based on adaptive estimates of lower-order moments. To evaluate the impact of the learning rate in the update process, we implement three different update schemes in 3.7.3, including standard gradient ascent (SGA) with large λ_θ, SGA with small λ_θ, and ADAM. Specifically, the SGA could be expressed as $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \lambda_\theta \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$.

3.6.3 Computational Complexity Analysis

In this section, we analyze the computational complexity for several variable selection policies, namely, the FSB, the RPB, and the GIRL. In general, the complexity of each policy mainly consists of two parts: the variable selection process and the node solving process.

For the FSB, in the variable selection process, if there are K fractional variables to branch, there are $2K$ NLPs to be solved, each with the complexity $\mathcal{O}(n^3)$ (n is the number of discrete variables). As such, the complexity for variable selection process is $\mathcal{C}_{\text{vs}}^{\text{FSB}} = \mathcal{O}(2n^3 K N_{\text{v}})$, where N_{v} is the number of times to select variables. It is also straightforward to see that, for the node solving process with N_{n} nodes in the enumeration tree, the complexity is $\mathcal{C}_{\text{ns}}^{\text{FSB}} = \mathcal{O}(n^3 N_{\text{n}})$. The RPB can also obtain an optimal solution with lower complexity than the FSB. Essentially, the RPB is based on the idea of strong branching initialization and then uses the pseudo-cost until the estimate of the variable is reliable [99]. The variable selection of the RPB features a lower complexity than the FSB ($\mathcal{C}_{\text{vs}}^{\text{RPB}} < \mathcal{C}_{\text{vs}}^{\text{FSB}}$). However, its enumeration tree is much larger than that of FSB, leading to higher complexity in the node solving process ($\mathcal{C}_{\text{ns}}^{\text{RPB}} > \mathcal{C}_{\text{ns}}^{\text{FSB}}$). Overall, we have $\mathcal{C}_{\text{vs}}^{\text{RPB}} + \mathcal{C}_{\text{ns}}^{\text{RPB}} < \mathcal{C}_{\text{vs}}^{\text{FSB}} + \mathcal{C}_{\text{ns}}^{\text{FSB}}$.

For the GIRL, its complexity lies in offline training and online inference. Offline training is a one-off process and its complexity can be ignored. After the offline training, we directly apply the optimal policy to select variables for a new MINLP problem instance in the online inference, and the complexity is only $\mathcal{O}(n)$. As such, the computational complexity is $\mathcal{C}_{\text{vs}}^{\text{GIRL}} = \mathcal{O}(nN_{\text{v}})$ for the whole variable selection process. Clearly, we have $\mathcal{C}_{\text{vs}}^{\text{GIRL}} \ll \mathcal{C}_{\text{vs}}^{\text{FSB}}$. Ideally, if the offline training is performed with a sufficient number of training samples, the GIRL would generate the same

enumeration tree as the FSB ($\mathcal{C}_{\text{ns}}^{\text{GIRL}} = \mathcal{C}_{\text{ns}}^{\text{FSB}}$). Overall, the complexity of the GIRL is much lower than FSB ($\mathcal{C}_{\text{vs}}^{\text{GIRL}} + \mathcal{C}_{\text{ns}}^{\text{GIRL}} \ll \mathcal{C}_{\text{vs}}^{FSB} + \mathcal{C}_{\text{ns}}^{\text{FSB}}$). In Section 3.7, the complexity of these three policies will be numerically compared.

3.7 Simulation Results

3.7.1 Experimental Setting

In Figs. 3.5-3.8 and Figs. 3.10-3.12, we consider a task offloading scenario with 20 MDs and 5 MSs. The distance DIS_1 between MD and its associated MS is sampled from the uniform distribution $\mathcal{U}(0.1, 0.5)$ (in kilometers). We assume that each MD has one D2D link with a peer MD, and the distance DIS_2 between MDs is sampled from the distribution $\mathcal{U}(0.01, 0.05)$ (in kilometers). The path-loss model between MDs and MSs follows $g = 128.1 + 37.6 \log(\text{DIS}_1)$ (DIS_1 in kilometers), and the path-loss model for the D2D link follows $g = 148.1 + 40 \log(\text{DIS}_2)$ (DIS_2 in kilometers) [116]. For task Q_i, its data size D_i is sampled uniformly from $\mathcal{U}(0.8, 1.2)$ (in Mb) and the number of required computation resources C_i is sampled uniformly as $\mathcal{U}(2, 5)$ (in Gigacycles). Besides, the other related parameters are summarized in Table 3.2.

3.7.2 Policies for Comparison and Performance Metrics

From the algorithm perspective, to illustrate the significance of adopting the variable selection policy in the B&B algorithm, we compare our proposed GIRL with the FSB and the RPB introduced in Section 3.3.2. A heuristic policy with random variable selection is also included to provide baseline performance for comparison. According to Remark 3.3.2, all the above variable selection policies can eventually

Table 3.2: MEC system parameters

Local transmission power p_i^{tr}	23 dBm
Local computation power p_i^{L}	0.5 W
Local computation capability f_i^{L}	0.6 GHz
Local energy capability $e_i^{\max}$	3 J
Computation capability of MS $f_k^{\max}$	1 GHz
Bandwidth capacity of MS $B_k^{\max}$	1 MHz
Round-trip transmission time d_{l,l_i}^{rrt}	0.5 s
Cloud computation capability $f_{i,c}$	20 GHz
Backhaul time d^{BC}	5 s
Noise power spectrum density σ^2	-174 dBm/Hz

reach the optimal node and obtain an optimal solution to the MINLP problem.

We first adopt accuracy as one of the algorithm performance metrics, which quantifies the similarity between other variable selection policies and the FSB in terms of the variable selection order. This will have an impact on the computational complexity rather than the optimality. The accuracy includes three representative indicators: top-one accuracy (acc@1), top-five accuracy (acc@5), and top-ten accuracy (acc@10). Generally speaking, the top-x accuracy is the percentage of the top-x ($x \in \{$one, five, ten$\}$) variables selected by other policies among all the unbranched variables. Specifically, for a given node, all unbranched variables are marked by the FSB. If a policy selects a variable (e.g., α_1) and its FSB score is the highest one among the variables, this variable will be marked as the top-one variable. If a policy selects a variable and its FSB score is one of the highest five among the unbranched variables, this variable will be marked as the top-five variable. To further evaluate the computational complexity, the number of solved NLPs is adopted as another metric. Note that the solved NLP number is related to both the variable selection process and node solving

process. In order to control the training and testing time, we set a number limitation (2000), beyond which the process of the B&B algorithm will be terminated.

From the network perspective, we compare our proposed full-dimensional task offloading scheme with the non-cooperation offloading scheme, the vertical offloading scheme, and the two-dimensional offloading scheme. For the non-cooperation scheme, the task Q_i of the MD i can only be offloaded to its associated MS l_i, and the computation resources $\mathbf{f}$ and bandwidth $\mathbf{b}$ are optimized with $x_{i,i} + y_{i,l_i} = 1$ for all $i \in \mathcal{N}$. For the vertical offloading scheme, Q_i can be offloaded to its associated MS l_i and CS. Here, $\mathbf{f}$ and $\mathbf{b}$ are optimized with $x_{i,i} + y_{i,l_i} + z_i = 1$ for all $i \in \mathcal{N}$. Furthermore, for the two-dimensional offloading scheme, Q_i not only can be offloaded vertically among MD i, MS j and CS but offloaded horizontally from MS j to MS $k \in \hat{\mathcal{M}}_i$, thus, we have $x_{i,i} + \sum_{k \in \mathcal{M}} y_{i,k} + z_i = 1$.

3.7.3 Simulation Results

Training Performance of the GIRL Variable Selection Policy

We first demonstrate the performance of the GIRL variable selection policy during the training process in our full-dimensional task offloading scheme. The performance is evaluated by the normalized loss. In Fig. 3.5, we compare the loss and convergence speed of the GIRL for three learning rate settings (e.g., SGA with large λ_θ (0.2), SGA with small λ_θ (0.001), and ADAM introduced in Section 3.6.2). We can observe that the SGA with the larger learning rate (0.2) leads to much faster convergence and a larger loss relative to the SGA with the smaller learning rate (0.001). It can also be seen that ADAM performs best in terms of loss, and its convergence speed is only slightly inferior to the learning rate 0.2 at the beginning. This demonstrates that

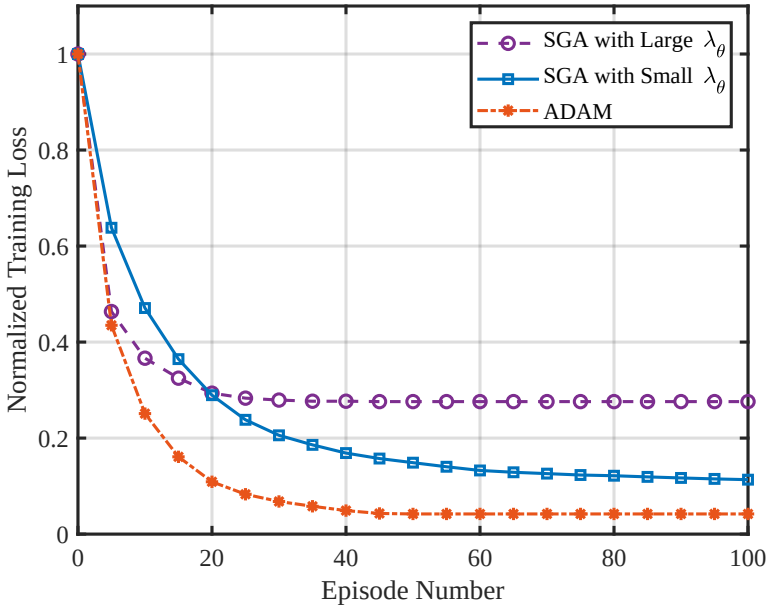


Figure 3.5: Loss and convergence speed comparison for different λ settings.

ADAM achieves an excellent training performance. A similar comparison is carried out in Fig. 3.6 for the expert and imperfect demonstrations. Note that the imperfect demonstration is mingled with some artificial data (noise). It is observed that the loss of the imperfect demonstration does not converge. In contrast, convergence is achieved for the expert demonstration, and its loss is always lower than that of the imperfect demonstration. This verifies the significant role that the demonstration plays in the learning process as discussed in Section 3.6.2.

Optimality Comparison

In Fig. 3.7, we compare the accuracy performance of the GIRL, the FSB, the RPB, and the heuristic policy. There are 40 online testing problem instances, and 3 out of 40 testing problem instances are randomly selected and presented in the figure

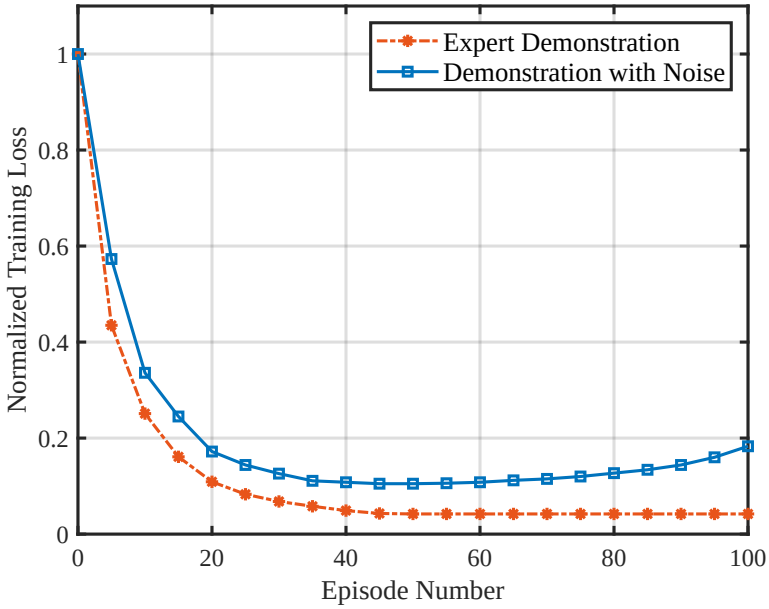


Figure 3.6: Loss and convergence speed comparison for different demonstrations.

as Setting 1-3. Note that the accuracy for the FSB is always 100%, as all the variable selection policies are to imitate the behavior of the FSB. It can be observed that the GIRL has higher acc@1, acc@5, and acc@10 than the RPB and the heuristic policy. Specifically, it achieves the accuracy of 85.2%, 93.6%, and 98.6% after averaging across the three settings for acc@1, acc@5, and acc@10, respectively. For the RPB, on average, the acc@10 approaches 90%, while acc@1 is about 50%. This clearly verifies the superiority of our proposed GIRL over the RPB in imitating the performance of the FSB. It is also clear that, for the heuristic policy, the acc@1 fluctuates significantly across the three settings (15.8%, 23.5%, and 43.7%), so it is with acc@5 and acc@10. Clearly, Fig. 3.7 leads to the conclusion that the GIRL best imitates the behavior of the FSB.

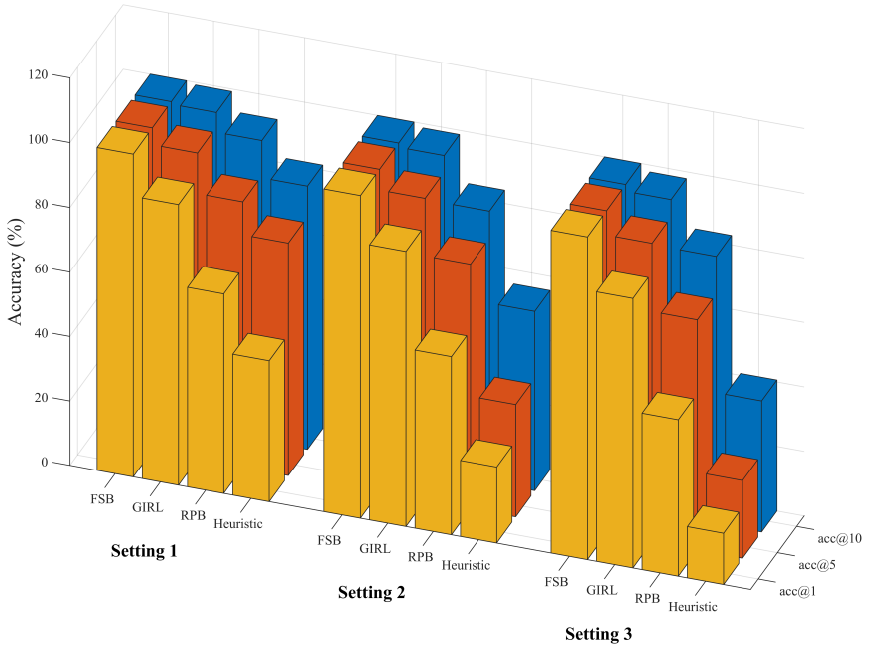


Figure 3.7: Accuracy for different variable selection policies.

Complexity Comparison

The computational complexity comparison is carried out for the GIRL, the FSB, the RPB, and the heuristic policies in Fig. 3.8 in terms of the number of the solved NLPs. Among 40 online testing problem instances, 4 are randomly selected and presented in both figures as Setting 1-4. In Fig. 3.8, it is observed that the GIRL significantly reduces the solved NLP number compared with the FSB across all four settings. A simple calculation finds that such a reduction stands at 84.8% on average. Compared with the RPB, the GIRL achieves a reduction of approximately 54.2% on average. Finally, it is clear that the heuristic policy has the highest complexity, and the number limitation is even triggered for Setting 2 and Setting 3.

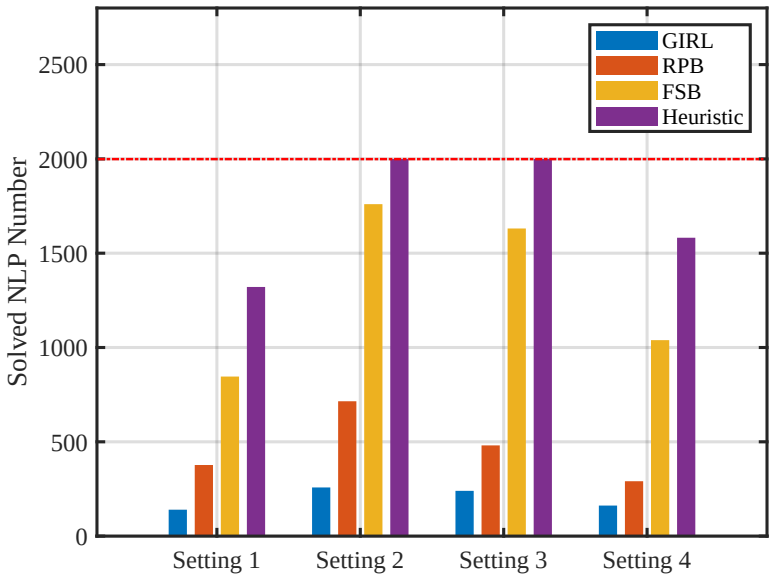


Figure 3.8: Complexity comparison of the four variable selection policies in terms of searched NLP number.

Model Generalization Capability of the GIRL Variable Selection Policy

Having considered the scenario with a fixed value for N, next, we compare the model generalization capability of these policies (c.f. Section 3.6.1). Here, the GIRL is offline trained with an original expert demonstration and three self-imitation demonstrations, and then directly infers the variable selection to solve the formulated problems with different settings. By contrast, RPB and FSB need to solve both the variable selection order and the sub-problems at each setting. Fig. 3.9 considers the original setting ($N = 20$) transferring to different MD numbers. Across the whole range of N, the GIRL has the lower average solved NLP number and the FSB has the highest average solved NLP number, consistent with the previous observation in Fig. 3.8. In Fig. 3.7, Setting 1 is the accuracy of Setting $N = 20$ in Fig. 3.9. It is

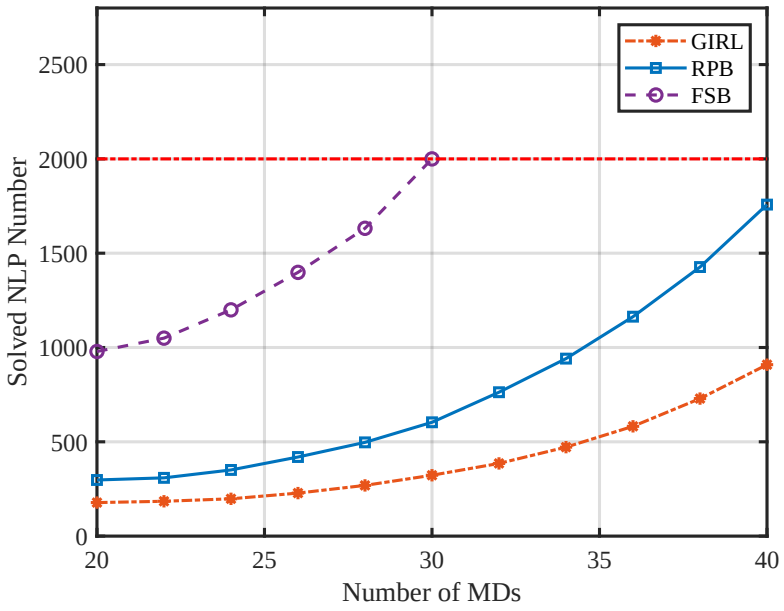


Figure 3.9: Average solved NLP numbers with different numbers of MDs.

observed from Figs. 3.7 and 3.9 that the GIRL has higher top-one accuracy than the RPB, and the GIRL achieves a lower complexity than the RPB. Note that the NLP number limitation has reached the FSB ($N = 30$). It is also observed that the average solved NLP number gap between the RPB and the GIRL significantly increases with N. In summary, the GIRL could directly infer the solutions to the problems with different settings without learning from scratch, which confirms the timeliness and effectiveness of the GIRL.

Average Delay Comparison

In Figs. 3.10-3.12, we compare our proposed full-dimensional task offloading scheme with the non-cooperation offloading scheme, the vertical offloading scheme, and the two-dimensional offloading scheme. The full-dimensional offloading schemes are achieved

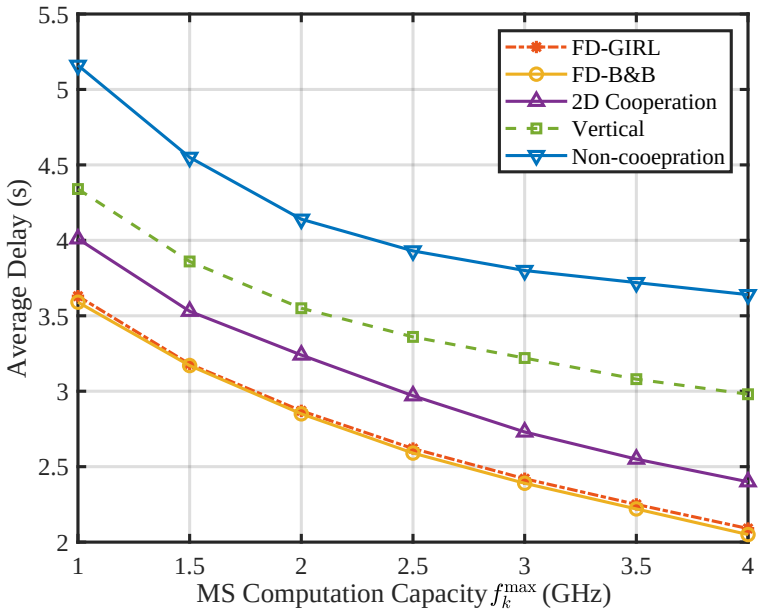


Figure 3.10: Average delay performance comparison versus different $f_k^{\max}$.

by either the B&B algorithm with the FSB or the accelerated B&B with the GIRL, which are denoted as FD-B&B and FD-GIRL, respectively.

In Fig. 3.10, we compare the variation of average delay with the MS computation capability $f_k^{\max}$. As expected, it is observed that the average delay decreases as $f_k^{\max}$ increases, due to the increased total computation capacity in the computing networks. Clearly, FD-B&B and FD-GIRL have similar performance, consistent with the observation in Fig. 3.7, and both of them achieve the shortest average delay among different offloading schemes. In particular, when $f_k^{\max} = 4$ GHz, improvements of the FD-GIRL are about 74.2%, 42.6% and 20.8% compared with the non-cooperation offloading, the vertical offloading, and the two-dimensional offloading schemes, respectively.

In Fig. 3.11, we compare the average delay versus the backhaul time d^{BC}. Except

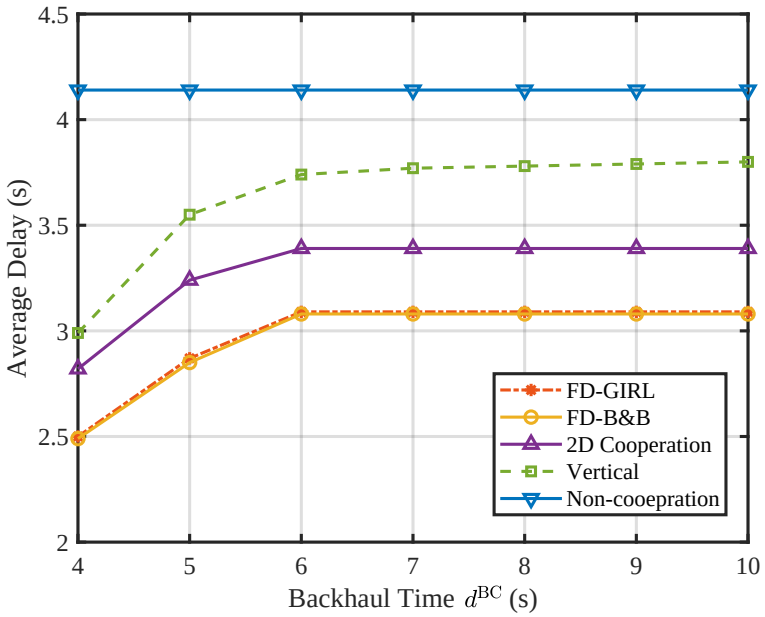


Figure 3.11: Average delay performance comparison versus different d^{BC}.

for the non-cooperation offloading scheme, the average delay of the other three offloading schemes increases with d^{BC} increases. It is clear that FD-B&B and FD-GIRL achieve the shortest average delay. In particular, when $d^{\mathrm{BC}} = 10$ s, improvements of the FD-B&B are about 34.2%, 22.9% and 10.5% compared with non-cooperation offloading, the vertical offloading, and the two-dimensional offloading schemes, respectively.

In Fig. 3.12, we compare the average delay versus the different maximum ranges of D2D links for the FD-GIRL and the two-dimensional offloading schemes [117]. Clearly, it is observed that FD-GIRL achieves a lower average delay than the two-dimensional offloading scheme. When the maximum ranges of D2D links increase, the average delay increases, the reason for this is MDs have a lower potential to offload tasks with the increased ranges of D2D links. It is also observed that a lower

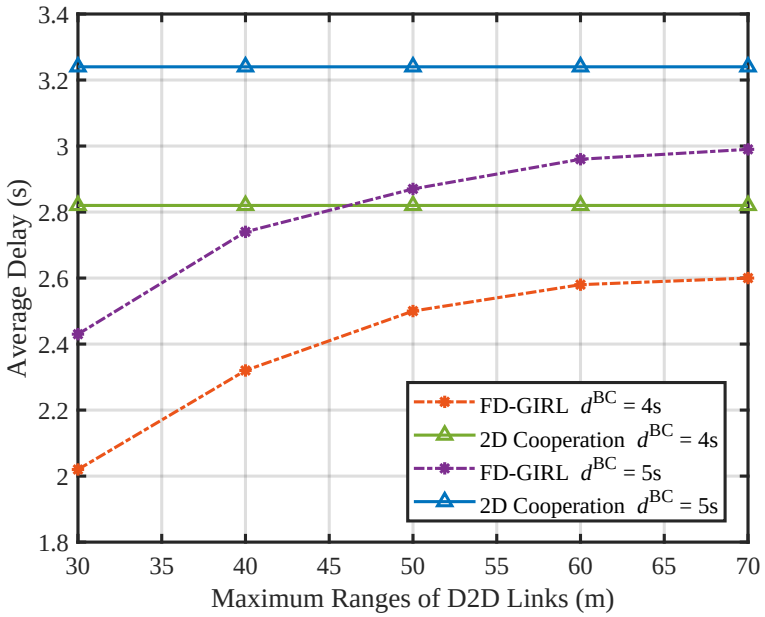


Figure 3.12: Average delay performance comparison versus different maximum ranges of D2D links.

average delay can be achieved with a lower backhaul time d^{BC} consistent with the observation in Fig. 3.11. In particular, when the maximum range of D2D links is 70 m, improvements of the FD-B&B are about 9.4% and 8.7% compared with the two-dimensional offloading scheme for $d^{\mathrm{BC}} = 4$ s and 5 s, respectively.

Overall, the simulations in Figs. 3.10-3.12 illustrate that our proposed full-dimensional offloading scheme can significantly improve the average delay performance by jointly optimizing task offloading decisions and communication/computation resources. In addition, it is verified that our GIRL could effectively solve the formulated problems with different dynamics.

3.8 Conclusion

In this chapter, we incorporated D2D communication into the multi-layer computing network and proposed a full-dimensional task offloading scheme by jointly optimizing task offloading decisions and communication/computation resources. We formulated it as an MINLP problem to minimize the average delay for all the MDs and proposed the GIRL that offers a global optimal solution to the MINLP problem by generating a new variable selection policy to accelerate the optimal B&B algorithm with significantly reduced complexity but without sacrificing the global optimality. It has been verified by simulations that the GIRL achieves a significantly lower computational complexity compared to the existing variable selection policies. Furthermore, the superiority of our proposed full-dimensional task offloading scheme over the existing schemes has been verified by simulation.

Chapter 4

Green Cell-free Massive MIMO: An Optimization Embedded Deep Reinforcement Learning Approach

In this chapter, we delve into strategies to optimize energy efficiency within the downlink cell-free massive MIMO systems, a tangible implementation of wireless IoT networks. To tackle this problem, we develop a green energy scheme by simultaneously optimizing power allocation and AP selection based on ZF beamforming. We formulate it as a non-convex MINLP, which is NP-hard. To address this challenging problem, we propose a new algorithm that embeds non-convex optimization into contemporary DRL, referred to as OSAC-G. OSAC-G enjoys the benefits of directly online inferring solutions for the non-convex problem with a much lower computational complexity compared to conventional non-convex optimization. To enable OSAC-G to adapt to the variations of class II parameters without learning from scratch, we further develop a GTN that extracts the underlying relationships between APs and UEs from a constructed heterogeneous graph. Simulation results demonstrate that the green energy scheme significantly decreases energy consumption compared to the existing ones.

4.1 Introduction

Cell-free massive MIMO has demonstrated great potential to enhance wireless networks by significantly increasing spectral efficiency and coverage area. The conventional MIMO system is limited by the cell-centric design with inherent inter-cell interference [118]. Cell-free massive MIMO can address degraded cell-edge performance by distributing a massive number of antennas over a large geographic area and fully eliminating inter-cell interference [59]. Specifically, a large number of APs equipped with single or multiple antennas are connected to a central processing unit (CPU) through high-capacity fronthaul links and jointly serve a number of UEs with the same time-frequency resource blocks [119]. In this case, the concept of "cell" and "cell edges" are no longer relevant, and inter-cell interference is eliminated. Consequently, significantly increased spectral efficiency and system capacity could be achieved. Besides, some antennas are potentially closer to the UEs in the cell-free massive MIMO configuration, which yields a higher degree of macro-diversity and lower path losses [120].

Several critical techniques have been proposed and progressively improved in cell-free massive MIMO. Beamforming aims to adjust signals transmitted by each AP to create a constructive interference pattern at the intended UE location, and appropriate beamforming techniques should be developed to mitigate interference. In the literature, conjugate beamforming (CB) is most considered due to its simple structure without much performance loss. The authors in [58] implemented CB on the downlink, which achieved low interference and high per-user throughput. In [121], the authors developed a modified CB for the downlink to eliminate self-interference and achieve good performance without pilots. In cell-free massive MIMO, usually, a large

number of distributed AP antennas are involved, and channel estimation is operated in large dimensions and highly challenging. Several efforts have been endeavored to develop efficient channel estimation methods. Single measurement vector based minimum mean square error (MMSE) estimation was proposed in [122] and centralized MMSE estimation was developed in [123]. In addition to the estimation method, pilot assignment is another crucial aspect. It ensures that each UE receives a sufficient number of pilots while minimizing interference between different UEs. Various techniques have been reported in the open literature. A weighted graphic framework-based approach and Hungarian algorithm-based approach have been proposed in [124] and [125], respectively.

In addition to the aforementioned key techniques, energy efficiency in cell-free massive MIMO is another critical issue. As a multitude of APs are distributed across a wide geographical area, a dedicated fronthaul link is required to connect each AP to a CPU [126], and the transmissions and hardware-related static power supplies of the APs consume huge energy, resulting in significantly degraded energy efficiency. Recently, some methods were proposed to optimize the energy efficiency in cell-free massive MIMO [30–32]. In [30], the authors optimized power allocation to maximize total energy efficiency constrained by per-user spectral efficiency and per-AP transmit power. Based on the power allocation, the AP selection was optimized to further increase energy efficiency. The authors in [31] optimized the association parameters to maximize energy efficiency subject to minimum rate constraints by modified deep deterministic policy gradients (DDPG) algorithm. In [32], the power allocation and load balancing were jointly optimized to minimize downlink power consumption subject to transmit powers and spectral requirements.

In this chapter, we develop a green energy scheme in downlink cell-free massive MIMO by simultaneously optimizing power allocation and AP selection based on local-based ZF, a more efficient beamforming technique compared to CB [127]. We consider a practical energy consumption model that includes energy consumed by fronthaul links and APs, where the former transfers data between APs and CPU [128]. We observe that some APs are potentially located far from the UEs, so it might not be necessary to activate all APs all the time. By jointly optimizing power allocation and AP selection, it is possible to identify a subset of APs that can be deactivated to sleep mode leading to reduced energy consumption without significantly impacting the quality of service, especially when the number of APs is relatively large. On this basis, we formulate a non-convex MINLP constrained by QoS requirements and maximum transmit power. This optimization problem involves a large number of mutually coupled discrete and continuous variables, which is NP-hard. It becomes difficult for the existing algorithms to handle a large mixed (discrete and continuous) space, and generally no optimal solution can be found with a polynomial-time algorithm [129].

To address this challenge, we first transform mixed variables of the formulated problem into continuous ones by exploiting a strong correlation between the activated APs and the power allocation coefficients. Despite its decreased computational complexity, the transformed problem is still very challenging due to its non-convexity. Therefore, we propose a new algorithm that embeds non-convex optimization into contemporary DRL [130], referred to as optimization-embedded soft actor-critic with graph transformer networks (OSAC-G). OSAC-G exploits soft actor-critic (SAC) [131] to learn the optimal mapping from the "state" (Class I parameters e.g., time-varying

wireless parameters) to the "action" (e.g., power allocation and AP selection) to maximize the "reward" through one-off offline training. This enjoys the benefits of directly online inferring solutions for the transformed non-convex problem with a much lower computation complexity. With the soft policy updates of SAC, OSAC-G can stabilize learning and improve convergence. However, SAC alone cannot guarantee that all constraints in the formulated problem can be strictly satisfied and cannot capture the complex data structure of solutions for the non-convex formulated problem. As a solution, we leverage the successive approximation method (SAM) to incorporate both optimization objectives and constraints into the design of the reward function. With the handcrafted reward function, it is verified that all the constraints can be satisfied with a high probability. The designed reward function involves solution performance evaluation, and the decision-making of joint power allocation and AP selection by OSAC-G is further improved.

Moreover, whenever the Class II parameters (e.g., the number of APs and the number of UEs) change, DRL models need to be retrained based on the new parameters. To avoid repetitive training, we develop a novel GTN in OSAC-G adaptive to the moderate variation of Class II parameters without learning from scratch. Under the GTN, a heterogeneous graph [132] is first constructed, where each AP-UE pair is mapped to one node of the graph. The hidden feature within each node, independent of different Class II parameters, is then extracted by the graph transformer and mapped to the action. In addition, multi-head attention [133] is introduced to assign a different weight to each node, which highlights the dependency of each AP-UE pair.

The subsequent sections of this chapter are structured in the following manner. We first introduce the system model and formulate a non-convex problem in Section

4.2. Then, we elaborate on the design of the proposed OSAC-G for fast online power allocation and AP selection in Section 4.3. Furthermore, we develop a GTN to enhance the model generalization capability in Section 4.4. On this basis, in Section 4.5, we provide the convergence and computational complexity analysis. In Section 4.6, the simulation results are presented and discussed. Finally, the conclusions are presented in Section 4.7.

4.2 System and Signal Transmission Model

4.2.1 System Model

We consider a cell-free massive MIMO system, where N APs, each with M antennas, simultaneously serve K single-antenna UEs within the same time-frequency resource ($NM \gg K$). The N APs are randomly distributed across a large geographical area and are connected to a CPU via fronthaul links. It is assumed that the uplink and downlink transmissions operate under the time-division duplex (TDD) mode. Additionally, a quasi-static block Rayleigh fading channel [134] is assumed, where channel coefficients remain constant during a transmission block but vary independently from block to block. Specifically, the channel coefficients $\mathbf{g}_{nk} \in \mathbb{C}^{M\times 1}$ between AP n and UE k are modeled as

$$\mathbf{g}_{nk} = \beta_{nk}^{1/2}\ \mathbf{h}_{nk}, \tag{4.2.1}$$

where $\beta_{n,k}$ is the large-scale fading coefficient consisting of path loss and shadowing, which is already known by each AP [127]. Besides, $\mathbf{h}_{nk} \sim \mathcal{CN}(\mathbf{0}, \mathbf{I}_M)$ are the small-scale fading coefficients.

In this chapter, we consider the downlink transmission. Based on the channel

reciprocity in the TDD mode, during a coherence interval τ_c, the uplink training is first used for CSI acquisition, and then beamforming based on the obtained CSI is used for downlink data transmission. Next, we elaborate on these two phases.

4.2.2 Uplink Training

During uplink training, UEs simultaneously transmit their pilot sequences. Utilizing the received pilot sequences, the APs can estimate the CSI through MMSE estimation. The time interval for the uplink training is denoted by τ_p, which is smaller than τ_c. It is assumed that τ_p ($\tau_p \leq K$) orthogonal pilots are available. Let $i_k \in \{1, \cdots, \tau_p\}$ be the index of the pilot used by UE k, and $\boldsymbol{\varphi}_k = \boldsymbol{\phi}_{i_k}$ is the pilot sequence sent by UE k with $\|\boldsymbol{\phi}_{i_k}\|^2 = 1, \forall k$. Note that when $\tau_p < K$, two different UEs might use the same pilot sequence, resulting in the pilot contamination effect [135].

The received signal matrix $\mathbf{Y}_n \in \mathbb{C}^{M \times \tau_p}$ from UE k to the AP n can be written as

$$\mathbf{Y}_n = \sqrt{\tau_p \delta_p} \sum_{k=1}^{K} \mathbf{g}_{nk} \boldsymbol{\varphi}_k^H + \mathbf{W}_n, \tag{4.2.2}$$

where δ_p is the uplink power of each pilot symbol, and $\mathbf{W}_n$ is the Gaussian noise matrix whose elements $[\mathbf{W}_n]_{i,j} \sim \mathcal{CN}(0, \sigma^2)$.

To estimate the channel coefficients $\mathbf{g}_{nk}$ for UE k, we denote the correlation between the received signal matrix $\mathbf{Y}_n$ and the pilot sequence $\boldsymbol{\varphi}_k$ by $\hat{\mathbf{y}}_{nk}$. Mathematically, we have

$$\hat{\mathbf{y}}_{nk} = \mathbf{Y}_n \boldsymbol{\varphi}_k = \sqrt{\tau_p \delta_p} \mathbf{g}_{nk} + \sqrt{\tau_p \delta_p} \sum_{k' \neq k}^{K} \mathbf{g}_{nk'} \boldsymbol{\varphi}_{k'}^H \boldsymbol{\varphi}_k + \mathbf{W}_n \boldsymbol{\varphi}_k. \tag{4.2.3}$$

Utilizing the MMSE [136] for estimation, the estimated channel coefficients, denoted

by $\hat{\mathbf{g}}_{nk}$, could be expressed as

$$\hat{\mathbf{g}}_{nk} = \mathbf{G}_{nk}\hat{\mathbf{Y}}_{nk}^{-1}\hat{\mathbf{y}}_{nk}, \tag{4.2.4}$$

where

$$\begin{aligned} \mathbf{G}_{nk} &\triangleq \mathbb{E}\Big\{\mathbf{g}_{nk}\hat{\mathbf{y}}_{nk}^{H}\Big\} = \sqrt{\tau_p\delta_p}\beta_{nk}\mathbf{I}_M, \\ \hat{\mathbf{Y}}_{nk} &\triangleq \mathbb{E}\Big\{\hat{\mathbf{y}}_{nk}\hat{\mathbf{y}}_{nk}^{H}\Big\} = \Big(\tau_p\delta_p\sum_{k'=1}^{K}\beta_{nk'}|\boldsymbol{\varphi}_{k'}^{H}\boldsymbol{\varphi}_k|^2 + \sigma^2\Big)\mathbf{I}_M. \end{aligned} \tag{4.2.5}$$

On this basis, $\hat{\mathbf{g}}_{nk}$ consists of M independent and identical distributed components and the variance of the m-th element of $\hat{\mathbf{g}}_{nk}$ is defined as

$$\gamma_{nk} = \mathbb{E}\Big\{|[\hat{\mathbf{g}}_{nk}]_m|^2\Big\} = \frac{\tau_p\delta_p\beta_{nk}^2}{\tau_p\delta_p\sum_{k'=1}^{K}\beta_{nk'}|\boldsymbol{\varphi}_{k'}^{H}\boldsymbol{\varphi}_k|^2 + \sigma^2}, \tag{4.2.6}$$

and the MMSE estimation error is denoted as $\tilde{\mathbf{g}}_{nk} = \mathbf{g}_{nk} - \hat{\mathbf{g}}_{nk}$ which follows $\mathcal{CN}(\mathbf{0}, (\beta_{nk} - \gamma_{nk})\mathbf{I}_M)$.

In this chapter, we define two classes of parameters: Class I and Class II. Class I parameters are related to the systems, e.g., the large-scale fading coefficient $\beta_{n,k}$ and the variance of the estimated channel coefficients $\gamma_{n,k}$. Class II parameters are related to the network scale, e.g., the number of APs N and the number of UEs K.

4.2.3 Downlink Data Transmission

Based on the estimated channel coefficients $\hat{\mathbf{g}}_{n,k}$, APs precode data and transmit them to UEs. Conventionally, most work [137–139] adopt CB schemes for beamforming. In this chapter, local-based ZF beamforming is adopted. Unlike conventional ZF, it does not need the CSI transmission from APs to the CPU, as each AP only needs local CSI for beamformer construction. Consequently local-based ZF and CB have the same fronthaul requirements. For $\tau_p \leq K$, some of the estimated channels are parallel, and the channel estimate $\hat{\mathbf{G}}_n = [\hat{\mathbf{g}}_{n,1}, \cdots, \hat{\mathbf{g}}_{n,K}] \in \mathbb{C}^{M\times K}$ is rank-deficient.

Therefore, we define a full-rank channel estimate at AP n as

$$\bar{\mathbf{G}}_n = \mathbf{Y}_n \mathbf{\Phi} \in \mathbb{C}^{M \times \tau_p}, \tag{4.2.7}$$

where $\mathbf{\Phi} = \left[\boldsymbol{\varphi}_1, \cdots \boldsymbol{\varphi}_{\tau_p}\right] \in \mathbb{C}^{\tau_p \times \tau_p}$. The estimated channel coefficients related to $\bar{\mathbf{G}}_n$ could be expressed as

$$\hat{\mathbf{g}}_{nk} = \mathbf{G}_{nk} \hat{\mathbf{Y}}_{nk}^{-1} \bar{\mathbf{G}}_n \mathbf{e}_k, \tag{4.2.8}$$

where $\mathbf{e}_k$ denotes the i_k-th column of $\mathbf{I}_{\tau_p}$. The normalized ZF beamforming vector $\mathbf{a}_{nk} \in \mathbb{C}^{M \times 1}$ of AP n associated with UE k is given by

$$\mathbf{a}_{nk} = \frac{\bar{\mathbf{G}}_n (\bar{\mathbf{G}}_n^H \bar{\mathbf{G}}_n)^{-1} \mathbf{e}_k}{\sqrt{\mathbb{E}\{\|\bar{\mathbf{G}}_n (\bar{\mathbf{G}}_n^H \bar{\mathbf{G}}_n)^{-1} \mathbf{e}_k\|^2\}}}. \tag{4.2.9}$$

Note that (4.2.9) requires $M > \tau_p$. Then, the transmitted data signal by AP n can be written as

$$\mathbf{x}_n = \sum_{k=1}^{K} \sqrt{p_{nk}} \mathbf{a}_{nk} c_k, \tag{4.2.10}$$

where c_k is the data symbol for UE k with $\mathbb{E}\{|c_k|^2\} = 1$, and p_{nk} is the power control coefficient allocated to UE k from AP n. Constrained by maximum transmitted power P_{TX} at each AP, the averaged power of $\mathbf{x}_n$ could be expressed as

$$\mathbb{E}\{\|\mathbf{x}_n\|^2\} = \sum_{k=1}^{K} p_{nk} \mathbb{E}\{\|\mathbf{a}_{nk}\|^2\} \mathbb{E}\{|c_k|^2\} \leq P_{\text{TX}}. \tag{4.2.11}$$

Substituting (4.2.9) and $\mathbb{E}\{|c_k|^2\} = 1$ into (4.2.11), we obtain the power constraint $\sum_{k=1}^{K} p_{nk} \leq P_{\text{TX}}$. With the transmitted signal $\mathbf{x}_n$ for AP n, the received signal at UE k could be formulated as

$$y_k = \sum_{n=1}^{N} \mathbf{g}_{nk}^H \mathbf{x}_n + w_k = \sum_{n=1}^{N} \sum_{k=1}^{K} \sqrt{p_{nk}} \mathbf{g}_{nk}^H \mathbf{a}_{nk} c_k + \sum_{n=1}^{N} \sum_{k' \neq k}^{K} \sqrt{p_{nk'}} \mathbf{g}_{nk'}^H \mathbf{a}_{nk'} c_{k'} + w_k. \tag{4.2.12}$$

Based on (4.2.12), UE k detects the desired data symbol c_k with channel statistics technique [140]. Specifically, there are no downlink pilots and we do not have knowledge about the effective channel gain $\sum_{n=1}^{N} \sqrt{p_{nk}} \mathbf{g}_{nk}^H \mathbf{a}_{nk}$. Instead, we employ the

first-order statistic $\sum_{n=1}^{N}\sqrt{p_{nk}}\mathbb{E}\Big\{\mathbf{g}_{nk}^{H}\mathbf{a}_{nk}\Big\}$ as the effective channel to demodulate c_k. The received signal at UE k on the downlink can be expressed as

$$y_k = \mathrm{DS}_k \cdot c_k + \mathrm{BU}_k \cdot c_k + \sum_{k' \neq k}^{K} \mathrm{UI}_{kk'} \cdot c_{k'} + w_k, \tag{4.2.13}$$

where

$$\begin{aligned}
\mathrm{DS}_k &= \sum_{n=1}^{N} \sqrt{p_{nk}}\ \mathbb{E}\Big\{\mathbf{g}_{nk}^{H}\mathbf{a}_{nk}\Big\}, \\
\mathrm{BU}_k &= \sum_{n=1}^{N} \Big(\sqrt{p_{nk}}\mathbf{g}_{nk}^{H}\mathbf{a}_{nk} - \sqrt{p_{nk}}\ \mathbb{E}\Big\{\mathbf{g}_{nk}^{H}\mathbf{a}_{nk}\Big\}\Big), \\
\mathrm{UI}_{kk'} &= \sum_{n=1}^{N} \sqrt{p_{nk'}}\mathbf{g}_{nk}^{H}\mathbf{a}_{nk'},
\end{aligned} \tag{4.2.14}$$

represent the desired signal, beamforming uncertainty, and inter-user interference, respectively. Following [140], the desired signal DS_k for UE k is deterministic and unrelated to the effective noise including BU_k, $\mathrm{UI}_{kk'}$ and w_k. According to uncorrelated Gaussian noise bound in [141], the effective achievable rate of UE k, denoted as R_k, based on signal-to-interference-plus-noise ratio of the UE k, denoted as SINR_k, could be expressed as

$$R_k = \frac{\tau_c - \tau_p}{\tau_c} B \log_2\big(1 + \mathrm{SINR}_k\big), \tag{4.2.15}$$

where

$$\begin{aligned}
\mathrm{SINR}_k &= \frac{|\mathrm{DS}_k|^2}{\mathbb{E}\{|\mathrm{BU}_k|^2\} + \sum\limits_{k' \neq k}^{K} \mathbb{E}\{|\mathrm{UI}_k|^2\} + \sigma^2} \\
&= \frac{(M - \tau_p)\left|\boldsymbol{p}_k^{H}\boldsymbol{\gamma}_k\right|^2}{(M - \tau_p)\sum\limits_{k'=1}^{K}|\boldsymbol{\varphi}_{k'}^{H}\boldsymbol{\varphi}_k|^2\left|\boldsymbol{p}_{k'}^{H}\boldsymbol{\gamma}_{k'}\right|^2 + \sum\limits_{k'=1}^{K}\boldsymbol{p}_{k'}^{H}\mathbf{B}_{k'}\boldsymbol{p}_{k'} + \sigma^2}.
\end{aligned} \tag{4.2.16}$$

where $\boldsymbol{p}_k = [\sqrt{p_{1k}}, \cdots, \sqrt{p_{Nk}}]^T$ and $\boldsymbol{\gamma}_k = [\sqrt{\gamma_{1k}}, \cdots, \sqrt{\gamma_{Nk}}]^T$. In addition, $\mathbf{B}_k \in \mathbb{C}^{N\times N}$ is the diagonal matrix whose n-th diagonal element is $\beta_{nk} - \gamma_{nk}$. Furthermore,

to evaluate the effective achievable rate performance for all UEs, we denote the sum-rate as $R_{\Sigma} = \sum_{k=1}^{K} R_k$.

4.2.4 Problem Formulation

Energy efficiency in cell-free massive MIMO is a critical issue to be considered. Compared with conventional massive MIMO systems, cell-free massive MIMO systems require a much larger number of APs that are spread out over a large geographical area, and more dedicated fronthaul links are required to connect each AP to the CPU, which potentially increases the total power consumption. Besides, electromagnetic radiation for signal transmission and hardware operation also accounts for a large portion of power consumption. The total energy consumption P_{total} could be formulated as $P_{\text{total}} = \sum_{n=1}^{N} P_n^{\text{A}} + \sum_{n=1}^{N} P_n^{\text{F}}$, where P_n^{A} and P_n^{F} represent power consumption by AP n and power consumption of the fronthaul link to transfer data between AP n and CPU, respectively.

In this chapter, we design a green energy efficiency optimization strategy in cell-free massive MIMO, aiming to minimize the total power consumption. Our key motivation is as follows. There exists a large number of APs distributed to systems. In this case, some APs can be deactivated for energy saving without significantly reducing transmission rates when UEs experience notable propagation loss due to the weak channels associated with these deactivated APs. Here, each AP within the network operates in either active or sleep (deactivated) mode. In active mode, a given AP n can transmit data in the downlink. As expected, P_n^{A} in the active mode is contingent upon the radiated power $P_n^{\text{A,tx}}$. Besides, P_n^{A} also depends on parameters such as the efficiency of the power amplifier, small-signal RF transceiver

power, baseband power, feeder losses, and cooling losses [142]. In sleep mode, the AP n enters a state of diminished power consumption where it remains operationally inactive yet not entirely powered off, thereby enabling prompt reactivation.

Despite the absence of radiation, there are components that are still active and consume power. Consequently, P_n^{A} could be formulated as

$$P_n^{\mathrm{A}} = \begin{cases} \dfrac{P_n^{\mathrm{A,tx}}}{\kappa_n^A} + P_n^{\mathrm{A,chain}} + P_n^{\mathrm{A,fix}}, & \text{Active mode,} \\ P_n^{\mathrm{A,fix}}, & \text{Sleep mode,} \end{cases} \tag{4.2.17}$$

where κ_n^{A} is the power amplifier efficiency, $P_n^{\mathrm{A,chain}}$ represents the power consumption of the circuitry associated with each RF chain, and $P_n^{\mathrm{A,fix}}$ represents the fixed AP power consumption.

The fronthaul links are established between the CPU and APs, and the authors in [30, 143] introduced the fronthaul power consumption model, where the fronthaul power is consumed to transmit power control coefficient $p_{n,k}$ and UEs data. On this basis, the power consumption model can be established for P_n^{F}, and we have

$$P_n^{\mathrm{F}} = \begin{cases} \xi^{\mathrm{F}} R_{\Sigma} + P_n^{\mathrm{F,fix}}, & \text{Active mode,} \\ P_n^{\mathrm{F,fix}}, & \text{Sleep mode,} \end{cases} \tag{4.2.18}$$

where ξ_n^{F} represents the traffic-dependent power coefficient (in W/bps), which is proportional to the sum effective achievable rate, and $P_n^{\mathrm{F,fix}}$ represents fixed fronthaul power consumption.

On this basis, by jointly optimizing power allocation and AP selection, a subset of APs are deactivated to sleep mode resulting in decreased energy consumption and improved energy efficiency. The set of the activated APs is denoted by $\mathcal{A} \subseteq \mathcal{N} = \{1, \cdots, N\}$. Accordingly, the total energy consumption could be expressed as

$$P_{\mathrm{total}} = \sum_{n=1}^{N} (P_n^{\mathrm{A,fix}} + P_n^{\mathrm{F,fix}}) + \sum_{n \in \mathcal{A}} P_n^{\mathrm{A,chain}} + \sum_{n \in \mathcal{A}} \xi^{\mathrm{F}} \sum_{k=1}^{K} R_k + \frac{1}{\kappa_n^{\mathrm{A}}} \sum_{n \in \mathcal{A}} \sum_{k=1}^{K} P_n^{\mathrm{A,tx}}. \tag{4.2.19}$$

When omitting the constant item $\sum_{n=1}^{N}(P_n^{\mathrm{A,fix}} + P_n^{\mathrm{B,fix}})$ in (4.2.19), the energy consumption minimization problem can be formulated by jointly optimizing power allocation and AP selection

$$\mathcal{P}_1 : \min_{p_{nk}\geq 0, \mathcal{A}} \sum_{n\in\mathcal{A}} P_n^{\mathrm{A,chain}} + \sum_{n\in\mathcal{A}} \xi^{\mathrm{F}} \sum_{k=1}^{K} R_k + \frac{1}{\kappa_n^{\mathrm{A}}} \sum_{n\in\mathcal{A}} \sum_{k=1}^{K} \gamma_{nk} p_{nk} \tag{4.2.20}$$

$$\text{s.t. } R_k \geq \eta_k, \tag{4.2.20a}$$

$$\sum_{k=1}^{K} p_{nk} \leq P_{\mathrm{TX}}, \tag{4.2.20b}$$

where η_k is QoS requirement at UE k. The objective function in (4.2.20) represents the optimized power consumption P_{opt} in the downlink based on activated APs in $\mathcal{A}$ and the power control coefficient p_{nk} subject to QoS requirements and transmit power limitations P_{TX}.

However, $\mathcal{P}_1$ is quite complicated due to the following observations.

- It can be shown that $\mathcal{P}_1$ is an MINLP problem combining discrete variables related to activated APs in $\mathcal{A}$ and continuous variables related to power control coefficients $\{p_{nk}\}$. Typically, the branch and bound (B&B) algorithm [87] could be implemented to solve the MINLP problems by conducting a systematic search over the discrete variables via an enumeration tree. However, the computational complexity is exponential with the increased number of discrete variables.

- In essence, $\mathcal{P}_1$ is NP-hard [144]. Here, $\mathcal{P}_1$ consists of $2^{|\mathcal{A}|}$ non-convex subproblems, where $|\mathcal{A}|$ is the number of APs in $\mathcal{A}$. In other words, its computational complexity increases exponentially with the dimension of $\mathcal{A}$, and generally, no optimal solution could be found by a polynomial-time algorithm.

- When Class II parameters such as the channel coefficients $\mathbf{g}_{nk}$ significantly

change, $\mathcal{P}_1$ needs to be resolved from the scratch.

This calls for efficient strategies that scale favorably with the problem size and adapt to dynamic network environments, enabling fast AP selection and power allocation.

4.2.5 Problem Transformation

Before introducing our proposed OSAC-G algorithm, we first simplify the complicated MINLP problem $\mathcal{P}_1$. Specifically, we consider the strong correlation between the activated APs in $\mathcal{A}$ and the power control coefficients $\{p_{nk}\}$. If all power control coefficients for AP n are zero, i.e., $\{p_{n,k} = 0, k = 1, \cdots, K\}$, AP n can be deactivated and goes into sleep mode, which will significantly increase energy efficiency. Therefore, the activated APs in $\mathcal{A}$ can be simultaneously captured by their power control coefficients. Here, we first define a power allocation coefficient as $q_{nk} = \sqrt{p_{nk}}$, and its corresponding matrix is denoted as $\mathbf{Q} \in \mathbb{R}^{N\times K}$ with $[\mathbf{Q}]_{nk} = q_{nk}$. For $\mathbf{Q}$, its k-th column vector $\boldsymbol{q}_k = [q_{1k}, \cdots, q_{Nk}]^T$ and its n-th row vector $\hat{\boldsymbol{q}}_n = [q_{n1}, \cdots, q_{nK}]^T$ represent power allocation vectors for UE k and AP n, respectively. Obviously, if we have $\|\hat{\boldsymbol{q}}_n\|_2 = 0$, the AP n is deactivated. Then, we further denote $\hat{\mathbf{Q}} = [\|\hat{\boldsymbol{q}}_1\|_2, \cdots, \|\hat{\boldsymbol{q}}_N\|_2]^T$, and norm-zero $\|\hat{\mathbf{Q}}\|_0$ could represent the number of activated APs. Accordingly, $\mathcal{P}_1$ could be reformulated as

$$\mathcal{P}_2 : \min_{q_{nk}\geq 0} \|\hat{\mathbf{Q}}\|_0 (P_n^{\mathrm{A,chain}} + \xi^{\mathrm{F}} \bar{B} \sum_{k=1}^{K} \bar{R}_k) + \frac{1}{\kappa_n^{\mathrm{A}}} \sum_{n=0}^{N} \sum_{k=1}^{K} \gamma_{nk} q_{nk}^2 \tag{4.2.21}$$

$$\text{s.t. } \bar{R}_k \geq \bar{\eta}_k, \tag{4.2.21a}$$

$$\sum_{k=1}^{K} q_{nk}^2 \leq P_{\mathrm{TX}}, \tag{4.2.21b}$$

where $\bar{\eta}_k = \frac{\eta_k}{B}$, $\bar{B} = \frac{\tau_c - \tau_p}{\tau_c} \cdot B$, and $\bar{R}_k = \log_2 \left(1 + \frac{(M-\tau_p)\left|\boldsymbol{p}_k^H \boldsymbol{\gamma}_k\right|^2}{(M-\tau_p) \sum\limits_{k'=1}^{K} |\boldsymbol{\varphi}_{k'}^H \boldsymbol{\varphi}_k|^2 \left|\boldsymbol{p}_{k'}^H \boldsymbol{\gamma}_{k'}\right|^2 + \sum\limits_{k'=1}^{K} \boldsymbol{p}_{k'}^H \mathbf{B}_{k'} \boldsymbol{p}_{k'} + \sigma^2}\right)$.

With $\|\hat{\mathbf{Q}}\|_0$, the MINLP problem $\mathcal{P}_1$ with mixed variables are converted to $\mathcal{P}_2$ only with continuous variables. Utilizing the continuous variable q_{nk}, power allocation and AP selection are jointly optimized. However, $\mathcal{P}_2$ is still very challenging due to its non-convexity for $\|\hat{\mathbf{Q}}\|_0$ and $\bar{R}_k$. Conventionally, SAM is a viable solution for non-convex optimization [145]. This method starts with an initial approximation of the root. Then a series of first-order approximations are implemented to the non-convex constraints for solving the convex approximations iteratively until the desired level of accuracy is achieved. However, this method suffers from three major limitations:

- (**L1**) Due to the non-convexity of $\mathcal{P}_2$, this method features high computational complexity as a large number of iterations are involved to converge to a satisfactory local optimal solution. The detailed complexity analysis will be elaborated on in Section 4.5.2

- (**L2**) Once Class I parameters (e.g., β_{nk} and γ_{nk}) vary, $\mathcal{P}_2$ has to be solved again, the complexity of which impedes the real-time power allocation and AP selection.

- (**L3**)Similarly, whenever Class II parameters (e.g., the number of APs N and the number of UEs K) change, $\mathcal{P}_2$ needs to be resolved. Moreover, with the increase of N or K, the computational complexity increases significantly.

4.3 The Proposed OSAC-G Algorithm

In this section, we first provide an overall picture of the proposed OSAC-G algorithm to solve the non-convex formulated problem in downlink cell-free massive MIMO. Then, we provide the preliminary and develop OSAC-G by embedding non-convex optimization into contemporary DRL methods. Finally, we introduce the components of OSAC-G, including pre-processing module, successive approximation module, critic network module, and GTN actor network module.

4.3.1 Overall Picture of OSAC-G Algorithm

DRL combines deep learning and reinforcement learning to take actions in complex situations with varying Class I parameters [146]. DRL learns a mapping from the input states (e.g., Class I parameters) to output actions (e.g., power allocation and AP selection), and a model is trained first by maximizing a predefined reward function. On this basis, DRL can directly online infer optimal power allocation and AP selection, resulting in a much lower computational complexity. This has the potential to address (**L1**) and (**L2**). However, DRL alone cannot guarantee that all constraints in the formulated problem can be strictly satisfied either during training or online inference, and cannot capture the complex data structure of solutions. Moreover, the conventional DRL cannot handle (**L3**), because the DRL model needs to be trained from scratch whenever there is a significant variation in the Class II parameters.

The above observation motivates us to develop OSAC-G, which embeds non-convex optimization (SAM) into the DRL method (SAC) to directly online infer solutions for the proposed non-convex problems. It features significantly lower complexity while guaranteeing constraints are satisfied in a large probability. We leverage

SAM to incorporate both optimization objectives and constraints into the design of the reward function. The designed reward function involves solution performance evaluation, and the decision-making of joint power allocation and AP selection by OSAC-G is further improved. Besides, incorporating the soft policy updates and entropy regularization, OSAC-G can also stabilize learning and improve convergence.

4.3.2 The Preliminary of OSAC-G Algorithm

As shown in Fig. 4.1, the proposed OSAC-G algorithm consists of two phases: the offline training phase and the online inference phase. In the offline training phase, a significant amount of time to converge is required, but this training is a one-off offline process and its complexity can be ignored from the overall perspective. Once the offline training is completed, online inference for directly mapping the state to the action (power allocation and AP selection) can be made in real time with very low complexity. Next, we provide the preliminary in this subsection, including state, action, reward function, and the soft value function.

State: For the channel between AP n and UE k at time step t, the state is denoted as $\boldsymbol{\chi}_{n,k}(t) = \{\beta_{n,k}(t), \gamma_{n,k}(t), c_{\mathrm{I}}'(t)\}$, where $c_{\mathrm{I}}'(t)$ represents Class I parameters, which includes the distance $d_{n,k}$ between AP n and UE k, QoS requirement η_k and maximum transmitted power P_{TX}, excluding $\beta_{n,k}(t)$ and $\gamma_{n,k}(t)$. Besides, we denote the state of the system as $\boldsymbol{\chi}(t)$ with $[\boldsymbol{\chi}(t)]_{n,k} = \boldsymbol{\chi}_{n,k}(t)$. Consequently, the state space could be represented as $\mathcal{X} = \{\boldsymbol{\chi}(1), \boldsymbol{\chi}(2), \boldsymbol{\chi}(3), \cdots\}$.

Action: The agent strategically takes an action for the channel between AP n and UE k, and the action is represented by $\varrho_{n,k}(t) = q_{n,k}(t)$ at time step t, and the action of the system is denoted as $\boldsymbol{\varrho}(t)$ with $[\boldsymbol{\varrho}(t)]_{n,k} = \varrho_{n,k}(t)$. Consequently, the

action space could be represented as $\mathcal{Y} = \{\boldsymbol{\varrho}(1), \boldsymbol{\varrho}(2), \boldsymbol{\varrho}(3), \cdots\}$.

Reward: We denote reward for the state-action pair $\{\boldsymbol{\chi}(t), \boldsymbol{\varrho}(t)\}$ as $Z(\boldsymbol{\chi}(t), \boldsymbol{\varrho}(t))$, which measures the performance of the action $\boldsymbol{\varrho}(t)$ and makes the agent take better decisions. Given $Z(\boldsymbol{\chi}(t), \boldsymbol{\varrho}(t))$, the DRL attempts to learn an optimal policy π^* capable of effectively mapping a given state $\boldsymbol{\chi}(t)$ to an optimal action $\boldsymbol{\varrho}(t)^*$. As $\boldsymbol{\varrho}(t)$ provides power control for each UE. Thus, $Z(\boldsymbol{\chi}(t), \boldsymbol{\varrho}(t))$ measures the performance of $\boldsymbol{\varrho}(t)$ based on whether the corresponding scheme minimizes the total power consumption in (4.2.21) subject to constraints (4.2.21a) - (4.2.21b).

Obviously, the objective function in (4.2.21) should be considered in the design of the reward function aimed at guiding the agent. At time step t, the actor of OSAC-G directly online infers the action $\boldsymbol{\varrho}(t)$ with $[\boldsymbol{\varrho}(t)]_{n,k} = q_{n,k}$. By substituting $\{q_{n,k} \mid n = 1, \cdots N,\ k = 1, \cdots, K\}$ into the objective of $\mathcal{P}_2$, we can obtain $P_{\text{opt}}(t)$. On this basis, the reward function related to the objective function is expressed as

$$Z_{\text{obj}}(t) = |P_{\text{opt}}(t) - P^{\text{s}}_{\text{opt}}(t)|, \tag{4.3.1}$$

where $P^{\text{s}}_{\text{opt}}(t)$ is the optimized power consumption solved by successive approximation module to be specified in Section 4.3.3. The reason why we incorporate (4.3.1) in the design of the reward function is that the solution online inferred by OSAC-G could asymptotically approach that calculated by the successive approximation module.

Besides, if power allocation coefficients inferred by OSAC-G approach those calculated by the SAM, an incentive is given to OSAC-G and we have

$$Z_{\text{pc}}(t) = \sum_{n=1}^{N} \sum_{k=1}^{K} \Big| q_{nk}(t) - q^{\text{s}}_{nk}(t) \Big|. \tag{4.3.2}$$

Due to the complex constraints in non-convex optimization problems, it is challenging for the DRL alone to learn a policy while strictly following all the constraints. Therefore, we take constraints into account in our reward function design. Specifically,

if the solution solved by OSAC-G cannot satisfy the constraints of the formulated problem, a penalty is given to OSAC-G, and we have

$$Z_{\text{cstr}}(t) = \sum_{k=1}^{K} \omega_{\eta} \text{sgn}(\bar{R}_k(t) - \bar{\eta}_k) + \omega_{\gamma} \text{sgn}(P_{\text{TX}} - \sum_{k=1}^{K} \gamma_{nk}(t) q_{nk}(t)^2), \tag{4.3.3}$$

where ω_{η} and ω_{γ} are the penalty coefficients for constraints (4.2.21a) and (4.2.21b), respectively, and $\text{sgn}(\cdot)$ is a sign function. The total reward of $\{\boldsymbol{\chi}(t), \boldsymbol{\varrho}(t)\}$ is given by

$$Z(\boldsymbol{\chi}(t), \boldsymbol{\varrho}(t)) = \omega_{\text{obj}} Z_{obj}(t) + \omega_{\text{pc}} Z_{\text{pc}}(t) + \omega_{\text{cstr}} Z_{\text{cstr}}(t), \tag{4.3.4}$$

where ω_{obj}, ω_{pc} and ω_{cstr} are the penalty coefficients. Clearly, our designed reward function comprehensively considers three factors: objective function, solved solutions, and constraints.

Soft Value Function: SAC is one of the leading-edge off-policy DRL algorithms, and it comprises a policy network acting as an actor and a Q-value network performing as a critic [147]. Different from the objective of standard DRL, SAC considers the objective with maximum entropy $\mathcal{H}(\pi(\boldsymbol{\varrho}(t)|\boldsymbol{\chi}(t))) = \mathbb{E}_{\boldsymbol{\varrho}(t)\sim\pi}[-\log \pi(\boldsymbol{\varrho}(t)|\boldsymbol{\chi}(t))]$ and the expected sum of reward, where $\pi(\boldsymbol{\varrho}(t)|\boldsymbol{\chi}(t))$ denotes the probability of taking action $\boldsymbol{\varrho}(t)$ based on $\boldsymbol{\chi}(t)$ under π. The optimal policy π^* is expressed as

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{(\boldsymbol{\chi}(t), \boldsymbol{\varrho}(t))\sim\rho_{\pi}} \left\{ \sum_{t=0}^{\infty} \gamma^t \left[Z(\boldsymbol{\chi}(t), \boldsymbol{\varrho}(t)) + \alpha \mathcal{H}(\pi(\boldsymbol{\varrho}(t)|\boldsymbol{\chi}(t))) \right] \right\}, \tag{4.3.5}$$

where ρ_{π} is the state-action marginals of the trajectory distribution induced by $\pi(\boldsymbol{\varrho}(t)|\boldsymbol{\chi}(t))$, $\gamma \in [0, 1]$ is the discount factor. In addition, $\alpha \in [0, \infty]$ is the temperature parameter [131] which determines the significance of the entropy term $(\mathcal{H}(\pi(\boldsymbol{\varrho}(t)|\boldsymbol{\chi}(t))))$ relative to the reward $(Z(\boldsymbol{\chi}(t), \boldsymbol{\varrho}(t)))$, and it controls the stochasticity of the optimal policy. With a larger α, the stochasticity of the policy increases, and the agent is more likely to explore and take random actions.

The objective of SAC is to learn three categories of neural networks including

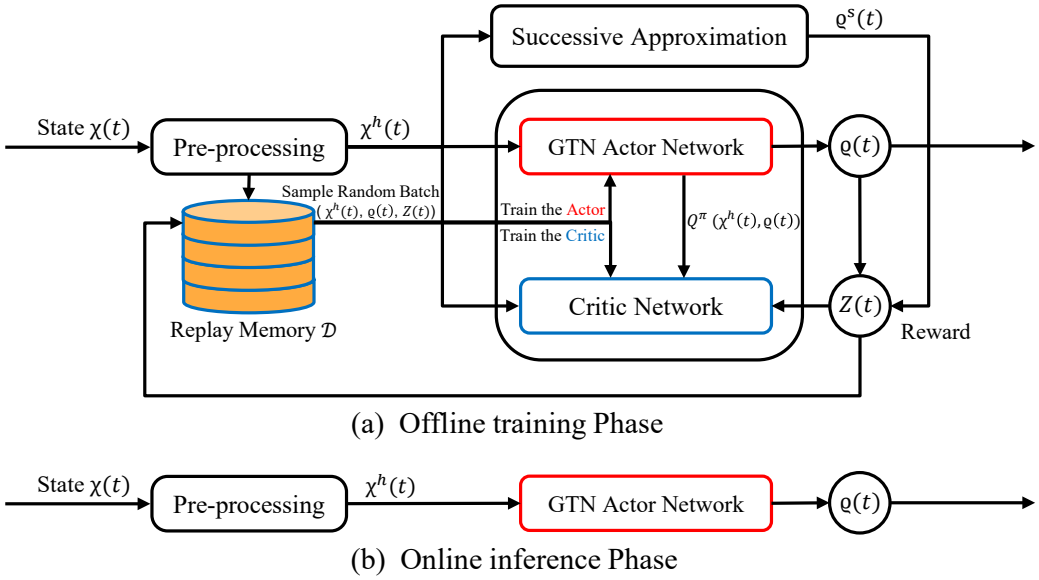


Figure 4.1: Our proposed OSAC-G Algorithm.

parameterized target value network $V_\psi(\boldsymbol{\chi}(t))$, the soft value network $Q_\phi(\boldsymbol{\chi}(t), \boldsymbol{\varrho}(t))$ (critic network) and the policy network $\pi_\theta(\boldsymbol{\varrho}(t)|\boldsymbol{\chi}(t))$ (actor network). The parameters ψ, ϕ, and θ are improved iteratively during the learning process, and their initialization is executed to fulfill the criteria of conforming to the standard normal distribution at the beginning of OSAC-G. Specifically, the soft Q-function parameters serve as an implicit parameterization for the target value function

$$V_\psi(\boldsymbol{\chi}(t)) = \mathbb{E}_{\boldsymbol{\varrho}(t)\sim\pi}\Big[Q(\boldsymbol{\chi}(t), \boldsymbol{\varrho}(t)) - \alpha \log \pi(\boldsymbol{\varrho}(t)|\boldsymbol{\chi}(t))\Big], \tag{4.3.6}$$

and target value network is trained through the minimization of the squared residual error with the objective

$$L_V(\psi) = \mathbb{E}_{\boldsymbol{\chi}(t)\sim\mathcal{D}}\Big[\frac{1}{2}\Big(V_\psi(\boldsymbol{\chi}(t)) - \mathbb{E}_{\boldsymbol{\varrho}(t)\sim\pi_\theta}[Q_\phi(\boldsymbol{\chi}(t), \boldsymbol{\varrho}(t)) - \log \pi_\theta(\boldsymbol{\varrho}(t)|\boldsymbol{\chi}(t))]\Big)^2\Big]. \tag{4.3.7}$$

Algorithm 2 Optimization-embedded Soft Actor-Critic with Graph Transformer Networks (OSAC-G) Algorithm.

Initialize target value network parameters ψ, $\bar{\psi}$; critic network parameters ϕ; actor network parameters θ.
for each iteration **do**
  for each environment step **do**
    Observe the environment (state) $\boldsymbol{\chi}(t)$ and compute $\boldsymbol{\chi}^h(t)$ with pre-processing.
    Calculate the mean and standard deviation $(\boldsymbol{\mu}^g(t), \boldsymbol{\sigma}^g(t))$ and take action $\boldsymbol{\varrho}(t)$ by the GTN actor network with θ.
    Compute $\boldsymbol{\varrho}^{\mathrm{s}}(t)$ by the successive approximation module, and receive the reward $Z(\boldsymbol{\chi}^h(t), \boldsymbol{\varrho}(t))$.
    Incorporate the following tuple into the replay memory buffer $\mathcal{D} \leftarrow \mathcal{D} \cup \{\big(\boldsymbol{\varrho}(t), \boldsymbol{\chi}^h(t), Z(\boldsymbol{\chi}^h(t), \boldsymbol{\varrho}(t))\big)\}$.
  end for
  for each step **do**
    Select a batch of samples $\big(\boldsymbol{\varrho}(t), \boldsymbol{\chi}^h(t), Z(\boldsymbol{\chi}^h(t), \boldsymbol{\varrho}(t))\big)$ from the replay memory buffer $\mathcal{D}$ in a randomized manner.
    Compute $L_V(\psi)$ and update ψ according to (4.3.7).
    Compute $L_Q(\phi)$ and update ϕ according to (4.3.14).
    Compute $L_\pi(\theta)$ and θ according to (4.3.18).
    Update target value parameters $\bar{\psi} \leftarrow \delta\psi + (1-\delta)\bar{\psi}$.
  end for
end for

4.3.3 OSAC-G Algorithm

Next, we provide the workflow of the offline training of OSAC-G, followed by elaboration on the design of its components[1]. Its workflow is summarized in **Algorithm 2**. Once the neural network parameters ψ, ϕ, and θ are initialized, each iterative step comprises two distinct phases: 1) acquisition of experience from the

[1]In OSAC-G, standard procedures of RL include critic network module and replay memory buffer, and our novel contributions include the pre-processing, successive approximation module, GTN, and the handcraft reward function., including pre-processing module, successive approximation module, critic network module, and GTN actor network module.

environment (Lines $3-8$), and 2) training and modification of the network parameters (Lines $9-15$). During the first phase, we pre-process the system state $\boldsymbol{\chi}(t)$ and obtain $\boldsymbol{\chi}^h(t)$, based on which the GTN actor with θ takes action $\boldsymbol{\varrho}(t)$. Interacting with the environment, the successive approximation module computes the solution $\boldsymbol{\varrho}^s(t) = \{\boldsymbol{q}^s(t), \mathbf{x}^s(t), \mathbf{y}^s(t), \mathbf{z}^s(t), u^s(t)\}$, and the reward will be duly noted and recorded within the replay memory buffer $\mathcal{D}$. During the second phase, a batch of samples is chosen randomly from $\mathcal{D}$ to facilitate the training process of the target value network, the critic network, and the GTN actor network. Then, the neural network parameters $(\psi, \bar{\psi}, \phi, \theta)$ are updated.

Pre-Processing Module

The pre-processing module prepares the raw input state $\boldsymbol{\chi}(t)$ as the input to the agent of OSAC-G (CPU). The input state $\boldsymbol{\chi}(t)$ is defined in Section 4.3.2. Clearly, the variables $\beta_{n,k}(t)$ and $\gamma_{n,k}(t)$ in $\boldsymbol{\chi}_{n,k}(t)$ are spread out over different ranges. Consequently, in order to avoid convergence speed degradation in the training process, we have to adopt a lower training rate, and complicated parameter initialization is needed. We implement the min-max normalization to the components $\beta_{n,k}(t)$ and $\gamma_{n,k}(t)$ which are given by

$$\begin{aligned} \beta'_{n,k}(t) &= \frac{\beta_{n,k}(t) - \beta_{\min}(t)}{\beta_{\max}(t) - \beta_{\min}(t)}, n \in \{1, \cdots N\}, k \in \{1, \cdots, K\}, \\ \gamma'_{n,k}(t) &= \frac{\gamma_{n,k}(t) - \gamma_{\min}(t)}{\gamma_{\max}(t) - \gamma_{\min}(t)}, n \in \{1, \cdots N\}, k \in \{1, \cdots, K\}, \end{aligned} \tag{4.3.8}$$

where $\beta_{\max}(t)$ and $\beta_{\min}(t)$ represent the maximum and minimum of large-scale fading coefficients at time step t, respectively, and $\gamma_{\max}(t)$ and $\gamma_{\min}(t)$ represent the maximum and minimum of variances of channel estimation at time step t, respectively. Here, $\beta_{\max}(t)$, $\beta_{\min}(t)$, $\gamma_{\max}(t)$ and $\gamma_{\min}(t)$ are selected from the state $[\boldsymbol{\chi}(t)]_{n,k} = \boldsymbol{\chi}_{n,k}(t)$.

Accordingly, we obtain the new state $\boldsymbol{\chi}'_{n,k}(t)$ with normalized $\beta'_{n,k}(t)$ and $\gamma'_{n,k}(t)$.

The integration of past states is a pivotal process in crafting a holistic representation that encapsulates the historical and current states. Then, we transform $\boldsymbol{\chi}(t)$ into

$$\begin{aligned} \boldsymbol{\chi}^h_{n,k}(t) &= (\boldsymbol{\chi}'_{n,k}(1), \cdots, \boldsymbol{\chi}'_{n,k}(t-T+1)), \\ [\boldsymbol{\chi}^h(t)]_{n,k} &= \boldsymbol{\chi}^h_{n,k}(t), \end{aligned} \tag{4.3.9}$$

where T denotes the historical state sequence length. As T increases, the agent's decision-making capabilities improve [148]. However, a substantial value of T can lead to a significantly larger state space, which, in turn, can impede the learning speed of OSAC-G. The choice of T should strike a harmonious equilibrium between enhancing decision-making capabilities and mitigating the associated complexity.

Successive Approximation Module

To design an efficient reward function, we embed the successive approximation module in our OSAC-G. In the successive approximation module, we implement SAM to solve $\mathcal{P}_2$. Specifically, we first introduce some auxiliary variables to $\mathcal{P}_2$. Then, we apply the first approximation to the constraints and obtain $\mathcal{P}_3$ to be detailed in the sequel. Finally, we solve $\mathcal{P}_3$ iteratively until a solution with good performance is obtained. In particular, we introduce auxiliary positive variables u, x_k, y_k, z_k into $\mathcal{P}_2$. Moreover, to handle the non-convex constraints, a first-order approximation is

utilized. Accordingly, $\mathcal{P}_2$ has been transformed into

$$\mathcal{P}_3 : \min_{\mathbf{q}_k, x_k, y_k, z_k, u} = u P_n^{\mathrm{A,chain}} + \xi^{\mathrm{B}} \bar{B} \sum_{k=1}^{K} x_k + \frac{1}{\kappa_n^{\mathrm{A}}} \sum_{n=0}^{N} \sum_{k=1}^{K} \gamma_{nk} q_{nk}^2 \tag{4.3.10}$$

$$\text{s.t.} \quad (M-\tau_p) \sum_{k'=1}^{K} |\boldsymbol{\varphi}_{k'}^H \boldsymbol{\varphi}_k|^2 \left|\boldsymbol{q}_{k'}^H \boldsymbol{\gamma}_{k'}\right|^2 + \sum_{k'=1}^{K} \boldsymbol{q}_{k'}^H \mathbf{B}_{k'} \boldsymbol{q}_{k'} + \sigma^2 \leq \frac{(M-\tau_p)}{\bar{\eta}_k} \left[2(\boldsymbol{q}_k^H \boldsymbol{\gamma}_k)(\dot{\boldsymbol{q}}_k^H \boldsymbol{\gamma}_k) - \left|\dot{\boldsymbol{q}}_k^H \boldsymbol{\gamma}_k\right|^2\right], \tag{4.3.10a}$$

$$(u + y_k)^2 \leq 2x_k + \dot{u}(2u - \dot{u}) + \dot{y}_k(2y_k - \dot{y}_k), \tag{4.3.10b}$$

$$\|\hat{\mathbf{Q}}\|_0 \leq u, \tag{4.3.10c}$$

$$1 + z_k \leq e^{y_k}(y_k - \dot{y}_k + 1), \tag{4.3.10d}$$

$$\frac{(M-\tau_p)\left|\boldsymbol{q}_k^H \boldsymbol{\gamma}_k\right|^2}{z_k} \leq -\sum_{k'=1}^{K} \dot{\boldsymbol{q}}_{k'}^H \mathbf{B}_{k'} \dot{\boldsymbol{q}}_{k'} 2(M-\tau_p) \sum_{k'=1}^{K} |\boldsymbol{\varphi}_{k'}^H \boldsymbol{\varphi}_k|^2 (\boldsymbol{q}_{k'}^H \boldsymbol{\gamma}_{k'})(\dot{\boldsymbol{q}}_{k'}^H \boldsymbol{\gamma}_{k'}) + 2\sum_{k'=1}^{K} \boldsymbol{q}_{k'}^H \mathbf{B}_{k'} \dot{\boldsymbol{q}}_{k'} - (M-\tau_p) \sum_{k'=1}^{K} |\boldsymbol{\varphi}_{k'}^H \boldsymbol{\varphi}_k|^2 \left|\dot{\boldsymbol{q}}_k^H \boldsymbol{\gamma}_k\right|^2 + \sigma^2, \tag{4.3.10e}$$

$$\sum_{k=1}^{K} q_{nk}^2 \leq P_{\mathrm{TX}}. \tag{4.3.10f}$$

It is clear that, in obtaining the objective function (4.3.10), $\bar{R}_k$ and $\|\hat{\mathbf{Q}}\|_0$ in (4.3.10) are substituted by u and x_k, respectively, and the equivalence of $\mathcal{P}_2$ and $\mathcal{P}_3$ is guaranteed with constraints (4.3.10b), (4.3.10c), (4.3.10d) and (4.3.10e). To further simplify the constraint (4.3.10c), we can write the block-sparsity norm $l_{2,1}$ of $\mathbf{L} \in \mathbb{R}^{N \times M}$ as $\|\mathbf{L}\|_{2,1} = \sum_{n=1}^{N} \sqrt{\sum_{n=1}^{M} L_{n,m}^2}$, and the constraint (4.3.10c) could be approximated as $\|\mathbf{Q}\|_{2,1} \leq \lambda u$, where λ is the scaling factor. Overall, the application of $\|\mathbf{L}\|_{2,1}$ results in the convex transformation of constraint (4.3.10c) along with the convex objective function (4.3.10).

The detailed workflow of the SAM is provided in **Algorithm 3**. The obtained solution is denoted as $\boldsymbol{\varrho}^{\mathrm{s}}(t) = \{\boldsymbol{q}^{\mathrm{s}}(t), \mathbf{x}^{\mathrm{s}}(t), \mathbf{y}^{\mathrm{s}}(t), \mathbf{z}^{\mathrm{s}}(t), u^{\mathrm{s}}(t)\}$, where $\boldsymbol{q}^{\mathrm{s}}(t) = [\boldsymbol{q}_1^{\mathrm{s}}(t), \cdots \boldsymbol{q}_K^{\mathrm{s}}(t)]^T$, $\mathbf{x}^{\mathrm{s}}(t) = [x_1^{\mathrm{s}}(t), \cdots x_K^{\mathrm{s}}(t)]^T$, $\mathbf{y}^{\mathrm{s}}(t) = [y_1^{\mathrm{s}}(t), \cdots y_K^{\mathrm{s}}(t)]^T$ and $\mathbf{z}^{\mathrm{s}}(t) = [z_1^{\mathrm{s}}(t), \cdots z_K^{\mathrm{s}}(t)]^T$.

Algorithm 3 Successive Approximation Method (SAM).

Initialize a feasible solution $\dot{\boldsymbol{q}}_k$, $\dot{y}_k$, $\dot{u}$; a tolerance ϵ^{s};
for each iteration **do**
  Solve the problem presented in (4.3.10) and derive the corresponding solution $\boldsymbol{q}_k^{\mathrm{s}}$, x_k^{s}, y_k^{s}, z_k^{s}, u^{s}, $\forall k$.
  Compute $P_{\mathrm{opt}}^{\mathrm{s}}(t)$.
  If $|P_{\mathrm{opt}}(t) - P_{\mathrm{opt}}^{\mathrm{s}}(t)| \leq \epsilon^{\mathrm{s}} \rightarrow$ Stop iteration.
  Update $\hat{\boldsymbol{q}}_k = \boldsymbol{q}_k^{\mathrm{s}}$; $\dot{y}_k = y_k^{\mathrm{s}}$; $\dot{u} = u^{\mathrm{s}}$; $P_{\mathrm{opt}}(t) = P_{\mathrm{opt}}^{\mathrm{s}}(t)$.
end for

With the solution above, we can calculate the optimized power consumption $P_{\mathrm{opt}}^{\mathrm{s}}$, and design the reward function based on (4.3.1) to obtain a more accurate guide than the cumulative reward. With $\boldsymbol{q}^{\mathrm{s}}(t)$, we can design the reward function based on (4.3.2) to directly guide the actor by the relationship between optimal solution q_{nk}^{s} and the action generated by the actor. With the aid of successive approximation module, the training of OSAC-G could be accelerated significantly, and a better policy could be learned.

Critic Network Module

Here, we design the critic module as shown in Fig. 4.2(b). Given the input $(\boldsymbol{\chi}^h(t), \boldsymbol{\varrho}(t))$, the critic generate the corresponding expectation of $Q^{\pi}(\boldsymbol{\chi}^h(t), \boldsymbol{\varrho}(t))$ including the long-term reward and entropy. The embedding layer is employed to encode the state $\boldsymbol{\chi}^h(t)$ into $\boldsymbol{\chi}^c(t)$, and normalize the action $\boldsymbol{\varrho}(t)$ a fixed dimension d_c

Then, we apply three fully connected layers, which are 2-layer perceptron with *relu* activation functions, to evaluate $Q^{\pi}(\boldsymbol{\chi}^h(t), \boldsymbol{\varrho}(t))$. Specifically, $Q^{\pi}(\boldsymbol{\chi}^h(t), \boldsymbol{\varrho}(t))$ is evaluated by the soft Q-value $Q_{\phi}(\boldsymbol{\chi}^h(t), \boldsymbol{\varrho}(t))$ parameterized by ϕ, and trained through the minimization of the soft Bellman residual with the objective

$$L_Q(\phi) = \mathbb{E}_{(\boldsymbol{\chi}^h(t), \boldsymbol{\varrho}(t)) \sim \mathcal{D}} \Big[\frac{1}{2} \Big(Q_{\phi}(\boldsymbol{\chi}^h(t), \boldsymbol{\varrho}(t)) - \hat{Q}(\boldsymbol{\chi}^h(t), \boldsymbol{\varrho}(t)) \Big)^2 \Big], \tag{4.3.11}$$

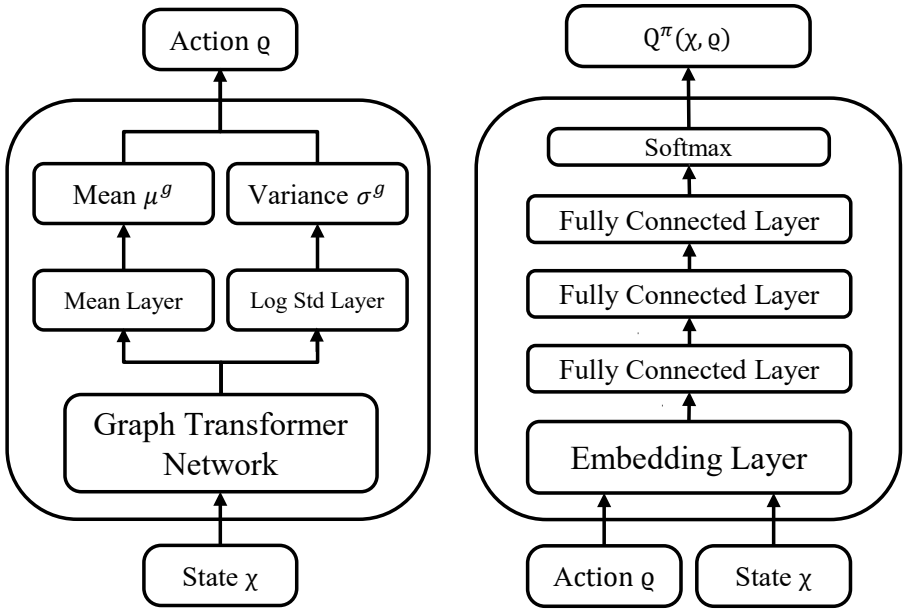


Figure 4.2: (a) The structure of actor. (b) The structure of critic.

where

$$\hat{Q}(\boldsymbol{\chi}^h(t), \boldsymbol{\varrho}(t)) = Z(\boldsymbol{\chi}^h(t), \boldsymbol{\varrho}(t)) + \gamma \mathbb{E}_{\boldsymbol{\chi}^h(t+1)}[V_{\bar{\psi}}(\boldsymbol{\chi}^h(t+1))] \tag{4.3.12}$$

is the target soft Q function and $V_{\bar{\psi}}$ is the target value to stabilize the learning process. Accordingly, ϕ undergo stochastic gradient updates

$$\begin{aligned}\hat{\nabla}_\phi L_Q(\phi) = \nabla_\phi Q_\phi(\boldsymbol{\chi}^h(t), \boldsymbol{\varrho}(t))\Big(Q_\phi(\boldsymbol{\chi}^h(t), \boldsymbol{\varrho}(t)) - (Z(\boldsymbol{\chi}^h(t), \boldsymbol{\varrho}(t)) \\ + \gamma(Q_{\bar{\phi}}(\boldsymbol{\chi}^h(t+1), \boldsymbol{\varrho}(t+1)) - \alpha \log \pi_\theta(\boldsymbol{\chi}^h(t+1), \boldsymbol{\varrho}(t+1)))\Big),\end{aligned} \tag{4.3.13}$$

$$\phi \leftarrow \phi - \lambda_c \hat{\nabla}_\phi L_Q(\phi), \tag{4.3.14}$$

where λ_c is the learning rate of the critic, and $Q_{\bar{\phi}}$ is the target soft Q function. Here, a replay memory buffer $\mathcal{D}$ with size S is maintained to store newly generated tuples $\{(\boldsymbol{\varrho}(t), \boldsymbol{\chi}^h(t), Z(\boldsymbol{\chi}^h(t), \boldsymbol{\varrho}(t)))\}$ at each environment step. Upon filling the replay memory, during each iteration, a mini-batch is sampled from $\mathcal{D}$, and the critic network is subjected to an update process whereby the minimization of $L_Q(\phi)$ is performed.

GTN Actor Network Module

We design the GTN actor network module as shown in Fig. 4.2(a). Given the input state $\boldsymbol{\chi}^h(t)$, the actor takes action and returns $\boldsymbol{\varrho}(t)$. The actor is composed of a GTN and an action generator. The former is designed to capture the graphical characteristics of APs and UEs and enhance the model generalization capability of OSAC-G. Given the input state $\boldsymbol{\chi}^h(t)$, the GTN generates the mean $\boldsymbol{\mu}^g(t) \in \mathbb{R}^{N\times K\times \mu^g}$ and the standard deviation $\boldsymbol{\sigma}^g(t) \in \mathbb{R}^{N\times K\times \sigma^g}$, where μ^g and σ^g are the size of the elements of $\boldsymbol{\mu}^g(t)$ and $\boldsymbol{\sigma}^g(t)$, respectively. The GTN is to be elaborated in Section 4.4. Then, the action generator generates actions utilizing the Gaussian distributions, as follows:

$$\begin{gathered} q_{11}(t) \sim \mathcal{CN}(\boldsymbol{\mu}^g_{11}(t), \boldsymbol{\sigma}^g_{11}(t)), \\ \cdots \\ q_{nk}(t) \sim \mathcal{CN}(\boldsymbol{\mu}^g_{nk}(t), \boldsymbol{\sigma}^g_{nk}(t)), \\ \cdots \\ q_{NK}(t) \sim \mathcal{CN}(\boldsymbol{\mu}^g_{NK}(t), \boldsymbol{\sigma}^g_{NK}(t)), \end{gathered} \tag{4.3.15}$$

where $\boldsymbol{\mu}^g_{nk}(t) = [\boldsymbol{\mu}^g(t)]_{nk}$ and $\boldsymbol{\sigma}^g_{nk}(t) = [\boldsymbol{\sigma}^g(t)]_{nk}$. Similar to the derivation of (4.3.11) in the critic network, we update π by minimizing

$$L_\pi(\theta) = \mathbb{E}_{\boldsymbol{\chi}(t)\sim\mathcal{D}}\Big[\mathbb{E}_{\boldsymbol{\varrho}(t)\sim\pi_\theta}[\alpha \log \pi_\theta(\boldsymbol{\chi}(t), \boldsymbol{\varrho}(t)) - Q_\phi(\boldsymbol{\chi}(t), \boldsymbol{\varrho}(t))]\Big]. \tag{4.3.16}$$

The GTN actor network parameters θ are updated with stochastic gradients, and we have

$$\hat{\nabla}_\theta L_\pi(\theta) = \nabla_\theta \alpha \log \pi_\theta(\boldsymbol{\chi}(t), \boldsymbol{\varrho}(t)) + \Big(\nabla_{\boldsymbol{\varrho}(t)} \alpha \log \pi_\theta(\boldsymbol{\chi}(t), \boldsymbol{\varrho}(t)) - Q_\phi(\boldsymbol{\chi}(t), \boldsymbol{\varrho}(t))\Big), \tag{4.3.17}$$

$$\theta \leftarrow \theta - \lambda_a \hat{\nabla}_\phi L_\pi(\theta), \tag{4.3.18}$$

where λ_a is the learning rate of the actor.

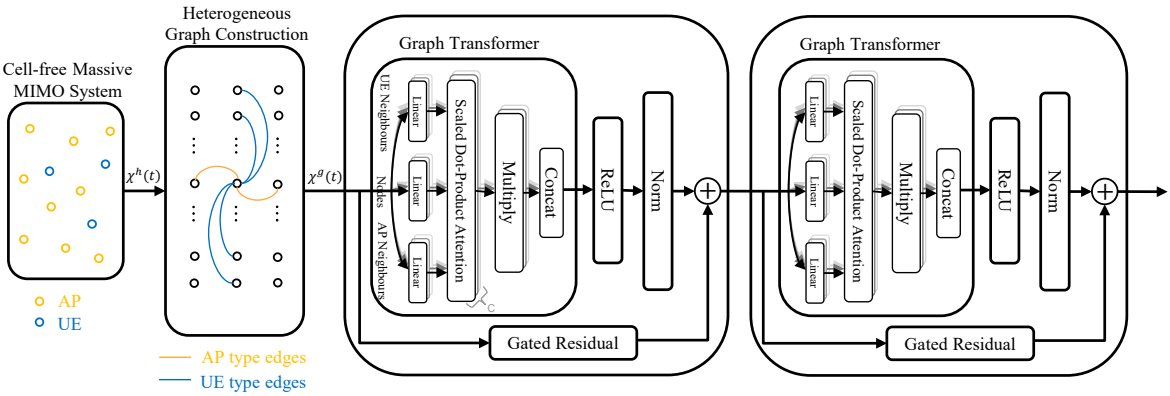


Figure 4.3: The functional structure of the proposed GTN.

Remark 4.3.1. *After joint optimization on power allocation and AP selection, there are two outcomes. 1) If* $\{p_{n,k} \approx 0, k = 1, \cdots, K\}$*, AP* n *is deactivated to the sleep model, resulting in significantly high energy efficiency. Case 2: If the AP* n *is in the active mode, for a certain UE* k*, we still might have* $p_{n,k} \approx 0$ *(For example, the channel between UE* k' *and AP* n *is weak). In this case, AP* n *is excluded from the service of UE* k*.*

4.4 The Proposed Graph Transformer Network

To enable OSAC-G to adapt to the moderate variations of Class II parameters without learning from scratch (**L3**), we develop the GTN as shown in Fig. 4.3, which is a component of the GTN actor network module. Specifically, we first construct a heterogeneous graph to map each AP-UE pair to one node of the graph. Then, the hidden feature within each node is extracted by the graph transformer.

4.4.1 Heterogeneous Graph Construction

In this subsection, we convert the pre-processed state $\boldsymbol{\chi}^h(t)$ to a heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, wherein $\mathcal{V}$ represents the collection of nodes and $\mathcal{E}$ represents the collection of edges. For a system with N APs and K UEs, we create NK nodes that represent NK AP-UE pairs. For the AP-UE pair (node) $(n, k) \in \{1, ..., N\} \times \{1, ..., K\}$, the state (n, k) at this node is collected, the node index is denoted as $i = I(n, k) \in \{1, ..., NK\}$. Clearly, there are NK nodes in the set $\mathcal{V}$. Each node i comprised within the set $\mathcal{V}$ is inherently linked with a corresponding node feature $\boldsymbol{\chi}_i(t)$ with the size χ, and the initial node features for the problem are $\boldsymbol{\chi}^h_{n,k}(t)$.

If one node and its neighbor are connected to either the same AP or the same UE, an edge e is generated between them, and we have $e \in \mathcal{E}$. For example, $I(\mathbf{1}, 2)$ and $I(\mathbf{1}, 3)$ are type-AP neighbors, and $I(1, \mathbf{2})$ and $I(2, \mathbf{2})$ are type-UE neighbors. This neighbor relationship (edge) is pre-determined and recorded in the process of collecting state. Each edge e is classified as belonging to either the type-AP if they share a common AP, or to the type-UE if they share a common UE. For $n, n' \in \{1, ..., N\}$ and $k, k' \in \{1, ..., K\}$, type-AP edge and type-UE edge are defined as

$$\begin{aligned} e^{\mathrm{AP}} &= (I(n, k), I(n, k')) \in \mathcal{E}^{\mathrm{AP}},\ n \neq n', \\ e^{\mathrm{UE}} &= (I(n, k), I(n', k)) \in \mathcal{E}^{\mathrm{UE}},\ k \neq k', \end{aligned} \tag{4.4.1}$$

where $\mathcal{E}^{\mathrm{AP}}$ and $\mathcal{E}^{\mathrm{UE}}$ are sets of AP edge and UE edge, respectively. Clearly, we have $\mathcal{E} = \mathcal{E}^{\mathrm{AP}} \cup \mathcal{E}^{\mathrm{UE}}$.

For node i, the set of its neighboring nodes is denoted as $\mathcal{N}(i) = \{j \in \mathcal{V} : (i, j) \in \mathcal{E}\}$, whereas the sets

$$\begin{aligned} \mathcal{N}^{\mathrm{AP}}(i) &= \{j \in \mathcal{V} : (i, j) \in \mathcal{E}^{\mathrm{AP}}\}, \\ \mathcal{N}^{\mathrm{UE}}(i) &= \{j \in \mathcal{V} : (i, j) \in \mathcal{E}^{\mathrm{UE}}\}. \end{aligned} \tag{4.4.2}$$

Here, $\mathcal{N}^{\mathrm{AP}}(i)$ includes all nodes that share the same AP with $i = I(n,k)$, and $\mathcal{N}^{\mathrm{UE}}(i)$ includes all nodes that share the same UE with $i = I(n,k)$. To simplify the expression, we implement $\circ \in \{\mathrm{AP}, \mathrm{UE}\}$ in the next subsection. After the heterogeneous graph construction, the pre-proposed state $\boldsymbol{\chi}^h(t)$ is transferred to the graphical state $\boldsymbol{\chi}^g(t) = \{\boldsymbol{\chi}_i, e^{\circ}_{i,j} | i \in \mathcal{V}, (i,j) \in \mathcal{E}\}$. Then, we design a GTN to capture the graphical information from $\boldsymbol{\chi}^g(t)$.

4.4.2 The Structure of Graph Transformer

The graph transformer is proposed to incorporate the graph structure of the constructed heterogeneous graph into the neural networks to explore the hidden features from nodes in $\mathcal{V}$ and edges in $\mathcal{E}$ [110]. Specifically, our graph transformer has multilayer structures [149] with each node aggregating features from its neighbors of type-UE and type-AP, and combining them with its own features in each layer. The graph transformer iteratively updates each node by aggregation and combination operations. Besides, it is apparent that the power consumption for each AP-UE pair could be potentially different with the same QoS requirement. Therefore, our proposed graph transformer adopts multi-head attention to assign different weights to the different neighbors of nodes.

Given the node features at time slot l, we adopt vanilla multi-head attention of the transformer and incorporate the two types of edges defined in (4.4.1). Among C multi-head attentions, the c-th attention of the edge between node i and its neighbor

j can be calculated as

$$\begin{aligned}\mathcal{L}^{\circ}_{c,i}(l) &= \mathbf{\Xi}^{\circ}_{c,i}(l) \cdot \boldsymbol{\chi}_i(l) + \mathbf{b}^{\circ}_{c,i}(l),\\ \mathcal{L}^{\circ}_{c,j}(l) &= \mathbf{\Xi}^{\circ}_{c,j}(l) \cdot \boldsymbol{\chi}_j(l) + \mathbf{b}^{\circ}_{c,j}(l),\\ \mathcal{L}^{\circ}_{c,ij}(l) &= \mathbf{\Xi}^{\circ}_{c,ij}(l) \cdot e^{\circ}_{i,j}(l) + \mathbf{b}^{\circ}_{c,ij}(l),\end{aligned} \tag{4.4.3}$$

where $\mathbf{\Xi}^{\circ}_{c,i}(l) \in \mathbb{R}^{\mathcal{L}\times\chi}$, $\mathbf{\Xi}^{\circ}_{c,j}(l) \in \mathbb{R}^{\mathcal{L}\times\chi}$ and $\mathbf{\Xi}^{\circ}_{c,ij}(l) \in \mathbb{R}^{\mathcal{L}\times 1}$ are trainable weight parameters; $\mathbf{b}^{\circ}_{c,i}(l) \in \mathbb{R}^{\mathcal{L}}$, $\mathbf{b}^{\circ}_{c,j}(l) \in \mathbb{R}^{\mathcal{L}}$ and $\mathbf{b}^{\circ}_{c,ij}(l) \in \mathbb{R}^{\mathcal{L}}$ are trainable bias parameters. Then, the c-th attention coefficient between node i and its neighbor j could be expressed as

$$\alpha^{\circ}_{c,ij}(l) = \frac{\langle \mathcal{L}^{\circ}_{c,i}(l), \mathcal{L}^{\circ}_{c,j}(l) + \mathcal{L}^{\circ}_{c,ij}(l)\rangle}{\sum_{k\in\mathcal{N}^{\circ}(i)} \langle \mathcal{L}^{\circ}_{c,i}(l), \mathcal{L}^{\circ}_{c,k}(l) + \mathcal{L}^{\circ}_{c,ik}(l)\rangle}, \tag{4.4.4}$$

where $\langle x, y\rangle = \exp\frac{x^T y}{\sqrt{d}}$ is the exponential scale dot product and d represents the hidden size of each head. With the graph multi-head attention, we make a message aggregation for node i from its neighbors $\mathcal{N}^{\circ}(i)$, and we have

$$\text{AGGREGATE}^{\circ}_{c,i}(l) = \sum_{j\in\mathcal{N}^{\circ}(i)} \alpha^{\circ}_{c,ij}(l) \times \mathcal{L}^{\circ}_{c,j}(l). \tag{4.4.5}$$

Then, we combine the feature of the node i and message aggregation from its neighbors in (4.4.5), resulting in

$$\text{COMBINE}^{\circ}_{i}(l) = \bigoplus_{c=1}^{C} \Big(\mathcal{L}^{\circ}_{c,i}(l) + \text{AGGREGATE}^{\circ}_{c,i}(l) \Big), \tag{4.4.6}$$

where $\bigoplus$ is the concatenation operation. Usually, $C = 2$ suffices as a further increase in C results in a diminishing return. Note that the trainable parameters in (4.4.4), (4.4.5), and (4.4.6) are different. Finally, combining $\text{COMBINE}^{\text{AP}}_{i}(l)$ and $\text{COMBINE}^{\text{UE}}_{i}(l)$, the update rule of the node i is given by

$$\text{UPDATE}_i(l) = \text{Norm}\Big(\text{ReLu}(\text{COMBINE}^{\text{AP}}_{i}(l) + \text{COMBINE}^{\text{UE}}_{i}(l)) \Big). \tag{4.4.7}$$

Inputs with different Class II parameters will be constructed as heterogeneous graphs, where the AP-UE pair is mapped to one node of the graph. It is clear that the operations (4.4.3) - (4.4.7) do not rely on any specific ordering of neighbors of each node.

Consequently, the output of GTN is guaranteed to exhibit permuted equivariance and is independent of different Class II parameters. Besides, due to the self-attention mechanism in GTN, a different weight is applied to each node, which highlights the dependency of each AP-UE pair. The computational complexity is only related to the graph density, so GTN is an ideal choice for processing the formulated problem $\mathcal{P}_2$.

4.5 Performance Analysis

4.5.1 Convergence Analysis

In this section, we investigate the asymptotic convergence of OSAC-G. To achieve this, we first investigate the performance of successive approximation module, which is a component of OSAC-G. As mentioned in Section 4.3.3, $\mathcal{P}_2$ is transformed to $\mathcal{P}_3$ by the first-order approximation, and then $\mathcal{P}_3$ is solved by the successive approximation module. Here, the process for solving $\mathcal{P}_3$ is considered as a function $\mathcal{F}$ with the input $\boldsymbol{\varrho}^{\mathrm{s}}(t)$ and the output $\boldsymbol{\varrho}^{\mathrm{s}}(t+1)$, which is denoted as $\boldsymbol{\varrho}^{\mathrm{s}}(t+1) = \mathcal{F}(\boldsymbol{\varrho}^{\mathrm{s}}(t))$. Note that the feasible solution to an optimization problem should satisfy all its constraints. For feasible solution $\boldsymbol{\varrho}^{s}(t)$ of $\mathcal{P}_3$, the solution $\boldsymbol{\varrho}^{s}(t+1)$ is feasible in the optimization problem $\mathcal{P}_3$ as mentioned in [150].

Theorem 4.5.1. *The solution $\boldsymbol{\varrho}^{\mathrm{s}}(t)$ converges to a local optimum or saddle point of the optimization problem $\mathcal{P}_3$.*

At iteration $t+1$, the constraints of $\mathcal{P}_3$ are first updated by the solution $\boldsymbol{\varrho}^{\mathrm{s}}(t)$, and then $\boldsymbol{\varrho}^{\mathrm{s}}(t+1)$ is solved by successive approximation module. Clearly, $\boldsymbol{\varrho}^{\mathrm{s}}(t)$ satisfies the constraints of $\mathcal{P}_3$ at iteration $t+1$, and $\boldsymbol{\varrho}^{\mathrm{s}}(t)$ is one of the solutions for $\mathcal{P}_3$ and

provides the upper bound. Therefore, we have $P_{\text{opt}}^{\text{s}}(t+1) \leq P_{\text{opt}}^{\text{s}}(t)$. Clearly, $P_{\text{opt}}^{\text{s}}(t)$ monotonically decreases with iteration. With enough iterations, $\boldsymbol{\varrho}^{\text{s}}(t)$ could converge to a local minimum or saddle point for $\mathcal{P}_2$, and Theorem 4.5.1 has been validated. Implementing escaping saddle points technique for successive approximation [151], we can find the local minimum for $\mathcal{P}_2$. However, the successive approximation module obtains the local minimum of $\mathcal{P}_2$ at the cost of a number of iterations. It is impractical to solve $\mathcal{P}_2$ within each time slot.

In the following, we analyze the convergence of OSAC-G. In our OSAC-G, we embed the successive approximation module to design an appropriate reward function $Z(\boldsymbol{\chi}(t), \boldsymbol{\varrho}(t))$ in (4.3.4). Once $\boldsymbol{\varrho}^{\text{s}}(t)$ solved by successive approximation module converges to the local minimum, accurate $Z(\boldsymbol{\chi}(t), \boldsymbol{\varrho}(t))$ would be generated. By providing better reward functions for $\boldsymbol{\chi}(t)$, the agent is more likely to learn the policy faster and more efficiently. Specifically, the loss $L_\pi(\theta)$ converges when $|P_{\text{opt}}(t) - P_{\text{opt}}^{\text{s}}(t)| \leq \epsilon_P$, where ϵ_P is the power tolerance. Consequently, with the trained policy, OSAC-G can directly infer the solution with significantly lower complexity.

4.5.2 Computational Complexity Analysis

As a typical optimization method, SAM can obtain a near-optimal solution to the non-convex problem but suffers from high computational complexity due to the introduction of multiple convex approximations. Given the interior-point method, the complexity required for solving the convex optimization problem on each iteration is $\mathcal{O}((NK)^3 N_c)$ [152], where N_c is the count of constraints. The total complexity of SAM is $\mathcal{O}((NK)^3 N_c N_s)$, where N_s signifies the count of iterations involved.

By contrast, the complexity of OSAC-G lies in training and online inference, and

training is a one-off offline process and its complexity can be ignored. After the offline training, we apply the trained GTN actor network to directly optimize the power allocation and AP selection in the downlink. Therefore, the complexity of the online inference is generated due to the GTN actor network. As shown in Fig. 4.3, the GTN builds the self-attention module with C heads. For simplicity, we analyze a one-head module and explore its complexity. A head takes the input $\boldsymbol{\chi}^g(t) = \{\boldsymbol{\chi}_i, e^\circ_{i,j} | i \in \mathcal{V}, (i,j) \in \mathcal{E}\}$, and applies (4.4.3) to obtain $\mathcal{L}^\circ_{c,i} \in \mathbb{R}^{\mathcal{L}\times\chi}$, $\mathcal{L}^\circ_{c,j} \in \mathbb{R}^{\mathcal{L}\times\chi}$ and $\mathcal{L}^\circ_{c,ij} \in \mathbb{R}^{\mathcal{L}\times\chi}$. Then, we obtain $\alpha^\circ_{c,ij}(l)$ following (4.4.4), and the complexity can be calculated as $\mathcal{O}(\mathcal{L}^2\chi)$ for each head. After implementing the aggregate, combine and update operations to C self-attention attention heads as shown in Section 4.4, the complexity for the GTN is $\mathcal{O}(C\mathcal{L}^2\chi)$ Overall, to solve the non-convex optimization problem with NK variables, the total complexity of the online interference is calculated as $\mathcal{O}(NKC\mathcal{L}^2\chi)$. For example, to solve $\mathcal{P}_2$ with $N = 40$ and $K = 10$, the typical count of iterations for SAM is $N_s = 50$, and the number of constraints is $N_c = 6$. Thus, the complexity for SAM is on the order of 10^{10}, which requires high computational complexity which is consistent with **L1**. By contrast, the complexity for OSAC-G to solve $\mathcal{P}_2$ is on the order of 10^6 with $C = 2$ multi-head attentions, the hidden size $\mathcal{L} = 32$ and the state size $\chi = 5$. Clearly, we can conclude that OSAC-G can significantly reduce the complexity compared to SAM.

4.6 Simulation Results

4.6.1 Experimental Settings

In the simulation, we consider the downlink in a cell-free massive MIMO system, where APs and UEs are located randomly within a square area, and the distance $d_{n,k}$ between AP n and UE k follows a uniform distribution $\mathcal{U}(1,1)$ (in kilometers). Each UE randomly chooses one pilot sequence from a pre-defined collection of mutually orthogonal pilot sequences, wherein each sequence has a duration of τ_p symbols. The modeling of large-scale fading coefficient $\beta_{n,k}$ can be represented by

$$\beta_{nk} = PL(d_{nk}) \cdot 10^{\frac{\sigma_{sh} z_{nk}}{10}}, \tag{4.6.1}$$

where $10^{\frac{\sigma_{sh} z_{nk}}{10}}$ is the log-normal shadowing with the standard deviation σ_{sh} and z_{nk} [153], and $PL(d_{nk})$ is the path loss (in dB) [154] which is expressed as

$$PL(d_{nk}) = \begin{cases} -L - 35\log_{10}(d_{nk}), \ d_{nk} \geq d_1, \\ -L - 15\log_{10}(d_1) - 20\log_{10}(d_{nk}), \ d_0 \leq d_{nk} \leq d_1, \\ -L - 15\log_{10}(d_1) - 20\log_{10}(d_0), \ d_{nk} \leq d_0, \end{cases} \tag{4.6.2}$$

where d_0 and d_1 are the distance constants and

$$L = 149.48 + 33.9\log_{10}(f) - (1.1\log_{10}(f) - 0.7)h_{\text{UE}} - 13,82\log_{10}(h_{\text{AP}}) + (1.56\log_{10}(f) - 0.8), \tag{4.6.3}$$

where h_{UE} and h_{AP} are the height of UE and AP, respectively.

4.6.2 Schemes for Comparison

For simplicity, our proposed green energy scheme with ZF beamforming is referred to as the green energy scheme-ZF hereafter. In the meantime, we also consider the green energy scheme with the CB, referred to as the green energy scheme-CB. To

Table 4.1: Simulation Parameters

Parameters	Value
Power amplifier efficiency κ_n^{A}, $\forall n$	0.4
Number of APs N	40
Number of antennas per AP M	20
Number of UEs K	10
QoS requirement η_k, $\forall n$	20 Mbps
Traffic-dependent power coefficient ξ^{B}	0.5 W/Gbps
RF chain power $P_n^{\mathrm{A,chain}}$, $\forall n$	0.3 W
Maximum transmitted power P_{TX}, $\forall n$	0.2 W

illustrate the superiority, two existing optimization schemes are included for comparison. The first one is the all-AP scheme, where all APs are activated with ZF beamforming. The second one separately handles power allocation and AP selection with ZF beamforming [30].

4.6.3 Simulation Results

Performance Comparison among Different Schemes

In Figs 4.4-4.6, the averaged optimized power consumption of the four schemes is compared under three different scenarios. We consider different optimization algorithms for the aforementioned four schemes: 1) all-AP scheme without power allocation, 2) proposed scheme [30] solved by SAM, 3) green energy scheme-CB solved by SAM, 4) green energy scheme-ZF solved by SAM, and 5) green energy scheme-ZF solved by OSAC-G.

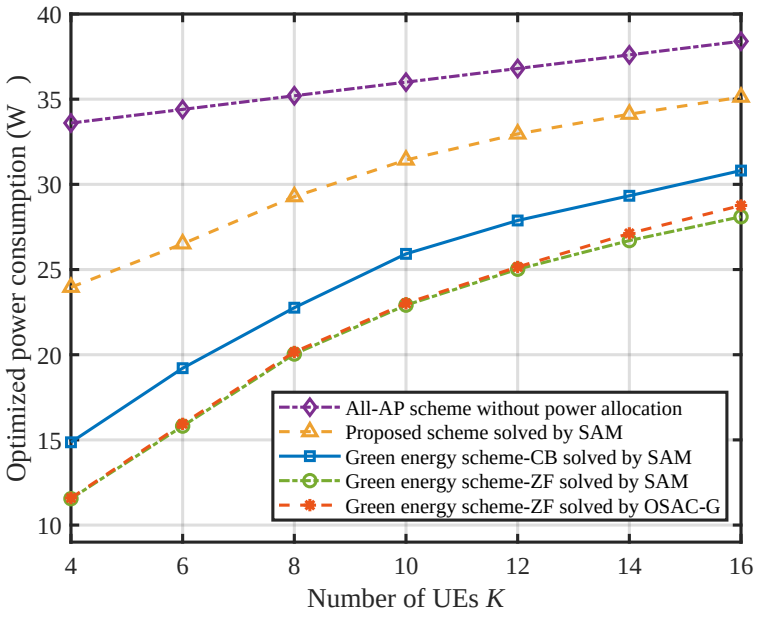


Figure 4.4: Optimized power consumption versus the number of UEs K with $\eta = 20$ Mbps and $\xi^{\mathrm{F}} = 0.5$ W/Gbps.

In Fig. 4.4, the comparison is carried out for different numbers of UEs. Note that OSAC-G here is trained offline and then implemented for low-complexity online inference for the same K. It is observed that the averaged optimized power consumption increases with K for all the schemes. Clearly, the green energy scheme-ZF solved by SAM and OSAC-G achieve a similar performance and outperform the other three schemes, significantly so when K is relatively small. Specifically, when $K = 4$, the averaged optimized power consumption reduction of the green energy scheme-ZF is 65.62%, 51.79% and 28.62% relative to the all-AP scheme, the proposed scheme [30], and the green energy scheme-CB, respectively. In Fig. 4.5, the comparison is carried out for different QoS requirement η. As expected, it is observed that the averaged optimized power consumption increases with η as fewer APs are activated in the system.

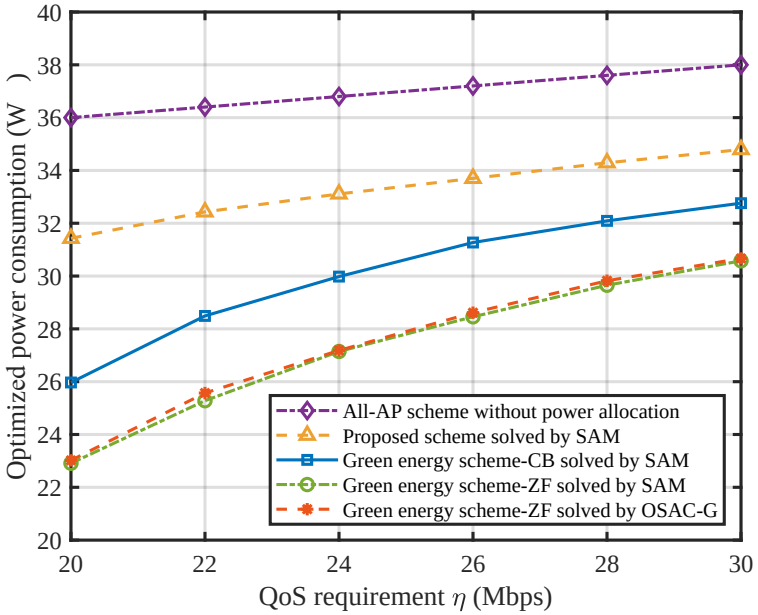


Figure 4.5: Optimized power consumption versus the QoS requirement η with $K = 10$ and $\xi^{\mathrm{F}} = 0.5$ W/Gbps.

The relative performance supremacy of these schemes is similar to that in Fig. 4.4. In Fig. 4.6, the comparison is carried out for different traffic-dependent power coefficient ξ^{F}. It can be observed that, for all the schemes, the averaged optimized power consumption grows almost linearly with ξ^{F}. In terms of the relative performance, a similar conclusion could be drawn to that in Fig. 4.4.

In summary, Figs 4.4-4.6 illustrate that our green energy scheme-ZF can significantly improve power consumption.

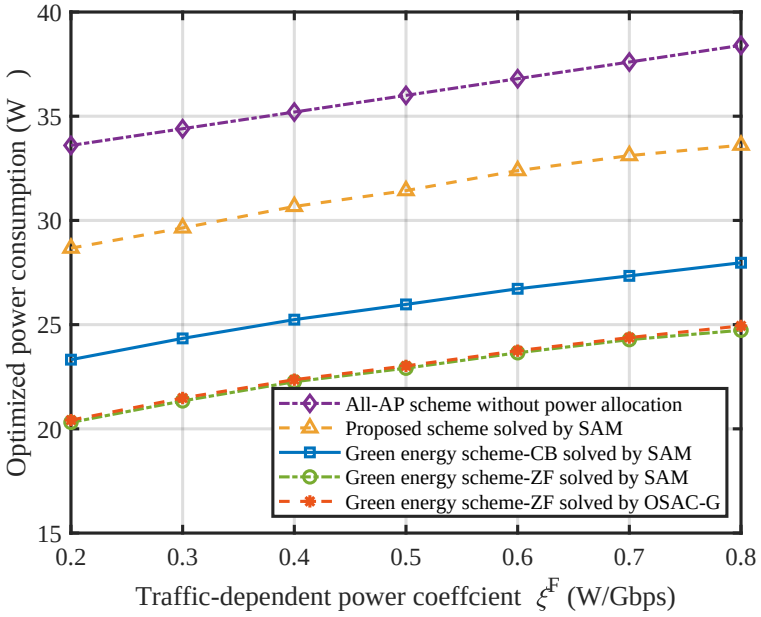


Figure 4.6: Optimized power consumption versus the traffic-dependent power coefficient ξ^{F} with $K = 10$ and $\eta = 20$ Mbps.

Performance Comparison between OSAC-G and SAC

To demonstrate the advantage of the proposed optimization-embedded DRL algorithm (OSAC-G) over the conventional DRL algorithm (SAC), in Figs. 4.7-4.8, we compare the performance for $K = 10$ and $K = 15$ under green energy scheme-ZF. Two metrics are adopted: the normalized loss for the offline training and the feasible ratio for the online inference, and they are presented in Fig. 4.7 and Fig. 4.8, respectively. The normalized loss is defined as a weighted sum of the policy loss $L_\pi(\theta)$, the Q-function loss $L_Q(\phi)$, and the target value loss $L_V(\psi)$. The feasible ratio is defined as the number of solutions that satisfy constraints to that of all test problem instances.

It is observed from Fig. 4.7 that, for $K = 10$, OSAC-G leads to a much faster

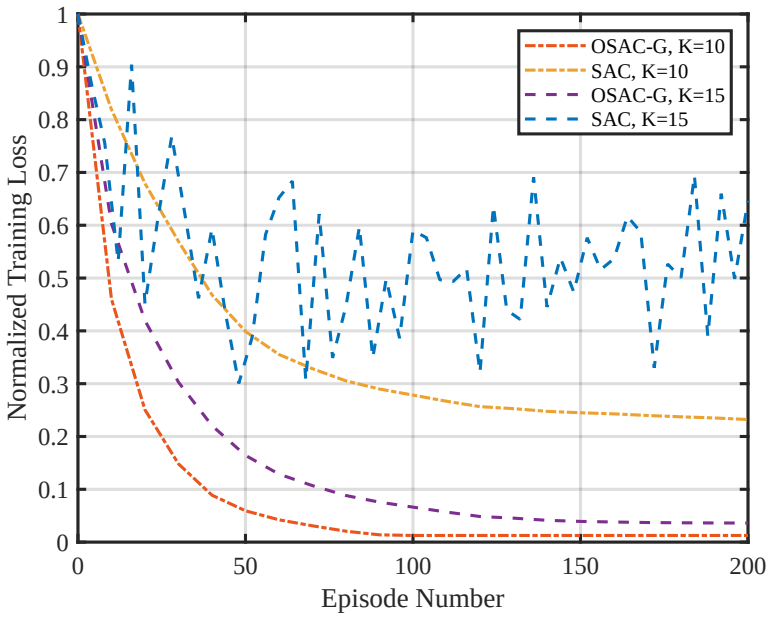


Figure 4.7: Offline Training Performance Comparison between OSAC-G and SAC.

convergence and a smaller loss relative to SAC. When $K = 15$, the offline training performance of OSAC-G is only slightly inferior to that of $K = 10$. In stark contrast, for SAC, the loss does not converge. In Fig. 4.8, OSAC-G achieves high feasible ratios, which are 97.2% and 96.3% for $K = 10$ and $K = 15$, respectively. In other words, the constraints of the formulated problem can be satisfied with a high probability in OSAC-G. In comparison, SAC only achieves 59.8% and 7.8% for $K = 10$ and $K = 15$, respectively. Clearly, Figs. 4.7-4.8 confirm that OSAC-G, through offline training and online inference, achieves significantly superior performances than those of the conventional DRL algorithm.

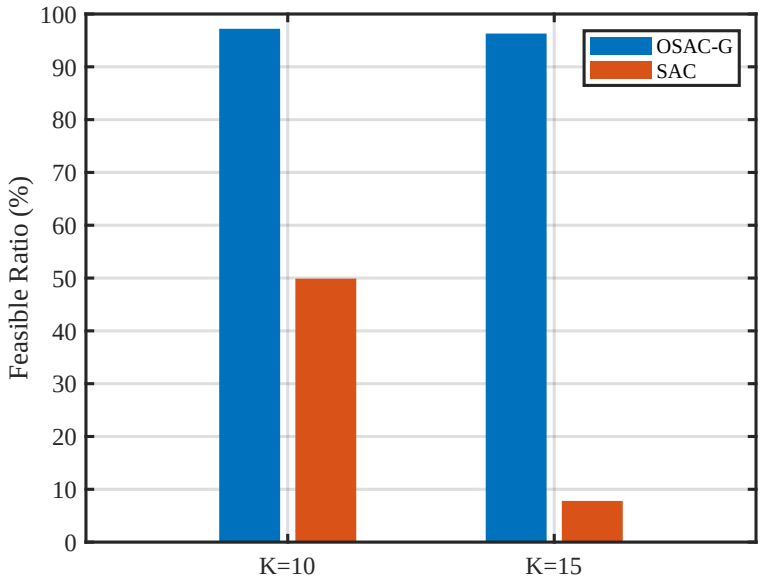


Figure 4.8: Online Training Performance Comparison between OSAC-G and SAC.

Cumulative Distribution Function Comparison

In Figs. 4.9-4.10, we compare the performance of OSAC-G, SAC, and SAM in terms of the cumulative distribution function (CDF) of the optimized power consumption under the green energy scheme-ZF. Note that if the online inferred solution of OSAC-G or SAC is not feasible, we adopt the all-AP scheme here. In Fig. 4.9, the comparison is carried out for $K = 10$ and $K = 15$. Note that the result of SAC for $K = 15$ is not available, as it does not converge following Fig. 4.7. For $K = 10$, it is observed that OSAC-G and SAM achieve a similar CDF and, compared with SAC, less average power consumption and a lower standard deviation. When $K = 15$, the CDF of OSAC-G is only slightly inferior to that of SAM. In Fig. 4.10, the comparison is carried out for $\xi^{\mathrm{F}} = 0.5$ and 0.8 W/Gbps. It is shown that, for

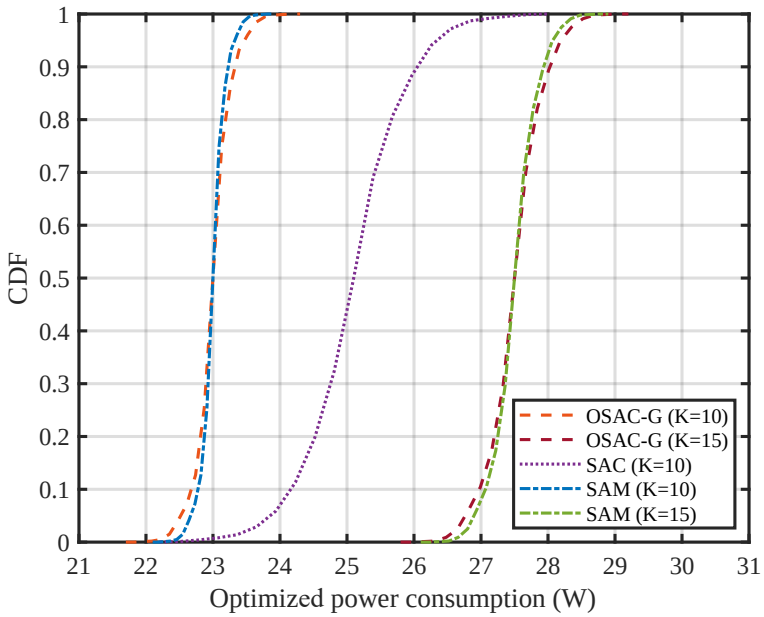


Figure 4.9: The CDF of optimized power consumption Versus Different number of UEs K.

both ξ^{F} values, OSAC-G has a similar CDF to SAM, and OSAC-G and SAM outperform SAC. Figs. 4.9-4.10 verify that OSAC-G achieves a similar performance to SAM, while features significantly reduced computational complexity as demonstrated in Section 4.5.2.

Model Generalization Capability of OSAC-G

As introduced in Section 4.4, the GTN is designed to enable OSAC-G to adapt to the variations of Class II parameters without learning from scratch. Next, we demonstrate the model generalization capability of OSAC-G under the green energy scheme-ZF. Here, the averaged optimized power consumption and the feasible ratio are adopted as metrics. In Fig. 4.11, OSAC-G is trained offline on the $K = 10$, and then directly implemented for online inference from $K = 10$ transferring to $K = 15$

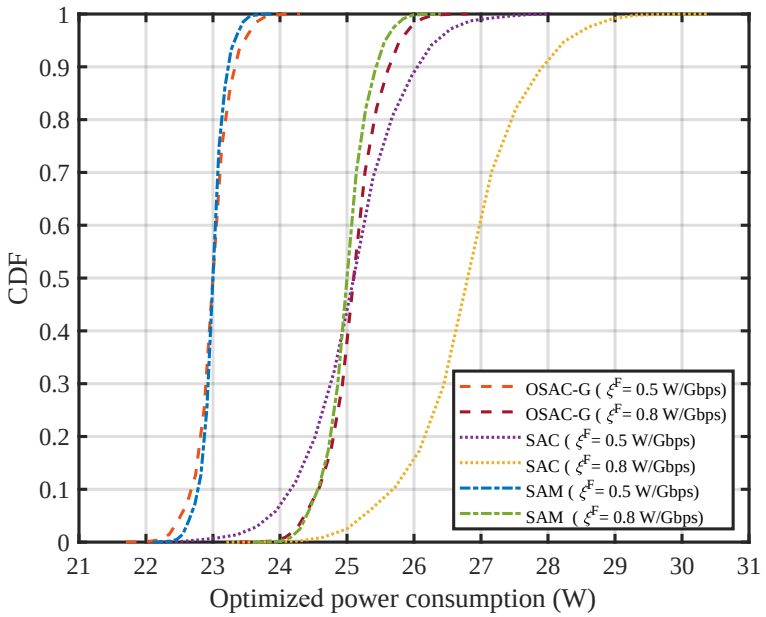


Figure 4.10: The CDF of optimized power consumption Versus Different traffic-dependent power coefficient ξ^{F}.

with a step 1. It can be seen that the power difference increases with K due to the increasing complexity of the non-convex problem. However, even for $K = 15$, the difference is only 4.15%. Meanwhile, the feasible ratio decreases with K. For $K = 15$, a reasonable value of 76.5% is still achieved. Furthermore, our experimental results show that when K is reduced, the feasible ratio slightly decreases. Clearly, Fig. 4.11 illustrates an outstanding model generalization capability of our OSAC-G.

4.7 Conclusion

Power allocation and AP selection play a vital role in the improvement of energy efficiency in downlink cell-free massive MIMO. We proposed a green energy scheme and formulated it as a non-convex MINLP problem, which is quite challenging to

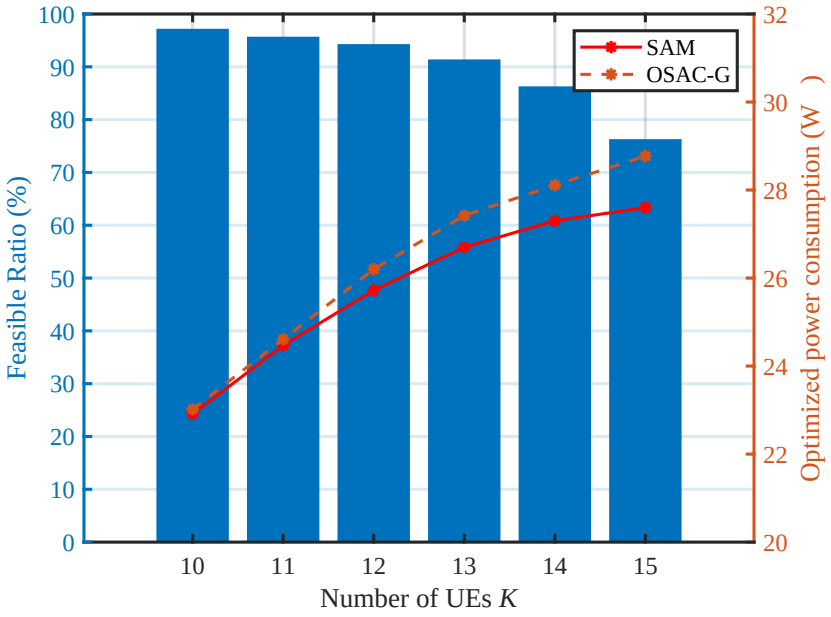


Figure 4.11: Model Generalization Performance of OSAC-G.

solve. To solve this problem, we proposed a novel OSAC-G algorithm by deeply embedding non-convex optimization into contemporary DRL methods. OSAC-G can directly infer solutions for the transformed non-convex problem with a much lower computation complexity, and all the constraints can be satisfied with a high probability due to our handcrafted reward function. It has been verified by simulations that the green energy scheme achieves lower power consumption than the existing ones.

Chapter 5

Partial NOMA-based Online Task Offloading For Multi-layer Computing Networks

In this chapter, we consider a multi-layer computing network with communication, computation, and energy requirements, which is a practical application of wireless IoT networks. To minimize the latency across all MDs, we develop a PNOMA-based task offloading scheme in a multi-layer computing network. PNOMA combines the high throughput of NOMA and low interference of OMA to achieve low-latency transmission. Besides, the PNOMA-based multi-layer collaborations enable fast task processing for different computing requirements. We formulate a non-convex mixed-integer optimization problem to minimize the average delay across all MDs. To address this, we propose the RPPO algorithm, which offers direct online solutions with significantly low complexity. We incorporate accumulated apriori information in RPPO for fast retraining and design a reward function and an evaluation phase to ensure the communication/computation constraints to be satisfied with a high probability. Simulation results demonstrate that the proposed scheme dramatically outperforms the existing ones.

5.1 Introduction

The soaring demands of MDs for elevated computation capability, low latency, and vast scalability have created an unprecedented challenge [155]. However, MDs are usually limited by their computation resources and battery capabilities. As a transformative technique, MEC has been proposed to provide more computation resources than MDs [156]. Typically, an MS is deployed close to MDs, which reduces transmission delay and energy consumption. The transmission between MDs and MS is originally established through OMA, where each orthogonal sub-channel is only occupied by one MD. NOMA in [36–38] has recently been introduced to two-layer (MD-MS) mobile computing task offloading. This allows multiple MDs to share the same sub-channel for MD-MS transmissions by multiplexing their tasks in the power domain[10], thereby increasing throughput compared to OMA. However, the cost is the introduction of interference between MDs.

In this chapter, we develop a partial NOMA based task offloading scheme in a multi-layer mobile computing network, where a task can be either executed locally or offloaded to MS or the cloud server (CS). Based on the number of available sub-channels, PNOMA divides MDs into non-orthogonal pairs and individual units, enabling transmission to MS via NOMA and OMA, respectively. This flexible approach leverages the advantages of NOMA's high throughput and OMA's low interference of OMA, resulting in low-latency transmission across various data sizes. Furthermore, the PNOMA-based multi-layer collaborations enable fast task processing for different computing requirements, thereby significantly reducing the computation delay. We optimize the average delay across all MDs by jointly optimizing task offloading and resource allocation. Mathematically, we formulate a non-convex optimization

problem involving a number of discrete and continuous variables.

However, the formulated problem is NP-hard. As a typical optimization method, successive approximation (SCA) can obtain a near-optimal solution but suffers from high complexity due to the multiple convex approximations [157]. Some heuristic algorithms, such as GA [91], feature low complexity, but the performance cannot be guaranteed. DRL [95] is able to achieve fast online inference for directly mapping state (e.g., bandwidth and computation capabilities) to action (problem solutions) by maximizing a reward via offline training. However, conventional DRL, such as vanilla actor-critic (AC) and deep Q-network (DQN), is not applicable to our formulated problem due to two main challenges: 1) The PNOMA-based multi-layer computing network environment is highly dynamic with parameters fluctuating across a wide range. Accordingly, the DRL needs to be retrained from scratch frequently. 2) Our formulated problem involves multiple communication and computation constraints that cannot be easily satisfied by the DRL.

To address the above challenges, we propose a reincarnating proximal policy optimization (RPPO) algorithm taking advantage of the DRL algorithm and the reincarnating technique [158]. RPPO can infer the optimal task offloading and resource allocation online with a very low complexity. In addition, it is adaptive to system parameter variations in the PNOMA-based multi-layer computing network. Specifically, RPPO involves several reincarnating training phases with different system parameters, resulting in less retraining and better generalization than the conventional DRL. When RPPO needs to be retrained, it utilizes the accumulated apriori information (e.g., learned value function, and learned parameters) from previous iterations, and therefore the new training would start with a lower loss, resulting in a more efficient

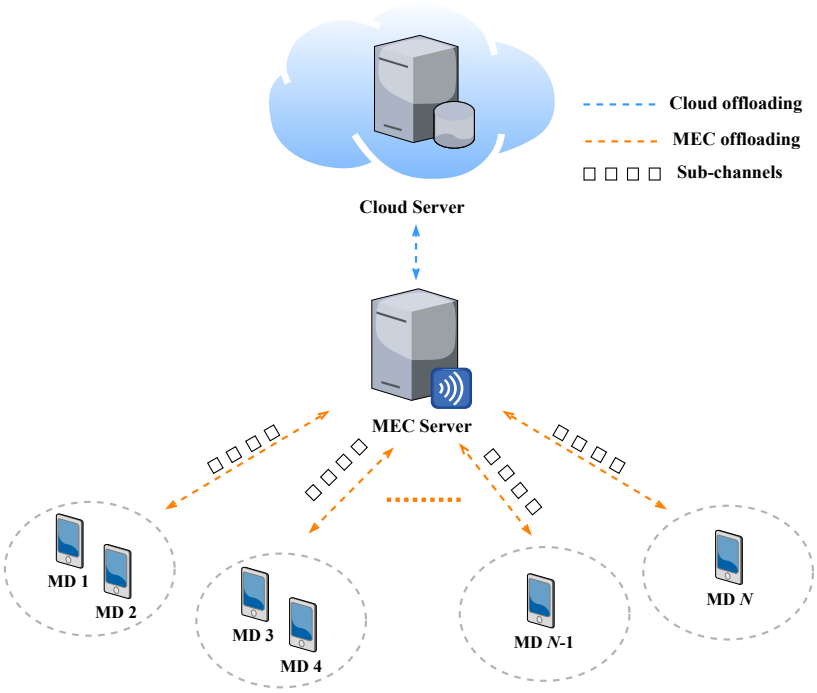


Figure 5.1: PNOMA-based multi-layer computing network.

retraining. Furthermore, by incorporating communication/computation constraints into the reward function design and evaluation phase of RPPO, the constraints could be satisfied with a high probability. Simulation results indicate that the proposed scheme achieves a much lower average delay than the existing ones, and it is shown that RPPO could significantly outperform the conventional DRL.

5.2 System Model and Problem Formulation

5.2.1 System Model for Partial NOMA

As shown in Fig. 5.1, we consider an uplink multi-layer computing network with N single-antenna MDs indexed by $\mathcal{N} = \{1, \cdots, N\}$, an MS, and a CS. Here, M

sub-channels, indexed by $\mathcal{M} = \{1, \cdots, M\}$, are allocated to MDs for uplink task offloading. In this chapter, we assume that the number of MDs is larger than that of sub-channels, i.e., $N > M$. Here, we develop a flexible and efficient approach, PNOMA. In PNOMA, we assume that each sub-channel can accommodate one MD or two MDs regarded as a non-orthogonal pair. According to M, N MDs are classified into N_p pairs and N_s individual units ($N = 2N_p + N_s$). Here, M_p ($M_p = N_p$) sub-channels, indexed by $\mathcal{M}_p = \{1, \cdots, M_p\}$, are allocated to N_p pairs while M_s ($M_s = N_s$) sub-channels, indexed by $\mathcal{M}_s = \{M_p + 1, \cdots, M\}$, are allocated to N_s individual units. In each time slot, the n-th MD ($n \in \mathcal{N}$) has a computation task $Q_n = \{D_n, C_n\}$, where D_n is the data size and C_n is the required computing resources. The task Q_n could be processed locally (Level 0 local computing) or offloaded to the MS (Level 1 edge computing). Alternatively, Q_n could be further offloaded to the CS from the MS (Level 2 cloud computing).

5.2.2 Communication Model

In this chapter, the communication between an MD and the MS is through a PNOMA-based wireless link, and the communication between the MS and the CS is through a wired backhaul link. For the wireless link, the channel power gain between the n-th MD and the MS via the m-th sub-channel ($m \in \mathcal{M}$) is denoted by $h_{n,m}$. We assume that the MS and MDs know the perfect channel state information. The SIC technique [73] is implemented to decode the received signal from the sub-channel allocated to a non-orthogonal pair. The decoding order for two MDs (n and n') in a pair is decided by the MS based on channel conditions, and the achievable rate for

the n-th MD via the m-th sub-channel is expressed as

$$\begin{aligned} r_{n,m} &= b_{n,m}\log_2\left(1+\frac{P^{\text{off}}_{n,m}h_{n,m}}{P^{\text{off}}_{n',m}h_{n',m}+\sigma^2}\right), h_{n,m}\geq h_{n',m}, \\ r_{n,m} &= b_{n,m}\log_2\left(1+\frac{P^{\text{off}}_{n,m}h_{n,m}}{\sigma^2}\right), h_{n,m}< h_{n',m}, \end{aligned} \tag{5.2.1}$$

where $b_{n,m}$ is the allocated bandwidth, $P^{\text{off}}_{n,m}$ is the transmission power, and σ^2 is the additive white Gaussian noise power.

In addition, each individual MD could offload data by a dedicated sub-channel and the data rate conforms to $r_{n,m} = b_{n,m}\log_2\left(1+\frac{P^{\text{off}}_{n,m}h_{n,m}}{\sigma^2}\right)$.

Level 1 Edge Computation

In Level 1, the task Q_n is offloaded to the MS through sub-channels, and then it can be processed by the MS or further offloaded to the CS. Here, we define $x_{n,m}\in\{0,1\}$ as the sub-channel allocation indicator. If $x_{n,m}=1$, the m-th sub-channel is allocated to the n-th MD for offloading. We further define $\alpha_{n,m}$ as the offloading ratio of the n-th MD via the m-th sub-channel, and we have

$$\alpha_n = \sum_{m=1}^{M} x_{n,m}\alpha_{n,m}. \tag{5.2.2}$$

Accordingly, the offloading delay from the n-th MD to the MS through the m-th sub-channel is calculated as

$$T^{\text{M,off}}_{n,m} = \frac{\alpha_{n,m}D_n}{r_{n,m}}. \tag{5.2.3}$$

The corresponding energy consumption for offloading could be obtained as $E^{\text{M,off}}_{n,m} = T^{\text{M,off}}_{n,m}P^{\text{off}}_{n,m}$. The computation delay could be calculated as

$$T^{\text{M,comp}}_{n,m} = \frac{\alpha_{n,m}(1-\beta_{n,m})C_n}{f^{\text{M}}_{n,m}}, \tag{5.2.4}$$

where $\beta_{n,m}$ is the offloading ratio from the MS to the CS, and $f^{\text{M}}_{n,m}$ is the allocated computation resources to the n-th MD via the m-th sub-channel (in CPU frequency).

Level 2 Cloud Computation

In Level 2, the task is first transmitted to the MS, and then to the CS. The associated transmission delay is denoted as T^{bc}, and the computation delay by the CS is

$$T_{n,m}^{\mathrm{C,comp}} = \frac{\alpha_{n,m}\beta_{n,m}C_n}{f^{\mathrm{C}}}, \tag{5.2.5}$$

where f^{C} is the cloud computation capability, which is a predetermined value [84]. The CS has a significantly larger capability than that of the MS, i.e., $f^{\mathrm{C}} \gg f_{n,m}^{\mathrm{M}}$.

5.2.3 Problem Formulation

Here, we aim to minimize the average delay across all MDs by jointly optimizing task offloading and resource allocation subject to communication, computation, and energy constraints. Accordingly, the optimization problem in the PNOMA-based multi-layer task offloading scheme is formulated as:

$$\mathbf{P}: \min_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{x},\mathbf{f},\mathbf{b}} \sum_{n=1}^{N} T_n^{\mathrm{L,comp}} + \sum_{n=1}^{N}\sum_{m=1}^{M} x_{n,m}\big(T_{n,m}^{\mathrm{M,off}} + T_{n,m}^{\mathrm{M,comp}} + \alpha_n\beta_{n,m}T^{\mathrm{bc}} + T_{n,m}^{\mathrm{C,comp}}\big), \tag{5.2.6}$$

$$\begin{aligned}
\text{s.t.}\quad & T_n^{\mathrm{L,comp}}P_n^{\mathrm{L}} + \sum_{m=1}^{M} x_{n,m}T_{n,m}^{\mathrm{M,off}}P_{n,m}^{\mathrm{off}} \le E_{n,max}, \forall n \in \mathcal{N}, \\
& \sum_{n=1}^{N}\sum_{m=1}^{M} f_{n,m}^{\mathrm{M}} \le f_{\max}^{\mathrm{M}}, \quad f_{n,m}^{\mathrm{M}} \ge 0, \sum_{n=1}^{N}\sum_{m=1}^{M} b_{n,m} \le B_{\max}, \quad b_{n,m} \ge 0, \\
& 0 \le \alpha_n \le 1,\ 0 \le \beta_{n,m} \le 1, \quad \forall n \in \mathcal{N}, \forall m \in \mathcal{M}, \\
& \sum_{n=1}^{N} x_{n,m} = 2, \forall m \in \mathcal{M}_p, \sum_{n=1}^{N} x_{n,m} = 1,\ \forall m \in \mathcal{M}_s, \\
& \sum_{m=1}^{M} x_{n,m} = 1, \forall n \in \mathcal{N},\ x_{n,m} \in \{0,1\},
\end{aligned} \tag{5.2.7}$$

where $E_{n,\max}$, $f_{\max}^{\mathrm{M}}$ and $B_{\max}$ represent the local energy limit, the MS computation capability, and the maximum available bandwidth from MDs to the MS, respectively. In addition, we have $\boldsymbol{\alpha} = \{\alpha_{n,m}\}$, $\boldsymbol{\beta} = \{\beta_{n,m}\}$, $\boldsymbol{x} = \{x_{n,m}\}$, $\boldsymbol{f} = \{f_{n,m}^{\mathrm{M}}\}$, and $\boldsymbol{b} = \{b_{n,m}\}$, where $n \in \mathcal{N}$ and $m \in \mathcal{M}$. Clearly, $\mathbf{P}$ is a mixed-integer optimization problem with discrete variables $\boldsymbol{x}$ and continuous variables $\{\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{f}, \mathbf{b}\}$. Furthermore, $\mathbf{P}$ is NP-hard due to the non-convexity of the objective in (5.2.6).

5.3 The Proposed RPPO Algorithm

Due to its non-convex nature, solving $\mathbf{P}$ requires significant computation resources for conventional optimization algorithms. By contrast, DRL learns mapping from the input states to output actions by maximizing a predefined reward function. The trained DRL can infer a solution to the optimization problem directly online resulting in much lower complexity. However, the conventional DRL requires frequent retraining from scratch due to the highly dynamic nature and wide parameter fluctuations of the PNOMA multi-layer computing environment. In addition, the problem involves multiple communication and computation constraints that the DRL cannot guarantee to satisfy easily.

The above observation motivates us to develop the RPPO algorithm taking advantage of the PPO algorithm [159] and the reincarnating technique. The PPO algorithm is introduced as a remarkable policy-gradient DRL algorithm where a batch of samples is subdivided into mini-batches and iteratively reused multiple times. When computing the gradient for strategy adjustment, PPO restricts the range of parameter updates to mitigate training instability effectively. This algorithm operates on

policy, exhibiting the desirable stability attributes associated with a randomized gradient strategy. The reincarnating technique allows the agent to accumulate apriori information from previous training iterations and avoids learning from scratch. By leveraging the PPO and the reincarnating technique, RPPO could directly online infer the optimal task offloading and resource allocation, and achieve fast retraining in the dynamic PNOMA-based multi-layer mobile computing network.

5.3.1 The Preliminary of RPPO Algorithm

Here, we provide the preliminary including state, action, and reward function.

State: For the task offloading from the n-th MD to the MS via the m-th subchannel at time step t, the state is encoded as $\mathbf{s}_{n,m}(t)$ which is composed of the communication, computation, and constraint information, such as the data rate $r_{n,m}$, the allocated computation resources $f^{\mathrm{M}}_{n,m}$ and local energy limit $E_{n,\max}$. Besides, we denote the state of the system as $\mathbf{s}(t)$ with $[\mathbf{s}(t)]_{nm} = \mathbf{s}_{n,m}(t)$. Consequently, the state space could be represented as $\mathcal{S} = \{\mathbf{s}(1), \mathbf{s}(2), \mathbf{s}(3), \cdots\}$.

Action: The agent strategically takes an action $\boldsymbol{\varrho}_{n,m}(t)$ for task offloading and resource allocation based on the state $\mathbf{s}_{n,m}(t)$, and the action is represented by $\boldsymbol{\varrho}_{n,m}(t) = \{\alpha_{n,m}, \beta_{n,m}, x_{n,m}, f^{\mathrm{M}}_{n,m}, b_{n,m}\}$ at time step t. The action of the system is denoted as $\boldsymbol{\varrho}(t)$ with $[\boldsymbol{\varrho}(t)]_{nm} = \boldsymbol{\varrho}_{n,m}(t)$. Consequently, the action space could be represented as $\mathcal{A} = \{\boldsymbol{\varrho}(1), \boldsymbol{\varrho}(2), \boldsymbol{\varrho}(3), \cdots\}$.

Reward: We denote reward for the state-action pair $\{\mathbf{s}(t), \boldsymbol{\varrho}(t)\}$ as $r(t) = r(\mathbf{s}(t), \boldsymbol{\varrho}(t))$, which measures the performance of the system action $\boldsymbol{\varrho}(t)$ and makes the agent take better decisions. Given $r(t)$, RPPO attempts to learn an optimal policy π^* capable of mapping a given state $\mathbf{s}(t)$ to an optimal action $\boldsymbol{\varrho}^*(t)$. As $\boldsymbol{\varrho}(t)$ provides task offloading

and resource allocation for each MD, $r(t)$ measures the performance of $\boldsymbol{\varrho}(t)$. Obviously, the objective function in $\mathbf{P}$ should be considered in the design of the reward function to better guide the agent in decision-making. The average delay associated with the time step t, denoted as $T_{\text{obj}}(t)$, could be calculated by substituting $\boldsymbol{\varrho}(t)$ generated by the agent into (5.2.6). Here, the objective-related reward function is expressed as $r_{\text{obj}}(t) = -T_{\text{obj}}(t)$. In addition, due to the complex constraints in the non-convex optimization problem $\mathbf{P}$, it is difficult for a conventional DRL to meet the constraints during both training and online inference phases. In this chapter, we consider constraints in our reward function design, which ensures constraints are met with a high probability. We design the constraint-related reward function at time step t as

$$\begin{aligned} r_{\text{cstr}}(t) = \omega_e \sum_{n=1}^{N} \text{sgn}(E_{n,\max} - E_{n,\text{total}}) + \omega_f \text{sgn}(f_{\max}^{\text{M}} - \sum_{n=1}^{N}\sum_{m=1}^{M} f_{n,m}^{\text{M}}) \\ + \omega_x \sum_{n=1}^{N}\sum_{m=1}^{M} \Lambda_x(x_{n,m}) + \omega_b \text{sgn}(b_{\max} - \sum_{n=1}^{N}\sum_{m=1}^{M} b_{n,m}) \\ + \omega_\alpha \sum_{n=1}^{N}\sum_{m=1}^{M} \Lambda_\alpha(\alpha_{n,m}) + \omega_\beta \sum_{n=1}^{N}\sum_{m=1}^{M} \Lambda_\beta(\beta_{n,m}), \end{aligned} \tag{5.3.1}$$

where $\omega_{(\cdot)}$ is the penalty coefficient for constraints, and $E_{n,\text{total}}$ is the energy consumption in n-th MD. In addition, $\text{sgn}(\cdot)$ is a sign function, and $\Lambda_{(\cdot)}(\cdot)$ is the self-designed check function. The total reward is given by

$$r(t) = \omega_{\text{obj}} r_{\text{obj}}(t) + \omega_{\text{cstr}} r_{\text{cstr}}(t), \tag{5.3.2}$$

where ω_{obj} and ω_{cstr} are the penalty coefficients. Clearly, our constructed reward function considers both the objective function and the constraints as delineated in (5.3.2). This approach ensures that the objective is pursued and the constraints in (5.2.6) are satisfied with high probability, thereby facilitating effective training.

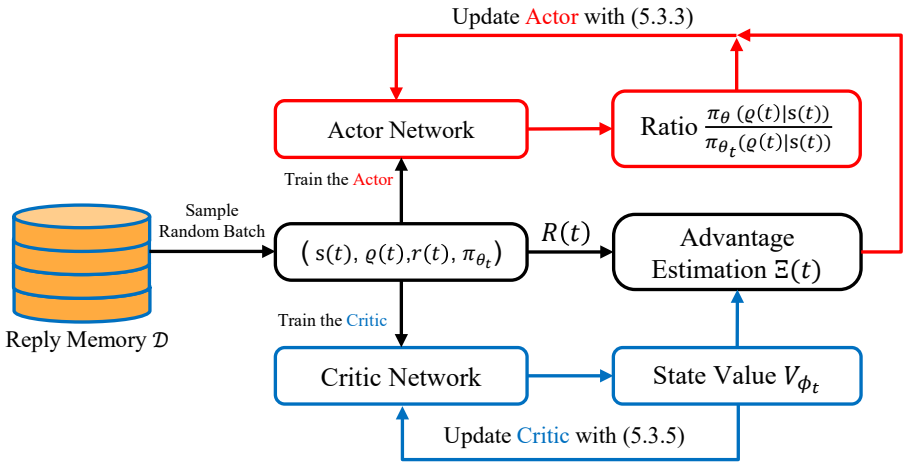
Algorithm 4 Reincarnating Proximal Policy Optimization (RPPO) Algorithm.

1: **Initialize** the policy π_{θ_0} with θ_0, and value function V_{ϕ_0} with ϕ_0.
2: Commence the preliminary training.
3: **for** each iteration **do**
4: Reincarnate the policy with the apriori knowledge and fine-tuning.
5: **for** each training step **do**
6: **for** each environment step **do**
7: Observe state $\mathbf{s}(t)$ and take action $\boldsymbol{\varrho}(t)$ with policy π_{θ_t}.
8: Compute the step reward $r(t)$ based on (5.3.2) considering the objective and constraints.
9: **end for**
10: Collect the replay memory buffer $\mathcal{D}_t$ from the environment steps.
11: Compute the cumulative reward $R(t)$ and the advantage estimation $\Xi(t)$.
12: Update the policy π_{θ_t} using stochastic gradient ascent according to (5.3.3).
13: Update the value function V_{ϕ_t} by minimizing the mean squared error according to (5.3.5).
14: **end for**
15: Evaluate the performance of the learned policy.
16: Accumulate the apriori information including the learned policy, the value function, and the related parameters.
17: **end for**

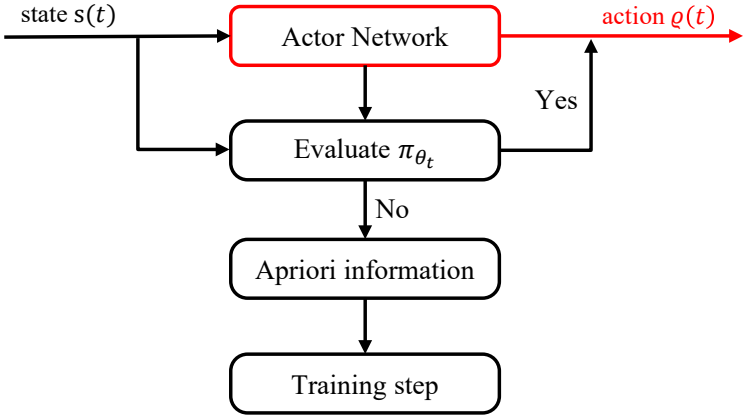
5.3.2 RPPO Algorithm

The workflow of our proposed RPPO algorithm is summarized in **Algorithm 4**. Upon parameters initialization, each iterative step comprises three phases: 1) reincarnating the training process with the apriori information and adjusting network parameters (Lines $3-14$), 2) evaluating learned policy performance (Line 15), and 3) accumulating the apriori information (Line 16). In phase 1), we reuse the accumulated apriori information as a starting point for training the policy over multiple epochs. In phase 2), we evaluate the policy performance in terms of average delay and constraint violation by solving new problem instances. In phase 3), if the policy performance is unsatisfactory, all knowledge - including the learned policy, the learned value function,

and the related parameters - is collected as apriori information. These phases repeat across multiple iterations to allow reincarnating learning from previous training.



(a) Training step



(b) Reincarnate step

Figure 5.2: The framework of the RPPO.

Next, we illustrate how the policy is trained. Interacting with the environment, the policy π_{θ_t} takes an action $\boldsymbol{\varrho}_{n,m}(t) = \{\alpha_{n,m}, \beta_{n,m}, x_{n,m}, f^{\mathrm{M}}_{n,m}, b_{n,m}\}$ given the state $\mathbf{s}_{n,m}(t)$. Specifically, the output of the actor network is a continuous set $\{\alpha_{n,m}, \beta_{n,m}, x^{\mathrm{C}}_{n,m}, f^{\mathrm{M}}_{n,m}, b_{n,m}\}$, where $x^{\mathrm{C}}_{n,m} \in [0, 1]$. Then, discretization is carried out

based on a threshold so that we have $x_{n,m} \in \{0, 1\}$. Accordingly, the reward $r(t)$ from (5.3.2) is computed. After the environment steps, we record states, actions, and rewards to the replay memory buffer $\mathcal{D}_t$. Based on $\mathcal{D}_t$, the cumulative reward $R(t)$ and advantage estimation $\Xi(t) = \Xi(\pi_\theta(\boldsymbol{\varrho}(t)|\mathbf{s}(t))) = R(t) - V_{\phi_t}$ are obtained. According to [160], we update the policy π_{θ_t} using a stochastic gradient ascent:

$$\theta_{t+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_t|T} \sum_{\mathcal{D}_t} \sum_{\tau=0}^{T} \min\left(\frac{\pi_\theta(\boldsymbol{\varrho}(\tau)|\mathbf{s}(\tau))}{\pi_{\theta_t}(\boldsymbol{\varrho}(\tau)|\mathbf{s}(\tau))} \Xi(t), \Psi(\epsilon, \Xi(t)) \right), \tag{5.3.3}$$

where $|\mathcal{D}_t|$ represents the number of elements in $\mathcal{D}_t$, and ϵ is the parameter to control the size of the trust region. In addition, $\Psi(\epsilon, \Xi(t))$ could be expressed as

$$\Psi(\epsilon, \Xi(t)) = \begin{cases} (1+\epsilon) \cdot \Xi(t), \ \Xi(t) \geq 0 \\ (1-\epsilon) \cdot \Xi(t), \ \Xi(t) < 0 \end{cases}. \tag{5.3.4}$$

Moreover, we update the value function V_{ϕ_t} by minimizing the mean squared error, and we have

$$\phi_{t+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_t|T} \sum_{\mathcal{D}_t} \sum_{\tau=0}^{T} \big(V_\phi(\mathbf{s}_\tau) - R(\tau)\big)^2. \tag{5.3.5}$$

5.3.3 Complexity Analysis

Although the SCA can obtain a near-optimal solution, it suffers from high complexity. As we have $N_v = 5NM$ variables in $\mathbf{P}$, the complexity required for solving the approximated convex problem at each iteration is $\mathcal{O}\big((N_v + N_c)^3\big)$, where N_c is the number of constraints. Consequently, the total complexity of the SCA is $\mathcal{O}\big((N_v + N_c)^3 N_s\big)$, where N_s signifies the number of iterations involved. By contrast, the complexity of RPPO lies in training and online inference; training is a one-off offline process, and its complexity can be ignored. The online inference complexity is $\mathcal{O}(N_v)$. Therefore, the complexity of RPPO is significantly lower than that of the SCA.

Table 5.1: MEC system parameters

Local computation power P_n^{L}	0.5 W
Local computation capability f_n^{L}	1 GHz
Local energy limit $E_{n,\max}$	5 J
Transmission power $P_{n,m}^{\mathrm{off}}$	23 dBm
MS Computation capability $f_{\max}^{\mathrm{M}}$	10 GHz
Maximum available bandwidth $B_{\max}$	5 MHz
Backhaul time T^{bc}	0.5 s
Cloud computation capability f^{C}	25 GHz
White Gaussian noise power σ^2	-97 dBm

5.4 Simulation Results

In the simulation, we consider the uplink in a multi-layer computing network, where $N = 20$ MDs are located randomly, and the distance d_n between the n-th MD and the MS follows a uniform distribution $\mathcal{U}(0.1, 0.5)$ (in kilometers). There are $M = 14$ sub-channels implemented for uplink task offloading, and we have 6 non-orthogonal pairs and 8 individual MDs. According to [153], the channel power gain is modeled as $h_{n,m} = 127 + 30\log(d_n)$. For the task Q_n, its data size D_n is sampled uniformly from $\mathcal{U}(1.5, 2.5)$ (in Mb) and the required computing resources C_n are sampled uniformly as $\mathcal{U}(3, 5)$ (in Gigacycles). The other related parameters are summarized in Table 5.1.

5.4.1 Average Delay Comparison among Different Schemes

In Fig. 5.3, the average delay is compared among seven schemes: 1) our proposed PNOMA-based multi-layer task offloading scheme, 2) NOMA-based multi-layer task

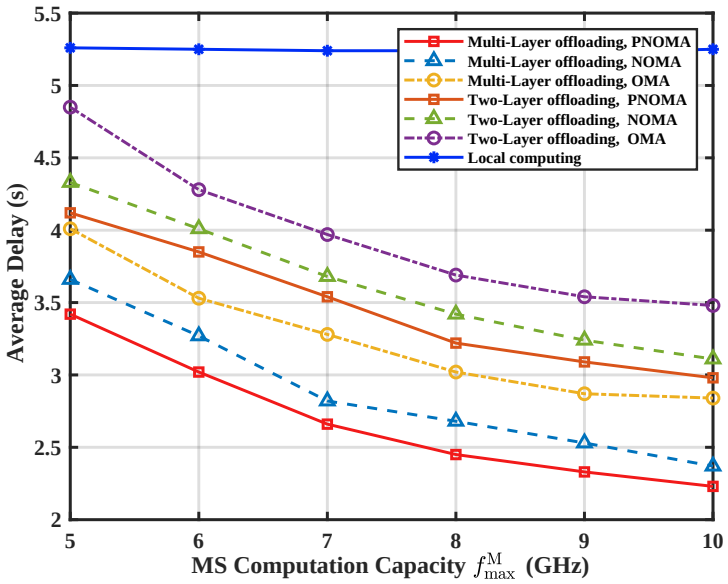


Figure 5.3: Average delay comparison among different schemes.

offloading scheme, 3) OMA-based multi-layer task offloading scheme, 4) PNOMA-based two-layer task offloading scheme, 5) NOMA-based two-layer task offloading scheme, 6) OMA-based two-layer task offloading scheme, and 7) local computing scheme. Note that formulated problems of these seven schemes are solved by RPPO, which is trained offline and then implemented for low-complexity online inference.

Clearly, the PNOMA-based multi-layer task offloading scheme achieves the lowest average delay among the seven schemes[1]. Specifically, when $f_{\max}^{M} = 10$ GHz, the average delay reduction is 7.3%, 21.5%, 25.2%, 28.3%, 35.9% and 57.5% relative to 2), 3), 4), 5), 6) and 7) introduced before. This confirms that our proposed scheme can significantly reduce the average delay.

[1]In OMA-based schemes, a sub-channel is occupied by only one MD. Due to insufficient sub-channels, some MDs can not offload their tasks and are forced to process them locally, resulting in higher computation delay compared to PNOMA. In NOMA-based schemes, all MDs are paired for transmissions to the MS, and some sub-channels are not used. This introduces more interference between MDs, resulting in higher transmission delay compared to PNOMA.

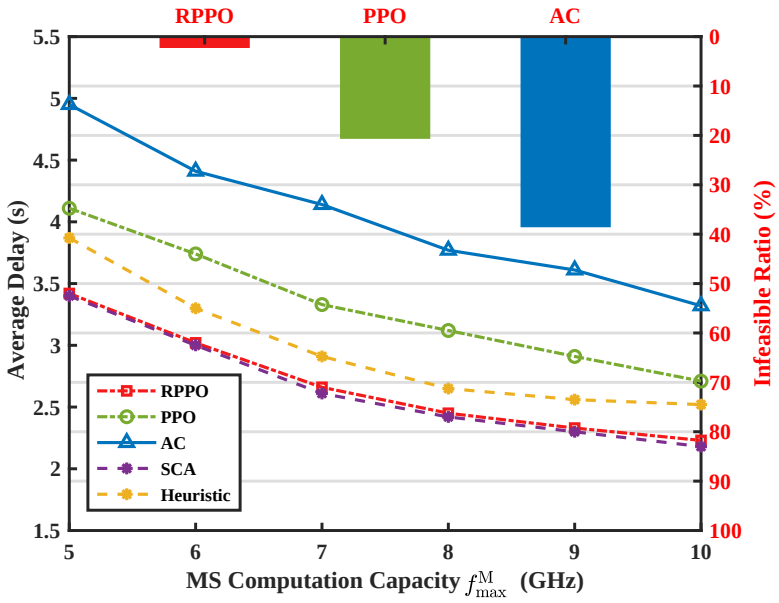


Figure 5.4: Average delay and infeasible ratio comparison among different algorithms.

5.4.2 Performance Comparison of Different Algorithms

In Fig. 5.4, we demonstrate the superiority of the proposed RPPO over two DRL-based algorithms, namely, the PPO and the AC, as well as two optimization algorithms which are the SCA and a heuristic algorithm (GA). Two metrics are adopted: the average delay and the infeasible ratio for the online inference, which quantifies the proportion of the solutions that fail to satisfy the constraints. Note that all solutions, solved by the optimization algorithms, are feasible, so we only provide the average delay here. For the DRL-based algorithms, if the solution is infeasible this task would be processed locally. It is clearly shown that, in terms of average delay, the proposed RPPO significantly outperforms the PPO, the AC, and the heuristic algorithm, and is only marginally inferior to the SCA. When it comes to the infeasible ratio, a similar trend could be observed. Overall, Fig. 5.4 confirms that, in terms of online inference

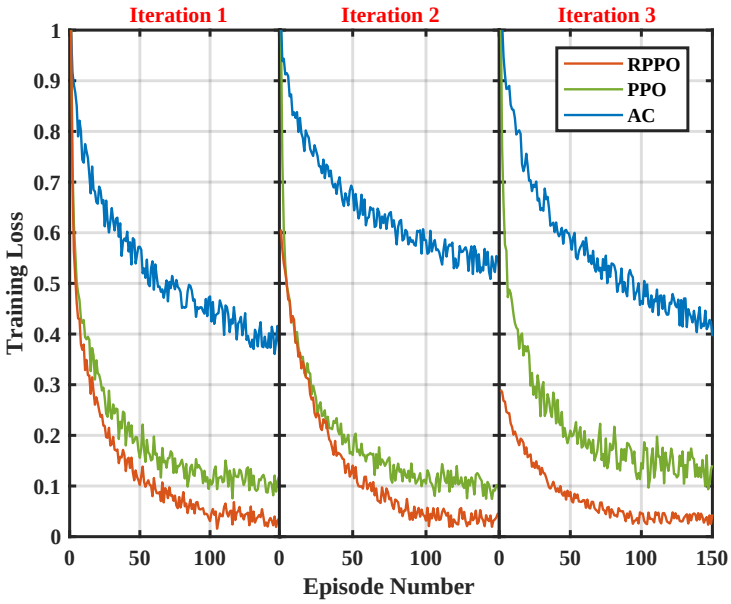


Figure 5.5: Training performance comparison among different algorithms.

performance, RPPO significantly outperforms the PPO and the AC.

5.4.3 Training Performance Comparison

Note that Figs. 5.3-5.4 only consider the scenario for a fixed number of MDs N. When there is a significant variation in N, the DRL models need to be retrained. In Fig. 5.5, we compare loss values of the proposed RPPO, the PPO, and the AC during the training process based on our proposed PNOMA-based multi-layer task offloading scheme. Specifically, we conduct three training iterations ($N = 20, 25,$ and 30), each solving a different problem with varying numbers of MDs. It can be seen that RPPO achieves the smallest loss and demonstrates a decreasing starting loss value along with faster convergence as interactions progress. This confirms that RPPO can achieve fast retraining and efficiently adapt to system parameter variations and expedite the training process.

5.5 Conclusion

We proposed a PNOMA-based multi-layer task offloading scheme by jointly optimizing task offloading and resource allocation. We formulated a non-convex mixed-integer optimization problem to minimize the average delay across all MDs. To solve this problem, we proposed an RPPO algorithm taking advantage of the DRL and the reincarnating technique. RPPO maps the input state to the optimal task offloading and resource allocation and achieves direct online inference with significantly low complexity. We incorporated accumulated apriori information in RPPO for fast retraining to adapt to the variations of the system parameters. In addition, we designed a reward function and an evaluation phase to guarantee the communication/computation constraints with a high probability. Simulations confirmed that the proposed scheme achieves a much lower average delay than the existing ones.

Chapter 6

Conclusions and Future Work

In this chapter, we summarize the pivotal findings from this thesis and its contributions. Following this, we proceed to contemplate potential avenues for advancing our research, offering suggestions for future works to further explore and extend our existing strategies.

6.1 Summary of Results and Insights

In Chapter 3, we considered the challenging problem of meeting the low-latency requirement of wireless IoT applications. We developed a full-dimensional task offloading scheme by integrating D2D communication into a multi-layer computing network, where tasks could be processed locally, or offloaded to adjacent MDs. Alternatively, tasks could be further offloaded to MSs and CS. This led to the development of a comprehensive task offloading scheme, which is optimized for both task offloading decisions and the utilization of communication and computation resources with the objective of minimizing the average delay across all MDs. We formulated it as an MINLP problem, which involves both discrete and continuous variables. With relaxation, the sub-problems of the formulated problem are convex. To solve

this problem, we introduced a method involving inverse reinforcement learning with graph neural networks, referred to as GIRL. This offers a unified solution to a broad class of this-type MINLP problems in wireless IoT networks by accelerating the optimal B&B algorithm with significantly reduced complexity but without sacrificing the global optimality. Simulations were conducted to validate these claims, revealing that GIRL outperforms existing variable selection policies in terms of computational complexity. Additionally, our simulations confirmed the superiority of our proposed full-dimensional task offloading scheme compared to existing ones.

In Chapter 4, we explored the vital aspects of power allocation and AP selection, with a focus on augmenting energy efficiency in downlink cell-free massive MIMO. Our exploration led to the introduction of a green energy scheme specifically crafted to minimize energy consumption, a goal achieved by formulating the power allocation and AP selection as a non-convex MINLP problem. This formulation is one of the most formidable classes of optimization problems to solve. Conventional non-convex optimization can obtain a near-optimal solution to the non-convex problem but suffers from high computational complexity due to the introduction of multiple convex approximations. To solve this challenging problem, we innovated a novel algorithm known as OSAC-G. This groundbreaking approach is distinct in its integration of non-convex optimization into state-of-the-art DRL techniques. The OSAC-G enables direct inference of solutions for the transformed non-convex problem, and it does so with a marked reduction in computational complexity. A key feature of our algorithm is the assurance that all constraints are met with high probability. This is achieved through the utilization of a meticulously designed reward function that balances both efficiency and compliance. The efficacy of our green energy scheme is corroborated

by simulation results, which conclusively demonstrate its superiority in achieving reduced power consumption compared to existing methods. The outcomes not only validate the effectiveness of our approach but also highlight its potential as a leading solution for sustainable energy management in wireless IoT networks.

Finally, in Chapter 5, we proposed a partial NOMA-based multi-layer task offloading scheme to minimize the average delay across all MDs by jointly optimizing task offloading and resource allocation. We formulated it as a non-convex mixed-integer problem whose discrete variables could be relaxed to 0 or 1. To solve this problem, we proposed an RPPO algorithm that leverages the strengths of DRL and an innovative reincarnating technique. RPPO maps the input state to the optimal task offloading and resource allocation. This leads to direct online inference, making the process highly efficient with significantly reduced complexity. A key feature of RPPO is the integration of accumulated apriori information, facilitating rapid retraining to adapt to the variations of the system parameters. Furthermore, we crafted a meticulously designed reward function and incorporated an evaluation phase. This was instrumental in upholding the communication and computation constraints, ensuring compliance with a high degree of probability. Simulation results provided concrete evidence that our proposed scheme outperformed existing methods, achieving a substantially reduced average delay. These findings underline the value and effectiveness of our approach, setting a new benchmark in the field of mobile computing task offloading.

6.2 Future Work

We propose the following potential advancements to the issues examined in this thesis as future avenues for exploration. These opportunities offer chances that may yield valuable results in our future work.

In Chapter 3, our full-dimensional task offloading scheme in a multi-layer computing network involves assigning computing, storage, and communication resources to the different layers of the network to improve its efficiency and performance. In practice, there are some implementation problems of different layers that should be considered in our future work. Specifically, the local layer comprises the devices, such as smartphones and tablets, that are close to the end users. The edge layer consists of small data centers, usually located at the edge of the network, that provide intermediate processing and storage capabilities. Finally, the cloud layer consists of large-scale data centers that offer massive processing and storage capabilities. There are several challenges in implementing our scheme, including determining traffic patterns [161], determining the resource requirements [162], and monitoring resource allocation [163]. Besides, from the algorithm perspective, we could integrate the newly proposed variable selection policy with the node pruning strategy. This integration will enhance the conventional B&B algorithm, and we intend to explore this in our future work.

In Chapter 4, we propose a green energy scheme in downlink cell-free massive MIMO by simultaneously optimizing power allocation and AP selection based on ZF beamforming. To further improve the energy efficiency of the system, the incorporation of energy harvesting technologies [164] into the green energy scheme could be explored. Chapter 4 focuses on single-objective optimization to increase energy efficiency, so we could investigate how to balance this with other objectives, such as

latency, reliability, or throughput, potentially leading to multi-objective optimization frameworks [165]. In addition, we focus on ZF beamforming. Future studies could explore how the approach could be adapted or extended to other beamforming techniques, evaluating the differences in performance. There is considerable scope for enhancing our existing model by exploring more effective non-convex optimization methods. Currently, non-convex optimization solutions have shown promise in various applications, and harnessing these could bring tangible improvements in efficiency and performance to our OSAC-G algorithm. By incorporating advanced non-convex optimization techniques, we not only expect to increase convergence speed but also potentially find more globally optimal solutions within complex search spaces. This exploration would necessitate a detailed analysis of the underlying mathematical principles, an evaluation of different non-convex optimization algorithms, and a comprehensive integration strategy to ensure compatibility with existing components of the OSAC-G algorithm.

In Chapter 5, partial NOMA task offloading results are based on a multi-layer computing network with MDs, an MS, and a CS. Within this context, there is a significant opportunity to augment the value of the network through the strategic deployment of additional MSs. Since the proposed scheme minimizes average delay, it would be interesting to investigate how energy efficiency can be integrated into the objective function. Balancing delay and energy consumption might be critical in some practical scenarios [166], especially for battery-powered devices. Additionally, there is a need to understand how the RPPO algorithm operates across various communication environments, including those characterized by high mobility or non-ideal

channels. Analyzing these factors can shed light on the method's adaptability to different multi-access computing architectures, enhancing its overall applicability. The potential combination of the proposed RPPO algorithm with other advanced machine learning and optimization techniques offers an exciting direction for further research. By incorporating methodologies like transfer learning [167] or meta-learning [168], it is possible to refine the algorithm's ability to adapt swiftly to new environments or tasks. Such improvements, as outlined here, could set the stage for more nuanced and efficient system operation in future work.

Bibliography

[1] L. Chettri and R. Bera, "A comprehensive survey on internet of things (IoT) toward 5G wireless systems," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 16–32, 2020.

[2] S. Painuly, P. Kohli, P. Matta, and S. Sharma, "Advance applications and future challenges of 5G IoT," in *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, 2020, pp. 1381–1384.

[3] M. Singh, S. Sachan, A. Singh, and K. K. Singh, "Chapter 7 - internet of things in pharma industry: possibilities and challenges," in *Emergence of Pharmaceutical Industry Growth with Industrial IoT Approach*, V. E. Balas, V. K. Solanki, and R. Kumar, Eds. Academic Press, 2020, pp. 195–216.

[4] Y. A. Qadri, A. Nauman, Y. B. Zikria, A. V. Vasilakos, and S. W. Kim, "The future of healthcare internet of things: A survey of emerging technologies," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 2, pp. 1121–1167, 2020.

[5] S. H. Sutar, R. Koul, and R. Suryavanshi, "Integration of smart phone and IoT for development of smart public transportation system," in *2016 International Conference on Internet of Things and Applications (IOTA)*, 2016, pp. 73–78.

[6] J. Ruan, H. Jiang, C. Zhu, X. Hu, Y. Shi, T. Liu, W. Rao, and F. T. S. Chan,

"Agriculture IoT: Emerging trends, cooperation networks, and outlook," *IEEE Wireless Communications*, vol. 26, no. 6, pp. 56–63, 2019.

[7] F. Tao, Y. Zuo, L. D. Xu, and L. Zhang, "IoT-based intelligent perception and access of manufacturing resource toward cloud manufacturing," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1547–1557, 2014.

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

[9] L. Lu, G. Y. Li, A. L. Swindlehurst, A. Ashikhmin, and R. Zhang, "An overview of massive MIMO: Benefits and challenges," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 5, pp. 742–758, 2014.

[10] S. M. R. Islam, N. Avazov, O. A. Dobre, and K.-s. Kwak, "Power-domain non-orthogonal multiple access (NOMA) in 5G systems: Potentials and challenges," *IEEE Commun Surv Tut*, vol. 19, no. 2, pp. 721–742, Oct. 2016.

[11] U. S. Shanthamallu, A. Spanias, C. Tepedelenlioglu, and M. Stanley, "A brief survey of machine learning methods and their sensor and IoT applications," in *2017 8th International Conference on Information, Intelligence, Systems and Applications (IISA)*, 2017, pp. 1–8.

[12] S. Chen, H. Xu, D. Liu, B. Hu, and H. Wang, "A vision of IoT: Applications, challenges, and opportunities with China perspective," *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 349–359, 2014.

[13] J. Ding, M. Nemati, C. Ranaweera, and J. Choi, "IoT connectivity technologies and applications: A survey," *IEEE Access*, vol. 8, pp. 67 646–67 673, 2020.

[14] J. Adu Ansere, G. Han, H. Wang, C. Choi, and C. Wu, "A reliable energy efficient dynamic spectrum sensing for cognitive radio IoT networks," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6748–6759, 2019.

[15] C. Cordeiro, K. Challapali, D. Birru, and S. Shankar, "Ieee 802.22: the first worldwide wireless standard based on cognitive radios," in *First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005.*, 2005, pp. 328–337.

[16] D. Porcino and W. Hirt, "Ultra-wideband radio technology: potential and challenges ahead," *IEEE Communications Magazine*, vol. 41, no. 7, pp. 66–74, 2003.

[17] A. V. Dastjerdi and R. Buyya, "Fog computing: Helping the internet of things realize its potential," *Computer*, vol. 49, no. 8, pp. 112–116, 2016.

[18] F. Pereira, R. Correia, P. Pinho, S. I. Lopes, and N. B. Carvalho, "Challenges in resource-constrained IoT devices: Energy and communication as critical success factors for future iot deployment," *Sensors*, vol. 20, no. 22, p. 6420, 2020.

[19] H. S. Hassanein and S. M. A. Oteafy, "Big sensed data challenges in the internet of things," in *2017 13th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2017, pp. 207–208.

[20] M. G. Kibria, K. Nguyen, G. P. Villardi, O. Zhao, K. Ishizu, and F. Kojima, "Big data analytics, machine learning, and artificial intelligence in next-generation wireless networks," *IEEE Access*, vol. 6, pp. 32 328–32 338, 2018.

[21] V. Angelakis, I. Avgouleas, N. Pappas, E. Fitzgerald, and D. Yuan, "Allocation of heterogeneous resources of an IoT device to flexible services," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 691–700, 2016.

[22] Y. Wang, X. Wen, Z. Hu, Z. Lu, J. Miao, C. Sun, and H. Qi, "Multi-UAV collaborative data collection for iot devices powered by battery," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, 2020, pp. 1–6.

[23] C. Wang, J. Zhang, S. Bai, D. Chang, and L. Duan, "A multiband compact flexible energy collector for wearable or portable IoT devices," *IEEE Antennas and Wireless Propagation Letters*, vol. 22, no. 5, pp. 1164–1168, 2023.

[24] A. Orsino, G. Araniti, L. Militano, J. Alonso-Zarate, A. Molinaro, and A. Iera, "Energy efficient IoT data collection in smart cities exploiting D2D communications," *Sensors*, vol. 16, no. 6, p. 836, 2016.

[25] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Transactions on Communications*, vol. 66, no. 4, pp. 1594–1608, 2018.

[26] Z. Zhao, S. Bu, T. Zhao, Z. Yin, M. Peng, Z. Ding, and T. Q. S. Quek, "On the design of computation offloading in fog radio access networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 7, pp. 7136–7149, 2019.

[27] K. Guo, M. Sheng, J. Tang, T. Q. S. Quek, and Z. Qiu, "Hierarchical offloading for delay-constrained applications in fog RAN," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4257–4270, Apr. 2020.

[28] Y. Wang, X. Tao, X. Zhang, P. Zhang, and Y. T. Hou, "Cooperative task offloading in three-tier mobile computing networks: An ADMM framework," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2763–2776, Mar. 2019.

[29] M. Waqas, Y. Niu, Y. Li, M. Ahmed, D. Jin, S. Chen, and Z. Han, "A comprehensive survey on mobility-aware D2D communications: Principles, practice and challenges," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 3, pp. 1863–1886, 2020.

[30] H. Q. Ngo, L.-N. Tran, T. Q. Duong, M. Matthaiou, and E. G. Larsson, "On the

total energy efficiency of cell-free massive MIMO," *IEEE trans. green commun. netw.*, vol. 2, no. 1, pp. 25–39, Mar. 2018.

[31] N. Ghiasi, S. Mashhadi, S. Farahmand, S. M. Razavizadeh, and I. Lee, "Energy efficient AP selection for cell-free massive MIMO systems: Deep reinforcement learning approach," *IEEE trans. green commun. netw.*, vol. 7, no. 1, pp. 29–41, Mar. 2023.

[32] T. Van Chien, E. Björnson, and E. G. Larsson, "Joint power allocation and load balancing optimization for energy-efficient cell-free massive MIMO networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6798–6812, Oct. 2020.

[33] M. Sheng, Y. Dai, J. Liu, N. Cheng, X. Shen, and Q. Yang, "Delay-aware computation offloading in NOMA MEC under differentiated uploading delay," *IEEE Transactions on Wireless Communications*, vol. 19, no. 4, pp. 2813–2826, 2020.

[34] K. Wang, H. Li, Z. Ding, and P. Xiao, "Reinforcement learning based latency minimization in secure NOMA-MEC systems with hybrid SIC," *IEEE Transactions on Wireless Communications*, vol. 22, no. 1, pp. 408–422, 2023.

[35] Z. Ding, D. Xu, R. Schober, and H. V. Poor, "Hybrid NOMA offloading in multi-user MEC networks," *IEEE Transactions on Wireless Communications*, vol. 21, no. 7, pp. 5377–5391, 2022.

[36] F. Fang, Y. Xu, Z. Ding, C. Shen, M. Peng, and G. K. Karagiannidis, "Optimal resource allocation for delay minimization in NOMA-MEC networks," *IEEE Trans Commun*, vol. 68, no. 12, pp. 7867–7881, Dec. 2020.

[37] Z. Song, Y. Liu, and X. Sun, "Joint task offloading and resource allocation for NOMA-enabled multi-access mobile edge computing," *IEEE Trans Commun*, vol. 69, no. 3, pp. 1548–1564, Mar. 2021.

[38] K. Wang, F. Fang, D. B. d. Costa, and Z. Ding, "Sub-channel scheduling, task assignment, and power allocation for OMA-based and NOMA-based MEC systems," *IEEE Trans Commun*, vol. 69, no. 4, pp. 2692–2708, Apr. 2021.

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

[40] S. Abolfazli, Z. Sanaei, E. Ahmed, A. Gani, and R. Buyya, "Cloud-based augmentation for mobile devices: Motivation, taxonomies, and open challenges," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 1, pp. 337–368, 2014.

[41] D. S. Nunes, P. Zhang, and J. Sá Silva, "A survey on human-in-the-loop applications towards an internet of all," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 2, pp. 944–965, 2015.

[42] P. K. Tysowski and M. A. Hasan, "Hybrid attribute- and re-encryption-based key management for secure and scalable mobile applications in clouds," *IEEE Transactions on Cloud Computing*, vol. 1, no. 2, pp. 172–186, 2013.

[43] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, 2010.

[44] A. u. R. Khan, M. Othman, S. A. Madani, and S. U. Khan, "A survey of mobile cloud computing application models," *IEEE Commun Surv Tut*, vol. 16, no. 1, pp. 393–413, 2014.

[45] C. Kotas, T. Naughton, and N. Imam, "A comparison of Amazon Web services and Microsoft Azure cloud platforms for high performance computing," in *2018 IEEE International Conference on Consumer Electronics (ICCE)*, 2018, pp. 1–4.

[46] Z. Xu, W. Gong, Q. Xia, W. Liang, O. F. Rana, and G. Wu, "NFV-enabled IoT service provisioning in mobile edge clouds," *IEEE Trans Mob Comput*, vol. 20, no. 5, pp. 1892–1906, May 2021.

[47] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, 2018.

[48] K. Zhang, Y. Zhu, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Deep learning empowered task offloading for mobile edge computing in urban informatics," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7635–7647, 2019.

[49] M. Tang and V. W. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Transactions on Mobile Computing*, vol. 21, no. 6, pp. 1985–1997, 2022.

[50] J. Peng, H. Qiu, J. Cai, W. Xu, and J. Wang, "D2D-assisted multi-user cooperative partial offloading, transmission scheduling and computation allocating for MEC," *IEEE Trans. Wirel. Commun.*, vol. 20, no. 8, pp. 4858–4873, Aug. 2021.

[51] T. Fang, F. Yuan, L. Ao, and J. Chen, "Joint task offloading, D2D pairing, and resource allocation in device-enhanced MEC: A potential game approach," *IEEE Internet Things J.*, vol. 9, no. 5, pp. 3226–3237, Mar. 2022.

[52] M. Sun, X. Xu, X. Tao, and P. Zhang, "Large-scale user-assisted multi-task online offloading for latency reduction in D2D-enabled heterogeneous networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2456–2467, Oct. 2020.

[53] I. F. Akyildiz, A. Kak, and S. Nie, "6G and beyond: The future of wireless communications systems," *IEEE access*, vol. 8, pp. 133 995–134 030, 2020.

[54] N. H. M. Adnan, I. M. Rafiqul, and A. Z. Alam, "Massive MIMO for fifth generation (5G): Opportunities and challenges," in *2016 International Conference on Computer and Communication Engineering (ICCCE)*, 2016, pp. 47–52.

[55] A. Ashikhmin, L. Li, and T. L. Marzetta, "Interference reduction in multi-cell massive MIMO systems with large-scale fading precoding," *IEEE Transactions on Information Theory*, vol. 64, no. 9, pp. 6340–6361, 2018.

[56] J. Guo, C.-K. Wen, and S. Jin, "Deep learning-based csi feedback for beamforming in single- and multi-cell massive MIMO systems," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 1872–1884, 2021.

[57] J. Zhang, E. Björnson, M. Matthaiou, D. W. K. Ng, H. Yang, and D. J. Love, "Prospective multiple antenna technologies for beyond 5G," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 8, pp. 1637–1660, 2020.

[58] H. Q. Ngo, A. Ashikhmin, H. Yang, E. G. Larsson, and T. L. Marzetta, "Cell-free massive MIMO versus small cells," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1834–1850, Mar. 2017.

[59] S. Elhoushy, M. Ibrahim, and W. Hamouda, "Cell-free massive MIMO: A survey," *IEEE Commun. Surv. Tutor.*, vol. 24, no. 1, pp. 492–523, Mar. 2022.

[60] X. Zhang, H. Qi, X. Zhang, and L. Han, "Energy-efficient resource allocation and data transmission of cell-free internet of things," *IEEE Internet of Things Journal*, vol. 8, no. 20, pp. 15 107–15 116, 2021.

[61] D. Goodman, R. Valenzuela, K. Gayliard, and B. Ramamurthi, "Packet reservation multiple access for local wireless communications," *IEEE Transactions on Communications*, vol. 37, no. 8, pp. 885–890, 1989.

[62] C. Ciochina and H. Sari, "A review of OFDMA and single-carrier FDMA," in *2010 European Wireless Conference (EW)*, 2010, pp. 706–710.

[63] D. Falconer, F. Adachi, and B. Gudmundson, "Time division multiple access methods for wireless personal communications," *IEEE Communications Magazine*, vol. 33, no. 1, pp. 50–57, 1995.

[64] J. Salehi, "Code division multiple-access techniques in optical fiber networks. i. fundamental principles," *IEEE Transactions on Communications*, vol. 37, no. 8, pp. 824–833, 1989.

[65] M. Morelli, C.-C. J. Kuo, and M.-O. Pun, "Synchronization techniques for orthogonal frequency division multiple access (OFDMA): A tutorial review," *Proceedings of the IEEE*, vol. 95, no. 7, pp. 1394–1427, 2007.

[66] L. Dai, B. Wang, Y. Yuan, S. Han, I. Chih-lin, and Z. Wang, "Non-orthogonal multiple access for 5g: solutions, challenges, opportunities, and future research trends," *IEEE Communications Magazine*, vol. 53, no. 9, pp. 74–81, 2015.

[67] I. Budhiraja, N. Kumar, S. Tyagi, S. Tanwar, Z. Han, M. J. Piran, and D. Y. Suh, "A systematic review on noma variants for 5G and beyond," *IEEE Access*, vol. 9, pp. 85 573–85 644, 2021.

[68] Z. Ding, X. Lei, G. K. Karagiannidis, R. Schober, J. Yuan, and V. K. Bhargava, "A survey on non-orthogonal multiple access for 5G networks: Research challenges and future trends," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 10, pp. 2181–2195, 2017.

[69] O. Shental, B. M. Zaidel, and S. S. Shitz, "Low-density code-domain NOMA: Better be regular," in *2017 IEEE International Symposium on Information Theory (ISIT)*, 2017, pp. 2628–2632.

[70] M. AL-Imari, M. A. Imran, R. Tafazolli, and D. Chen, "Performance evaluation of low density spreading multiple access," in *2012 8th International Wireless*

Communications and Mobile Computing Conference (IWCMC), 2012, pp. 383–388.

[71] H. Nikopour and H. Baligh, "Sparse code multiple access," in *2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2013, pp. 332–336.

[72] S. Chen, B. Ren, Q. Gao, S. Kang, S. Sun, and K. Niu, "Pattern division multiple access—a novel nonorthogonal multiple access for fifth-generation radio networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3185–3196, 2017.

[73] M. Wildemeersch, T. Q. S. Quek, M. Kountouris, A. Rabbachin, and C. H. Slump, "Successive interference cancellation in heterogeneous networks," *IEEE Trans Commun*, vol. 62, no. 12, pp. 4440–4453, Dec. 2014.

[74] J. M. Hamamreh, H. M. Furqan, and H. Arslan, "Classifications and applications of physical layer security techniques for confidentiality: A comprehensive survey," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 2, pp. 1773–1828, 2019.

[75] H. Shariatmadari, R. Ratasuk, S. Iraji, A. Laya, T. Taleb, R. Jäntti, and A. Ghosh, "Machine-type communications: current status and future perspectives toward 5G systems," *IEEE Communications Magazine*, vol. 53, no. 9, pp. 10–17, 2015.

[76] R. I. Ansari, C. Chrysostomou, S. A. Hassan, M. Guizani, S. Mumtaz, J. Rodriguez, and J. J. P. C. Rodrigues, "5G D2D networks: Techniques, challenges, and future prospects," *IEEE Syst J*, vol. 12, no. 4, pp. 3970–3984, Dec. 2018.

[77] M. A. Sedaghat and R. R. Müller, "On user pairing in uplink NOMA," *IEEE Transactions on Wireless Communications*, vol. 17, no. 5, pp. 3474–3486, 2018.

[78] H. Zhang, F. Fang, J. Cheng, K. Long, W. Wang, and V. C. M. Leung, "Energy-efficient resource allocation in NOMA heterogeneous networks," *IEEE Wireless Communications*, vol. 25, no. 2, pp. 48–53, 2018.

[79] F. Fang, H. Zhang, J. Cheng, and V. C. Leung, "Energy-efficient resource scheduling for NOMA systems with imperfect channel state information," in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 1–5.

[80] R. Gopal and N. BenAmmar, "Framework for unifying 5G and next generation satellite communications," *IEEE Network*, vol. 32, no. 5, pp. 16–24, 2018.

[81] J. A. Ansere, G. Han, L. Liu, Y. Peng, and M. Kamal, "Optimal resource allocation in energy-efficient internet-of-things networks with imperfect CSI," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5401–5411, Jun. 2020.

[82] A. Kamilaris and A. Pitsillides, "Mobile phone computing and the internet of things: A survey," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 885–898, Dec. 2016.

[83] A. A. Adebayo, D. B. Rawat, and M. Song, "Energy-efficient multivariate privacy-aware rf spectrum reservation in wireless virtualization for wireless internet of things," *IEEE Trans. on Green Commun. and Netw.*, vol. 5, no. 2, pp. 682–692, Jun. 2021.

[84] C.-H. Hong and B. Varghese, "Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms," *ACM Comput. Surv.*, vol. 52, no. 5, Sep. 2019.

[85] M.-H. Chen, B. Liang, and M. Dong, "Multi-user multi-task offloading and resource allocation in mobile cloud systems," *IEEE Trans. Wirel. Commun.*, vol. 17, no. 10, pp. 6790–6805, Oct. 2018.

[86] A. Naouri, H. Wu, N. A. Nouri, S. Dhelim, and H. Ning, "A novel framework for mobile-edge computing by optimizing task offloading," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 13 065–13 076, Aug. 2021.

[87] Z. Yan, P. Cheng, Z. Chen, B. Vucetic, and Y. Li, "Two-dimensional task offloading for mobile networks: An imitation learning framework," *IEEE/ACM Trans. Netw.*, vol. 29, no. 6, pp. 2494–2507, Dec. 2021.

[88] C. Suraci, S. Pizzi, A. Molinaro, and G. Araniti, "MEC and D2D as enabling technologies for a secure and lightweight 6G eHealth system," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 11 524–11 532, Jul. 2022.

[89] C. V. Anamuro, N. Varsier, J. Schwoerer, and X. Lagrange, "Distance-aware relay selection in an energy-efficient discovery protocol for 5G D2D communication," *IEEE Trans. Wirel. Commun.*, vol. 20, no. 7, pp. 4379–4391, Jul. 2021.

[90] A. H. Land and A. G. Doig, "An automatic method for solving discrete programming problems," *Springer Berlin Heidelberg*, pp. 105–132, 2010.

[91] X. Li, L. Huang, H. Wang, S. Bi, and Y.-J. A. Zhang, "An integrated optimization-learning framework for online combinatorial computation offloading in MEC networks," *IEEE Wirel Commun*, vol. 29, no. 1, pp. 170–177, Feb. 2022.

[92] M. Feng, M. Krunz, and W. Zhang, "Task partitioning and user association for latency minimization in mobile edge computing networks," in *IEEE INFOCOM 2021*, May. 2021, pp. 1–6.

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

[94] R. Yu, G. Xue, Y. Wan, J. Tang, D. Yang, and Y. Ji, "Robust resource provisioning in time-varying edge networks," in *Mobihoc '20*. New York, NY, USA: Association for Computing Machinery, Oct. 2020, p. 21–30.

[95] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[96] F. Jiang, K. Wang, L. Dong, C. Pan, W. Xu, and K. Yang, "Deep-learning-based joint resource scheduling algorithms for hybrid MEC networks," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6252–6265, Jul. 2020.

[97] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans Mob Comput*, vol. 19, no. 11, pp. 2581–2593, Nov. 2020.

[98] F. Jiang, K. Wang, L. Dong, C. Pan, and K. Yang, "Stacked autoencoder-based deep reinforcement learning for online resource scheduling in large-scale MEC networks," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9278–9290, Oct. 2020.

[99] T. Achterberg, T. Koch, and A. Martin, "Branching rules revisited," *Oper. Res. Lett.*, vol. 33, no. 1, pp. 42–54, Jan. 2005.

[100] C. Sonmez, A. Ozgovde, and C. Ersoy, "Edgecloudsim: An environment for performance evaluation of edge computing systems," *Trans. Emerg. Telecommun. Technol.*, vol. 29, no. 11, p. e3493, Aug. 2018.

[101] W. Xiang, J. Li, Y. Zhou, P. Cheng, J. Jin, and K. Yu, "Digital twin empowered industrial IoT based on credibility-weighted swarm learning," *IEEE Trans Industr Inform*, pp. 1–10, 2023.

[102] D. Castanheira and A. Gameiro, "Low complexity and high-resolution line spectral estimation using cyclic minimization," *IEEE Trans. Signal Process*, vol. 67, no. 24, pp. 6285–6300, Dec. 2019.

[103] A. Mokhtari and A. Koppel, "High-dimensional nonconvex stochastic optimization by doubly stochastic successive convex approximation," *IEEE Transactions on Signal Processing*, vol. 68, pp. 6287–6302, 2020.

[104] S. A. Lippman, *Dynamic Programming and Markov Decision Processes.* The New Palgrave Dictionary of Economics, 1987.

[105] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Comput. Surv.*, vol. 50, no. 2, Apr. 2017. [Online]. Available: https://doi.org/10.1145/3054912

[106] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process Mag*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[107] B. Jang, M. Kim, G. Harerimana, and J. W. Kim, "Q-learning algorithms: A comprehensive classification and applications," *IEEE Access*, vol. 7, pp. 133 653–133 667, Sep. 2019.

[108] M. Wulfmeier, P. Ondruska, and I. Posner, "Maximum entropy deep inverse reinforcement learning," *arXiv preprint arXiv:1507.04888*, Jul. 2015.

[109] E. T. Jaynes, "Information theory and statistical mechanics," *Phys. Rev.*, vol. 106, no. 4, p. 620, May 1957.

[110] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans Neural Netw Learn Syst*, vol. 32, no. 1, pp. 4–24, Jan 2021.

[111] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four

research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[112] H. Cheng, J. T. Zhou, W. P. Tay, and B. Wen, "Graph neural networks with triple attention for few-shot learning," *IEEE Trans Multimedia*, pp. 1–15, 2023.

[113] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Networks*, vol. 20, no. 1, pp. 61–80, Jan. 2009.

[114] L. Ruiz, F. Gama, and A. Ribeiro, "Gated graph recurrent neural networks," *IEEE Trans. Signal Process*, vol. 68, pp. 6303–6318, Oct. 2020.

[115] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, Jan. 2014.

[116] R. Zhang, Y. Li, Y. Ruan, H. Zhang, W. Wang, and W. Wang, "CQI-based interference management scheme for D2D communication underlaying cellular networks," in *IEEE WCNCW 2015*, Mar. 2015, pp. 347–351.

[117] Y. He, J. Ren, G. Yu, and Y. Cai, "Joint computation offloading and resource allocation in D2D enabled MEC networks," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[118] J. Zheng, J. Zhang, E. Björnson, Z. Li, and B. Ai, "Cell-free massive MIMO-OFDM for high-speed train communications," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 10, pp. 2823–2839, Oct. 2022.

[119] H. He, X. Yu, J. Zhang, S. Song, and K. B. Letaief, "Cell-free massive mimo for 6G wireless communication networks," *J. Commun. Netw.*, vol. 6, no. 4, pp. 321–335, Dec. 2021.

[120] H. A. Ammar, R. Adve, S. Shahbazpanahi, G. Boudreau, and K. V. Srinivas, "User-centric cell-free massive MIMO networks: A survey of opportunities, challenges and solutions," *IEEE Commun. Surv. Tutor.*, vol. 24, no. 1, pp. 611–652, Mar. 2022.

[121] M. Attarifar, A. Abbasfar, and A. Lozano, "Modified conjugate beamforming for cell-free massive MIMO," *IEEE Wireless Commun. Lett.*, vol. 8, no. 2, pp. 616–619, Apr. 2019.

[122] M. Guo and M. C. Gursoy, "Joint activity detection and channel estimation in cell-free massive MIMO networks with massive connectivity," *IEEE Trans. Commun.*, vol. 70, no. 1, pp. 317–331, Jan. 2022.

[123] E. Björnson and L. Sanguinetti, "Making cell-free massive MIMO competitive with MMSE processing and centralized implementation," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 77–90, Jan. 2020.

[124] W. Zeng, Y. He, B. Li, and S. Wang, "Pilot assignment for cell-free massive MIMO systems using a weighted graphic framework," *IEEE Trans. Veh. Technol.*, vol. 70, no. 6, pp. 6190–6194, Jun. 2021.

[125] S. Buzzi, C. D'Andrea, M. Fresia, Y.-P. Zhang, and S. Feng, "Pilot assignment in cell-free massive MIMO based on the hungarian algorithm," *IEEE Wireless Commun. Lett.*, vol. 10, no. 1, pp. 34–37, Jan. 2021.

[126] F. Guo, H. Lu, and Z. Gu, "Joint power and user grouping optimization in cell-free massive MIMO systems," *IEEE Trans. Wireless Commun.*, vol. 21, no. 2, pp. 991–1006, Feb. 2022.

[127] G. Interdonato, M. Karlsson, E. Björnson, and E. G. Larsson, "Local partial zero-forcing precoding for cell-free massive MIMO," *IEEE Trans. Wireless Commun.*, vol. 19, no. 7, pp. 4758–4774, Jul. 2020.

[128] G. Femenias, N. Lassoued, and F. Riera-Palou, "Access point switch on/off strategies for green cell-free massive MIMO networking," *IEEE Access*, vol. 8, pp. 21 788–21 803, Jan. 2020.

[129] J. Lee and S. Leyffer, *Mixed integer nonlinear programming*. Springer Science & Business Media, Dec. 2011, vol. 154.

[130] P. Cheng, Y. Chen, M. Ding, Z. Chen, S. Liu, and Y.-P. P. Chen, "Deep reinforcement learning for online resource allocation in IoT networks: Technology, development, and future challenges," *IEEE Communications Magazine*, vol. 61, no. 6, pp. 111–117, 2023.

[131] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, Jul. 2018, pp. 1861–1870.

[132] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Adv. Neural Inf. Process. Syst.*, vol. 30, Dec. 2017.

[133] K. Pratik, B. D. Rao, and M. Welling, "RE-MIMO: Recurrent and permutation equivariant neural MIMO detection," *IEEE Trans. Signal Process.*, vol. 69, no. Dec., pp. 459–473, 2021.

[134] J. Gao, Y. Wu, S. Shao, W. Yang, and H. V. Poor, "Energy efficiency of massive random access in MIMO quasi-static rayleigh fading channels with finite blocklength," *IEEE Trans. Inf. Theory*, vol. 69, no. 3, pp. 1618–1657, Mar. 2023.

[135] O. Elijah, C. Y. Leow, T. A. Rahman, S. Nunoo, and S. Z. Iliya, "A comprehensive survey of pilot contamination in massive MIMO—5G system," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 2, pp. 905–923, 2016.

[136] S. M. Kay, *Fundamentals of statistical signal processing: estimation theory.* Prentice-Hall, Inc., Mar. 1993.

[137] T. T. Vu, D. T. Ngo, N. H. Tran, H. Q. Ngo, M. N. Dao, and R. H. Middleton, "Cell-free massive MIMO for wireless federated learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6377–6392, Oct. 2020.

[138] M. Farooq, H. Q. Ngo, E.-K. Hong, and L.-N. Tran, "Utility maximization for large-scale cell-free massive MIMO downlink," *IEEE Trans. Commun.*, vol. 69, no. 10, pp. 7050–7062, Oct. 2021.

[139] T. C. Mai, H. Q. Ngo, and L.-N. Tran, "Energy efficiency maximization in large-scale cell-free massive MIMO: A projected gradient approach," *IEEE Trans. Wireless Commun.*, vol. 21, no. 8, pp. 6357–6371, Aug. 2022.

[140] H. Yang and T. L. Marzetta, "Capacity performance of multicell large-scale antenna systems," in *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Oct. 2013, pp. 668–675.

[141] E. Nayebi, A. Ashikhmin, T. L. Marzetta, H. Yang, and B. D. Rao, "Precoding and power optimization in cell-free massive MIMO systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 7, pp. 4445–4459, Jul. 2017.

[142] C. Desset, B. Debaillie, V. Giannini, A. Fehske, G. Auer, H. Holtkamp, W. Wajda, D. Sabella, F. Richter, M. J. Gonzalez, H. Klessig, I. Gódor, M. Olsson, M. A. Imran, A. Ambrosy, and O. Blume, "Flexible power modeling of LTE base stations," in *2012 IEEE Wireless Communications and Networking Conference (WCNC)*, Apr. 2012, pp. 2858–2862.

[143] Y. Shi, J. Zhang, and K. B. Letaief, "Group sparse beamforming for green cloud-ran," *IEEE Transactions on Wireless Communications*, vol. 13, no. 5, pp. 2809–2823, 2014.

[144] R. Kannan and C. L. Monma, "On the computational complexity of integer programming problems," in *Optimization and Operations Research: Proceedings of a Workshop Held at the University of Bonn, October 2–8, 1977*. Springer, 1978, pp. 161–172.

[145] A. Liu, V. K. N. Lau, and B. Kananian, "Stochastic successive convex approximation for non-convex constrained stochastic optimization," *IEEE Trans. Signal Process.*, vol. 67, no. 16, pp. 4189–4203, Aug. 2019.

[146] X. Wang, S. Wang, X. Liang, D. Zhao, J. Huang, X. Xu, B. Dai, and Q. Miao, "Deep reinforcement learning: A survey," *IEEE Trans. Neural Networks Learn. Syst.*, pp. 1–15, Sep. 2022.

[147] Y. Xu, Z. Zhao, P. Cheng, Z. Chen, M. Ding, B. Vucetic, and Y. Li, "Constrained reinforcement learning for resource allocation in network slicing," *IEEE Commun. Lett.*, vol. 25, no. 5, pp. 1554–1558, May 2021.

[148] S. Wang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning with communication transformer for adaptive live streaming in wireless edge networks," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 1, pp. 308–322, Jan. 2022.

[149] W. Zhang, Z. Yin, Z. Sheng, Y. Li, W. Ouyang, X. Li, Y. Tao, Z. Yang, and B. Cui, "Graph attention multi-layer perceptron," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Aug. 2022, pp. 4560–4570.

[150] G. Scutari, F. Facchinei, and L. Lampariello, "Parallel and distributed methods for constrained nonconvex optimization—part i: Theory," *IEEE Trans. Signal Process.*, vol. 65, no. 8, pp. 1929–1944, Apr. 2017.

[151] A. S. Bedi, K. Rajawat, V. Aggarwal, and A. Koppel, "Escaping saddle points

for successive convex approximation," *IEEE Trans. Signal Process.*, vol. 70, pp. 307–321, 2022.

[152] S. Bubeck *et al.*, "Convex optimization: Algorithms and complexity," *Found. Trends Mach. Learn.*, vol. 8, no. 3-4, pp. 231–357, Nov. 2015.

[153] Z. Wang, E. K. Tameh, and A. R. Nix, "Joint shadowing process in urban peer-to-peer radio channels," *IEEE Trans. Veh. Technol.*, vol. 57, no. 1, pp. 52–64, Jan. 2008.

[154] A. Tang, J. Sun, and K. Gong, "Mobile propagation loss with a low base station antenna for nlos street microcells in urban area," in *IEEE VTS 53rd Vehicular Technology Conference, Spring 2001. Proceedings (Cat. No.01CH37202)*, vol. 1, 2001, pp. 333–336 vol.1.

[155] G. Wang, P. Cheng, Z. Chen, W. Xiang, B. Vucetic, and Y. Li, "Inverse reinforcement learning with graph neural networks for IoT resource allocation," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5.

[156] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun Surv Tut*, vol. 19, no. 4, pp. 2322–2358, Aug. 2017.

[157] M. Hong, Q. Li, and Y.-F. Liu, "Decomposition by successive convex approximation: A unifying approach for linear transceiver design in heterogeneous networks," *IEEE Trans. Wirel. Commun.*, vol. 15, no. 2, pp. 1377–1392, Feb. 2016.

[158] R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville, and M. Bellemare, "Reincarnating reinforcement learning: Reusing prior computation to accelerate

progress," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35. Curran Associates, Inc., Nov. 2022, pp. 28 955–28 971.

[159] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, Aug. 2017.

[160] A. Al-Hilo, M. Samir, C. Assi, S. Sharafeddine, and D. Ebrahimi, "UAV-assisted content delivery in intelligent transportation systems-joint trajectory planning and cache management," *IEEE trans Intell Transp Syst*, vol. 22, no. 8, pp. 5155–5167, Aug. 2021.

[161] Y.-H. Liu and K. C.-J. Lin, "Traffic-aware resource allocation for multi-user beamforming," *IEEE Trans Mob Comput*, vol. 22, no. 6, pp. 3677–3690, Jun. 2023.

[162] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang, "Edge intelligence for energy-efficient computation offloading and resource allocation in 5G beyond," *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 12 175–12 186, Oct. 2020.

[163] Y. He, W. Zhu, and L. Guan, "Optimal resource allocation for pervasive health monitoring systems with body sensor networks," *IEEE Trans Mob Comput*, vol. 10, no. 11, pp. 1558–1575, Nov. 2011.

[164] X. Lu, P. Wang, D. Niyato, D. I. Kim, and Z. Han, "Wireless networks with RF energy harvesting: A contemporary survey," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 2, pp. 757–789, 2015.

[165] Z. Fei, B. Li, S. Yang, C. Xing, H. Chen, and L. Hanzo, "A survey of multi-objective optimization in wireless sensor networks: Metrics, algorithms, and open problems," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 1, pp. 550–586, 2017.

[166] A. Santiago, H. J. F. Huacuja, B. Dorronsoro, J. E. Pecero, C. G. Santillan, J. J. G. Barbosa, and J. C. S. Monterrubio, "A survey of decomposition methods for multi-objective optimization," *Recent advances on hybrid approaches for designing intelligent systems*, pp. 453–465, 2014.

[167] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 2010, pp. 242–264.

[168] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 1126–1135.