Deep Reinforcement Learning for Resource Allocation in Beyond 5G Systems

by

Rui Huang

M.Sc., Shanghai Jiao Tong University, 2018
B.Sc., Chongqing University, 2015

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

The Faculty of Graduate and Postdoctoral Studies

(Electrical and Computer Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

December 2022

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the dissertation entitled:

Deep Reinforcement Learning for Resource Allocation in Beyond 5G Systems

submitted by **Rui Huang** in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical and Computer Engineering.

Examining Committee:

Vincent W.S. Wong, Professor, Electrical and Computer Engineering, UBC
Supervisor

Vijay Bhargava, Professor, Electrical and Computer Engineering, UBC
Supervisory Committee Member

Lutz Lampe, Professor, Electrical and Computer Engineering, UBC
Supervisory Committee Member

Ryozo Nagamune, Professor, Mechanical Engineering, UBC
University Examiner

Jane Z. Wang, Professor, Electrical and Computer Engineering, UBC
University Examiner

Hai Jiang, Professor, Electrical and Computer Engineering, University of Alberta
External Examiner

Abstract

With the rapid development of wireless network-enabled applications, the beyond fifth generation (B5G) wireless systems are required to support a large number of mobile and Internet of things (IoT) devices. Moreover, the growing demand for applications with high data rate requirements, including virtual reality (VR), brings new challenges to the B5G wireless systems. While several emerging physical layer and medium access control techniques, including grant-free multiple access (GFMA), intelligent reflecting surface (IRS), and rate-splitting (RS), introduce additional degrees of freedom (DoF) to the B5G wireless systems, novel resource allocation algorithms are required to fully exploit their potentials. In this thesis, we propose deep reinforcement learning (DRL)-based algorithms to efficiently optimize the DoF and improve the performance of B5G wireless systems. First, we propose a distributed pilot sequence selection scheme for GFMA systems. The proposed scheme maximizes the aggregate throughput by mitigating pilot sequence selection collisions. In the proposed scheme, a distributed pilot sequence selection policy is obtained by using a multiagent DRL technique. Second, we propose a joint user scheduling, phase shift control, and beamforming optimization algorithm for IRS-aided systems. We formulate a joint optimization problem for maximizing the aggregate throughput and achieving the proportional fairness in IRS-aided systems. The proposed algorithm exploits neural combinatorial optimization (NCO) to determine user scheduling, and uses curriculum learning (CL) and deep deterministic policy gradient (DDPG) to optimize the beamforming vectors and IRS phase shifts. Third, we propose a novel IRS-aided RS VR streaming system. We formulate an optimization problem for maximizing the achievable bitrate of the 360-degree video subject to the quality of experience (QoE) constraints of the users. We propose a deep

deterministic policy gradient with imitation learning (Deep-GRAIL) algorithm, in which we leverage DRL and the human expert knowledge to optimize the IRS phase shifts, RS parameters, beamforming vectors, and bitrate selection of 360-degree videos. Simulation results show that the proposed DRL-based algorithms improve the performance of B5G wireless systems by efficiently optimizing the DoF. Our results also demonstrate the effectiveness of empowering the DRL techniques with the human expert knowledge of wireless systems.

Lay Summary

In order to support the growing number of wireless devices and the emerging high data-rate applications, the additional degrees of freedom (DoF) introduced by novel physical layer and medium access control techniques can be exploited to improve the performance of the beyond fifth generation (B5G) wireless systems. While conventional optimization methods can be applied to optimize the DoF, they are in general computationally intensive and difficult to deploy in practical systems. To tackle this issue, we design deep reinforcement learning (DRL)-based algorithms to efficiently optimize the DoF. We also propose to exploit the human expert knowledge in DRL-based algorithms design to facilitate the learning process. With the proposed DRL-based algorithms, the B5G wireless systems can benefit from the additional propagation channels created by intelligent reflecting surfaces (IRSs), the multiplexing gain introduced by rate-splitting (RS), and throughput improvement obtained from the proposed grant-free multiple access (GFMA) scheme.

Preface

Chapters 2−4 encompass the work which has been published or is currently under review. The corresponding papers are under the supervision of Prof. Vincent Wong. Prof. Robert Schober has co-authored the papers corresponding to Chapters 2 and 4, and provided valuable and constructive comments.

For all chapters, I hereby declare that I am the first author of the corresponding papers. I conducted the literature survey on related topics, formulated the problems, designed the solution approaches, and performed simulations to evaluate the performance. I prepared all the paper drafts. My supervisor, Prof. Vincent Wong, guided the research, validated the results, and provided comments on improving the manuscripts. The following publications describe the work completed in this thesis.

Journal Papers, Published or Accepted

- R. Huang, V. W.S. Wong, and R. Schober, "Throughput optimization for grant-free multiple access with multiagent deep reinforcement learning," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 228−242, Jan. 2021.

- R. Huang and V. W.S. Wong, "Joint user scheduling, phase shift control, and beamforming optimization in intelligent reflecting surface-aided systems," *IEEE Trans. Wireless Commun.*, vol. 21, no 9, pp. 7521−7535, Sept. 2022.

- R. Huang, V. W.S. Wong, and R. Schober, "Rate-splitting for intelligent reflecting surface-aided multiuser VR streaming," accepted for publication in *IEEE J. Sel. Areas Commun.*, Dec. 2022.

Conference Papers, Published

- R. Huang, V. W.S. Wong, and R. Schober, "Throughput optimization in grant-free NOMA with deep reinforcement learning," in *Proc. of IEEE Global Commun. Conf. (GLOBECOM)*, Waikoloa, HI, Dec. 2019.

- R. Huang and V. W.S. Wong, "Neural combinatorial optimization for throughput maximization in IRS-aided systems," in *Proc. of IEEE Global Commun. Conf. (GLOBECOM)*, Taipei, Taiwan, Dec. 2020.

- R. Huang and V. W.S. Wong, "Towards reliable communications in intelligent reflecting surface-aided cell-free MIMO systems," in *Proc. of IEEE Global Commun. Conf. (GLOBECOM)*, Madrid, Spain, Dec. 2021.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

2-D	Two-dimensional
3-D	Three-dimensional
ACB	Access class barring
ACK	Acknowledgement
AO	Alternating optimization
AoI	Age of information
AR	Augmented reality
AWGN	Additive white Gaussian noise
B5G	Beyond fifth generation
CDF	Cumulative distribution function
CL	Curriculum learning
CL-DDPG	Curriculum learning deep deterministic policy gradient
CNN	Convolutional neural network
CSI	Channel state information
CTDE	Centralized training distributed execution
DCO	Differentiable convex optimization
DDPG	Deep deterministic policy gradient
Deep-GRAIL	Deep deterministic policy gradient with imitation learning
DNN	Deep neural network
DoF	Degrees of freedom
DRL	Deep reinforcement learning
DRQN	Deep recurrent Q-network

DUPB	DRL-based user scheduling, phase shift control, and beamforming optimization
FC	Fully connected
FoV	Field-of-view
FP	Fractional programming
GFMA	Grant-free multiple access
HMD	Head-mounted device
HMIMOS	Holographic multiple-input multiple-output surface
IoT	Internet of things
IRS	Intelligent reflecting surface
LTE	Long Term Evolution
LoS	Line-of-sight
LSTM	Long short-term memory
MA-DRL	Multiagent deep reinforcement learning
Mbps	Mega bits per second
MDP	Markov decision process
MIMO	Multiple-input multiple-output
MLP	Multi-layer perceptron
MU-MISO	Multiuser multiple-input single-output
NCO	Neural combinatorial optimization
PRB	Physical resource block
QFI	Quality of service flow identifier
QoE	Quality of experience
QoS	Quality of service
ReLU	Rectified linear unit
ResNet	Residual network
RL	Reinforcement learning
RNN	Recurrent neural network
RRC	Radio resource control

RS	Rate-splitting
RSMA	Rate-splitting multiple access
RS-NOUM	Rate-splitting non-orthogonal unicast and multicast
SCA	Successive convex approximation
SDR	Semidefinite relaxation
SGD	Stochastic gradient descent
SIC	Successive interference cancellation
SINR	Signal-to-interference-plus-noise ratio
SL	Supervised learning
TD	Temporal difference
UAV	Unmanned aerial vehicle
UE	User equipment
USD	United States dollars
VR	Virtual reality
WMMSE	Weighted minimum mean square error

Acknowledgements

I would like to thank my supervisor, Prof. Vincent Wong, for his support and guidance throughout my Ph.D. program. He has provided numerous comments, suggestions, and advice which helped me to constantly improve the quality of my work, and encouraged me to think critically. Prof. Wong has also applied and arranged the resources that are crucial to my Ph.D. program. I would also like to thank Prof. Robert Schober for his constructive and insightful comments on my research projects. I have learned a lot from Prof. Wong and Prof. Schober during my Ph.D. program.

My thanks also go to the rest of my committee members, Prof. Vijay Bhargava and Prof. Lutz Lampe, and my committee Chair, Prof. Cyril Leung, for their valuable comments and questions. They have kindly helped me to proceed with my Ph.D. program smoothly.

I am grateful for the computational services provided by the Digital Research Alliance of Canada. My work was also supported in part by the University of British Columbia and the Natural Sciences and Engineering Research Council of Canada.

Many thanks to my fellow labmates in K4090. They are all excellent researchers and they always surprise me with innovative ideas. Many thanks to my friends, especially Bowen, Joey, Ronghua, Yunshu, and Yuxi, for sharing experiences and opinions with me.

I thank Vancouver for its beautiful mountains and sea. They have brought me countless outdoor adventures and offered me inner peace. I also thank my neighbors and the local communities in Vancouver for their kindness and inclusiveness.

Most importantly, I would like to thank my mother for her support all the time. She is also an excellent engineer who gives me the motivation to pursue my career.

Dedication

To my mother

Chapter 1

Introduction

The beyond fifth generation (B5G) wireless systems is a key technique to support various types of wireless network-enabled applications, including smart home, smart city, intelligent transportation systems, and eHealth [1]. It is estimated that by 2027, there will be 9.1 billion mobile subscriptions and 30.2 billion connections from Internet of things (IoT) devices [2]. Besides, due to the growing popularity of emerging cellular network services such as mobile video streaming, multiplayer mobile gaming, augmented reality (AR), and virtual reality (VR) [3, 4], the global data traffic is expected to more than double by 2025 [2]. The B5G wireless systems are required to offer a high spectral efficiency and deliver a large amount of data to support both IoT devices and mobile users.

In IoT applications, different IoT devices may have different data rate requirements depending on their applications and services [5]. For example, the IoT devices for road traffic monitoring (e.g., cameras) in intelligent transportation systems may need to send the real-time sensing data back to the control center to monitor and control traffic conditions. Meanwhile, those IoT devices for sensing environmental parameters, such as greenhouse temperature and CO2 concentration in smart agriculture, may only need to send data packets in a sporadic and less frequent manner [5]. To tackle the heterogeneity of the IoT applications, the network resources in the B5G wireless systems need to be properly allocated to the IoT devices based on their specific applications and corresponding data rate requirements. One solution is to allow the base stations in B5G wireless systems to inform the IoT devices about the allocated resources by sending downlink control signals. However, this centralized resource management incurs a significant amount of signaling overhead when the number of IoT devices is large. Another solution is to design *distributed* resource

allocation algorithms for IoT devices. In distributed resource allocation algorithms, the control signaling overhead can be alleviated by allowing each IoT device to make its own decision on which resources to use. However, without the centralized scheduling from the base stations in B5G wireless systems and the information exchange between IoT devices, interference and collisions may occur when multiple IoT devices choose to occupy the same resources, which can lead to performance degradation. Hence, distributed resource allocation algorithms for IoT devices with different service and data rate requirements need to be properly designed to tackle the lack of centralized scheduling and information exchange.

For cellular and mobile use cases of B5G wireless systems, mobile multimedia streaming, including the streaming of 360-degree videos in VR applications, creates new business models for various industries such as telecommunication, retail, education, and entertainment. The increasing number of VR users and the growing demand for VR streaming introduce new challenges to the current wireless systems. First, the bitrate of a high-resolution 360-degree video can be much higher than that of conventional multimedia applications. For example, a 4K 360-degree video may have a bitrate of 78 Mega bits per second (Mbps) [6]. In addition, a VR user may experience motion sickness when the motion-to-photon delay, i.e., the delay between the head movement and the requested 360-degree video segments being rendered at the head-mounted device (HMD) of this user, is larger than 20 milliseconds (ms) [7]. To mitigate these issues, the data transmission of 360-degree videos should be accomplished within a short downlink transmission window. Hence, the B5G wireless systems have to be able to support a high data transmission rate to meet the requirement of VR streaming.

An effective way to improve the spectral efficiency of wireless systems is to design resource allocation algorithms to control the degrees of freedom (DoF) in wireless systems. Various algorithms have been proposed to solve resource allocation problems in wireless systems by using conventional optimization methods. During the past two decades, convex optimization [8] has been widely applied to optimize the DoF, such as transmit power and

beamforming vectors, in wireless systems. The popularity of convex optimization comes from the fact that it can obtain the optimal solution for those resource allocation problems in wireless systems that are convex. In addition, some nonconvex resource allocation problems may have the *hidden convexity*, which allows these problems to be transformed or relaxed into convex optimization problems. Apart from convex optimization, alternating optimization (AO) [9] is a powerful tool for jointly optimizing several sets of coupled variables by optimizing each set of these variables iteratively. AO has been widely applied to solve resource allocation problems in wireless systems where several sets of DoF have to be jointly optimized. Moreover, other optimization tools, such as fractional programming (FP) [10] and semidefinite relaxation (SDR) [11], have been used in the existing resource allocation algorithms design for wireless systems [12–14]. However, the conventional optimization methods in general have high computational complexity, and they may require relatively long runtimes to obtain the solutions. In addition, the computational complexity of the conventional optimization methods increases significantly with respect to the number of DoF of the wireless systems. For example, solving a convex optimization problem in a wireless system with n optimization variables using the interior point method may incur a computational complexity of $\mathcal{O}(n^{3.5})$ [8]. Hence, for B5G wireless systems with a large number of DoF, the aforementioned conventional optimization methods can be computationally intensive to implement. Computationally efficient algorithms need to be designed for resource allocation in B5G wireless systems.

Deep reinforcement learning (DRL) [15–18] is a learning technique based on deep neural networks (DNNs) [19]. DRL does not rely on a pre-established system model and is a powerful tool for solving optimization problems with large decision spaces. DRL-based solutions can be generalized and applied to problems with different system models and objectives, without necessarily relying on the (hidden) convexity of the optimization problems [20, 21]. Moreover, after the training process, the learning agent in DRL can obtain solutions of the optimization problems with significantly lower computational complexity than that of the conventional optimization methods. The aforementioned advantages of

DRL in terms of learning capability and computational efficiency make it a promising technique for designing resource allocation algorithms for B5G wireless systems.

In this thesis, we aim to design resource allocation algorithms for optimizing the DoF of B5G wireless systems to improve the system performance in a computationally efficient manner. In particular, we propose a multiagent DRL (MA-DRL)-based pilot sequence selection scheme for grant-free multiple access (GFMA) systems [22, 23]. The proposed algorithm exploits the capability of MA-DRL to learn a distributed scheme that fosters collaboration between IoT devices. We also propose DRL-based algorithms for joint optimization of the DoF in intelligent reflecting surface (IRS)-aided systems [24, 25] and rate-splitting (RS) systems [26–28]. In addition, we propose to use curriculum learning (CL) [29, 30] and imitation learning [31, 32] to improve the performance of DRL-based algorithms in B5G wireless systems by learning from the hidden convexity of the resource allocation problems.

The rest of this chapter is organized as follows. In Section 1.1, we provide an overview of the GFMA systems. In Section 1.2, we present the concept of the IRS-aided multiuser systems. In Section 1.3, we introduce the RS systems, and describe the benefits of combining RS with IRS in multiuser VR streaming systems. An overview of DRL and its application for algorithms design in wireless systems are presented in Section 1.4. The results and contributions of this thesis are summarized in Section 1.5. The thesis organization is provided in Section 1.6.

1.1 GFMA Systems

GFMA is a multiple access technique in B5G wireless systems for reducing the access delay of IoT devices [22, 23]. In GFMA, an IoT device selects a pilot sequence from a pre-allocated resource pool, and transmits its data to the base station without sending an access request to the base station *a priori*. The base station sends an acknowledgement (ACK) to the device upon successful decoding. The timing sequence diagram for GFMA

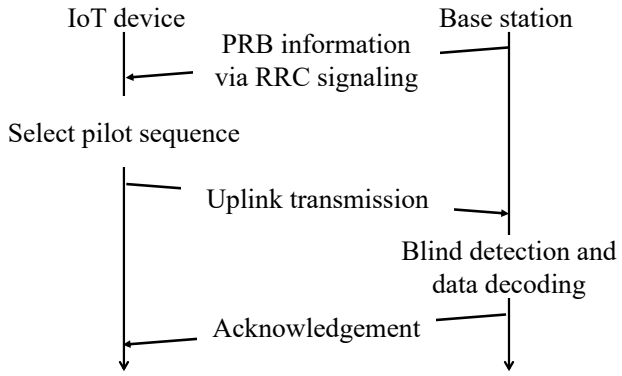


Figure 1.1: The timing sequence diagram for GFMA. The base station configures the PRB via radio resource control (RRC) signaling. Each IoT device then selects a pilot sequence and transmits its packet to the base station. After decoding, the base station informs the device about the decoding state by sending an ACK.

is illustrated in Fig. 1.1. Compared with the four-step grant-based random access in Long Term Evolution (LTE) systems, GFMA has a two-step access procedure. Hence, GFMA incurs a lower signaling overhead and reduces the access delay of IoT devices during the random access procedure. Moreover, by sharing the same physical resource block (PRB), multiple IoT devices can transmit their packets simultaneously to the base station in GFMA systems [33].

To fully exploit the benefits of GFMA, two challenges have to be overcome. First, due to the lack of centralized scheduling, packet collisions occur when multiple IoT devices select the same pilot sequence, which can lead to decoding failure and throughput degradation. Therefore, each device should choose a specific pilot sequence that distinguishes its signal from the signals of other devices to ensure successful channel estimation and decoding at the receiver [33]. Second, IoT devices cannot coordinate their transmissions or exchange information with each other. Each device selects a pilot sequence independently without knowing the selection decisions of the other devices. Moreover, due to the lack of knowledge of the throughput requirements of the other devices, an IoT device may greedily occupy too many network resources such that the throughput requirements of the other devices cannot be satisfied.

Various schemes have been proposed to resolve collisions in the pilot sequence selection

in GFMA systems [34–36]. Han *et al.* in [34] proposed an ACK-based scheduling scheme, where a device that experienced a packet collision selects a new pilot sequence from the remaining pilot sequences that have not been selected by other devices, and retransmits the packet. Shen *et al.* in [35] proposed that the base station reserves some of the pilot sequences for the retransmissions of devices that have suffered a packet collision. The base station then informs the devices about the reserved pilot sequences by broadcasting an ACK. For ACK-based solutions [34, 35], although collisions are resolved in the retransmission phase, collisions can still occur when a device transmits a packet for the first time since the scheduling is performed only after a collision has occurred. Sun *et al.* in [36] proposed a pilot sequence allocation scheme, where the base station pre-assigns pilot sequences to those devices that have higher probabilities of transmitting packets in the next time slot. However, the allocation scheme in [36] requires centralized scheduling and accurate estimation of the transmit probability. In addition, the aforementioned schemes do not take into account any throughput requirements, and therefore may not be able to simultaneously support multiple IoT applications with different data rate requirements.

DRL has been applied to design distributed spectrum access schemes in cognitive radio systems [37–39]. Due to the similarity between the distributed spectrum access in cognitive radio and the pilot sequence selection in GFMA systems, the aforementioned works can offer some insights into the design of DRL-based pilot sequence selection schemes. Wang *et al.* in [37] used DRL to design a distributed multi-channel access scheme to reduce the collision probability and maximize the channel utilization. DRL was employed to study cooperative and non-cooperative channel access of multiple users in [38]. Naparstek *et al.* in [38] showed that, by properly designing a cooperative reward function, DRL-based channel access schemes may yield a better performance in terms of proportional fairness compared with using individual reward functions. Yu *et al.* in [39] proposed a DRL-based channel access scheme for heterogeneous wireless networks. The results in [39] showed that, by using an MA-DRL framework along with a cooperative reward function, the aggregate throughput can be improved and proportional fairness can be achieved. While

the aggregate throughput maximization and proportional fairness, which are two special cases of α-fairness, have been investigated in [38, 39], the DRL frameworks proposed in the aforementioned studies cannot be applied to wireless systems where the users may have different throughput requirements. The latter case can be regarded as a generalized case of proportional fairness, where the users are prioritized based on their specific throughput requirements. Compared with α-fairness, in this case, it is more difficult for the learning agent in DRL to learn the pilot sequence selection policies that satisfy the user-specific throughput requirements.

To address the aforementioned issues, in this thesis, we propose an MA-DRL based distributed pilot sequence selection scheme for aggregate throughput maximization in GFMA systems, where we take the throughput requirements of different IoT devices into account. Each IoT device does not know the pilot sequence selection decisions or the throughput requirements of the other devices. Due to the lack of global information in each device, it is challenging to design a distributed scheme that fosters collaboration between IoT devices such that the throughput requirements of different devices can be satisfied. We exploit recurrent neural networks (RNNs) to handle the incomplete information of the underlying decision process and investigate their capability of learning the pilot sequence selection policies such that the aggregate throughput is maximized while the throughput requirements of different users are satisfied.

1.2 IRS-aided Systems

The presence of the line-of-sight (LoS) channels between the base station and users is critical to ensure high data transmission rates in wireless systems. The users may experience poor channel conditions and low data transmission rates when the LoS channels are blocked by physical obstacles, e.g., trees, buildings, and vehicles. The spectral efficiency degradation due to the blockages of LoS channels can be more significant when high-frequency carrier signals are used in B5G wireless systems.

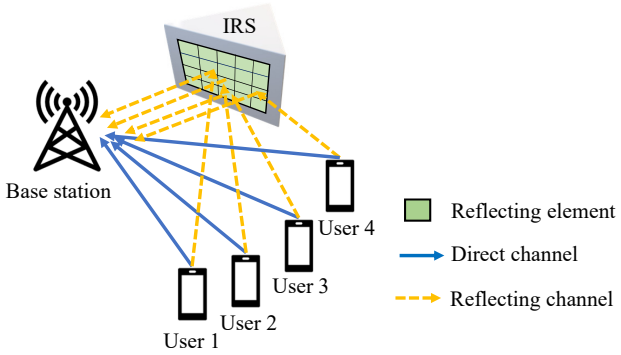


Figure 1.2: An IRS-aided system with four users sharing one uplink PRB. The direct channels are denoted by solid blue lines, while the reflecting channels are denoted by dashed yellow lines.

IRS [24, 25] can effectively tackle the aforementioned issue in B5G wireless systems. IRS is a reconfigurable planar surface with multiple passive reflecting elements. Each reflecting element can perform a phase shift to the incident signal independently and reflect the shifted signal to a receiver. An IRS-aided system serving four users to perform uplink transmissions is shown in Fig. 1.2. Apart from the direct channels between the base station and users, IRS introduces additional propagation channels in B5G wireless systems. When the LoS channels between the base station and users are blocked by obstacles, deploying an IRS can create virtual LoS channels to facilitate data transmission and improve the coverage of the base station [40]. By properly controlling the phase shifts of the reflecting elements on IRS, the base station can mitigate interference and allow multiple users to share a PRB for uplink transmission. IRS can be categorized as a passive holographic multiple-input multiple-output surface (HMIMOS) since the reflecting elements can be powered by an energy harvesting module [41]. IRS can be combined with other physical layer techniques, including full-duplex communications and energy harvesting communications [23]. Potential applications of IRS include unmanned aerial vehicle (UAV) networks and VR [42, 43].

Existing research on resource allocation in IRS-aided systems mostly focus on the beamforming optimization at the base station and the phase shift control of IRS [12, 13, 44–51]. The beamforming and phase shift optimization for IRS-aided systems with a single user

is studied in [12, 44]. Huang *et al.* in [13] studied the joint phase shift and power control problem for maximizing the energy efficiency of the IRS-aided systems. Jia *et al.* in [45] determined the ergodic rate of an IRS-aided system with interference from a secondary user and proposed a parallel coordinate descent-based algorithm to optimize the phase shifts of the IRS. Abeywickrama *et al.* in [46] studied multiuser beamforming with practical amplitude variation in an IRS-aided system. An AO-based algorithm is proposed to solve the joint optimization problem. Xu *et al.* in [47] investigated the aggregate throughput maximization problem in IRS-aided full-duplex systems. They solved the joint phase shift control, power control, and beamforming optimization problem using AO with successive convex approximation (SCA). Moreover, Ma *et al.* in [48] investigated the joint beamforming and phase shift control in an IRS-aided multiuser system under both perfect and imperfect channel information. The beam pattern and channel estimation in a terahertz massive multiple-input multiple-output (MIMO) system with holographic IRS were investigated in [49]. The joint beamforming and phase shift control for maximizing the physical layer security in IRS-aided systems has been studied in [50]. Huang *et al.* in [51] studied the application of IRS in cell-free MIMO to support reliable data transmission. In addition, FP [10] was applied in [50] to develop low-complexity beamforming and phase shift control algorithms. Although the aforementioned works studied the optimization of beamforming and phase shift control, the uplink user scheduling problem in IRS-aided systems has not been investigated. For an IRS-aided system with multiple users, it is beneficial for the base station to properly schedule the transmission of the users, such that the interference between the users can be mitigated. Moreover, in the existing AO-based approaches, the iterative optimization process has to be invoked whenever the base station observes a change in the channel state information (CSI). This may lead to high computational complexity.

To address the aforementioned issues, in this thesis, we investigate the joint optimization of user scheduling, phase shift control, and beamforming vectors in IRS-aided systems. We consider both maximizing the aggregate throughput and achieving proportional fairness as objectives. Obtaining the optimal solution of the joint optimization problem is

challenging since the optimization variables are coupled and the formulated problem is nonconvex. To tackle the challenges, we propose a DRL-based user scheduling, phase shift control, and beamforming optimization (DUPB) algorithm, in which we first use neural combinatorial optimization (NCO) [52, 53] to determine the user scheduling. The DNNs with attention mechanism [54] are trained to learn a stochastic user scheduling policy using the REINFORCE algorithm [55]. We then solve the phase shift control and beamforming subproblem by proposing a curriculum learning deep deterministic policy gradient (CL-DDPG) algorithm, in which we exploit the DDPG [17, 56], CL [29, 30], and the hidden convexity of the joint problem to optimize the phase shift and beamforming variables.

1.3 IRS-aided RS VR Streaming Systems

1.3.1 Exploiting the Shared Interests in Multiuser VR Streaming Systems with RS

VR streaming provides the users with an immersive experience by rendering 360-degree videos using HMDs. VR is considered as one of the important use cases of B5G wireless systems [3, 4]. Via wireless connectivity, VR users can move and interact freely without being restricted by the cable that connects the HMDs and VR server. Driven by the development of VR technology, there are emerging applications of VR streaming in different industries, including entertainment, retail, and education. It is estimated that the global VR market size will increase from $4.42 billion US dollars (USD) in 2020 to $84.09 billion USD by 2028, with a compound annual growth rate of 44.8% [57].

In multiuser VR streaming, the same 360-degree video segment may be requested by multiple users due to their shared interests. As an example, for the streaming of a 360-degree soccer match video, the supporters of a particular soccer team may frequently share those field-of-views (FoVs) that include the players of their team. Using RS, the shared interests of users can be exploited to achieve a significant multiplexing gain for data trans-

mission, and improve the spectral efficiency of VR streaming systems. The unique feature of RS is that the base station constructs a common message that needs to be decoded by all users [26–28]. The common message in RS can be used to transmit the data of the video segments that are requested by multiple users. By doing this, the data transmitted using the common message can be received by multiple users simultaneously, thereby achieving multiplexing gains in multiuser VR streaming systems. In particular, using RS, the information intended for the users is split into two parts, namely a common message and private messages [26–28]. Each user needs to decode the common message first by treating the private messages as interference. The user then subtracts the common message from the received signal using successive interference cancellation (SIC) and subsequently decodes its private message [27]. These features of RS make it a promising physical layer technique for multiuser VR streaming systems since (a) the data related to the shared interests of the VR users can be encoded into the common message to reap the multiplexing gain, and (b) the unique data requested by each VR user can be encoded in its private message. In this thesis, we show that the quality of experience (QoE) in a multiuser VR streaming system can be significantly improved with a properly designed RS scheme that facilitates the exploitation of the shared interests of the users.

Some existing works studied RS systems in which the data for different users are independent and uncorrelated [28, 58–60]. However, in VR streaming, different users may request the data of the same 360-degree video segment due to their shared interests. In this case, it becomes important to take the shared interests of the users into account when designing the RS scheme. However, the shared interests of the users have not been exploited in [28, 58–60]. The RS VR streaming system we consider in this thesis is related to the RS multicast systems [61, 62]. Joudeh *et al.* in [61] studied RS multigroup multicast systems, in which the same message is requested by the users of the same group. Mao *et al.* in [62] considered an RS non-orthogonal unicast and multicast (RS-NOUM) system, where a multicast message needs to be received by all the users in the system and each user's private message is being sent via unicast. Although the RS schemes considered in [61, 62]

exploited the multiplexing gain of RS-based multicasting, they assumed that a part of the information is requested by every user in the same group (as in [61]) or in the system (as in [62]). However, this assumption may not always hold in RS VR streaming systems when the FoVs of some users do not overlap. Moreover, while it is possible to divide the users into groups based on their FoVs and apply the multigroup RS algorithm in [61] to support multiuser VR streaming, finding the optimal user grouping (i.e., determining the number of groups and the number of users within each group) is non-trivial and computationally intensive. To tackle these issues, the RS parameters in RS VR streaming systems need to be optimized based on the FoVs and the shared interests of the VR users.

1.3.2 Combining IRS with RS in Multiuser VR Streaming Systems

Besides RS, the considered multiuser VR streaming system also exploits IRSs. IRSs introduce additional propagation channels and DoF that can be exploited to mitigate interference [63]. Moreover, in this thesis, we show that IRSs can improve the performance of RS systems by increasing the minimum signal-to-interference-plus-noise ratio (SINR) experienced by the common message at different users. Existing research has confirmed the benefits of employing IRSs in conventional multiuser wireless communication systems without RS [13, 43, 48, 64, 65]. Chaccour *et al.* in [43] showed that IRSs can improve both the sum-rate and reliability of data transmission in VR applications. Physics-based modeling of IRS and codebook design for scalable IRS phase shift optimization were studied in [64]. Besser *et al.* in [65] proposed a phase hopping algorithm for IRS-aided systems to improve the reliability of data transmission without requiring CSI. However, the aforementioned works have not investigated the benefits of combining IRS with RS. Bansal *et al.* in [60] proposed an IRS-aided rate-splitting multiple access (RSMA) system and designed an on-off control scheme to adjust the phase shifts of the IRSs. Fu *et al.* in [66] proposed an AO algorithm to maximize the minimum achievable rate of an IRS-aided multiuser multiple-input single-output (MU-MISO) RSMA system. However, the algorithm for opti-

mizing the RS parameters based on the shared interests of the users has not been studied in [60, 66]. Moreover, in multiuser VR streaming systems, the joint optimization of the IRS phase shifts, RS parameters, beamforming vectors, and bitrate selection of the 360-degree videos based on the FoVs and CSI of the VR users is crucial for achieving satisfying performance.

In this thesis, we propose an IRS-aided RS VR streaming system, where RS is applied to exploit the shared VR streaming interests of the users, and IRSs are used to improve the minimum SINR experienced by the common message across the users and the system sum-rate. We aim to maximize the achievable bitrate of the 360-degree video by optimizing the IRS phase shifts, RS parameters, beamforming vectors, and individual bitrates. Solving such a problem using conventional optimization methods (e.g., AO) can be computationally expensive and time-consuming. To tackle this issue, we propose a deep deterministic policy gradient with imitation learning (Deep-GRAIL) algorithm, which can efficiently solve the formulated constrained optimization problem with low computational complexity. In the proposed Deep-GRAIL algorithm, we use imitation learning [31, 32], which allows the learning agent to learn not only from its own exploration, but also from the solutions obtained with conventional optimization methods.

1.4 DRL for Resource Allocation Algorithms Design in Wireless Systems

1.4.1 Basics of Reinforcement Learning (RL) and DRL

RL is a technique for learning a *policy* that maximizes the expected discounted reward based on the interaction between the learning agent and the environment [67, Section 2.1]. In a Markov decision process (MDP) which is defined by the state, action, state transition probability function, and reward, a policy is a mapping from states to actions. In each decision epoch of the MDP, the learning agent first observes the current state of the MDP,

and chooses an action based on the learned policy. After the action has been executed under the current state, a reward is received by the learning agent, which measures how good is the action taken by the learning agent. The action taken by the learning agent may also incur a state transition, causing the state of the MDP to change from the current state to a new state. The target of RL is to learn the optimal policy that achieves the maximum expected discounted reward based on the historical actions taken by the learning agent, and the corresponding state transition and received rewards.

For the policy learning in RL, the learning agent needs to approximate the *state-action value function*, which is a function of a state-action pair that estimates the expected discounted reward that can be obtained by taking a given action under a given state [67, Chapter 3]. While the approximation of the state-action value function provides the foundation for policy learning in RL, it becomes difficult to obtain an accurate approximation of the state-action value function when the state and action spaces are large [67, Chapter 9]. In DRL, this challenge is tackled by using DNNs as *universal function approximators* [68] to approximate the state-action value function. Using DRL, the state-action value function in high-dimensional state and action spaces can be approximated by exploiting the representational capability of the DNNs [19]. Moreover, different DRL algorithms can be utilized to tackle either the continuous or discrete optimization variables, making DRL applicable to various types of resource allocation problems in wireless systems. In particular, the existing DRL algorithms for the optimization of discrete variables include deep Q-learning [15], double deep Q-learning [69], and NCO [52, 53]. For optimization of continuous variables, DRL algorithms such as DDPG [17], twin delayed DDPG [18], and soft actor-critic [70] can be applied. In addition, proximal policy optimization [71] and trust region policy optimization [72] can be applied to optimize both discrete and continuous variables. Many successful applications of DRL algorithms have demonstrated their capabilities of achieving state-of-the-art performance in decision-making problems [15, 16, 73] and classic optimization problems [52, 53].

1.4.2 DRL for Algorithms Design in Wireless Systems

Due to the appealing features of DRL, various resource allocation algorithms for wireless systems have been proposed in recent years. Apart from the previously mentioned works on GFMA, IRS-aided, and RS systems, DRL has been applied for the algorithms design in other wireless systems, including UAV systems [74–77], vehicular networks [78–80], small cell systems [81, 82], edge computing and caching [83–86], cell-free systems [87, 88], as well as energy harvesting wireless systems [89, 90].

Despite the policy learning capability of DRL, there are still several challenges that need to be overcome for the successful applications of DRL in B5G wireless systems.

Nonstationarity and Lack of Global Information

For the DRL-based algorithms design for IoT devices in B5G wireless systems, due to the limited communication range and hardware capability, each IoT device can only obtain its local information without having access to the global information of the system. This incurs the following challenges for the policy learning of DRL in IoT systems. First, due to the lack of global information, each IoT device can only obtain a part of the global state information of the underlying MDP. Policy learning under partial information is known to be more difficult than the one with global information [91, 92]. Moreover, the policies learned by the IoT devices may change throughout the learning process, making the underlying MDP become nonstationary. Without being able to obtain knowledge of the policies of other IoT devices, the policy learned by each IoT device may suffer from divergence [93]. In Chapter 2 of this thesis, we tackle the nonstationarity and the lack of global information in IoT devices during the policy learning phase by using a centralized training distributed execution (CTDE) framework [94, 95]. The proposed CTDE framework not only takes advantage of the global information available at the base station to facilitate policy learning, but also allows IoT devices to skip the computationally intensive training process. In addition, we use the factorization technique [94] to ensure that the policies learned during the centralized training phase can be executed by IoT devices in a distributed manner

without requiring global information.

Curse of Dimensionality

The curse of dimensionality was originally introduced by Bellman in [96], which refers to the phenomenon that the computational requirements of solving dynamic programming problems grow exponentially with the number of optimization variables. In the context of RL, Sutton *et al.* in [67] have used the curse of dimensionality to refer to those issues that are caused by the increase in the dimensionality of the state and action spaces. The performance of DRL algorithms also suffers from the curse of dimensionality due to the following reasons. First, since the improvement of the learned policy is based on the exploration of the learning agent in DRL, the learning agent may need to visit a large number of states and take a large number of actions in order to explore the state and action spaces of the underlying MDP, and learn a policy with reasonable performance [97]. When the state and action spaces are large, the learning agent may not be able to efficiently explore those good states and actions that can lead to relatively high rewards without having a good policy beforehand [31]. Hence, the performance of the learned policy may be affected by the inefficient exploration of the learning agent. Second, policy learning becomes more difficult when the number of decision variables increases. A decision-making problem with a large number of coupled decision variables can be highly nonconvex and have a complex landscape for optimization. From the perspective of policy learning, such an issue may cause the policy learned based on gradient descent algorithms (e.g., [98, 99]) to be local optimal (or suboptimal), and make the optimal policy more difficult to obtain by the learning agent.

Unfortunately, the curse of dimensionality may exist in B5G wireless systems with high DoF. For the IRS-aided systems, the curse of dimensionality can be caused by a large number of reflecting elements that are required in order to reap the performance gain of the IRSs [100]. For the DRL-based phase shift control algorithms design, having more reflecting elements on the IRSs makes the state and action spaces of the underlying MDP

larger. This is because with more reflecting elements, the state of the MDP needs to include more subchannels, while the action of the MDP has to include more phase shift variables. Besides, the number of antennas and users can also increase the DoF of the IRS-aided systems. In Chapters 3 and 4 of this thesis, we propose to tackle the curse of dimensionality by exploiting the knowledge of the hidden convexity of the resource allocation problems in B5G wireless systems. We use CL [29, 30] and imitation learning [31, 32] to facilitate the exploration and policy learning of the learning agent in DRL. In the proposed DRL-based algorithms, we enhance the vanilla random exploration scheme, which has been applied in various existing DRL algorithms (such as deep Q-learning [15, 69] and DDPG [17, 18]), by guiding the exploration of the learning agent based on the hidden convexity of resource allocation problems. By doing this, those states (and actions) that lead to relatively high rewards can be visited (and discovered) by the learning agent more often than when the vanilla random exploration scheme is used. The proposed DRL-based algorithms improve the exploration efficiency and allow the learning agent to discover better policies during the early stage of learning.

1.5 Summary of Results and Contributions

This thesis proposes DRL-based algorithms for resource allocation in B5G wireless systems. In particular, the DRL-based resource allocation algorithms for GFMA systems, IRS-aided multiuser systems, and IRS-aided RS VR streaming systems are proposed in this thesis. The results are divided into three main chapters. The results and contributions in each chapter are as follows.

1. In Chapter 2, we study the pilot sequence selection problem for throughput optimization in GFMA systems with average throughput constraints for IoT devices. We propose an MA-DRL based pilot sequence selection scheme, where the DNNs are trained to learn the pilot sequence selection policies from the transition history of the underlying MDP for the IoT devices. We propose a deep recurrent Q-network

(DRQN) to learn the temporal correlations of system transitions in time series learning problems. We propose a CTDE framework, in which the DRQNs of all IoT devices are jointly trained by leveraging global information at the base station, whereas the policies learned during the centralized training phase can be executed in a distributed manner. We show that DNNs are capable of learning near-optimal pilot sequence selection policies for IoT devices. For the considered system, the proposed scheme can achieve an aggregate throughput that is within 85% of the optimum. The aggregate throughput of the proposed scheme is 31%, 128%, and 162% higher than that of an ACK-based scheme [34], dynamic access class barring (ACB) [101], and a random selection scheme, respectively. The pilot sequence selection policies learned via MA-DRL can also accommodate different throughput requirements of the IoT devices. Part of the work of Chapter 2 has been presented in [102] and the full paper has been published in *IEEE Transactions Wireless Communications* [103].

2. In Chapter 3, we investigate the performance gain that can be obtained by using an IRS in multiuser B5G wireless systems in terms of aggregate throughput and proportional fairness. The DoF in the considered IRS-aided systems include user scheduling, IRS phase shifts, and beamforming vectors. We propose the DUPB algorithm to efficiently optimize the aforementioned DoF. The first part of the proposed DUPB algorithm comprises an NCO-based user scheduling algorithm, in which we exploit the DNN with attention mechanism to learn a stochastic policy for determining which users should be scheduled. In the second part of the DUPB algorithm, we combine CL [29, 30] with the DDPG algorithm [17, 56] to jointly optimize phase shift control and beamforming variables in IRS-aided systems. Using CL, we tackle the challenge of learning to optimize a large number of coupled decision variables by providing the learning agent with the knowledge of the hidden convexity of the optimization problem. Results show that the proposed DUPB algorithm can outperform the AO-based algorithms and greedy scheduling in terms of aggregate throughput. Our results also indicate that a higher performance gain can be achieved with more

reflecting elements on the IRS. Moreover, our results validate the advantage of the proposed DUPB algorithm in terms of computational complexity over AO algorithms. We have presented part of the work of Chapter 3 in [104]. The full paper has been published in *IEEE Transactions Wireless Communications* [105].

3. In Chapter 4, we propose a novel IRS-aided RS VR streaming system, in which the shared interests of the VR users are exploited by RS and IRS to achieve better QoE. In the proposed system, RS facilitates the exploitation of the shared interests of the users in VR streaming, and IRS creates additional propagation channels to support the transmission of high-resolution 360-degree videos. IRS also enhances the capability to mitigate the performance bottleneck caused by the requirement that all RS users have to be able to decode the common message. We formulate an optimization problem for the maximization of the achievable bitrate of the 360-degree video subject to the QoE constraints of the users. We propose a Deep-GRAIL algorithm, in which we leverage DRL and imitation learning to optimize the IRS phase shifts, RS parameters, beamforming vectors, and bitrate selection of 360-degree videos. We also propose RavNet, which is a DNN customized for policy learning in the proposed Deep-GRAIL algorithm. Performance evaluation based on a real-world VR streaming dataset shows that the proposed IRS-aided RS VR streaming system outperforms several baseline schemes in terms of system sum-rate, achievable bitrate of the 360-degree videos, and online execution runtime. Our results also reveal the respective performance gains obtained from RS and IRS for improving the QoE in multiuser VR streaming systems. The work of Chapter 4 has been accepted for publication in *IEEE Journal on Selected Areas in Communications* [106].

1.6 Thesis Organization

The rest of the thesis is organized as follows. In Chapter 2, the pilot sequence selection problem in GFMA systems with average throughput constraints of the IoT devices is for-

mulated. An MA-DRL based algorithm using a CTDE framework is proposed to learn the distributed pilot sequence selection policies. In Chapter 3, we solve the joint user scheduling, phase shift control, and beamforming optimization problem in IRS-aided systems using the proposed DUPB algorithms. In Chapter 4, we propose an IRS-aided RS VR streaming system, and formulate the problem for maximizing the achievable bitrate of the 360-degree video. We develop a Deep-GRAIL algorithm to optimize the DoF of the IRS-aided RS VR streaming systems with low computational complexity. Conclusions as well as the discussions on limitations and potential future work are given in Chapter 5. Chapters 2−4 are self-contained and included in separate journal or conference papers. The notations are defined separately for Chapters 2−4.

Chapter 2

Throughput Optimization for Grant-Free Multiple Access with Multiagent Deep Reinforcement Learning

2.1 Introduction

GFMA can reduce the signaling overhead as well as the access delay of the users in B5G wireless systems. The design of the pilot sequence selection scheme is critical to fully reap the benefits of GFMA [22, 23]. This is because packet collisions occur in GFMA systems when multiple users choose the same pilot sequence, making the base station unable to decode the signal of each user from the received superimposed signal [33]. However, due to the lack of coordination between the users, designing a distributed pilot sequence selection scheme for mitigating packet collisions in GFMA systems is challenging. In this chapter, we exploit the potential of MA-DRL techniques to learn the distributed pilot sequence selection policies for maximizing the aggregate throughput in GFMA systems. We incorporate the user-specific quality of service (QoS) requirements in our problem formulation, and design an MA-DRL based algorithm to tackle the QoS requirements without requiring excessive information exchange between the users. To tackle the nonstationarity in MA-DRL, we propose a CTDE framework, in which the DNNs of all users are jointly trained to learn the pilot sequence selection policies with the help of global information available at the base

station. Using the technique of *factorization* [95, 107], the policies learned during joint training can be executed in a distributed manner. Our contributions are as follows:

- We formulate the pilot sequence selection problem for throughput optimization in GFMA systems with average throughput constraints for the IoT devices. We apply stochastic network optimization [108] and propose an algorithm to obtain the optimal solution. While relying on centralized scheduling, the optimal solution serves as a benchmark when evaluating the performance of the proposed DRL-based scheme.

- We model the pilot sequence selection process as an MDP and propose a pilot sequence selection scheme, where the DNNs are trained based on DRL to learn the pilot sequence selection policies from the transition history of the underlying MDP for the IoT devices. We propose a DRQN to take advantage of the capability of RNNs to efficiently learn the temporal correlations of system transitions in time series learning problems.

- Using the factorization technique, we propose a CTDE framework. In the proposed training framework, the DRQNs of all IoT devices are jointly trained by leveraging global information at the base station, whereas the policies learned during the centralized training phase can be executed in a distributed online manner.

- We conduct simulations to evaluate the performance of the proposed scheme. Our results show that DNNs are capable of learning near-optimal pilot sequence selection policies for IoT devices. For the considered system, the proposed scheme can achieve an aggregate throughput that is within 85% of the optimum. The aggregate throughput of the proposed scheme is 31%, 128%, and 162% higher than that of an ACK-based scheme [34], dynamic ACB scheme [101], and a random selection scheme, respectively. The pilot sequence selection policies learned via DRL can also accommodate the throughput requirements of different IoT devices. Hence, the proposed scheme is capable of supporting different IoT applications in GFMA systems.

The remainder of this chapter is organized as follows. The system model and problem formulation are introduced in Section 2.2. The DRL framework for pilot sequence selection is presented in Section 2.3. The DNN architectures and training algorithm are presented in Section 2.4. Simulation results are provided in Section 2.5. Section 2.6 summarizes this chapter.

2.2 System Model and Problem Formulation

We consider a GFMA system with one base station serving multiple users[1]. The base station and each user are equipped with one antenna. Time is slotted into intervals of equal duration. The time interval $[t, t+1)$ is referred to as time slot t, where $t \in \mathcal{T} = \{0, 1, 2, \ldots, T-1\}$. In each time slot, the base station assigns one PRB, i.e., one time-frequency resource block, for GFMA transmission. We assume the base station assigns K pilot sequences to the PRB in each time slot, and $\mathcal{K} = \{1, 2, \ldots, K\}$ is the set of pilot sequence indices. Welch bound equality sequences, Grassmannian sequences, or other types of sparse spreading sequences can be used as pilot sequences.

There are in total N users in the considered GFMA system. The set of users is denoted by $\mathcal{N} = \{1, 2, \ldots, N\}$. We have the following assumptions for the considered GFMA system. First, we assume that all data packets are transmitted using GFMA. Second, we assume the saturated case where each user always has packets to send. At the beginning of each time slot, the base station informs the users about the PRB and the K available pilot sequences via radio resource control (RRC) signaling. When a user decides to transmit, it selects one of the K available pilot sequences and performs uplink transmission. We define binary variable $g_{nk}(t) \in \{0, 1\}$, where $g_{nk}(t)$ is equal to 1 if user $n \in \mathcal{N}$ selects the k-th pilot sequence, $k \in \mathcal{K}$, in time slot $t \in \mathcal{T}$. Otherwise, $g_{nk}(t)$ is equal to 0. Since a user can

[1]In the remainder of this chapter, we use the terms users and IoT devices interchangeably.

select at most one pilot sequence in each time slot, we have

$$\sum_{k\in\mathcal{K}} g_{nk}(t) \leq 1, \quad n \in \mathcal{N},\, t \in \mathcal{T}. \tag{2.1}$$

User n does not transmit in time slot t if $\sum_{k\in\mathcal{K}} g_{nk}(t) = 0$. We further define $\boldsymbol{g}_n(t) \triangleq (g_{n1}(t), g_{n2}(t), \ldots, g_{nK}(t))$ as the pilot sequence selection vector of user n in time slot t. We define $n_k(t)$ as the number of users that select the k-th pilot sequence in time slot t. We have

$$n_k(t) \triangleq \sum_{n\in\mathcal{N}} g_{nk}(t), \quad k \in \mathcal{K},\, t \in \mathcal{T}. \tag{2.2}$$

Furthermore, $\mathcal{S}(t)$ denotes the set of users who select a pilot sequence that is not chosen by other users in time slot t. That is,

$$\mathcal{S}(t) \triangleq \Big\{ n \,\Big|\, \sum_{k\in\mathcal{K}} \mathbb{1}(n_k(t) = 1)\, g_{nk}(t) \;=\; 1,\; n \in \mathcal{N} \Big\}, \quad t \in \mathcal{T}, \tag{2.3}$$

where $\mathbb{1}(\cdot)$ is the indicator function.

At the receiver side, the base station estimates the channels based on the received pilot sequences and then decodes the packets of the users. We have the following assumptions for the receiver at the base station. For the users who select a pilot sequence that is not chosen by other users, we assume the base station can perform perfect channel estimation and apply multiuser detection to mitigate the interference between the users [33, 109] and decode their packets successfully. However, when multiple users select the same pilot sequence, the base station cannot estimate the channels of the users, and hence cannot decode their packets [110]. That is, for the receiver at the base station, we assume it can only successfully decode the data of the users in set $\mathcal{S}(t)$. For user $n \in \mathcal{N}$, we define $r_n(t)$ to be a binary indicator that specifies whether its signal is successfully decoded in time

slot t. We have

$$r_n(t) \triangleq \begin{cases} 1, & \text{if } n \in \mathcal{S}(t), \\ 0, & \text{otherwise.} \end{cases} \tag{2.4}$$

The base station sends an ACK to the user if its packet has been successfully decoded.

We use the time average expectation of the number of packets that are successfully received by the base station as the performance metric to measure the average aggregate throughput of the considered GFMA system. This metric has also been adopted in [39]. Since there are K pilot sequences, the maximum aggregate throughput of the considered system is K packets per time slot, which is the achievable aggregate throughput of the considered system in the absence of pilot sequence selection collisions. We denote $\mu_n(t)$ as the average throughput of user n up to time slot t. We have

$$\mu_n(t) \triangleq \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}[r_n(\tau)], \quad n \in \mathcal{N},\, t \in \mathcal{T} \setminus \{0\}, \tag{2.5}$$

and $\mu_n(0) \triangleq 0$, where $\mathbb{E}[\cdot]$ is the expectation with respect to the randomness of the pilot sequence selections of the other users. Given (2.5), we have $\mu_n(t) \in [0, 1]$ for $n \in \mathcal{N}$, $t \in \mathcal{T}$. Moreover, we assume that each user knows its own average throughput but not that of the other users. This has two main reasons. First, we consider the average throughput of a particular user to be private information. Hence, from the perspective of the wireless network operator, the base station should not broadcast the average throughput of the users in order to avoid the potential leakage of private information. Second, broadcasting the average throughput of the users incurs an additional signaling overhead, which scales with the number of users K.

We denote user n's average required throughput by μ_n^{req}. Hence, the constraint with respect to the average throughput requirement of user n is given by

$$\limsup_{T\to\infty} \mu_n(T) = \limsup_{T\to\infty} \frac{1}{T} \sum_{\tau=0}^{T-1} \mathbb{E}[r_n(\tau)] \geq \mu_n^{\text{req}}, \quad n \in \mathcal{N}. \tag{2.6}$$

We assume that the average required throughputs of the IoT applications can be categorized into different QoS levels, and each of the QoS levels is pre-assigned with a QoS flow identifier (QFI), which is a positive integer [111, Section 5.7]. When sending a packet to the base station, a user indicates its QFI in the packet header, and the base station is informed about the average required throughput of the user by checking the packet header [111, Section 5.8].

Our objective is to maximize the average aggregate throughput in GFMA systems subject to user-specific throughput constraints. This leads to the following optimization problem

$$\begin{aligned} \underset{\boldsymbol{g}_n(t),\, n\in\mathcal{N},\, t\in\mathcal{T}}{\text{maximize}} \quad & \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{n\in\mathcal{N}} \mathbb{E}[r_n(t)] \\ \text{subject to} \quad & \sum_{k\in\mathcal{K}} g_{nk}(t) \leq 1,\ n \in \mathcal{N},\ t \in \mathcal{T}, \\ & \limsup_{T\to\infty} \frac{1}{T} \sum_{\tau=0}^{T-1} \mathbb{E}[r_n(\tau)] \geq \mu_n^{\text{req}},\ n \in \mathcal{N}. \end{aligned} \tag{2.7}$$

Problem (2.7) is a combinatorial optimization problem that can be solved using stochastic network optimization with virtual queues [108]. The details of this approach can be found in Appendix A. However, the optimal solution of problem (2.7) requires global information and centralized computation. Besides, centralized scheduling is necessary for implementing the optimal solution of problem (2.7) in GFMA systems. Hence, in this chapter, we propose an MA-DRL based pilot sequence selection scheme to obtain a suboptimal solution of problem (2.7) in a distributed manner.

2.3 MA-DRL Framework for GFMA Systems

With the recent advances in machine learning and artificial intelligence, DRL has emerged as a powerful tool for developing solutions for combinatorial optimization problems. In this section, we model the pilot sequence selection problem as an MDP and propose an MA-DRL based framework to solve problem (2.7) efficiently.

2.3.1 Modeling with MDP

As the pilot sequence selection of the N users in the considered GFMA system is essentially a multiagent decision-making process, we model it as an MDP, which can be defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$. The details are as follows:

State $\mathcal{S}$

We define variable $d_k(t) \in \{-1, 0, 1\}$, $k \in \mathcal{K}$, as the indicator for the decoding state of the k-th pilot sequence at the beginning of the current time slot $t \in \mathcal{T}$. $d_k(t)$ depends on the pilot sequence selection of the users in the previous time slot $t-1$. Specifically, $d_k(t)$ is equal to 1 if the signal using the k-th pilot sequence was decoded successfully in time slot $t-1$. We set $d_k(t)$ to -1 if the signal using the k-th pilot sequence was not decoded successfully in time slot $t-1$. $d_k(t)$ is equal to 0 if no signal using the k-th pilot was transmitted in time slot $t-1$. In time slot t, the *global* state $\boldsymbol{s}(t)$ consists of the decoding states of all pilot sequences and the average throughput of all users up to time slot t. We have

$$\boldsymbol{s}(t) \triangleq (d_1(t), \ldots, d_K(t), \mu_1(t), \ldots, \mu_N(t)), \quad t \in \mathcal{T}, \tag{2.8}$$

where $\mu_n(t)$ is given in (2.5). After decoding the packets of all users, the base station knows the decoding states of all pilot sequences $d_1(t), \ldots, d_K(t)$, and it can determine the average throughput of all users $\mu_1(t), \ldots, \mu_N(t)$. Therefore, global state $\boldsymbol{s}(t)$ is available at the base station in each time slot. We define $\mathcal{S} \subseteq \{-1, 0, 1\}^K \times [0, 1]^N$ to be the set of all the possible global states.

We assume the base station includes the information on $d_1(t), \ldots, d_K(t)$ in the RRC signaling and broadcasts this information to the users at the beginning of time slot t. Each $d_k(t)$, $k \in \mathcal{K}$ can be encoded using 2 bits. Hence, for a GFMA system with K pilot sequences, this incurs an overhead of $2K$ bits for downlink signaling. Apart from knowing the decoding states of all K available pilot sequences, user n only knows its

average throughput $\mu_n(t)$. Therefore, we define the *local* state of user n in time slot t as

$$\boldsymbol{s}_n(t) \triangleq (d_1(t), \ldots, d_K(t), \mu_n(t)), \quad t \in \mathcal{T}. \tag{2.9}$$

We define $\widehat{\mathcal{S}} \subseteq \{-1, 0, 1\}^K \times [0, 1]$ as the set of all possible local states of user $n \in \mathcal{N}$. We note that as the state includes the average throughput of the users, the state in one particular time slot depends on the state of the previous time slot and the pilot sequence selections of the users.

Joint Action $\mathcal{A}$

In time slot t, the action profile of user n is its own pilot sequence selection $\boldsymbol{g}_n(t)$. To reduce the dimensionality of the action space, we use the index of the pilot sequence selected by user n in time slot t to indicate its action. That is

$$a_n(t) \in \widehat{\mathcal{A}} = \{0, 1, \ldots, K\}, \quad n \in \mathcal{N},\, t \in \mathcal{T}, \tag{2.10}$$

where $a_n(t) = 0$ means user n does not transmit in time slot t. The joint action of all users in time slot t is given by

$$\boldsymbol{a}(t) \triangleq (a_1(t), \cdots, a_N(t)),\, t \in \mathcal{T}. \tag{2.11}$$

The joint action space $\mathcal{A}$ is equal to $\widehat{\mathcal{A}}^N$.

State Transition Probability $\mathcal{P}$

The state transition probability $\mathbb{P}(\boldsymbol{s}(t+1) \,|\, \boldsymbol{s}(t), \boldsymbol{a}(t)),\, t \in \mathcal{T}$ is the probability that given current state $\boldsymbol{s}(t)$ and joint action $\boldsymbol{a}(t)$, the next state is $\boldsymbol{s}(t+1)$, where $\mathbb{P}(\cdot)$ denotes the probability of event $(\cdot)$. In the considered GFMA system, the next state $\boldsymbol{s}(t+1)$ can be determined with probability (w.p.) one if the current state $\boldsymbol{s}(t)$ and joint action $\boldsymbol{a}(t)$ are given. We denote the state transition probabilities as $\mathcal{P} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$.

Reward $\mathcal{R}$

A reward is a real number that is determined based on the state and joint action of all users. In the considered MDP, given the global state $\boldsymbol{s}(t)$ and joint action $\boldsymbol{a}(t)$, all users receive an aggregate reward, which is defined as follows

$$R(\boldsymbol{s}(t), \boldsymbol{a}(t)) \triangleq \sum_{n \in \mathcal{N}} \Big(\hat{r}_n(t) - \lambda_n(t)(\mu_n^{\text{req}} - \mu_n(t)) \Big),\ t \in \mathcal{T}, \tag{2.12}$$

where the term $\hat{r}_n(t)$ is given by

$$r_n(t) = \begin{cases} 1, & \text{the packet of user } n \text{ has been successfully received,} \\ -1, & \text{the packet of user } n \text{ collided with those of other users,} \\ 0, & \text{otherwise.} \end{cases} \tag{2.13}$$

The term $\lambda_n(t)(\mu_n^{\text{req}} - \mu_n(t))$ is the penalty resulting from violating the throughput constraint of user $n \in \mathcal{N}$. We set $\lambda_n(t)$ to a positive constant C when $\mu_n(t) < \mu_n^{\text{req}}$, and equal to zero when the throughput constraint is satisfied. That is,

$$\lambda_n(t) = \begin{cases} C, & \text{if } \mu_n(t) < \mu_n^{\text{req}}, \\ 0, & \text{otherwise.} \end{cases} \tag{2.14}$$

We define $\mathcal{R} \subseteq \mathbb{R}$ to be the set of rewards.

2.3.2 Global Q-Value Approximation with MA-DRL

To maximize the aggregate reward in the considered MDP, the optimal joint action *may* be determined with conventional Q-learning [67]. In Q-learning, the expected cumulative discounted reward of taking the joint action $\boldsymbol{a}$ under state $\boldsymbol{s}$ is given by the *global Q-value*

$Q^{\mathrm{G}}(\boldsymbol{s}, \boldsymbol{a})$, which is given by [67, Section 3.5]

$$Q^{\mathrm{G}}(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}\Bigg[\sum_{k=0}^{\infty} \gamma^k \, R(\boldsymbol{s}(t+k), \boldsymbol{a}(t+k)) \,\Bigg|\, \boldsymbol{s}(t) = \boldsymbol{s},\ \boldsymbol{a}(t) = \boldsymbol{a} \Bigg], \tag{2.15}$$

where $\gamma \in [0, 1]$ is a discount factor. The global Q-value is updated during the decision process. Given state $\boldsymbol{s}(t)$ in time slot t, the users choose joint action $\boldsymbol{a}(t)$, receive an aggregate reward $R(\boldsymbol{s}(t), \boldsymbol{a}(t))$, and the next state is $\boldsymbol{s}(t+1)$. Then, the global Q-value is updated as follows

$$Q^{\mathrm{G}}(\boldsymbol{s}(t), \boldsymbol{a}(t)) \leftarrow \mathbb{E}\Bigg[R(\boldsymbol{s}(t), \boldsymbol{a}(t)) + \gamma \max_{\boldsymbol{a}' \in \mathcal{A}} Q^{\mathrm{G}}(\boldsymbol{s}(t+1), \boldsymbol{a}') \,\Bigg|\, \boldsymbol{s}(t) = \boldsymbol{s},\ \boldsymbol{a}(t) = \boldsymbol{a} \Bigg]. \tag{2.16}$$

Estimating the global Q-value is beneficial in a multiagent decision process as the global state $\boldsymbol{s}(t)$ and joint action $\boldsymbol{a}(t)$ contain the information about all the users. Hence, the global Q-value can capture the impact of the action of a user on other users, and therefore can handle the nonstationary case of a multiagent decision process [112, 113]. To determine the action $\boldsymbol{a}(t)$ that maximizes the expected cumulative discounted reward, in conventional Q-learning, a Q-table is required to store the global Q-values of all possible actions for a given state $\boldsymbol{s}(t)$. However, the size of the Q-table increases with the cardinalities of the action and state spaces. This makes Q-learning costly in terms of computation and memory, especially for IoT devices.

Deep Q-learning [15] has been proposed to tackle the aforementioned issues. In deep Q-learning, the Q-value is approximated by DNNs through the establishment of a mapping between a given state and the corresponding Q-values of all possible actions. A DNN module with learnable parameters $\boldsymbol{\Phi}$ can be employed to approximate the global Q-values. In fact, $\boldsymbol{\Phi}$ is a vector that collects the weights and biases of the neurons within the DNN module.

In deep Q-learning, the learnable parameters $\boldsymbol{\Phi}$ are updated based on the system transition history to improve the accuracy of the Q-value approximation. This is referred to as the *training phase* of the DNN. In current time slot t, the system transition history

consists of the *system transition tuples* $(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau), R(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)), \boldsymbol{s}(\tau+1))$ with *time index* $\tau \in \{0, 1, \ldots, t-1\}$, which describes the system transition from time slot τ to time slot $\tau + 1$. We denote the global Q-value for $\boldsymbol{s}(\tau)$ and $\boldsymbol{a}(\tau)$, which is approximated by a DNN with parameters $\boldsymbol{\Phi}$, as $Q^{\mathrm{G}}_{\boldsymbol{\Phi}}(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))$. In each training iteration, we store the parameters of the DNN resulting from the previous training iteration, which we denote as $\widehat{\boldsymbol{\Phi}}$. Then, the target of global Q-value approximation is determined based on the Bellman equation [15], i.e., $R(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)) + \gamma \max_{\boldsymbol{a} \in \mathcal{A}} Q^{\mathrm{G}}_{\widehat{\boldsymbol{\Phi}}}(\boldsymbol{s}(\tau+1), \boldsymbol{a})$. The update of $\boldsymbol{\Phi}$ is determined by minimizing the temporal difference (TD) error between the target and the approximated global Q-value [15]. That is,

$$\arg\min_{\boldsymbol{\Phi}} \quad \frac{1}{2}\big(R(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)) + \gamma \max_{\boldsymbol{a} \in \mathcal{A}} Q^{\mathrm{G}}_{\widehat{\boldsymbol{\Phi}}}(\boldsymbol{s}(\tau+1), \boldsymbol{a}) - Q^{\mathrm{G}}_{\boldsymbol{\Phi}}(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))\big)^2. \tag{2.17}$$

To solve problem (2.17), we apply a stochastic gradient descent (SGD) algorithm to update parameters $\boldsymbol{\Phi}$. Specifically, parameters $\boldsymbol{\Phi}$ are updated as follows [98]

$$\boldsymbol{\Phi} \leftarrow \boldsymbol{\Phi} - \alpha \nabla Q^{\mathrm{G}}_{\boldsymbol{\Phi}}(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)) \left(R(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)) + \gamma \max_{\boldsymbol{a} \in \mathcal{A}} Q^{\mathrm{G}}_{\widehat{\boldsymbol{\Phi}}}(\boldsymbol{s}(\tau+1), \boldsymbol{a}) - Q^{\mathrm{G}}_{\boldsymbol{\Phi}}(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))\right), \tag{2.18}$$

where α is the learning rate. The gradient $\nabla Q^{\mathrm{G}}_{\boldsymbol{\Phi}}(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))$ in (2.18) is determined using the backpropagation algorithm [19, Chapter 6]. After updating parameters $\boldsymbol{\Phi}$ with a sufficient number of system transition tuples, the optimal joint action can be determined as follows

$$\boldsymbol{a}(t) = \arg\max_{\boldsymbol{a} \in \mathcal{A}} Q^{\mathrm{G}}_{\boldsymbol{\Phi}}(\boldsymbol{s}(t), \boldsymbol{a}). \tag{2.19}$$

To obtain the optimal joint action based on (2.19), we feed $\boldsymbol{s}(t)$ into the DNN and determine $\boldsymbol{a}(t)$ based on the output of the DNN.

2.3.3 Local Q-Value based on Factorization

To estimate the global Q-value, it is necessary for the users to know the joint action $\boldsymbol{a}(t)$ and the global state $\boldsymbol{s}(t)$ [112, 113]. However, as communication (or coordination) between

the users in the considered GFMA system is not possible, a user cannot obtain knowledge about the actions of the other users or their local states to estimate the global Q-value directly. This means that an MA-DRL algorithm that relies solely on the global Q-value cannot be implemented in the considered GFMA system in a distributed manner.

To tackle the aforementioned difficulties, rather than directly estimating the global Q-value, we propose that each user maintains a *local DNN module* to estimate the impact of its own action on the global Q-value. With the local DNN module, which is characterized by learnable parameters $\mathbf{\Phi}_n$, each user $n \in \mathcal{N}$ can obtain a *local Q-value* denoted by $Q_{\mathbf{\Phi}_n}(\boldsymbol{s}_n(\tau), a_n(\tau))$ based on its local information, i.e., its action $a_n(\tau)$ and its local state $\boldsymbol{s}_n(\tau)$.

To ensure that the local Q-value estimated by the local DNN module of user n can capture the impact of user n's action on the global Q-value, a mapping between the global Q-value and the local Q-value should be learned. To obtain such a mapping, we adopt the idea of *factorization* in multiagent systems [107, 114, 115], according to which, for a given state, an action of a user that leads to a larger global Q-value should also result in a larger local Q-value. In particular, let $\boldsymbol{a}_{-n}(\tau)$ denote the actions of all users in set $\mathcal{N}$ except for user n in time slot τ. Given the global state $\boldsymbol{s}(\tau)$ and the local state $\boldsymbol{s}_n(\tau)$, the local Q-value approximated by the local DNN module of user n is a factorization of the global Q-value if [114, Chapter 3]

$$\begin{aligned} &Q_{\mathbf{\Phi}_n}(\boldsymbol{s}_n(\tau), a_n(\tau)) > \ Q_{\mathbf{\Phi}_n}(\boldsymbol{s}_n(\tau), a'_n(\tau)) \\ &\Leftrightarrow Q^{\mathrm{G}}_{\mathbf{\Phi}}(\boldsymbol{s}(\tau), (a_n(\tau), \boldsymbol{a}_{-n}(\tau))) > \ Q^{\mathrm{G}}_{\mathbf{\Phi}}(\boldsymbol{s}(\tau), (a'_n(\tau), \boldsymbol{a}_{-n}(\tau))), \end{aligned} \tag{2.20}$$

where $a_n(\tau), a'_n(\tau) \in \widehat{\mathcal{A}}$ and $\boldsymbol{a}_{-n}(\tau) \in \widehat{\mathcal{A}}^{N-1}$. To obtain such a factorization, we use a monotonic function $F(\cdot)$ such that

$$Q^{\mathrm{G}}_{\mathbf{\Phi}}(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)) = F\big(Q_{\mathbf{\Phi}_1}(\boldsymbol{s}_1(\tau), a_1(\tau)), \cdots, Q_{\mathbf{\Phi}_N}(\boldsymbol{s}_N(\tau), a_N(\tau))\big). \tag{2.21}$$

The monotonic function in (2.21) should satisfy the following condition

$$\frac{\partial F\big(Q_{\boldsymbol{\Phi}_1}(\boldsymbol{s}_1(\tau), a_1(\tau)), \cdots, Q_{\boldsymbol{\Phi}_N}(\boldsymbol{s}_N(\tau), a_N(\tau))\big)}{\partial Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}_n(\tau), a_n(\tau))} > 0. \tag{2.22}$$

Any monotonic function $F(\cdot)$ that satisfies constraint (2.22) guarantees that the resulting local Q-value is a factorization of the global Q-value as specified in (2.20).

We emphasize that the factorization of the global Q-value is the key to the design of a distributed pilot sequence selection scheme based on MA-DRL. The factorization of the global Q-value in (2.21) leads to the following property

$$\arg\max_{\boldsymbol{a}\in\mathcal{A}} Q^{\mathrm{G}}_{\boldsymbol{\Phi}}(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)) = \big(\arg\max_{a_1\in\widehat{\mathcal{A}}} Q_{\boldsymbol{\Phi}_1}(\boldsymbol{s}_1(\tau), a_1), \ldots, \arg\max_{a_N\in\widehat{\mathcal{A}}} Q_{\boldsymbol{\Phi}_N}(\boldsymbol{s}_N(\tau), a_N)\big). \tag{2.23}$$

Equation (2.23) shows that the joint action that maximizes the global Q-value corresponds to the individual actions that maximize the local Q-value of each user $n \in \mathcal{N}$. In other words, each user can determine its own action that maximizes the global Q-value by greedily selecting the action that maximizes its local Q-value, which is

$$a_n(t) = \arg\max_{a_n\in\widehat{\mathcal{A}}} Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}_n(t), a_n). \tag{2.24}$$

To obtain $a_n(t)$ in (2.24), user n feeds $\boldsymbol{s}_n(t)$ into its local DNN module and determines $a_n(t)$ based on the output of the DNN. This does not require knowledge of the actions of the other users or the global state, and therefore can be implemented in a distributed manner.

2.4 DNN Architecture and Proposed Scheme

In this section, we propose an architecture for the DNN modules to approximate the aforementioned global and local Q-values in GFMA systems. We also present an online pilot sequence selection scheme, in which the pilot sequence selections are determined based

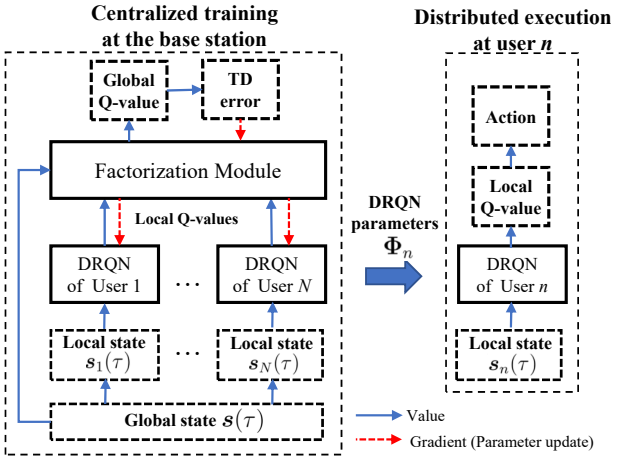


Figure 2.1: The overall network architecture. The DRQNs of all users are jointly trained at the base station. The base station sends the updated parameters of the pre-trained DRQNs to the users. The users can then select pilot sequences in a distributed manner. The forward propagations for determining the Q-values and the TD error are denoted by solid blue arrows, while the backpropagations for updating the learnable parameters are denoted by dashed red arrows.

on the outputs of pre-trained DNN modules.

2.4.1 Overall Network Architecture

The overall network architecture of the proposed DNN module for approximating the global and local Q-values is illustrated in Fig. 2.1. The DNN module consists of two parts:

- DRQN: DRQN is an RNN-based DNN module that can aggregate experience from the system transition history. We use DRQNs to generate the local Q-values based on the local states of the users.

- Factorization Module: The factorization module is a multi-layer perceptron (MLP) module that guarantees that the local Q-values of the users are the factorization of the approximated global Q-value. To this end, the factorization module is trained to approximate the monotonic function $F(\cdot)$ in (2.21) by taking advantage of the global information, i.e., the global state $\boldsymbol{s}(t)$ that is available at the base station.

In particular, the training of the DRQNs of all users is performed by the base station. During the centralized training phase at the base station (left-hand side of Fig. 2.1), the DRQNs of all users are jointly trained based on global information. The advantages of using centralized training in wireless systems are two-fold. First, the base station, which is equipped with powerful hardware, can efficiently train the DNNs for the users. Second, the users (i.e., IoT devices) can prolong their battery lifetime by not being involved in the energy-consuming training phase. The factorization module is employed only during the training phase to ensure that the global Q-value can be properly factorized. After training, the base station sends the learnable parameters of the DRQNs to the users. The users select the pilot sequences based on the outputs of the DRQNs in a distributed manner based on its local information (right-hand side of Fig. 2.1). Note that the base station does not know which pilot sequence selections of the users lead to collisions. Moreover, the base station cannot differentiate between the users that remain silent and the users that suffer from collisions. This prevents the base station from knowing the joint action $\boldsymbol{a}(\tau)$. Hence, in practice, user n may include the history of its action $a_n(\tau)$ in its data packet, so that the base station can obtain the complete system transition history.

The proposed training framework falls into the category of CTDE frameworks in distributed DRL and MA-DRL [116, 117]. We also combine the factorization technique with MA-DRL in the proposed framework. The advantage of the proposed framework is that the distributed and user-specific pilot sequence selection policies can be learned during the centralized training process. Moreover, compared with the conventional tabular Q-learning [67], the proposed learning framework is able to better handle the large joint action space with the help of DNNs. The details of the network architecture are presented in the next two subsections.

2.4.2 DRQN Design

The DRQNs of all users have the same network architecture as illustrated in Fig. 2.2. For each user $n \in \mathcal{N}$, the DNN layers in the DRQN and their functionalities are as follows:

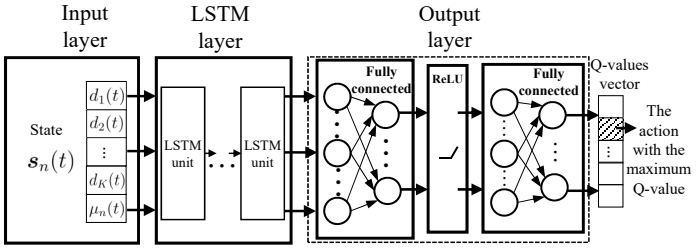


Figure 2.2: The proposed network structure for DRQN, which consists of an input layer, an LSTM layer, and an output layer.

Input Layer

The input layer collects the local state and feeds it to the DNN. For the DRQN of user n, the input is the local state $\boldsymbol{s}_n(t)$, which is a vector of size $K+1$.

Long Short-Term Memory (LSTM) Layer

The local state $\boldsymbol{s}_n(t)$ collected by the input layer is fed into an LSTM layer. LSTM layer is a type of RNN. Using the LSTM layer, the information about the system transition history can be learned and stored in its hidden states. Therefore, LSTM layer can learn the temporal correlations of the system transitions more efficiently than feedforward neural networks. For the same reason, the LSTM layer is capable of overcoming the lack of complete information in the underlying MDP [117]. In particular, we use an LSTM layer with D hidden units to aggregate the information from the system transition history and approximate the local Q-values. The outputs of the LSTM layer are its hidden states.

Output Layer

The LSTM layer is connected to the output layer to generate the local Q-values. Specifically, we use two fully connected (FC) layers with one rectified linear unit (ReLU) layer to generate the local Q-values based on the hidden states of the LSTM layer. The output is a vector of size $K+1$. For $k \in \mathcal{K}$, the $(k+1)$-th entry of the output vector is the local Q-value for selecting the k-th pilot sequence in time slot t for local state $\boldsymbol{s}_n(t)$, while the

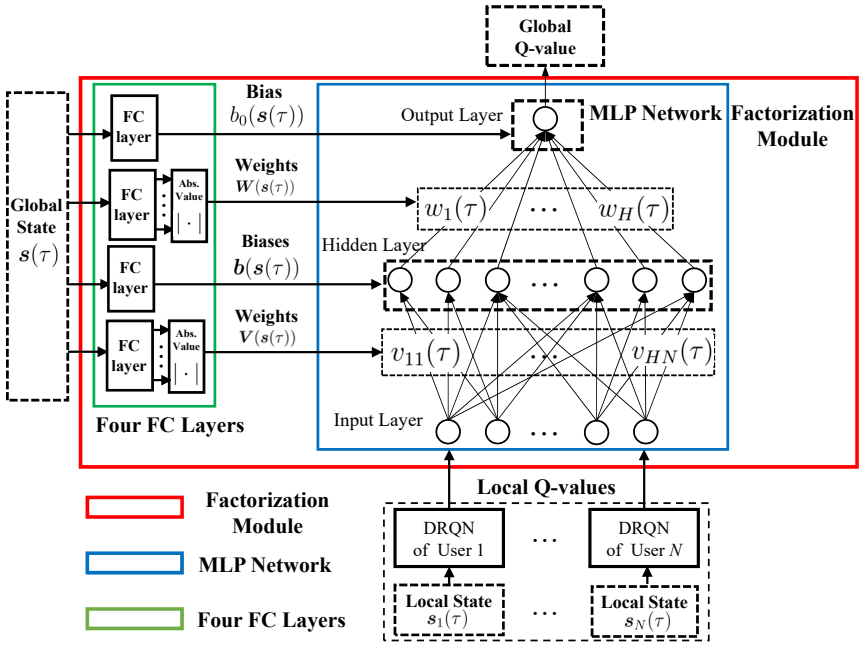


Figure 2.3: DNN architecture for the factorization module located at the base station. The factorization module, which is indicated by the red box in the figure, consists of two parts: (i) a universal approximator for obtaining the monotonic function $F(\cdot)$, which is an MLP network with one hidden layer as indicated by the blue box in the figure, and (ii) four FC layers, i.e., the green box in the figure, for generating the weights and biases of the MLP network. Each FC layer in the factorization module takes the global state as input.

first entry is the local Q-value for not transmitting in time slot t.

2.4.3 Factorization Module Design

The DNN architecture of the factorization module is shown in Fig. 2.3. To obtain the desired function $F(\cdot)$ for global Q-value factorization, we use an MLP network-based universal approximator (see the blue box in Fig. 2.3) to approximate function $F(\cdot)$, while the weights and biases of the MLP network are generated by four FC layers (see the green box Fig. 2.3). As shown in [68], an MLP network with one hidden layer can serve as a universal approximator. In particular, the input layer of the MLP network has N neurons and each neuron takes respectively one of the local Q-values approximated by the DRQNs of the N users as input. The hidden layer of the MLP network has H neurons. There is

one neuron in the output layer as the global Q-value is a scalar. We denote the weight between the j-th neuron of the hidden layer and the n-th neuron of the input layer and the bias of the j-th neuron of the hidden layer in time slot τ as $v_{jn}(\tau)$ and $b_j(\tau)$, respectively. We denote the weight between the j-th neuron of the hidden layer and the neuron of the output layer in time slot τ as $w_j(\tau)$. The bias of the neuron of the output layer in time slot τ is denoted by $b_0(\tau)$. Then, function $F(\cdot)$ obtained with the MLP network can be expressed as follows [68]

$$\begin{aligned} &F(Q_{\boldsymbol{\Phi}_1}(\boldsymbol{s}_1(\tau),\boldsymbol{a}_1(\tau)),\ldots,Q_{\boldsymbol{\Phi}_N}(\boldsymbol{s}_N(\tau),\boldsymbol{a}_N(\tau))) \\ &\quad= b_0(\tau)+\sum_{j=1}^{H} w_j(\tau)h\Big(b_j(\tau)+\sum_{n\in\mathcal{N}} v_{jn}(\tau)Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}_n(\tau),a_n(\tau))\Big), \end{aligned} \tag{2.25}$$

where $h(\cdot)$ is the sigmoid function.

Recall that, in order to ensure that the local Q-value is a factorization of the global Q-value, function $F(\cdot)$ should satisfy condition (2.22). This means that function $F(\cdot)$ should be entry-wise monotonic increasing with respect to the inputs $Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}_n(\tau),a_n(\tau))$, $n\in\mathcal{N}$. For the function approximated by the universal approximator in (2.25), since the sigmoid function $h(\cdot)$ is monotonically increasing, condition (2.22) is satisfied if the weights, i.e., $w_j(\tau)$ and $v_{jn}(\tau)$, in time slot $\tau\in\mathcal{T}$ are all positive [68]. Hence, we emphasize that the key to obtaining a function $F(\cdot)$ which possesses the desired factorization property is to enforce the positivity of the weights of the MLP network during the training phase.

As the global Q-value $Q^{\mathrm{G}}(\boldsymbol{s}(\tau),\boldsymbol{a}(\tau))$ depends on the global state $\boldsymbol{s}(\tau)$, it is intuitive to let the approximated monotonic function $F(\cdot)$ depend on the global state as well. Therefore, we use DNNs to generate the weights and biases of the universal approximator in time slot τ based on the global state $\boldsymbol{s}(\tau)$. By doing this, we let the weights and biases of the universal approximator depend on the specific state in the time slot, so that the monotonic function $F(\cdot)$ can be adjusted with respect to different states, e.g., the different achieved throughputs of the users. This makes the universal approximator more flexible. To take the effect of state $\boldsymbol{s}(\tau)$ on the weights and biases into account, we replace τ in

their arguments in (2.25) by $\boldsymbol{s}(\tau)$. The universal approximator (2.25) can then be written in the following matrix notation

$$\begin{aligned} &F\big(Q_{\boldsymbol{\Phi}_1}(\boldsymbol{s}_1(\tau),\boldsymbol{a}_1(\tau)),\ldots,Q_{\boldsymbol{\Phi}_N}(\boldsymbol{s}_N(\tau),\boldsymbol{a}_N(\tau))\big) \\ &= \boldsymbol{W}(\boldsymbol{s}(\tau))\; h\big(\boldsymbol{V}(\boldsymbol{s}(\tau))\;\boldsymbol{Q}(\boldsymbol{s}(\tau),\boldsymbol{a}(\tau)) + \boldsymbol{b}(\boldsymbol{s}(\tau))\big) + b_0(\boldsymbol{s}(\tau)), \end{aligned} \tag{2.26}$$

where function $h(\cdot)$ in (2.26) is the element-wise sigmoid function and

$$\boldsymbol{Q}(\boldsymbol{s}(\tau),\boldsymbol{a}(\tau)) = \Big[Q_{\boldsymbol{\Phi}_1}(\boldsymbol{s}_1(\tau),\boldsymbol{a}_1(\tau))\cdots\;\; Q_{\boldsymbol{\Phi}_N}(\boldsymbol{s}_N(\tau),\boldsymbol{a}_N(\tau))\Big]^\top \in \mathbb{R}^N, \tag{2.27}$$

$$\boldsymbol{V}(\boldsymbol{s}(\tau)) = \begin{bmatrix} v_{11}(\boldsymbol{s}(\tau)) & \cdots & v_{1N}(\boldsymbol{s}(\tau)) \\ \vdots & \ddots & \vdots \\ v_{H1}(\boldsymbol{s}(\tau)) & \cdots & v_{HN}(\boldsymbol{s}(\tau)) \end{bmatrix} \in \mathbb{R}^{H\times N}, \tag{2.28}$$

$$\boldsymbol{b}(\boldsymbol{s}(\tau)) = \Big[b_1(\boldsymbol{s}(\tau))\; b_2(\boldsymbol{s}(\tau))\;\cdots\; b_H(\boldsymbol{s}(\tau))\Big]^\top \in \mathbb{R}^H, \tag{2.29}$$

and

$$\boldsymbol{W}(\boldsymbol{s}(\tau)) = \Big[w_1(\boldsymbol{s}(\tau))\; w_2(\boldsymbol{s}(\tau))\;\cdots\; w_H(\boldsymbol{s}(\tau))\Big] \in \mathbb{R}^{1\times H}. \tag{2.30}$$

The weights $\boldsymbol{V}(\boldsymbol{s}(\tau))$, $\boldsymbol{W}(\boldsymbol{s}(\tau))$ and biases $\boldsymbol{b}(\boldsymbol{s}(\tau))$, $b_0(\boldsymbol{s}(\tau))$ in time slot τ are generated by multiple DNNs separately to improve the representational capacity of the universal approximator [94].

In particular, we use one FC layer to generate the weights $\boldsymbol{V}(\boldsymbol{s}(\tau))$. The input of this FC layer is the global state $\boldsymbol{s}(\tau)$, and the output is a vector of size HN. The output vector is then rearranged into an $H \times N$ matrix. The weights $\boldsymbol{V}(\boldsymbol{s}(\tau))$ are obtained by taking the absolute values of all elements of the output matrix. We use a similar method to obtain the weights $\boldsymbol{W}(\boldsymbol{s}(\tau))$. Note that by taking the absolute values, we ensure that the weights in (2.25) are all positive, and therefore the desired factorization property is achieved. We employ one FC layer to obtain the biases $\boldsymbol{b}(\boldsymbol{s}(\tau))$. This FC layer takes $\boldsymbol{s}(\tau)$ as input, and

Algorithm 1 Training Algorithm for MA-DRL Framework in Each Training Iteration

1: Sample a minibatch of system transition tuples $(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau), R(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)), \boldsymbol{s}(\tau+1)), \tau \in \mathcal{T}_b'$.
2: **for** $n \in \mathcal{N}$ **do**
3: Determine $Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}_n(\tau), a_n(\tau))$ and $\max_{a_n \in \widehat{\mathcal{A}}} Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}_n(\tau+1), a_n)$ with the DRQN of user n.
4: **end for**
5: Determine the approximated global Q-value $Q^{\mathrm{G}}_{\boldsymbol{\Phi}}(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))$ based on (2.31).
6: Determine the target global Q-value $y^{\mathrm{G}}_{\boldsymbol{\Phi}}(\boldsymbol{s}(\tau+1), R(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)))$ based on (2.32).
7: Update the parameters $\boldsymbol{\Phi}$ using SGD (2.35).

the output is a vector $\boldsymbol{b}(\boldsymbol{s}(\tau))$ of size H. It is not necessary to make the biases positive as the signs of the biases will not affect the monotonicity. We use a similar method to obtain the bias $b_0(\boldsymbol{s}(\tau))$. The output dimension of the FC layer for $b_0(\boldsymbol{s}(\tau))$ is equal to one.

2.4.4 Joint Training Algorithm

We use $\boldsymbol{\Phi}_0$ to denote the vector that contains the learnable parameters of the factorization module. Since the monotonic function $F(\cdot)$ in time slot τ depends on both the global state $\boldsymbol{s}(\tau)$ and the parameters $\boldsymbol{\Phi}_0$, we denote it as $F_{\boldsymbol{s}(\tau),\boldsymbol{\Phi}_0}(\cdot)$. Based on the proposed network architecture, the parameters $\boldsymbol{\Phi}$ used to approximate global Q-value $Q^{\mathrm{G}}_{\boldsymbol{\Phi}}(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))$ are now given by $\boldsymbol{\Phi} = (\boldsymbol{\Phi}_0, \boldsymbol{\Phi}_1, \ldots, \boldsymbol{\Phi}_N)$, which includes the parameters of the N DRQNs and the parameters of the factorization module. The training algorithm is summarized in **Algorithm 1**. In Line 5, the approximated global Q-value for state $\boldsymbol{s}(\tau)$ and joint action $\boldsymbol{a}(\tau)$ is given by

$$Q^{\mathrm{G}}_{\boldsymbol{\Phi}}(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)) = F_{\boldsymbol{s}(\tau),\boldsymbol{\Phi}_0}\big(Q_{\boldsymbol{\Phi}_1}(\boldsymbol{s}_1(\tau), a_1(\tau)), \ldots, Q_{\boldsymbol{\Phi}_N}(\boldsymbol{s}_N(\tau), a_N(\tau))\big). \tag{2.31}$$

To obtain $Q^{\mathrm{G}}_{\boldsymbol{\Phi}}(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))$, we first feed the local state $\boldsymbol{s}_n(\tau), n \in \mathcal{N}$, in time slot τ into the DRQN of user n. $Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}_n(\tau), a_n(\tau))$ can be obtained by selecting the value of the entry of the output that corresponds to $a_n(\tau)$. Then, we feed the global state $\boldsymbol{s}(\tau)$ into the factorization module. With the outputs of the four FC layers, the weights $\boldsymbol{V}(\boldsymbol{s}(\tau)), \boldsymbol{W}(\boldsymbol{s}(\tau))$ and the biases $\boldsymbol{b}(\boldsymbol{s}(\tau)), b_0(\boldsymbol{s}(\tau))$ which yield the monotonic function $F_{\boldsymbol{s}(\tau),\boldsymbol{\Phi}_0}(\cdot)$. $Q^{\mathrm{G}}_{\boldsymbol{\Phi}}(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))$

can now be obtained based on (2.26) with $Q_{\mathbf{\Phi}_n}(\boldsymbol{s}_n(\tau), a_n(\tau)), n \in \mathcal{N}$, as inputs of the monotonic function.

The base station can now jointly train the factorization module and the DRQNs based on the TD error of the global Q-value approximation. In Line 6, for the system transition tuple with time index τ, the target of the global Q-value approximation is

$$\begin{aligned}
&y^{\mathrm{G}}_{\mathbf{\Phi}}(\boldsymbol{s}(\tau+1), R(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))) \\
&= R(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)) + \gamma \max_{\boldsymbol{a}\in\mathcal{A}} Q^{\mathrm{G}}_{\mathbf{\Phi}}(\boldsymbol{s}(\tau+1), \boldsymbol{a}) \\
&= R(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)) + \gamma \max_{\boldsymbol{a}\in\mathcal{A}} F_{\boldsymbol{s}(\tau+1),\mathbf{\Phi}_0}\big(Q_{\mathbf{\Phi}_1}(\boldsymbol{s}_1(\tau+1), a_1), \ldots, Q_{\mathbf{\Phi}_N}(\boldsymbol{s}_N(\tau+1), a_N)\big) \\
&\overset{(a)}{=} R(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)) + \gamma\, F_{\boldsymbol{s}(\tau+1),\mathbf{\Phi}_0}\big(\max_{a_1\in\widehat{\mathcal{A}}} Q_{\mathbf{\Phi}_1}(\boldsymbol{s}_1(\tau+1), a_1), \ldots, \max_{a_N\in\widehat{\mathcal{A}}} Q_{\mathbf{\Phi}_N}(\boldsymbol{s}_N(\tau+1), a_N)\big).
\end{aligned} \tag{2.32}$$

Equality (a) holds since given state $\boldsymbol{s}(\tau+1)$ and parameters $\mathbf{\Phi}_0$, the weights and biases in monotonic function $F_{\boldsymbol{s}(\tau+1),\mathbf{\Phi}_0}(\cdot)$ are constants. Given the local state $\boldsymbol{s}_n(\tau+1)$ and parameters $\mathbf{\Phi}_n$, the output of the DRQN of user n is a constant vector of size $K+1$, meaning that we only have $K+1$ options for the value of $Q_{\mathbf{\Phi}_n}(\boldsymbol{s}_n(\tau+1), a_n)$ for user n. As function $F_{\boldsymbol{s}(\tau+1),\mathbf{\Phi}_0}(\cdot)$ is entry-wise monotonically increasing, the maximum can be obtained by selecting the maximum element, $\max_{a_n\in\widehat{\mathcal{A}}} Q_{\mathbf{\Phi}_n}(\boldsymbol{s}_n(\tau+1), a_n)$, in the output vector of user n's DRQN as the input of function $F_{\boldsymbol{s}(\tau+1),\mathbf{\Phi}_0}(\cdot)$.

To obtain the target global Q-value (2.32), we feed the local states of time slot $\tau+1$ into the DRQNs of the users accordingly and the value of $\max_{a_n\in\widehat{\mathcal{A}}} Q_{\mathbf{\Phi}_n}(\boldsymbol{s}_n(\tau+1), a_n), n \in \mathcal{N}$, can be obtained by selecting the action that corresponds to the entry with the maximum value in the output. Then, we feed the global state $\boldsymbol{s}(\tau+1)$ into the factorization module. Based on the outputs of the four FC layers, we obtain the weights $\boldsymbol{V}(\boldsymbol{s}(\tau+1)), \boldsymbol{W}(\boldsymbol{s}(\tau+1))$ and biases $\boldsymbol{b}(\boldsymbol{s}(\tau+1)), b_0(\boldsymbol{s}(\tau+1))$, and determine the monotonic function $F_{\boldsymbol{s}(\tau+1),\mathbf{\Phi}_0}(\cdot)$. $F_{\boldsymbol{s}(\tau+1),\mathbf{\Phi}_0}(\max_{a_1\in\widehat{\mathcal{A}}} Q_{\mathbf{\Phi}_1}(\boldsymbol{s}_1(\tau+1), a_1), \cdots, \max_{a_N\in\widehat{\mathcal{A}}} Q_{\mathbf{\Phi}_N}(\boldsymbol{s}_N(\tau+1), a_N))$ can now be obtained by using $\max_{a_n\in\widehat{\mathcal{A}}} Q_{\mathbf{\Phi}_n}(\boldsymbol{s}_n(\tau+1), a_n), n \in \mathcal{N}$, as the input of the monotonic function. Then, the TD error for the global Q-value approximation with respect to the system transition tuple

with time index τ is given by

$$\mathcal{L}_G(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau), R(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)), \boldsymbol{s}(\tau+1)) = \frac{1}{2}\big(y^{\mathrm{G}}_{\boldsymbol{\Phi}}(\boldsymbol{s}(\tau+1), R(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))) - Q^{\mathrm{G}}_{\boldsymbol{\Phi}}(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))\big)^2. \tag{2.33}$$

In practical systems, the base station maintains the system transition history, which is referred to as the *experience replay* in the DRL literature [15, 93]. In each training iteration, the base station samples a minibatch that contains multiple system transition tuples from the experience replay. The base station determines the TD error in (2.33) for the tuples within the minibatch, and then updates the learnable parameters $\boldsymbol{\Phi}$. To efficiently train the LSTM layer, instead of randomly sampling the system transition tuples from the experience replay, we sample the system transition tuples of m consecutive time slots and feed the corresponding system transition history sequentially into the DNN module to update the parameters. By doing this, the hidden states of the LSTM layer can be carried forward throughout the entire training iteration to learn the temporal correlations from the consecutive system transition history.

In addition, training the LSTM layer with minibatches sampled from the replay is known to suffer from *initial recurrent state mismatch* [117], which may lead to inaccurate Q-value approximation. In particular, the minibatches sampled in two consecutive training iterations (i.e., the previous training iteration and the current training iteration) may represent the system transition history of two different time periods. This means the temporal correlations within the system transition history sampled in the previous training iteration may significantly differ from the ones sampled in the current training iteration. Therefore, the hidden states of the LSTM layer generated based on the system transition history sampled in the previous training iteration may not be able to accurately approximate the Q-values with respect to the system transition history sampled in the current training iteration.

To overcome the initial recurrent state mismatch, we use the first l consecutive system transition tuples within the sampled minibatch to initialize the hidden states of the LSTM

layer, so that the hidden states can be initialized based on the temporal correlations of the system transition history sampled in the current training iteration (which is also referred to as the *burn-in method* [117]). This is accomplished by sequentially feeding the state history of the first l time slots into the DRQNs. Thus, the hidden states can be initialized and carried forward. Subsequently, we use the state history of the remaining $m-l$ consecutive time slots to update the learnable parameters. The TD errors of global Q-value approximation are determined only for the states of the remaining $m-l$ time slots and then averaged. We use $\mathcal{T}_b'$ to denote the set of the time indices of the system transition tuples within the minibatch, excluding the first l transition tuples. For the sampled minibatch, the TD error of the global Q-value approximation is

$$\mathcal{L}_G = \frac{1}{m-l} \sum_{\tau \in \mathcal{T}_b'} \mathcal{L}_G(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau), R(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)), \boldsymbol{s}(\tau+1)). \tag{2.34}$$

Then, in Line 7, all learnable parameters $\boldsymbol{\Phi}$ are updated using SGD [95]:

$$\boldsymbol{\Phi} \leftarrow \boldsymbol{\Phi} - \frac{\alpha}{m-l} \sum_{\tau \in \mathcal{T}_b'} (y_{\boldsymbol{\Phi}}^{\mathrm{G}}(\boldsymbol{s}(\tau+1), R(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))) - Q_{\boldsymbol{\Phi}}^{\mathrm{G}}(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))) \nabla Q_{\boldsymbol{\Phi}}^{\mathrm{G}}(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)). \tag{2.35}$$

The training in (2.35) is an *end-to-end* learning process. That is, in each training iteration, the DRQNs of all users, as well as the factorization module, are jointly trained based on the TD error of global Q-value approximation.

2.4.5 Proposed Online Scheme

As shown in (2.23) and (2.24), a user can determine its pilot sequence selection in a distributed manner by leveraging its local information and the pre-trained DRQN. In particular, the base station sends the learnable parameters of user n's DRQN, i.e., $\boldsymbol{\Phi}_n$, back to user $n \in \mathcal{N}$. Each user maintains a local DRQN. Upon receiving $\boldsymbol{\Phi}_n$ from the base station, user n updates the learnable parameters of its local DRQN to be the same as $\boldsymbol{\Phi}_n$. User n then applies an ϵ-greedy policy to select its pilot sequence. Specifically, the pilot

sequence selection of user n in the current time slot t is determined by [15]

$$a_n(t) = \begin{cases} \arg\max_{a_n \in \widehat{\mathcal{A}}} Q_{\mathbf{\Phi}_n}(\boldsymbol{s}_n(t), a_n), & \text{with probability } 1-\epsilon \\ \text{random selection}, & \text{with probability } \epsilon, \end{cases} \tag{2.36}$$

where $\epsilon = \epsilon_{\min} + (\epsilon_{\max} - \epsilon_{\min})e^{-G/\epsilon_{\text{decay}}}$, and $\epsilon_{\min}$, $\epsilon_{\max}$, and ϵ_{decay} are constants. G is the training iteration counter. The ϵ-greedy policy is adopted to avoid overfitting. To determine action $a_n(t) = \arg\max_{a_n \in \widehat{\mathcal{A}}} Q_{\mathbf{\Phi}_n}(\boldsymbol{s}_n(t), a_n)$ in the current time slot t, user n feeds the local state $\boldsymbol{s}_n(t)$ into the DRQN, and determines the pilot sequence selection in time slot t based on the output of the DRQN.

2.4.6 Discussion

The framework in Fig. 2.3 is scalable as the number of DRQN modules can be varied according to the number of users without changing the subsequent training algorithm. We note that the learnable parameters of the DNNs have to be updated if the values of N or K change. However, the base station can effectively avoid frequent re-training of the DNNs by using user clustering. Consider the case where the base station serves N' users with K' pilot sequences, but has pre-trained DNNs for N users and K pilot sequences. In this case, the base station can divide the users into $\lceil \frac{N'}{N} \rceil$ clusters, where each cluster contains N users. The base station then allocates K *orthogonal* pilot sequences to each cluster of users. This means that different clusters are allocated with different pilot sequences, and hence there is no pilot sequence collision between users in different clusters. This requires $\lceil \frac{N'}{N} \rceil K$ orthogonal pilot sequences. By doing this, the pre-trained DNNs can be re-used within each cluster of users, without invoking re-training.

We note that one user cluster may not have exactly N users when N' is not a multiple of N or N' is less than N. Moreover, one user cluster may not be allocated with exactly K pilot sequences when K' is not a multiple of K or K' is less than K. In both cases, the DNNs of the users within this particular user cluster have to be re-trained. The reason is

that, as the number of users or pilot sequences in a particular cluster changes, the pilot sequence selection policies of the users within this cluster should be updated in order to adapt to the new network setting. To this end, the DNNs have to be re-trained to learn new pilot sequence selection policies.

2.5 Performance Evaluation

In this section, we evaluate the performance of the proposed scheme. We simulate a GFMA system serving users with three different throughput requirements. Therefore, the users can be categorized into three groups (or types), i.e., Group I, Group II, and Group III, based on their throughput requirements. Throughout the simulations, we set the numbers of users in Group I, Group II, and Group III to be $\eta_1 N$, $\eta_2 N$, and $\eta_3 N$, respectively, where the coefficients η_1, η_2, $\eta_3 \in (0, 1)$ and $\eta_1 + \eta_2 + \eta_3 = 1$. As the maximum aggregate throughput for the considered system is equal to K packets per time slot, we assume the total numbers of successful packet transmissions per time slot required by Group I, Group II and Group III users to be $\zeta_1 K$, $\zeta_2 K$, and $\zeta_3 K$, respectively, where the coefficients ζ_1, ζ_2, $\zeta_3 \in (0, 1)$ and $\zeta_1 + \zeta_2 + \zeta_3 \leq 1$. Hence, given N and K, the throughputs required by each user in Group I, Group II, and Group III are given by $\frac{\zeta_1 K}{\eta_1 N}$, $\frac{\zeta_2 K}{\eta_2 N}$, and $\frac{\zeta_3 K}{\eta_3 N}$, respectively. We set $\frac{\zeta_1}{\eta_1} > \frac{\zeta_2}{\eta_2} > \frac{\zeta_3}{\eta_3}$ such that Group I users have the highest throughput requirement, while Group III users have the lowest throughput requirement. The number of users is at least two times as large as the number of pilot sequences in the considered GFMA system. In each time slot, the base station trains the DNN modules for 20 iterations. The detailed parameter settings used for simulations are shown in Table 2.1.

We compare the proposed MA-DRL based scheme with the following benchmark and baseline schemes[2]:

- **The optimal scheme**: In this scheme, the pilot sequence selections are determined

[2]While a DRL-based spectrum access scheme is proposed [38], the system model and problem setting of [38] are different from the framework we considered in this chapter. Hence, we have not included a performance comparison with [38] in this chapter.

Table 2.1: Simulation Parameters

Parameter	**Value**
Number of hidden units in LSTM D	64
Discount factor γ	0.995
Dimensions of the first and second FC layers in the DRQN	64×128 and $64 \times (K+1)$
Step size used in SGD α	0.01
Number of neurons in the hidden layer of the MLP network H	64
Number of tuples in minibatch m	80
Number of tuples in minibatch for initializing hidden states l	40
Constant in the reward function C	50
$\epsilon_{\min}$, $\epsilon_{\max}$, and ϵ_{decay} in ϵ-greedy policy	0.05, 0.95, and 2000
Coefficients η_1, η_2, and η_3 of the numbers of users in Groups I, II, and III	0.25, 0.5, and 0.25
Coefficients ζ_1, ζ_2, and ζ_3 for the throughput requirements	0.35, 0.3, and 0.075

by the Lyapunov drift-plus-penalty algorithm as shown in Appendix A. We set $V = 1$ when evaluating the performance. With the optimal scheme, the maximum aggregate throughput can be achieved and the throughput requirements of all the users can be satisfied.

- **Dynamic ACB with optimal parameter setting** [101]: In this scheme, the base station first determines the number of pilot sequences allocated to Group I users, such that the pilot sequence collision probability of Group I users is minimized. The base station continues to do the same for Group II users and then allocates the remaining pilot sequences to Group III users.

- **ACK-based scheme** [35]: In this scheme, the base station reserves the pilot sequences that have been selected by multiple users in the previous time slot for the retransmissions of users that have suffered a packet collision. We assume no collision will happen in the retransmissions after the ACK has been sent to the users.

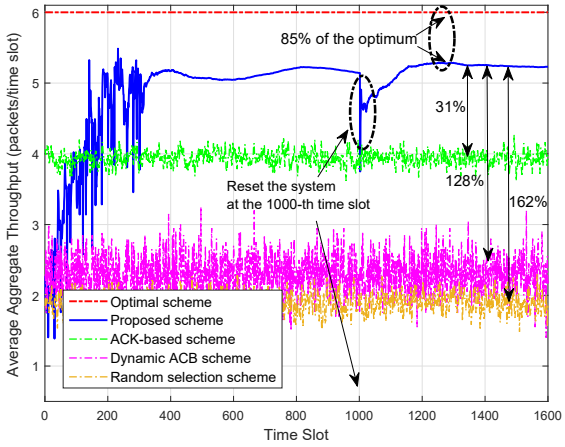
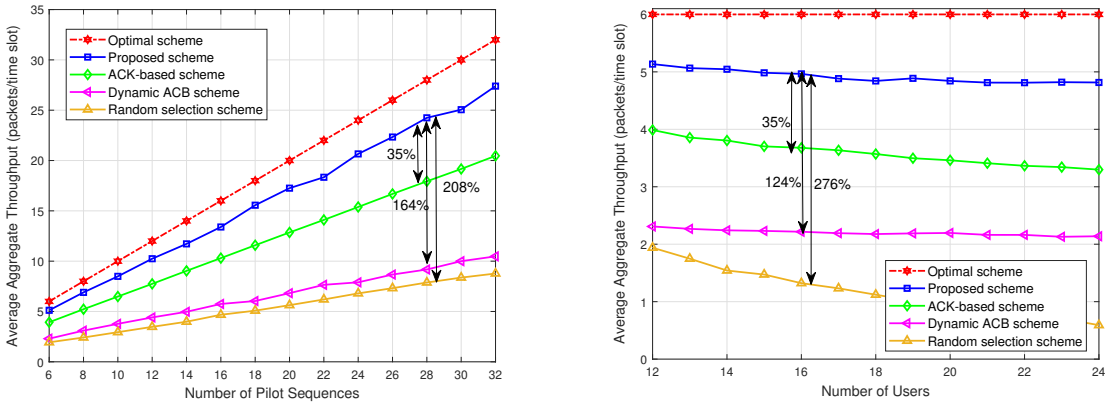


Figure 2.4: Average aggregate throughput versus time slots. We set $N = 12$ and $K = 6$. The proposed scheme achieves 85% of the optimal average aggregate throughput.

- **Random selection scheme**: Here, each user randomly selects one pilot sequence from the available ones.

2.5.1 Average Aggregate Throughput

Fig. 2.4 shows the evolution of the average aggregate throughput versus the time slots. We simulate $N = 12$ users sharing $K = 6$ pilot sequences. The throughput requirements of three of the users are set to $\frac{\zeta_1 K}{\eta_1 N} = 0.7$ packets per time slot (Group I users), six of the users require $\frac{\zeta_2 K}{\eta_2 N} = 0.3$ packets per time slot (Group II users), and the remaining three users require $\frac{\zeta_3 K}{\eta_3 N} = 0.15$ packets per time slot (Group III users). The proposed scheme achieves 85% of the optimal average aggregate throughput after 800 time slots, which is 31%, 128%, and 162% higher than that of the ACK-based scheme, the ACB-based scheme, and the random selection scheme, respectively. A gap exists between the aggregate throughput of the proposed scheme and the optimal scheme due to errors in the Q-value approximation and the ϵ-greedy policy in (2.36). In the 1000th time slot, we reset the system, i.e., we set the average throughput of all users back to zero. We observe that the proposed scheme quickly recovers from the new initial state after 150 time slots and retains the average aggregate throughput which is 85% of the optimum.



(a) (b)

Figure 2.5: Average aggregate throughput versus (a) the number of pilot sequences and (b) the number of users. The performance of the proposed scheme is evaluated after the DNNs have been trained for 1500 time slots.

Fig. 2.5(a) illustrates the average aggregate throughput versus the number of pilot sequences K, where we set the number of users to $2K$. The performance of the proposed scheme is evaluated after the DNNs have been trained for 1500 time slots. The proposed scheme can achieve an average aggregate throughput that is 35%, 164%, and 208% higher than that of the ACK-based scheme, the ACB-based scheme, and the random selection scheme, respectively, when the number of pilot sequences is 28. The near-linear increases in the average aggregate throughputs show that the probabilities of pilot sequence selection collision of all schemes do not change significantly as long as the ratio of the number of users to the number of pilot sequences is fixed.

Fig. 2.5(b) shows the impact of the number of users on the average aggregate throughput when the number of pilot sequences is $K = 6$. The performance of the proposed scheme is evaluated after the DNNs have been trained for 1500 time slots. All schemes except for the optimal scheme suffer from a performance degradation as the number of users N increases. The reason for this is that the probabilities of pilot sequence collisions increase with N for all these schemes when the number of pilot sequences K is fixed. However, compared with the ACK-based scheme, the ACB-based scheme, and the random selection scheme, a lower collision probability can be achieved under the proposed scheme. In fact,

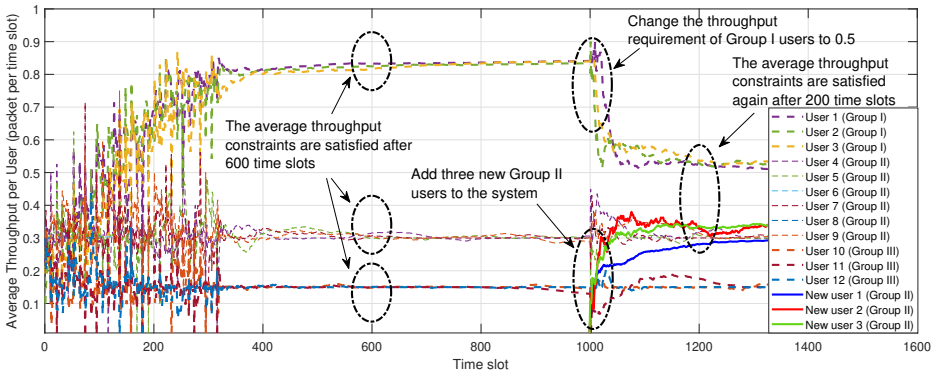


Figure 2.6: Average throughput per user versus the number of time slots for the proposed scheme. At the 1000th time slot we add three new Group II users and change the throughput requirement of Group I users to 0.5 packets per time slot.

the proposed scheme can achieve an average aggregate throughput that is 35%, 124%, and 276% higher than that of the ACK-based scheme, the ACB-based scheme, and the random selection scheme, respectively, when the number of users is 16.

2.5.2 Average Throughput of the Users

Fig. 2.6 shows the evolution of the achievable average throughput per user of the proposed scheme versus the time slots. We simulate $N = 12$ users sharing $K = 6$ pilot sequences. The results in Fig. 2.6 show that after convergence the average throughput requirements of all users are satisfied. In the proposed scheme, as all users receive the same aggregate reward, the successful transmission of a user benefits all users, while all users receive a penalty if a collision occurs or the throughput requirements are violated. Under the factorization-based MA-DRL framework, the DRQNs of the users are encouraged to learn the cooperative pilot sequence selection policies, such that users can estimate the impact of their actions on other users from a system-level perspective. Fig. 2.6 also shows that joint training based on the factorization module can ensure that the policies learned during the centralized training phase can be efficiently executed in a distributed manner without having to rely on global information. Furthermore, in the 1000th time slot, we have added three new Group II users into the system to illustrate the capability of the proposed scheme

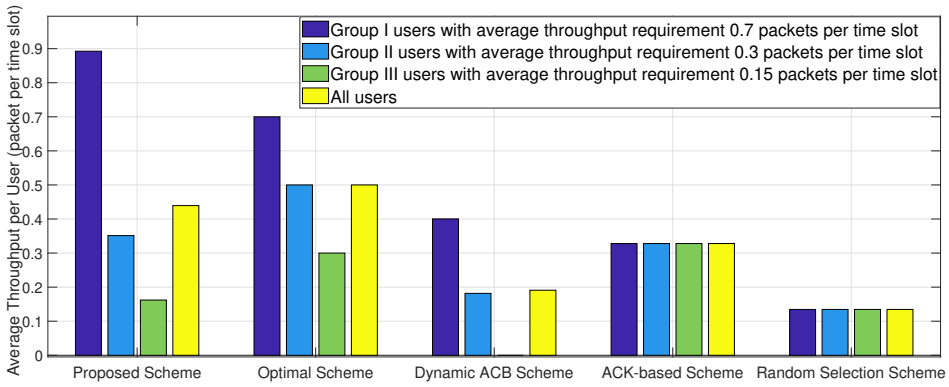


Figure 2.7: Average throughput per user for users with different throughput requirements for different schemes. We set $N = 12$ and $K = 6$. The optimal solution and the proposed scheme can satisfy all average throughput requirements.

to adapt to a new network setting. We also change the throughput requirement of Group I users from 0.7 packets per time slot to 0.5 packets per time slot. We observe that, after the DNN modules have been trained for 200 time slots, the average throughputs of the three new users reach approximately 0.3 packets per time slot. Moreover, the average throughput of the Group I users decreases to approximately 0.5 packets per time slot. These results show that the proposed scheme can adapt to new network topologies and new throughput requirements.

In Fig. 2.7, we compare the average throughput of the users for different schemes in the aforementioned setting. The performance of the proposed scheme is evaluated after the DNNs have been trained for 1500 time slots. For both the optimal scheme and the proposed scheme, all users can satisfy their average throughput requirements. For the ACB-based scheme, Group I users can achieve a higher average throughput than Group II users as the base station allocates more pilot sequences to Group I users. However, the users in Group I and II consume too many resources such that there is no resource left for the users in Group III to transmit. For both the ACK-based scheme and the random selection scheme, all users achieve the same average throughput as the throughput requirements are not taken into account in these two schemes.

We note that the user-specific throughput requirement has not been addressed in the

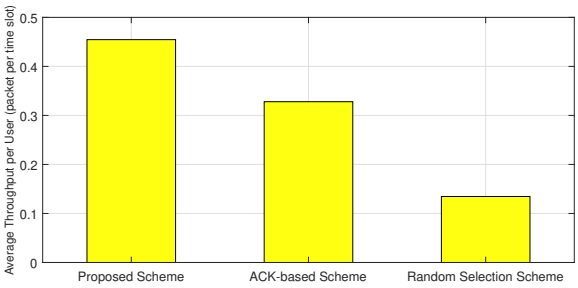


Figure 2.8: Average throughput per user for different pilot sequence selection schemes for the case without user-specific throughput requirement. We set $N = 12$ and $K = 6$.

ACK-based scheme and the random selection scheme. Hence, for a comprehensive performance comparison, we evaluate the performance of the proposed scheme, the ACK-based scheme, and the random selection scheme for the case when there is no user-specific throughput requirement. For the proposed scheme, we set coefficients $\lambda_n(t)$ in the reward function in (2.12) to zero. Hence, the reward function is now given by

$$R(\boldsymbol{s}(t), \boldsymbol{a}(t)) \triangleq \sum_{n \in \mathcal{N}} r_n(t), \ t \in \mathcal{T}. \tag{2.37}$$

The reward function corresponds to the achievable aggregate throughput (i.e., the number of successfully transmitted packets per time slot) of the system. The results for the average throughput per user of all these schemes are shown in Fig. 2.8. We observe that the proposed scheme achieves the highest average throughput per user among all considered schemes. In particular, the average throughput per user of the proposed scheme is 34.6% and 222.1% higher than that of the ACK-based scheme and the random selection scheme, respectively.

Next, we investigate the average throughput of the users for different numbers of pilot sequences, and the results for the users with the highest throughput requirement, i.e., Group I users, are shown in Fig. 2.9. We set the number of users to $2K$. For the optimal scheme and the proposed scheme, the throughput requirements of all users are satisfied for all considered numbers of pilot sequences. Because of the centralized scheduling, for

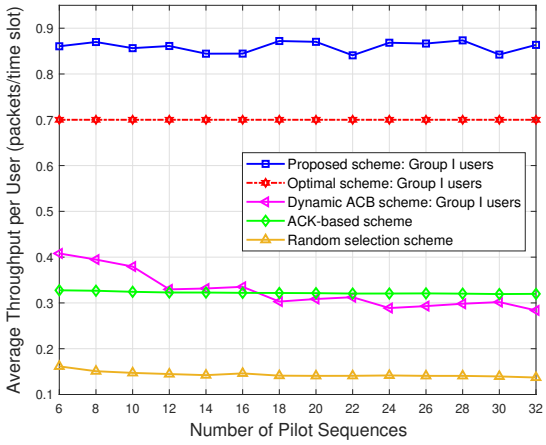


Figure 2.9: Average throughput per user for different pilot sequence selection schemes versus the number of pilot sequences. We show the results of the users in Group I, while the performances of the users in Group II and Group III show a similar behavior.

the optimal scheme, the average throughput of Group I users is exactly the same as the requirement. For the proposed scheme, Group I users tend to consume slightly more resources compared with the optimal scheme and therefore have a higher average throughput. For the ACK-based scheme, the ACB-based scheme, and the random selection scheme, the requirements of Group I users cannot be satisfied due to the relatively high probability of pilot sequence collision. We observe that, in the proposed MA-DRL based scheme, the DRQNs learn the near-optimal pilot sequence selection policies for a given number of pilot sequences. This also demonstrates the advantages of DRL as a model-free learning technique.

2.6 Summary

In this chapter, we proposed an MA-DRL based scheme for maximizing the aggregate throughput of GFMA systems, while taking into account the throughput requirements of the users. We first formulated the aggregate throughput maximization problem, where we accounted for user-specific throughput constraints. To obtain a distributed solution, we

proposed an MA-DRL based scheme under a CTDE framework. By fully exploiting global information, such as the states and actions of all the users, during the centralized training phase, the DRQNs of the users are jointly trained to learn the cooperative pilot sequence selection policies. By combining the factorization technique with MA-DRL, the pilot sequence selection policies learned via centralized training can be efficiently executed by each user in a distributed manner. Our results showed that the proposed scheme can achieve a significantly higher aggregate throughput than the three previously reported schemes. The proposed scheme can also accommodate users with heterogeneous throughput requirements, and therefore has the potential to be deployed in GFMA systems supporting different IoT applications and services.

Chapter 3

Joint User Scheduling, Phase Shift Control, and Beamforming Optimization in Intelligent Reflecting Surface-Aided Systems

3.1 Introduction

IRS introduces additional propagation channels with controllable phase shifts to B5G wireless systems. By properly adjusting the phase shifts of the IRS, the inter-user interference can be mitigated and the SINR at the receiver can be improved. Therefore, IRS is a promising physical layer technique for improving the spectral efficiency of B5G wireless systems. In this chapter, we investigate the performance improvement in terms of aggregate throughput and proportional fairness that can be obtained by properly controlling the DoF of IRS-aided multiuser systems. In particular, we aim to jointly optimize the user scheduling, phase shift control, and beamforming vectors to improve the spectral efficiency of IRS-aided multiuser systems. Optimizing the aforementioned DoF leads to a mixed-integer nonlinear optimization problem with a large number of coupled optimization variables, which can be computationally intensive to solve by using conventional optimization methods. In this chapter, we propose a DUPB algorithm to solve the joint optimization problem by exploiting the learning capability of DRL. In order to tackle the mixture of discrete and continuous variables in the joint optimization problem, we design two DRL

modules for optimizing different types of variables in the proposed DUPB algorithm. These two DRL modules are jointly trained in the proposed learning framework to improve the performance of learned policies. The contributions of this chapter are as follows:

- We formulate a joint optimization problem for maximizing the aggregate throughput and achieving the proportional fairness in IRS-aided systems as a mixed-integer nonlinear optimization problem. We decompose the problem into a subproblem for user scheduling, and a subproblem for joint phase shift control and beamforming optimization.

- We use NCO to learn the stochastic policy for user scheduling in an online manner. We employ two DNN modules, i.e., an encoder module and a decoder module. The encoder module learns the high-dimensional representations of the CSI of the users, and these representations are used by the decoder module to determine the stochastic policy. The DNNs are trained based on RL without requiring the optimal solution of the problem during the training phase.

- We further propose a CL-DDPG algorithm to solve the joint phase shift control and beamforming optimization subproblem. The proposed CL-DDPG algorithm employs an actor-critic method to learn a policy for optimizing the phase shift and beamforming variables. We then propose the DUPB algorithm, which integrates the NCO and CL-DDPG algorithms to jointly optimize the DoF of IRS-aided multiuser systems.

- Simulation results show that the proposed DUPB algorithm achieves an aggregate throughput that is higher than the AO-based algorithms and greedy scheduling. The throughput fairness among the users can be improved by using the proposed DUPB algorithm under the proportional fairness objective. Moreover, the proposed DUPB algorithm requires a lower runtime than the AO-based algorithms when the number of reflecting elements is large. We evaluate the impact of imperfect channel estimation on the achievable throughput of the proposed DUPB algorithm. Our results also show that both modules (i.e., the NCO algorithm for user scheduling and

the CL-DDPG algorithm for phase shift control and beamforming optimization) in the proposed DUPB algorithm contribute to an aggregate throughput that is higher than the AO-based algorithm with FP.

The remainder of this chapter is organized as follows. The system model and problem formulation are presented in Section 3.2. In Section 3.3, we apply the NCO technique to solve the user scheduling subproblem. In Section 3.4, we present the CL-DDPG algorithm and the overall framework of the DUPB algorithm. Simulation results are shown in Section 3.5. The summary of this chapter is presented in Section 3.6.

Notations: In this chapter, we use upper-case and lower-case boldface letters to denote matrices and column vectors, respectively. $\mathbb{C}^{M\times N}$ denotes the set of $M\times N$ complex-valued matrices. $\boldsymbol{A}^H$ denotes the conjugate transpose of matrix $\boldsymbol{A}$. $\mathrm{diag}(\boldsymbol{x})$ returns a diagonal matrix where the diagonal elements are given by the elements of vector $\boldsymbol{x}$.

3.2 System Model

We consider an IRS-aided system with one base station, one IRS, and N users. The base station is equipped with K antennas, while each user equipment (UE) has only one antenna. The set of users is denoted by $\mathcal{N} = \{1, 2, \ldots, N\}$. Time is divided into intervals of equal duration. The time interval $[t, t+1)$ is referred to as time slot t, where $t \in \mathcal{T} = \{1, 2, \ldots\}$. An IRS with L_R reflecting elements is deployed to facilitate the uplink transmission of the users. We assume the base station is allocated with one PRB to serve the users in each time slot $t \in \mathcal{T}$. In each time slot $t \in \mathcal{T}$, the base station schedules M users to perform uplink transmission using one PRB. We use binary control variable $x_n(t) \in \{0, 1\}$ to indicate whether user $n \in \mathcal{N}$ is scheduled for uplink transmission in time slot t. We set $x_n(t) = 1$ if user n is scheduled in time slot t, and $x_n(t) = 0$ otherwise. We have

$$x_n(t) \in \{0, 1\},\ n \in \mathcal{N}, \tag{3.1}$$

$$\sum_{n\in\mathcal{N}} x_n(t) = M. \tag{3.2}$$

We use vector $\boldsymbol{x}(t) = (x_1(t),\, x_2(t), \ldots,\, x_N(t))$ to collect $x_n(t)$, $n \in \mathcal{N}$, in time slot t.

We have the following assumptions on the channel model and channel estimation. First, we assume block fading channels, where the block length is larger than the time period of the PRB. Second, we assume perfect channel estimation at the base station[3].

Let $\boldsymbol{h}_{D,n}(t) \in \mathbb{C}^K$ and $\boldsymbol{h}_{R,n}(t) \in \mathbb{C}^{L_R}$ denote the channel response between user $n \in \mathcal{N}$ and the base station (i.e., the direct channel) and the channel response between user n and the IRS (i.e., the reflecting channel) in time slot $t \in \mathcal{T}$, respectively. The channel response between the IRS and the base station in time slot t is denoted by matrix $\boldsymbol{G}(t) \in \mathbb{C}^{L_R \times K}$. We use matrix $\boldsymbol{\Psi}(t)$ to denote the phase shift matrix of the IRS in time slot t. We have

$$\boldsymbol{\Psi}(t) = \mathrm{diag}(e^{j\psi_1(t)}, \cdots, e^{j\psi_{L_R}(t)}) \in \mathbb{C}^{L_R \times L_R}, \tag{3.3}$$

where $\psi_l(t)$, $l \in \{1, \ldots, L_R\}$ is the phase shift of the l-th reflecting element on the IRS in time slot t. We have the following constraint on the phase shift of the reflecting element

$$\psi_l(t) \in [0,\, 2\pi), \;\; l \in \{1, \ldots, L_R\}. \tag{3.4}$$

We assume the scheduled users always use the maximum transmit power $P^{\max}$ to transmit their signals. The received signal of user $n \in \mathcal{N}$ at the base station in time slot t is given by

$$\boldsymbol{y}_n(t) = \; x_n(t)\sqrt{P^{\max}}\left(\boldsymbol{h}_{D,n}(t)\, s_n(t) + \boldsymbol{G}^H(t)\boldsymbol{\Psi}(t)\boldsymbol{h}_{R,n}(t) s_n(t)\right) + \boldsymbol{f}_n(t) + \boldsymbol{\varrho}, \tag{3.5}$$

where $s_n(t) \in \mathbb{C}$ is the symbol of user n in time slot t with unit power, $\boldsymbol{\varrho}$ is the complex Gaussian noise with zero mean and variance σ^2, and $\boldsymbol{f}_n(t)$ is the interference from the

[3]Note that perfect channel information may be difficult to obtain in practical systems. The potential impact of imperfect channel estimation on the studied problem is evaluated in Section 3.5.4.

remaining users to user n in time slot t. $\boldsymbol{f}_n(t)$ is given by

$$\boldsymbol{f}_n(t) = \sum_{j \in \mathcal{N} \setminus \{n\}} x_j(t) \sqrt{P^{\max}} \left(\boldsymbol{h}_{D,j}(t)\, s_j(t) + \boldsymbol{G}^H(t) \boldsymbol{\Psi}(t) \boldsymbol{h}_{R,j}(t) s_j(t) \right). \tag{3.6}$$

The SINR of user n in time slot t is given by:

$$\Gamma_n(\boldsymbol{x}(t), \boldsymbol{\Psi}(t), \boldsymbol{b}_n(t)) = \frac{x_n(t) P^{\max} \left| \boldsymbol{b}_n^H(t) \boldsymbol{h}_{D,n}(t) + \boldsymbol{b}_n^H(t) \boldsymbol{G}^H(t) \boldsymbol{\Psi}(t) \boldsymbol{h}_{R,n}(t) \right|^2}{\sum\limits_{j \in \mathcal{N} \setminus \{n\}} x_j(t) P^{\max} \left| \boldsymbol{b}_n^H(t) \boldsymbol{h}_{D,j}(t) + \boldsymbol{b}_n^H(t) \boldsymbol{G}^H(t) \boldsymbol{\Psi}(t) \boldsymbol{h}_{R,j}(t) \right|^2 + \sigma^2 \left\| \boldsymbol{b}_n(t) \right\|_2^2}. \tag{3.7}$$

In equation (3.7), $\boldsymbol{b}_n(t) \in \mathbb{C}^K$ is the beamforming vector of user n's signal in time slot t at the base station. We use $\boldsymbol{b}(t) = (\boldsymbol{b}_1(t), \ldots, \boldsymbol{b}_N(t))$ to collect the beamforming vectors of all users in time slot t. The achievable throughput (bits/(time slot)/Hz) of user n can be determined as follows:

$$R_n(\boldsymbol{x}(t), \boldsymbol{\Psi}(t), \boldsymbol{b}_n(t)) = \log_2(1 + \Gamma_n(\boldsymbol{x}(t), \boldsymbol{\Psi}(t), \boldsymbol{b}_n(t))). \tag{3.8}$$

When proportional fairness is used as the objective, we determine the time average of the achievable throughput of user n up to time slot t based on the following moving average [118]:

$$\overline{R}_n(t) = \begin{cases} 1, & \text{if } t = 1, \\ \frac{1}{t-1} \sum_{\tau=1}^{t-1} R_n(\boldsymbol{x}(\tau), \boldsymbol{\Psi}(\tau), \boldsymbol{b}_n(\tau)), & \text{if } t = \{2, 3, \ldots, T_c\}, \\ (1 - \frac{1}{T_c}) \overline{R}_n(t-1) + \frac{1}{T_c} R_n(\boldsymbol{x}(t-1), \boldsymbol{\Psi}(t-1), \boldsymbol{b}_n(t-1)), & \text{otherwise}, \end{cases} \tag{3.9}$$

where T_c is the moving window size. We set $\overline{R}_n(1) = 1$, $n \in \mathcal{N}$, such that all users are considered to have equal average throughput in the first time slot.

In this chapter, we consider the joint optimization problem of user scheduling, phase shift control, and beamforming in IRS-aided systems. The optimization problem in time

slot $t \in \mathcal{T}$ is formulated as follows:

$$\begin{aligned} \underset{\boldsymbol{x}(t),\, \boldsymbol{\Psi}(t),\, \boldsymbol{b}(t)}{\text{maximize}} \quad & \sum_{n \in \mathcal{N}} w_n(t)\, R_n(\boldsymbol{x}(t), \boldsymbol{\Psi}(t), \boldsymbol{b}_n(t)) \\ \text{subject to} \quad & \text{constraints } (3.1), (3.2), (3.4). \end{aligned} \tag{3.10}$$

The optimization problem with the maximum aggregate throughput objective can be obtained by setting $w_n(t) = 1$ for all $n \in \mathcal{N}$ in problem (3.10). For the proportional fairness objective, we set $w_n(t) = \frac{1}{\bar{R}_n(t)}$ for $n \in \mathcal{N}$ [10, 118]. The optimal solution of problem (3.10) is difficult to obtain since the problem is nonconvex due to the weighted fractional objective function and the phase shift constraint (3.4). Moreover, problem (3.10) has binary control variables $\boldsymbol{x}(t)$ and the optimization variables $\boldsymbol{x}(t)$, $\boldsymbol{\Psi}(t)$, $\boldsymbol{b}(t)$ are coupled.

In time slot $t \in \mathcal{T}$, we can decompose problem (3.10) into a user scheduling subproblem and a subproblem for joint phase shift control and beamforming optimization. In particular, given $\boldsymbol{\Psi}(t)$ and $\boldsymbol{b}(t)$, the subproblem for user scheduling in time slot t is given by

$$\begin{aligned} \underset{\boldsymbol{x}(t)}{\text{maximize}} \quad & \sum_{n \in \mathcal{N}} w_n(t)\, R_n(\boldsymbol{x}(t)) \\ \text{subject to} \quad & \text{constraints (3.1) and (3.2).} \end{aligned} \tag{3.11}$$

Subproblem (3.11) is a combinatorial optimization problem with a total of $\binom{N}{M}$ feasible user scheduling selections in each time slot. Given the user scheduling vector $\boldsymbol{x}(t)$, the joint subproblem for phase shift control and beamforming optimization in time slot t is as follows:

$$\begin{aligned} \underset{\boldsymbol{\Psi}(t),\boldsymbol{b}(t)}{\text{maximize}} \quad & \sum_{n \in \mathcal{N}} w_n(t)\, R_n(\boldsymbol{\Psi}(t), \boldsymbol{b}(t)) \\ \text{subject to} \quad & \text{constraint (3.4).} \end{aligned} \tag{3.12}$$

Subproblem (3.12) is a nonconvex optimization problem with a multi-ratio fractional objective function. A suboptimal solution of problem (3.10) can be obtained by solving subproblem (3.12) for all $\binom{N}{M}$ user scheduling selections. This approach is computationally expensive, especially when problem (3.10) is required to be solved in each time slot and

the number of users is large. In the following sections, we propose a DUPB algorithm to solve problem (3.10) with lower computational complexity.

3.3 NCO-based Algorithm for User Scheduling

The first part of the proposed DUPB algorithm is an NCO-based algorithm for determining the user scheduling in an online manner. In the proposed NCO-based algorithm, we solve the user scheduling subproblem using an RL framework. A stochastic user scheduling policy is learned by the DNNs with attention mechanism. For notational simplicity, we drop the time index t in the subsequent sections.

3.3.1 MDP Formulation and Stochastic Policy

We first model the decision process for solving the user scheduling subproblem (3.11) using an MDP. The state, action, and reward of the MDP are defined as follows.

State

We first concatenate the rows of channel gain matrix $\boldsymbol{G}$ to obtain a vector of size KL_R, which is denoted by $\boldsymbol{g}$. Then, we use vector $\boldsymbol{v}_n$ to collect the CSI of user n, along with the weight w_n. In particular, vector $\boldsymbol{v}_n$ is given by

$$\boldsymbol{v}_n = [\boldsymbol{h}_{D,n},\, \boldsymbol{h}_{R,n},\, \boldsymbol{g},\, w_n],\ n \in \mathcal{N}, \tag{3.13}$$

where $[\cdot,\, \cdot,\, \cdot]$ denotes the concatenation operator. The size of vector $\boldsymbol{v}_n$ is $K+L_R+KL_R+1$.

State $\mathcal{S}$ consists of the CSI and the weights w_n of all N users. It is given by

$$\mathcal{S} = \{\boldsymbol{v}_1,\, \ldots,\, \boldsymbol{v}_N\}. \tag{3.14}$$

State $\mathcal{S}$ is a *set* that collects the global information [119, 120]. In the remainder of this chapter, we use the terms state $\mathcal{S}$ and set $\mathcal{S}$ interchangeably.

Action

The action is to schedule M users to perform uplink transmission. Given state $\mathcal{S}$, the action is equivalent to finding a subset $\mathcal{U}$ which consists of M different vectors (or elements) from $\mathcal{S}$. That is,

$$\mathcal{U} = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_M\}, \tag{3.15}$$

where $\boldsymbol{u}_l \in \mathcal{S}$ and $\boldsymbol{u}_l \neq \boldsymbol{u}_{l'}$, $l, l' \in \{1, \ldots, M\}$, and $l \neq l'$. Given subset $\mathcal{U}$, we obtain the corresponding user scheduling vector $\boldsymbol{x}^\star$ by setting $x_n = 1$ if $\boldsymbol{v}_n \in \mathcal{U}$. If $\boldsymbol{v}_n \notin \mathcal{U}$, then we set $x_n = 0$.

Reward

After determining subset $\mathcal{U}$, the reward $r(\mathcal{U}) \in \mathbb{R}$ is determined by solving the joint phase shift and beamforming optimization problem in (3.12) for fixed $\boldsymbol{x}^\star$. We propose a CL-DDPG algorithm for solving problem (3.12). The details of the CL-DDPG algorithm will be presented in Section 3.4. We denote the phase shift matrix and beamforming vector that are obtained by solving problem (3.12) as $\boldsymbol{\Psi}^\star$ and $\boldsymbol{b}^\star$, respectively. Then, the reward $r(\mathcal{U})$ is determined as follows:

$$r(\mathcal{U}) = \sum_{n \in \mathcal{N}} w_n \, R_n(\boldsymbol{x}^\star, \boldsymbol{\Psi}^\star, \boldsymbol{b}_n^\star). \tag{3.16}$$

Let $p(\cdot)$ denote the probability of event $(\cdot)$. The stochastic policy for choosing the action, i.e., scheduling M users, can be determined by the conditional probability of selecting a particular subset $\mathcal{U}$ given state $\mathcal{S}$, i.e., $p(\mathcal{U} \,|\, \mathcal{S})$. Using the chain rule, this probability can be factorized as follows [53]:

$$p(\mathcal{U} \,|\, \mathcal{S}) = \prod_{l=1}^{M} p(\boldsymbol{u}_l \,|\, \mathcal{S}, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_{l-1}). \tag{3.17}$$

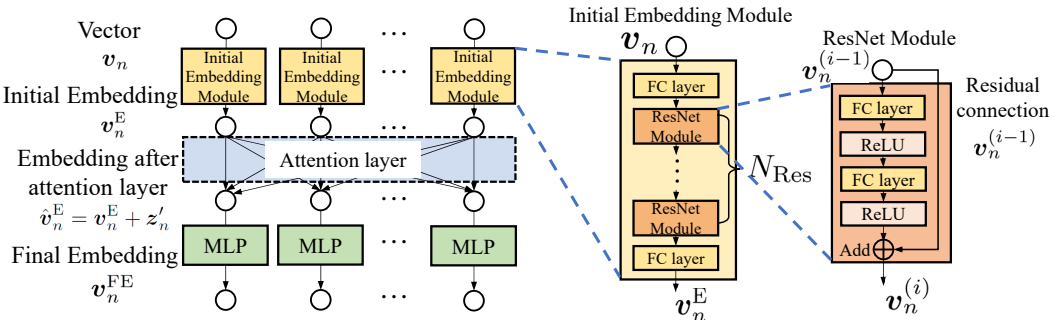


Figure 3.1: The network structure of the encoder module. The encoder module learns the embeddings of the input vectors with an initial embedding module, an attention layer, and an MLP module.

To obtain the optimal user scheduling, we aim to find the optimal stochastic policy that leads to the maximum expected reward. In particular, the optimal stochastic policy is given by

$$p^{\star}(\mathcal{U} \,|\, \mathcal{S}) = \arg\max \; \mathbb{E}_{\mathcal{U}' \sim p(\mathcal{U} \,|\, \mathcal{S})} \big[r(\mathcal{U}') \big]. \tag{3.18}$$

To this end, we propose an NCO-based algorithm to learn the optimal stochastic policy in (3.18). The details of the proposed algorithm are presented in the following subsections.

3.3.2 Encoder Module

We denote the learnable parameters of the DNN modules that are employed to learn the stochastic user scheduling policy as $\boldsymbol{\Phi}$. The parameterized stochastic user scheduling policy is denoted as $p_{\boldsymbol{\Phi}}(\mathcal{U} \,|\, \mathcal{S})$. In order to learning the policy $p_{\boldsymbol{\Phi}}(\mathcal{U} \,|\, \mathcal{S})$, we use an *encoder* DNN module to learn the underlying structures and abstraction of the information in set $\mathcal{S}$, which are referred to as the *embedding* of the users [53]. The DNN structure for the encoder module is shown in Fig. 3.1. For vector $\boldsymbol{v}_n \in \mathcal{S}$, we use an *initial embedding module* to obtain its embedding $\boldsymbol{v}_n^{\mathrm{E}}$. As shown on the right-hand side of Fig. 3.1, the initial embedding module consists of FC layers and residual network (ResNet) modules. N_{Res} ResNet modules are stacked in order to increase the depth of the initial embedding module and provide a higher representation capacity. Each ResNet module has two FC

layers, two ReLU layers, and one residual connection. The residual connection is added to overcome the vanishing gradient issue due to the depth of the network [121], and to facilitate the learning by eliminating the singularities of FC layers [122].

In the initial embedding module, the first FC layer projects the input vector $\boldsymbol{v}_n$ to a d_{Res}-dimensional space, where d_{Res} is a constant. The output of the first FC layer is given by:

$$\boldsymbol{v}_n^{(0)} = \boldsymbol{W}^{\text{FC1}}\, \boldsymbol{v}_n + \boldsymbol{b}^{\text{FC1}}, \tag{3.19}$$

where the weights $\boldsymbol{W}^{\text{FC1}} \in \mathbb{R}^{d_{\text{Res}} \times d_i}$ and biases $\boldsymbol{b}^{\text{E}} \in \mathbb{R}^{d_{\text{Res}}}$ are learnable parameters, and $d_i = K + L_R + KL_R + 1$ is the size of vector $\boldsymbol{v}_n$. The vector $\boldsymbol{v}_n^{(0)}$ is then fed into the subsequent ResNet modules. We denote $\boldsymbol{W}_1^{(i)}$ and $\boldsymbol{W}_2^{(i)} \in \mathbb{R}^{d_{\text{Res}} \times d_{\text{Res}}}$ as the weights, and $\boldsymbol{b}_1^{(i)}$ and $\boldsymbol{b}_2^{(i)} \in \mathbb{R}^{d_{\text{Res}}}$ as the biases of the first and second FC layers in the i-th ResNet module, respectively, for $i = 1, \ldots, N_{\text{Res}}$. Given vector $\boldsymbol{v}_n^{(0)},\ n \in \mathcal{N}$, the output of the i-th ResNet module is given by:

$$\boldsymbol{v}_n^{(i)} = \text{ ReLU}\Big(\boldsymbol{W}_2^{(i)}\big(\text{ReLU}(\boldsymbol{W}_1^{(i)}\boldsymbol{v}_n^{(i-1)} + \boldsymbol{b}_1^{(i)})\big) + \boldsymbol{b}_2^{(i)}\Big) + \boldsymbol{v}_n^{(i-1)},\ \ i = 1, \ldots, N_{\text{Res}}, \tag{3.20}$$

where $\text{ReLU}(\cdot)$ denotes the ReLU activation function. The output of the N_{Res}-th (i.e., the last) ResNet module is passed through another FC layer with weights $\boldsymbol{W}^{\text{FC2}} \in \mathbb{R}^{d_h \times d_{\text{Res}}}$ and biases $\boldsymbol{b}^{\text{FC2}} \in \mathbb{R}^{d_h}$ to obtain the initial embedding $\boldsymbol{v}_n^{\text{E}}$. We have $\boldsymbol{v}_n^{\text{E}} = \boldsymbol{W}^{\text{FC2}}\, \boldsymbol{v}_n^{(N_{\text{Res}})} + \boldsymbol{b}^{\text{FC2}}$. The learnable parameters of the initial embedding module $\boldsymbol{\Phi}^{\text{IEM}}$ are given by

$$\boldsymbol{\Phi}^{\text{IEM}} = \Big(\boldsymbol{W}^{\text{FC1}}, \boldsymbol{W}_1^{(i)}, \boldsymbol{W}_2^{(i)}, \boldsymbol{W}^{\text{FC2}}, \boldsymbol{b}^{\text{FC1}}, \boldsymbol{b}_1^{(i)}, \boldsymbol{b}_2^{(i)}, \boldsymbol{b}^{\text{FC2}}\Big),\ i = 1, \ldots, N_{\text{Res}}. \tag{3.21}$$

After obtaining the initial embedding, we use an *attention mechanism* [54] to capture the user interference and the combinatorial structure of the user scheduling subproblem. The attention mechanism can be considered as an information exchange process between the embeddings of vectors. By exchanging the information, the embedding $\boldsymbol{v}_n^{\text{E}}$ not only provides the multidimensional representations of the CSI and weight w_n of user n, but

also shows how it relates to the other users in the systems. We generate three additional vectors, namely, *key* $\boldsymbol{k}_n$, *query* $\boldsymbol{q}_n$, and *value* $\boldsymbol{z}_n$, for each vector based on the embedding $\boldsymbol{v}_n^{\mathrm{E}}$ as follows:

$$\boldsymbol{k}_n = \boldsymbol{W}_{\mathrm{en}}^{\mathrm{K}}\, \boldsymbol{v}_n^{\mathrm{E}}, \;\; \boldsymbol{q}_n = \boldsymbol{W}_{\mathrm{en}}^{\mathrm{Q}}\, \boldsymbol{v}_n^{\mathrm{E}}, \;\; \boldsymbol{z}_n = \boldsymbol{W}_{\mathrm{en}}^{\mathrm{Z}}\, \boldsymbol{v}_n^{\mathrm{E}}, \tag{3.22}$$

where matrices $\boldsymbol{W}_{\mathrm{en}}^{\mathrm{K}}$, $\boldsymbol{W}_{\mathrm{en}}^{\mathrm{Q}} \in \mathbb{R}^{d_k \times d_h}$, and $\boldsymbol{W}_{\mathrm{en}}^{\mathrm{Z}} \in \mathbb{R}^{d_z \times d_h}$ are learnable parameters, whereas d_k and d_z are constants. Using attention mechanism, the embedding of a vector may receive values from the embeddings of other vectors. We compute the *compatibility* $\delta_{n,j} \in \mathbb{R}$ of the two vectors as $\delta_{n,j} = \frac{\boldsymbol{q}_n^T \boldsymbol{k}_j}{\sqrt{d_k}}$. The attention weights $a_{n,j} \in [0,\, 1]$ can be obtained using the softmax function as $a_{n,j} = \frac{e^{\delta_{n,j}}}{\sum_{j' \in \mathcal{N}} e^{\delta_{n,j'}}}$. Let $\boldsymbol{z}'_n$ denote the value that embeddings $\boldsymbol{v}_n^{\mathrm{E}}$ received from the embeddings of other vectors. We have

$$\boldsymbol{z}'_n = \sum_{j \in \mathcal{N}} a_{n,j} \boldsymbol{z}_j. \tag{3.23}$$

We construct a new embedding $\hat{\boldsymbol{v}}_n^{\mathrm{E}}$ of vector $\boldsymbol{v}_n$ by combining $\boldsymbol{v}_n^{\mathrm{E}}$ with the received $\boldsymbol{z}'_n$, which is $\hat{\boldsymbol{v}}_n^{\mathrm{E}} = \boldsymbol{v}_n^{\mathrm{E}} + \boldsymbol{z}'_n$. The final embedding of vector $\boldsymbol{v}_n$, which is denoted by $\boldsymbol{v}_n^{\mathrm{FE}}$, is obtained by passing the embedding $\hat{\boldsymbol{v}}_n^{\mathrm{E}}$ through an MLP module. The dimension of the output layer of the MLP module is d_h. We denote the learnable parameters of MLP modules as $\boldsymbol{\Phi}^{\mathrm{MLP}}$. The parameters in the encoder module $\boldsymbol{\Phi}_{\mathrm{en}}$ are given by $\boldsymbol{\Phi}_{\mathrm{en}} = (\boldsymbol{\Phi}^{\mathrm{IEM}}, \boldsymbol{W}_{\mathrm{en}}^{\mathrm{K}}, \boldsymbol{W}_{\mathrm{en}}^{\mathrm{Q}}, \boldsymbol{W}_{\mathrm{en}}^{\mathrm{Z}}, \boldsymbol{\Phi}^{\mathrm{MLP}})$. The outputs of the encoder module are the final embeddings of the users.

3.3.3 NCO-based Algorithm: Context Embedding and Decoder

The network structure of the decoder module is shown in Fig. 3.2. In the decoder module, we first combine the final embeddings of all vectors in state $\mathcal{S}$ to obtain the aggregate embedding using the weighted summation $\boldsymbol{v}_{\mathrm{G}}^{\mathrm{E}} = \sum_{n \in \mathcal{N}} \eta_n \boldsymbol{v}_n^{\mathrm{FE}}$, where $\eta_n \in (0,\, 1)$ is the weight of the final embedding of user n. Ideally, the weight η_n should be adjusted flexibly to reflect the contribution of embedding $\boldsymbol{v}_n^{\mathrm{E}}$ (i.e., user n) to the reward. Inspired by the feature aggregation in deep multiple instance learning [120, 123], we employ an embedding

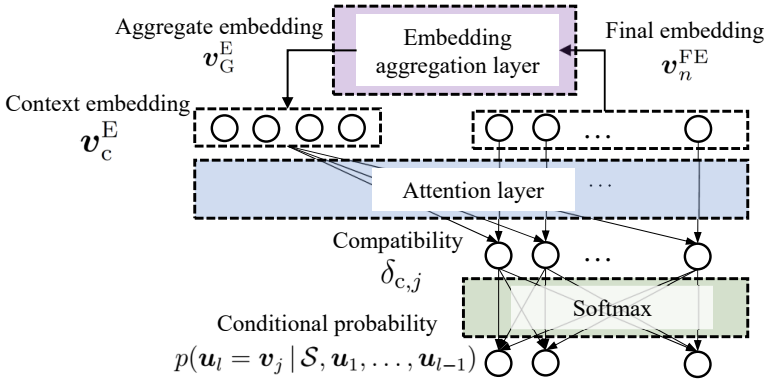


Figure 3.2: The network structure of the decoder module. The decoder module generates the conditional probabilities based on the final embeddings provided by the encoder module.

aggregation layer to determine the weight as $\eta_n = \frac{e^{\beta_n}}{\sum_{n' \in \mathcal{N}} e^{\beta_{n'}}}$, where $\beta_n = \boldsymbol{\nu}^T \tanh(\boldsymbol{W}^{\mathrm{A}} \boldsymbol{v}_n^{\mathrm{FE}})$ [123]. Vector $\boldsymbol{\nu} \in \mathbb{R}^{d_a}$ and matrix $\boldsymbol{W}^{\mathrm{A}} \in \mathbb{R}^{d_a \times d_h}$ are learnable parameters of the embedding aggregation layer, and d_a is a constant. Using the aggregate embedding $\boldsymbol{v}_{\mathrm{G}}^{\mathrm{E}}$, the decoder module constructs a *context embedding* for generating the conditional probability $p_{\boldsymbol{\Phi}}(\boldsymbol{u}_l \,|\, \mathcal{S}, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_{l-1})$, which is given by

$$\boldsymbol{v}_{\mathrm{c}}^{\mathrm{E}} = \begin{cases} [\boldsymbol{v}_{\mathrm{G}}^{\mathrm{E}},\, \boldsymbol{v}_{\boldsymbol{u}_0}^{\mathrm{FE}},\, \boldsymbol{w},\, M], & \text{if } l = 1, \\ [\boldsymbol{v}_{\mathrm{G}}^{\mathrm{E}},\, \boldsymbol{v}_{\mathcal{U}_{l-1}}^{\mathrm{E}},\, \boldsymbol{w},\, M - l + 1\,], & \text{if } l > 1, \end{cases} \tag{3.24}$$

where $\boldsymbol{v}_{\mathcal{U}_{l-1}}^{\mathrm{E}}$ is the aggregate embedding of the previously selected vectors. It is given by

$$\boldsymbol{v}_{\mathcal{U}_{l-1}}^{\mathrm{E}} = \frac{1}{l-1} \sum_{n \in \{n' \,|\, \boldsymbol{v}_{n'}^{\mathrm{E}} \in \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{l-1}\}\}} \boldsymbol{v}_n^{\mathrm{FE}}, \tag{3.25}$$

and vector $\boldsymbol{w} = (w_1, \ldots, w_N)$. $\boldsymbol{v}_{\boldsymbol{u}_0}^{\mathrm{FE}}$ serves as a placeholder to maintain the constant size of $\boldsymbol{v}_{\mathrm{c}}^{\mathrm{E}}$ when the base station determines the first scheduled user. The last element in (3.24), i.e., $M - l + 1$, is the number of remaining vectors that can be added into the subset. The size of the context embedding is given by $d_{\mathrm{c}} = 2d_h + N + 1$.

To obtain the stochastic policy, we compute the compatibility of the context embedding

in (3.24) with the final embedding of each of the remaining users that can potentially be scheduled. We obtain the query of the context embedding, the keys and values of the vectors as follows:

$$\boldsymbol{q}_{\mathrm{c}} = \boldsymbol{W}_{\mathrm{de}}^{\mathrm{Q}} \boldsymbol{v}_{\mathrm{c}}^{\mathrm{E}}, \quad \boldsymbol{k}_n = \boldsymbol{W}_{\mathrm{de}}^{\mathrm{K}} \boldsymbol{v}_n^{\mathrm{FE}}, \quad \boldsymbol{z}_n = \boldsymbol{W}_{\mathrm{de}}^{\mathrm{Z}} \boldsymbol{v}_n^{\mathrm{FE}}, \tag{3.26}$$

where matrices $\boldsymbol{W}_{\mathrm{de}}^{\mathrm{Q}} \in \mathbb{R}^{d_h \times d_c}$, $\boldsymbol{W}_{\mathrm{de}}^{\mathrm{K}} \in \mathbb{R}^{d_h \times d_k}$, and $\boldsymbol{W}_{\mathrm{de}}^{\mathrm{Z}} \in \mathbb{R}^{d_h \times d_z}$ project the context embedding and the final embeddings of the vectors to d_h dimensions. The compatibilities of context embedding with the final embedding of the remaining users can be determined by

$$\delta_{\mathrm{c},j} = \begin{cases} \delta^{\max} \tanh\left(\frac{\boldsymbol{q}_{\mathrm{c}}^T \boldsymbol{k}_j}{\sqrt{d_h}}\right), & \text{if } \boldsymbol{v}_j \text{ has not been selected,} \\ -\infty, & \text{otherwise,} \end{cases} \tag{3.27}$$

where $\delta^{\max}$ is a constant. The conditional probability for selecting user j as the l-th scheduled user in the subset is given by

$$p(\boldsymbol{u}_l = \boldsymbol{v}_j \,|\, \mathcal{S}, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_{l-1}) = \frac{e^{\delta_{\mathrm{c},j}}}{\sum_{j' \in \mathcal{N}} e^{\delta_{\mathrm{c},j'}}}. \tag{3.28}$$

With the conditional probability in (3.28), the stochastic policy $p_{\boldsymbol{\Phi}}(\mathcal{U} \,|\, \mathcal{S})$ can be obtained based on (3.17). The learnable parameters in the decoder module is given by $\boldsymbol{\Phi}_{\mathrm{de}} = (\boldsymbol{\nu}, \boldsymbol{W}^{\mathrm{A}}, \boldsymbol{W}_{\mathrm{de}}^{\mathrm{K}}, \boldsymbol{W}_{\mathrm{de}}^{\mathrm{Q}}, \boldsymbol{W}_{\mathrm{de}}^{\mathrm{Z}})$. The overall learnable parameters are collected as $\boldsymbol{\Phi} = (\boldsymbol{\Phi}_{\mathrm{en}}, \boldsymbol{\Phi}_{\mathrm{de}})$.

3.3.4 Learning Algorithm

The stochastic policy generated by the encoder and decoder modules is characterized by the learnable parameters $\boldsymbol{\Phi}$, i.e., $p_{\boldsymbol{\Phi}}(\mathcal{U} \,|\, \mathcal{S})$. We aim to find the learnable parameters $\boldsymbol{\Phi}$, such that the expected reward of policy $p_{\boldsymbol{\Phi}}(\mathcal{U} \,|\, \mathcal{S})$ is maximized. This leads to the following

optimization problem:

$$\min_{\mathbf{\Phi}} \ \mathcal{L}(\mathbf{\Phi} \,|\, \mathcal{S}) \triangleq \mathbb{E}_{\mathcal{U}' \sim p_{\mathbf{\Phi}}(\mathcal{U} \,|\, \mathcal{S})} \big[- r(\mathcal{U}') \big], \tag{3.29}$$

where $\mathcal{L}(\mathbf{\Phi} \,|\, \mathcal{S})$ is referred to as the *loss function*. We use the REINFORCE algorithm [55] to perform gradient descent and optimize $\mathbf{\Phi}$. The gradient estimation is given by

$$\nabla \mathcal{L}(\mathbf{\Phi} \,|\, \mathcal{S}) = \mathbb{E}_{\mathcal{U}' \sim p_{\mathbf{\Phi}}(\mathcal{U} \,|\, \mathcal{S})} \big[(b(\mathcal{S}) - r(\mathcal{U}')) \, \nabla \log p_{\mathbf{\Phi}}(\mathcal{U}' \,|\, \mathcal{S}) \big], \tag{3.30}$$

where $b(\mathcal{S})$ is the reward obtained by a baseline policy on set $\mathcal{S}$. We use the best policy learned by the DNN modules so far during the training phase to serve as the baseline. We compare the latest learned policy $p_{\mathbf{\Phi}}(\mathcal{U} \,|\, \mathcal{S})$ with the baseline every N_e training iterations, where N_e is a positive integer. The baseline will be replaced by the latest policy $p_{\mathbf{\Phi}}(\mathcal{U} \,|\, \mathcal{S})$ if $p_{\mathbf{\Phi}}(\mathcal{U} \,|\, \mathcal{S})$ achieves a higher expected reward compared with the previous baseline. We refer to the N_e training iterations between two baseline updates as a *training episode*. With the gradient in (3.30), we use the Adam optimizer [124] to update the learnable parameters $\mathbf{\Phi}$.

The training algorithm in each training episode is shown in **Algorithm 2**. In each training iteration, we train the DNN module using one minibatch, which includes B different channel realizations of the users. For each of the channel realizations in the minibatch, we determine the corresponding set $\mathcal{S}'$. We use set $\mathcal{S}_B$ to collect all the B vector sets in the minibatch. That is, set $\mathcal{S}'$ is an element in set $\mathcal{S}_B$. The loss function is first determined for each $\mathcal{S}' \in \mathcal{S}_B$, and is then averaged over the whole minibatch. After each episode, the performance of the learned policy and the baseline policy is evaluated over B_e new channel realizations.

Algorithm 2 NCO-based User Scheduling Algorithm: Training Phase

1: **for** Training iteration counter $= 1, \ldots, N_e$ **do**
2: Obtain a minibatch consisting of B channel realizations, and determine the corresponding vector set $\mathcal{S}_B$.
3: **for** each $\mathcal{S}' \in \mathcal{S}_B$ **do**
4: Feed the vectors in $\mathcal{S}'$ into the DNN modules and determine the subset $\mathcal{U}$.
5: Determine the reward $r(\mathcal{U})$ by solving subproblem (3.12).
6: Given state $\mathcal{S}'$, determine the reward of the baseline policy $b(\mathcal{S}')$.
7: Determine the loss function as $\mathcal{L}(\boldsymbol{\Phi} \,|\, \mathcal{S}') \leftarrow b(\mathcal{S}') - r(\mathcal{U})$.
8: **end for**
9: Determine the aggregate loss over the minibatch as $\mathcal{L}(\boldsymbol{\Phi} \,|\, \mathcal{S}_B) \leftarrow \frac{1}{B} \sum_{\mathcal{S}' \in \mathcal{S}_B} \mathcal{L}(\boldsymbol{\Phi} \,|\, \mathcal{S}')$.
10: Determine the gradient based on (3.30), and update $\boldsymbol{\Phi}$ by solving problem (3.29) using Adam optimizer [124].
11: **end for**
12: Update the baseline policy if the current learned policy outperforms the previous baseline.
13: **return** The updated learnable parameters $\boldsymbol{\Phi}$.

3.3.5 Online Algorithm

Algorithm 3 shows our proposed NCO-based algorithm, which determines the user scheduling given the online information in time slot $t \in \mathcal{T}$. In Line 1, we first determine the weights of all users in time slot t. In Line 2, we obtain state $\mathcal{S}(t)$ based on the CSI and the weights of the users. We then feed the vectors in $\mathcal{S}(t)$ into the pre-trained DNN modules, and determine the subset $\mathcal{U}(t)$ based on the learned policy $p_{\boldsymbol{\Phi}}(\mathcal{U}(t) \,|\, \mathcal{S}(t))$. With subset $\mathcal{U}(t)$, we can obtain the user scheduling vector $\boldsymbol{x}^\star(t)$. Now, with the given $\boldsymbol{x}^\star(t)$, we jointly optimize the phase shift matrix and beamforming vectors, and obtain $\boldsymbol{\Psi}^\star(t)$ and $\boldsymbol{b}^\star(t)$ using the CL-DDPG algorithm. The details of the proposed CL-DDPG algorithm will be presented in the next section.

Algorithm 3 Online Execution of the Proposed NCO-based User Scheduling Algorithm in Time Slot $t \in \mathcal{T}$

1: Determine the weights $w_n(t)$, $n \in \mathcal{N}$, based on (3.9).
2: Obtain vectors $\boldsymbol{v}_n(t)$, $n \in \mathcal{N}$ and state $\mathcal{S}(t)$ based on the channel information and the weights $w_n(t)$, $n \in \mathcal{N}$ in time slot t.
3: Feed the vectors in $\mathcal{S}(t)$ into the DNN modules, and determine the subset $\mathcal{U}(t)$ based on the learned policy $p_{\boldsymbol{\Phi}}(\mathcal{U}(t) \,|\, \mathcal{S}(t))$.
4: Obtain the corresponding user scheduling selection $\boldsymbol{x}^{\star}(t)$ in time slot t from $\mathcal{U}(t)$.

3.4 CL-DDPG Algorithm for Phase Shift Control and Beamforming Optimization

Given the user scheduling vector, we propose a CL-DDPG algorithm to solve the joint phase shift control and beamforming optimization subproblem.

3.4.1 CL-DDPG Algorithm: State, Action, and Reward

The joint phase shift control and beamforming optimization in an arbitrary time slot $t \in \mathcal{T}$ can be formulated as a sequential decision process with a maximum of $I_{\max}$ decision steps, as shown in Fig. 3.3. Since the CSI and weights of the users are constant within a particular time slot, we drop the time index t for notation simplicity. We define the state in the i-th decision step as follows:

$$\boldsymbol{s}^{\mathrm{PB}}(i) = [\boldsymbol{s}_1^{\mathrm{PB}}(i), \boldsymbol{s}_2^{\mathrm{PB}}(i), \ldots, \boldsymbol{s}_M^{\mathrm{PB}}(i), \boldsymbol{g}\,], \quad i = 1, 2, \ldots, I_{\max}, \tag{3.31}$$

where $\boldsymbol{s}_n^{\mathrm{PB}}(i) = [\boldsymbol{h}_{D,n},\, \boldsymbol{h}_{R,n},\, w_n R_n(\boldsymbol{a}^{\mathrm{PB}}(i-1))]$, $n \in \mathcal{M}$, $\boldsymbol{a}^{\mathrm{PB}}(i-1)$ is the action taken in the previous decision step, and $\boldsymbol{a}^{\mathrm{PB}}(0)$ is the initialized action. Recall that vector $\boldsymbol{g}$ in (3.31) is obtained by concatenating the rows of channel gain matrix $\boldsymbol{G}$. The action vector is defined as

$$\boldsymbol{a}^{\mathrm{PB}}(i) = [\boldsymbol{b}(i), \boldsymbol{\psi}(i)], \quad i = 1, 2, \ldots, I_{\max}, \tag{3.32}$$

where $\boldsymbol{\psi}(i) = (\psi_1(i), \ldots, \psi_{L_R}(i))$.

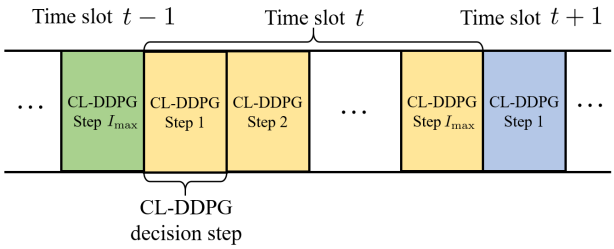


Figure 3.3: Illustration of the sequential decision process of the CL-DDPG algorithm for joint phase shift control and beamforming optimization.

Remark 3.1: Since the elements in action vector $\boldsymbol{a}^{\mathrm{PB}}(i)$ are continuous variables, we choose DDPG [17] as the foundation of our algorithm design. However, the vanilla DDPG algorithm suffers from exploration inefficiency when the dimensionality of the state and action space is large (i.e., the curse of dimensionality as we discussed in Section 1.4). For the problem studied in this chapter, as the dimensionality of the state and action space increases substantially with the number of reflecting elements L_R, the performance of the vanilla DDPG algorithm may degrade. To address these challenges and improve the efficiency of the vanilla DDPG algorithm, we use CL [29, 30] to design the reward function of the underlying MDP. Our reward function design can facilitate the policy learning by providing the learning agent with the knowledge of the structure of the joint problem.

CL is a technique for modifying the system transition history observed by the learning agent such that the performance of the learning algorithm can be improved [29]. The rationale of CL is to begin the learning process with a simple objective and gradually change the objective towards a more difficult one, with the intuition that the knowledge learned with the simple objective can facilitate the learning with a more difficult objective [30, 125]. In our proposed CL-DDPG algorithm, during the early stage of the learning process, we choose an objective of reducing the *distance* between the suboptimal solution. We employ the following average distance measurement:

$$\frac{||\boldsymbol{a}^{\mathrm{PB}}(i) - \boldsymbol{a}^{\mathrm{PBSub}}||_2^2}{MK + L_R}, \tag{3.33}$$

where $\boldsymbol{a}^{\text{PBSub}} = \left[\boldsymbol{b}^{\text{sub}}, \boldsymbol{\psi}^{\text{sub}}\right]$ denotes the suboptimal solution of the joint problem (3.12), which can be obtained using an AO-based algorithm (e.g., [13, 47, 104]). The squared ℓ_2 norm is divided by $MK + L_R$ to obtain the average results for each element in $\boldsymbol{a}^{\text{PB}}(i)$.

Following the CL-based approach, we progressively change the objective towards the original objective, which is to maximize the weighted aggregate throughput. To this end, the reward function in the proposed CL-DDPG algorithm is given by:

$$r^{\text{PB}}(i) = \sum_{n \in \mathcal{M}} w_n R_n(\boldsymbol{a}^{\text{PB}}(i)) - \beta(i) \frac{||\boldsymbol{a}^{\text{PB}}(i) - \boldsymbol{a}^{\text{PBSub}}||_2^2}{MK + L_R}, \tag{3.34}$$

where $\beta(i)$ is the weight representing the importance of minimizing the average distance measurement. The value $\beta(i)$ needs to be adjusted properly throughout the training phase, such that the second term in (3.34) can facilitate the learning and meanwhile the performance of the learning algorithm will not be limited by the suboptimal solution $\boldsymbol{a}^{\text{PBSub}}$. To achieve such goals in the proposed CL-DDPG algorithm, the value of $\beta(i)$ will be decayed by multiplying with $1 - \theta$, where θ is a positive constant, if either of the following two conditions is satisfied:

- Condition 1: The distance measurement between the learned solution and the suboptimal solution is less than or equal to a given threshold ϵ. That is

$$\frac{||\boldsymbol{a}^{\text{PB}}(i) - \boldsymbol{a}^{\text{PBSub}}||_2^2}{MK + L_R} \leq \epsilon. \tag{3.35}$$

- Condition 2: The learned solution achieves a weighted aggregate throughput that is higher than that obtained from the suboptimal solution. That is

$$\sum_{n \in \mathcal{N}} w_n R_n(\boldsymbol{a}^{\text{PB}}(i)) > \sum_{n \in \mathcal{N}} w_n R_n(\boldsymbol{a}^{\text{PBSub}}). \tag{3.36}$$

If neither of the aforementioned two conditions is satisfied, we increment the value of β by multiplying with $1 + \theta$ to offer a higher reward for approaching a suboptimal solution.

Finally, we clip the value of $\beta(i)$ to be within $(0, \beta_{\max}]$ to avoid numerical issues. The overall mechanism for adjusting the value of β is shown in the following equation:

$$\beta(i+1) = \begin{cases} (1-\theta)\,\beta(i), & \text{if inequality (3.35) or (3.36) is satisfied,} \\ \min\{(1+\theta)\beta(i), \beta_{\max}\}, & \text{otherwise.} \end{cases} \tag{3.37}$$

3.4.2 Actor-Critic Method and Learning Algorithm

We employ the actor-critic method to learn the policy for solving the joint phase shift control and beamforming optimization problem. The actor network learns a policy $\pi_{\boldsymbol{\Phi}_{\text{act}}}$, which is parameterized by the learnable parameters $\boldsymbol{\Phi}_{\text{act}}$. The policy $\pi_{\boldsymbol{\Phi}_{\text{act}}}$ defines a mapping from a state to an action. That is $\boldsymbol{a}^{\text{PB}}(i) = \pi_{\boldsymbol{\Phi}_{\text{act}}}(\boldsymbol{s}^{\text{PB}}(i))$. In addition, the critic network learns a state-action value function $Q_{\boldsymbol{\Phi}_{\text{crt}}}$, which is parameterized by $\boldsymbol{\Phi}_{\text{crt}}$. The state-action value function $Q_{\boldsymbol{\Phi}_{\text{crt}}}(\boldsymbol{s}^{\text{PB}}(i), \boldsymbol{a}^{\text{PB}}(i))$ estimates the discounted total reward of selecting action $\boldsymbol{a}^{\text{PB}}(i)$ under state $\boldsymbol{s}^{\text{PB}}(i)$. That is,

$$Q_{\boldsymbol{\Phi}_{\text{crt}}}(\boldsymbol{s}^{\text{PB}}(i), \boldsymbol{a}^{\text{PB}}(i)) = \mathbb{E}_{\boldsymbol{s}^{\text{PB}} \sim p_{\pi_{\boldsymbol{\Phi}_{\text{act}}}}, \boldsymbol{a}^{\text{PB}} \sim \pi_{\boldsymbol{\Phi}_{\text{act}}}} \left[\sum_{\ell=i}^{I_{\max}} \gamma^{\ell-i} r^{\text{PB}}(\boldsymbol{s}^{\text{PB}}(\ell), \boldsymbol{a}^{\text{PB}}(\ell)) \right], \tag{3.38}$$

where $p_{\pi_{\boldsymbol{\Phi}_{\text{act}}}}$ denotes the distribution of the state transition as the result of taking action based on the actor's policy $\pi_{\boldsymbol{\Phi}_{\text{act}}}$, $I_{\max}$ is the decision horizon, and $\gamma \in [0, 1]$ is the discount factor.

The goal of the actor-critic method is to learn the actor policy which maximizes the discounted total reward [17]. We have

$$\underset{\boldsymbol{\Phi}_{\text{act}}}{\text{maximize}}\ J(\boldsymbol{\Phi}_{\text{act}}) \triangleq \mathbb{E}_{\boldsymbol{s}^{\text{PB}} \sim p_{\pi_{\boldsymbol{\Phi}_{\text{act}}}}, \boldsymbol{a}^{\text{PB}} \sim \pi_{\boldsymbol{\Phi}_{\text{act}}}} \left[\sum_{\ell=1}^{I_{\max}} \gamma^{\ell-1} r^{\text{PB}}(\boldsymbol{s}^{\text{PB}}(\ell), \boldsymbol{a}^{\text{PB}}(\ell)) \right]. \tag{3.39}$$

To solve problem (3.39), we use the deterministic policy gradient algorithm [126] to update

the learnable parameters of the actor network $\mathbf{\Phi}_{\text{act}}$ with the following gradient:

$$\nabla J(\mathbf{\Phi}_{\text{act}}) = \mathbb{E}_{\boldsymbol{s}^{\text{PB}} \sim p_{\pi_{\mathbf{\Phi}_{\text{act}}}}} \Big[\nabla Q_{\mathbf{\Phi}_{\text{crt}}}(\boldsymbol{s}^{\text{PB}}, \boldsymbol{a}^{\text{PB}}) \,|_{\,\boldsymbol{a}^{\text{PB}} = \pi_{\mathbf{\Phi}_{\text{act}}}(\boldsymbol{s}^{\text{PB}})} \nabla \pi_{\mathbf{\Phi}_{\text{act}}}(\boldsymbol{s}^{\text{PB}}) \Big]. \tag{3.40}$$

Moreover, the learnable parameters of the critic network $\mathbf{\Phi}_{\text{crt}}$ is updated based on the TD error of value approximation [67, Chapter 6]. In particular, we first determine the target of the state-action value function approximation as

$$y(\boldsymbol{s}^{\text{PB}}, \pi_{\mathbf{\Phi}_{\text{act}}}(\boldsymbol{s}^{\text{PB}})) = r^{\text{PB}}(\boldsymbol{s}^{\text{PB}}, \pi_{\mathbf{\Phi}_{\text{act}}}(\boldsymbol{s}^{\text{PB}})) + \gamma \, Q_{\mathbf{\Phi}_{\text{crt}}}(\hat{\boldsymbol{s}}^{\text{PB}}, \pi_{\mathbf{\Phi}_{\text{act}}}(\hat{\boldsymbol{s}}^{\text{PB}})), \tag{3.41}$$

where $\hat{\boldsymbol{s}}^{\text{PB}}$ is the next state as a result of taking action $\pi_{\mathbf{\Phi}_{\text{act}}}(\boldsymbol{s}^{\text{PB}})$ in state $\boldsymbol{s}^{\text{PB}}$. Then, $\mathbf{\Phi}_{\text{crt}}$ is updated by solving the following minimization problem [67, Chapter 6]:

$$\underset{\mathbf{\Phi}_{\text{crt}}}{\text{minimize}} \; \mathcal{L}(\mathbf{\Phi}_{\text{crt}}) \triangleq \mathbb{E}_{\boldsymbol{s}^{\text{PB}} \sim p_{\pi_{\mathbf{\Phi}_{\text{act}}}}, \boldsymbol{a}^{\text{PB}} \sim \pi_{\mathbf{\Phi}_{\text{act}}}} \Big[\big(Q_{\mathbf{\Phi}_{\text{crt}}}(\boldsymbol{s}^{\text{PB}}, \pi_{\mathbf{\Phi}_{\text{act}}}(\boldsymbol{s}^{\text{PB}})) - y(\boldsymbol{s}^{\text{PB}}, \pi_{\mathbf{\Phi}_{\text{act}}}(\boldsymbol{s}^{\text{PB}})) \big)^2 \Big]. \tag{3.42}$$

We can update $\mathbf{\Phi}_{\text{crt}}$ using gradient descent with the following gradient:

$$\begin{aligned} \nabla \mathcal{L}(\mathbf{\Phi}_{\text{crt}}) = \mathbb{E}_{\boldsymbol{s}^{\text{PB}} \sim p_{\pi_{\mathbf{\Phi}_{\text{act}}}}, \boldsymbol{a}^{\text{PB}} \sim \pi_{\mathbf{\Phi}_{\text{act}}}} \Big[& 2 \nabla Q_{\mathbf{\Phi}_{\text{crt}}}(\boldsymbol{s}^{\text{PB}}, \pi_{\mathbf{\Phi}_{\text{act}}}(\boldsymbol{s}^{\text{PB}})) \\ & \big(Q_{\mathbf{\Phi}_{\text{crt}}}(\boldsymbol{s}^{\text{PB}}, \pi_{\mathbf{\Phi}_{\text{act}}}(\boldsymbol{s}^{\text{PB}})) - y(\boldsymbol{s}^{\text{PB}}, \pi_{\mathbf{\Phi}_{\text{act}}}(\boldsymbol{s}^{\text{PB}})) \big) \Big]. \end{aligned} \tag{3.43}$$

To overcome the overestimation issue [18] of the conventional actor-critic method, we introduce two critics, where each of them performs an independent estimation of the state-action value function. We denote the learnable parameters of the two critics as $\mathbf{\Phi}_{\text{crt1}}$ and $\mathbf{\Phi}_{\text{crt2}}$, respectively. We upper bound the approximated state-action value function by the minimum approximation of the two critics to alleviate the impact of overestimation. This results in the following modification of the target of the expected discounted reward

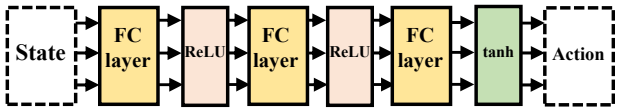
Algorithm 4 CL-DDPG: Training Algorithm

1: Set episode counter $k \leftarrow 0$.
2: Initialize the threshold δ.
3: Initialize the learnable parameters $\boldsymbol{\Phi}_{\text{act}}$, $\boldsymbol{\Phi}_{\text{crt1}}$, and $\boldsymbol{\Phi}_{\text{crt2}}$.
4: Conduct $T_{\text{warm-up}}$ episodes of exploration and store the transition history.
5: **while** $k \leq T_{\max}$ **do**
6: Observe new channel realizations and the weights of the users.
7: Initialize $i \leftarrow 0$ and $\beta(i) \leftarrow \beta_0$.
8: Determine the suboptimal solution as $\boldsymbol{a}^{\text{PBSub}}$ using a baseline algorithm.
9: **while** $i \leq I_{\max}$ **do**
10: Determine the action $\boldsymbol{a}^{\text{PB}}(i) \leftarrow \pi_{\boldsymbol{\Phi}_{\text{act}}}(\boldsymbol{s}^{\text{PB}}(i)) + \boldsymbol{\varrho}_{\text{epl}}$.
11: Determine the reward $r^{\text{PB}}(i)$ based on (3.34).
12: Update $\beta(i)$ based on (3.37) and store the transition history.
13: Sample a mini-batch of $\widehat{B}$ transition tuples.
14: Update $\boldsymbol{\Phi}_{\text{act}}$, $\boldsymbol{\Phi}_{\text{crt1}}$, and $\boldsymbol{\Phi}_{\text{crt2}}$ using the Adam optimizer and (3.45).
15: $i \leftarrow i + 1$.
16: **end while**
17: $k \leftarrow k + 1$
18: **end while**

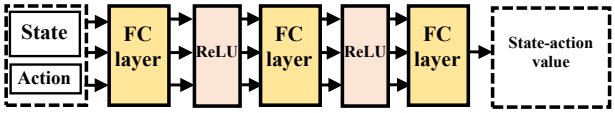
approximation in (3.41):

$$\begin{aligned} &y(\boldsymbol{s}^{\text{PB}}, \pi_{\boldsymbol{\Phi}_{\text{act}}}(\boldsymbol{s}^{\text{PB}})) \\ &= R(\boldsymbol{s}^{\text{PB}}, \pi_{\boldsymbol{\Phi}_{\text{act}}}(\boldsymbol{s}^{\text{PB}})) + \gamma \min\Big\{Q_{\boldsymbol{\Phi}_{\text{crt1}}}(\hat{\boldsymbol{s}}^{\text{PB}}, \pi_{\boldsymbol{\Phi}_{\text{act}}}(\hat{\boldsymbol{s}}^{\text{PB}})), Q_{\boldsymbol{\Phi}_{\text{crt2}}}(\hat{\boldsymbol{s}}^{\text{PB}}, \pi_{\boldsymbol{\Phi}_{\text{act}}}(\hat{\boldsymbol{s}}^{\text{PB}}))\Big\}. \end{aligned} \tag{3.44}$$

The overall learning algorithm is shown in **Algorithm 4**. The learning is performed in an episodic fashion. The learning progress consists of $T_{\max}$ episodes. Each episode has $I_{\max}$ decision steps. The same channel realization is used within an episode, while different (and independent) channel realizations are used across different episodes. To improve the exploration of the action space, we introduce a warm-up stage that consists of $T_{\text{warm-up}}$ episodes. Within the warm-up stage, the actions are chosen by sampling from a uniform distribution over the feasible actions. New channel realization and the weights of the users are observed at the beginning of each episode. In Line 8, we determine a suboptimal solution of the joint problem using a baseline algorithm, e.g., the AO-based algorithms [13, 47, 104]. In the i-th decision step, the action is determined as $\boldsymbol{a}^{\text{PB}}(i) =$



(a)



(b)

Figure 3.4: The network structure of (a) the actor network and (b) the critic network. The actor network determines the action based on the state. The critic network approximates the state-action value given the state and action.

$\pi_{\boldsymbol{\Phi}_{\text{act}}}(\boldsymbol{s}^{\text{PB}}(i)) + \boldsymbol{\varrho}_{\text{epl}}$, where $\boldsymbol{\varrho}_{\text{epl}}$ is the Gaussian noise with zero mean and variance σ^2_{epl}. In Lines 10−12, after selecting an action, we update the value of $\beta(i)$ based on (3.37). The transition tuple $(\boldsymbol{s}^{\text{PB}}(i), \boldsymbol{a}^{\text{PB}}(i), r^{\text{PB}}(i), \boldsymbol{s}^{\text{PB}}(i+1))$ is stored in the experience replay. In Line 13, we sample a mini-batch of $\widehat{B}$ transition tuples from the experience replay. After determining the gradient for each transition tuple within the mini-batch, we obtain the new learnable parameters $\boldsymbol{\Phi}'_{\text{act}}$, $\boldsymbol{\Phi}'_{\text{crt1}}$, and $\boldsymbol{\Phi}'_{\text{crt2}}$ by performing one step of gradient descent (and ascent) using the Adam optimizer. We then update the learnable parameters using the following soft update:

$$\begin{aligned}
\boldsymbol{\Phi}_{\text{act}} &\leftarrow \kappa\boldsymbol{\Phi}'_{\text{act}} + (1-\kappa)\boldsymbol{\Phi}_{\text{act}}, \\
\boldsymbol{\Phi}_{\text{crt1}} &\leftarrow \kappa\boldsymbol{\Phi}'_{\text{crt1}} + (1-\kappa)\boldsymbol{\Phi}_{\text{crt1}}, \\
\boldsymbol{\Phi}_{\text{crt2}} &\leftarrow \kappa\boldsymbol{\Phi}'_{\text{crt2}} + (1-\kappa)\boldsymbol{\Phi}_{\text{crt2}},
\end{aligned} \tag{3.45}$$

where κ is a constant and is between zero and one.

3.4.3 Network Structure of the Actor and Critic

The proposed actor network structure is shown in Fig. 3.4(a). The actor network consists of three FC layers. The first and second FC layers are followed by the ReLU activation layers. The output of the last FC layer is passed through a `tanh` activation layer to ensure

that each of the output values is always within the interval $(-1, 1)$. We note that setting a minimum and a maximum value for the action facilitates the learning of both the actor and critic networks. For the output of the actor network that corresponds to the phase shift variables, we multiply the output value by $(1 + \lambda)\pi$, where $\lambda > 0$, to obtain the phase shift value[4]. The sizes of the input and output of the actor network are given by $d_{\text{act\_in}} = M(K + L_R + KL_R + 1)$ and $d_{\text{act\_out}} = MK + L_R$, respectively. The sizes of the three FC layers in the actor network are $d_{\text{act\_in}} \times d_{\text{fc1}}$, $d_{\text{fc1}} \times d_{\text{fc2}}$, and $d_{\text{fc2}} \times d_{\text{act\_out}}$, respectively. The remaining outputs are directly used as the real and imaginary parts of the beamforming vectors. This is because the magnitude of the beamforming vector at the receiver does not affect the optimality of the solution. An arbitrary beamforming vector $\boldsymbol{b}_n$ can be scaled as $\frac{\boldsymbol{b}_n}{(1+\lambda)||\boldsymbol{b}_n||_2}$ so that the real and imaginary parts of each element of the vector are within the interval $(-1, 1)$. Both critics use the network structure as shown in Fig. 3.4(b). Each critic network has three FC layers and two ReLU activation layers. The input of the critic network is the concatenation of the state and action. The size of the input of the critic networks is given by $d_{\text{crt\_in}} = M(K + L_R + KL_R + 1) + MK + L_R$. Given the state and action, the output of the critic networks is $Q_{\boldsymbol{\Phi}_{\text{crt}}}(\boldsymbol{s}^{\text{PB}}, \pi_{\boldsymbol{\Phi}_{\text{act}}}(\boldsymbol{s}^{\text{PB}}))$. The sizes of the three FC layers in the critic networks are $d_{\text{crt\_in}} \times d_{\text{fc1}}$, $d_{\text{fc1}} \times d_{\text{fc2}}$, and $d_{\text{fc2}} \times 1$, respectively.

3.4.4 CL-DDPG: Online Algorithm

The online algorithm of the CL-DDPG algorithm is shown in **Algorithm 5**. In Line 2, given the CSI and the weights of the scheduled users, we obtain the suboptimal solution $\boldsymbol{a}^{\text{PBSub}}$ using a baseline algorithm. In Line 6, we conduct $I_{\text{warm-up}}$ decision steps of exploration to collect system transition tuples. In Lines 7−17, we update the learnable parameters for $I_{\max}$ decision steps based on the online information to further improve the learned policy. In Lines 12−15, the action that achieves the maximum weighted aggregate

[4]We choose the multiplier $(1 + \lambda)\pi > \pi$ since the output value of the `tanh` activation layer cannot be equal to either 1 or -1. Using a multiplier that is less than or equal to π will affect the optimality of the learned solution when the optimal solution is either $\psi_l^\star = \pi$ or $\psi_l^\star = -\pi$.

Algorithm 5 CL-DDPG: Online Execution Algorithm

Observe the channel realizations and the weights of the scheduled users.
Determine the suboptimal solution $\boldsymbol{a}^{\mathrm{PBSub}}$ using a baseline algorithm.
Initialize the threshold δ.
Initialize $i \leftarrow 0$ and $\beta(i) \leftarrow \beta_0$.
Initialize the maximum achievable weighted aggregate throughput $R_{\max} \leftarrow 0$.
Conduct $I_{\text{warm-up}}$ decision steps of exploration and store the transition history.
while $i \leq I_{\max}$ **do**
  Determine the action $\boldsymbol{a}^{\mathrm{PB}}(i) \leftarrow \pi_{\boldsymbol{\Phi}_{\mathrm{act}}}(\boldsymbol{s}^{\mathrm{PB}}(i)) + \boldsymbol{\varrho}_{\mathrm{epl}}$.
  Determine the reward $r^{\mathrm{PB}}(i)$ based on (3.34).
  Update $\beta(i)$ based on (3.37) and store the transition history.
  Sample a mini-batch and update $\boldsymbol{\Phi}_{\mathrm{act}}$, $\boldsymbol{\Phi}_{\mathrm{crt1}}$, and $\boldsymbol{\Phi}_{\mathrm{crt2}}$ using the Adam optimizer and (3.45).
  if $\sum_{n\in\mathcal{M}} w_n(t) R_n(\boldsymbol{a}^{\mathrm{PB}}(i)) \geq R_{\max}$ **then**
    $\boldsymbol{a}^{\mathrm{out}} \leftarrow \boldsymbol{a}^{\mathrm{PB}}(i)$.
    $R_{\max} \leftarrow \sum_{n\in\mathcal{M}} w_n(t) R_n(\boldsymbol{a}^{\mathrm{PB}}(i))$.
  end if
  $i \leftarrow i + 1$.
end while
Return $\boldsymbol{a}^{\mathrm{out}}$.

throughput throughout the $I_{\max}$ decision steps of the online execution is chosen as the output.

3.4.5 Overall Framework of the Proposed DUPB Algorithm

The overall framework of the proposed DUPB algorithm is shown in Fig. 3.5. The forward propagations for determining the rewards, state-action value function approximation, and TD error are denoted by blue solid arrows. The backpropagations of the gradients are denoted by dashed arrows. Note that, both the reward functions used in the NCO-based and CL-DDPG algorithms depend on the weighted aggregate throughput, which corresponds to the objective of the original problem (3.10). With the reward function in (3.16), we use the REINFORCE algorithm to determine the gradient for updating the learnable parameters of the encoder and decoder modules in the NCO-based algorithm. We then use the deterministic policy gradient and TD error to obtain the gradients for the training of

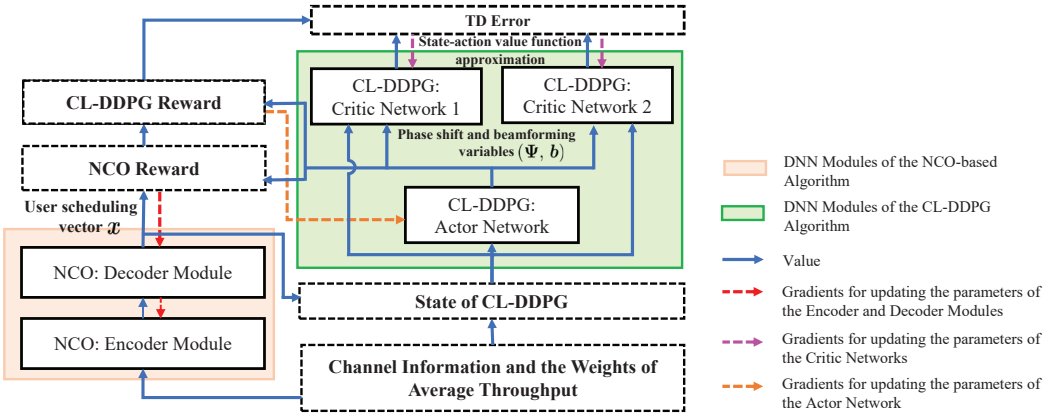


Figure 3.5: The overall architecture of the proposed DUPB algorithm. The proposed DUPB algorithm consists of (i) the encoder and decoder modules employed in the NCO-based user scheduling algorithm as indicated by the orange box in the figure, and (ii) the actor and critic networks, i.e., the green box in the figure, for determining the phase shift and beamforming variables.

the actor and critic networks in the CL-DDPG algorithm, respectively. Recall that for the reward function in (3.16), the solutions of phase shift control $\boldsymbol{\Psi}^\star$ and beamforming vectors $\boldsymbol{b}_n^\star$, $n \in \mathcal{N}$, are determined by the phase shift control and beamforming policy $\pi_{\boldsymbol{\Phi}_{\text{act}}}$ that is learned by the CL-DDPG algorithm. Therefore, $b(\mathcal{S})$ and $r(\mathcal{U}')$ in (3.30) depend on policy $\pi_{\boldsymbol{\Phi}_{\text{act}}}$. This indicates that the gradient for updating the user scheduling policy depends on the phase shift control and beamforming policy $\pi_{\boldsymbol{\Phi}_{\text{act}}}$. Thus, the learning of user scheduling policy depends on $\pi_{\boldsymbol{\Phi}_{\text{act}}}$.

For the proposed DUPB algorithm, the computational complexity of obtaining the user scheduling vector depends on the computational complexity of forward propagation through the encoder and decoder modules. The forward propagation through the initial embedding module in the encoder module has a complexity of $\mathcal{O}\Big(N\big((K+L_R+KL_R+1)d_{\text{Res}}+N_{\text{Res}}d_{\text{Res}}^2+d_{\text{Res}}d_h\big)\Big)$. The forward propagation through the attention layers, the MLP module, and the embedding aggregate layer incurs a complexity of $\mathcal{O}(Nd_zd_h+Nd_kd_h+N^2d_k+Nd_h^2+Nd_ad_h+d_cd_h+Nd_h)$. Hence, obtaining the user scheduling incurs the following

computational complexity:

$$\mathcal{O}_{\text{US}} = \mathcal{O}\Big(N\big((K + L_R + KL_R + 1)d_{\text{Res}} + N_{\text{Res}}d_{\text{Res}}^2 + d_{\text{Res}}d_h\big) + Nd_zd_h + Nd_kd_h + N^2d_k + Nd_h^2 + Nd_ad_h + d_cd_h + Nd_h\Big).$$

For the ease of analysis, given the M scheduled users, we assume that the AO-based algorithm with FP is employed to obtain a suboptimal solution of the joint phase shift and beamforming optimization problem. This incurs the following computational complexity [11]:

$$\mathcal{O}(C_{\text{AO}}(C_{\text{PS}}L_R^{4.5}\log(1/\epsilon_{\text{SDR}}) + MK^3)), \tag{3.46}$$

where C_{AO} is the number of iterations to solve the joint phase shift and beamforming optimization problem with a fixed user scheduling vector, C_{PS} is the number of iterations to solve the phase shift subproblem using FP and SDR, and ϵ_{SDR} is the solution accuracy of SDR. With the obtained suboptimal solution, the online execution of the CL-DDPG algorithm incurs $\widehat{B}I_{\max}$ forward propagations and $I_{\max}$ backpropagations. Each forward propagation has a computational complexity of $\mathcal{O}_{\text{PB\_forward}} = \mathcal{O}\Big((d_{\text{act\_in}} + d_{\text{crt\_in}})d_{\text{fc1}} + d_{\text{fc1}}d_{\text{fc2}} + (d_{\text{act\_out}} + 1)d_{\text{fc2}}\Big)$, while each backpropagation incurs a computational complexity of

$$\mathcal{O}_{\text{PB\_backprop}} = \mathcal{O}\Big((d_{\text{act\_in}} + d_{\text{crt\_in}} + d_{\text{fc2}} + 1)d_{\text{fc1}} + d_{\text{fc2}} + (d_{\text{act\_out}} + 1)d_{\text{fc2}} + d_{\text{act\_out}}\Big). \tag{3.47}$$

The overall computational complexity of the proposed DUPB algorithm is given by:

$$\mathcal{O}_{\text{US}} + \mathcal{O}(C_{\text{AO}}(C_{\text{PS}}L_R^{4.5}\log(1/\epsilon_{\text{SDR}}) + MK^3)) + \widehat{B}I_{\max}\mathcal{O}_{\text{PB\_forward}} + I_{\max}\mathcal{O}_{\text{PB\_backprop}}. \tag{3.48}$$

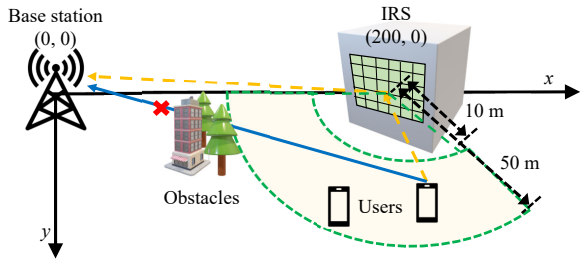


Figure 3.6: Simulation setup. The base station is located at $(0, 0)$, while the IRS is located at $(200, 0)$. The users are distributed within a 120° annulus sector where the IRS is located at the center of the circle.

3.5 Performance Evaluation

We simulate an IRS-aided system where the distance between the IRS and the base station is 200 meters. The users are randomly and uniformly distributed within [10, 50] meters of the IRS. The simulation setup is shown in Fig. 3.6. We assume the direct channels between the users and the base station are blocked [127]. Note that when the direct channels are present, the input dimensionality of the linear projection layers in the encoder module should be increased accordingly to incorporate the extra channel information. The reflecting channels follow Rician fading distribution. We set the Ricean K-factor to 6 with a path loss exponent equal to 3.5 [128]. The maximum transmit power $P^{\max}$ is set to 30 dBm, and the noise power is set to -90 dBm. The window size of the exponential moving average T_C in equation (3.9) is set to 20. The parameters for the proposed DUPB algorithm are shown in Table 3.1. We use PyTorch [129] for simulation. In our simulation, the channel realizations for DNN training and evaluation are generated independently. We conduct the simulation using a Linux-based computing server with an Intel E5-2683 Broadwell @ 2.1GHz CPU, and an NVIDIA Tesla P100 Pascal GPU with 12 GB memory. The average training time per episode of the proposed DUPB algorithm on the computing server is 132.7 min for the system setting $N = 10$, $M = K = 4$, and $L_R = 80$.

The performance of the proposed DUPB algorithm is evaluated after 200 training episodes. We first compare the performance of the proposed DUPB algorithm with the following three algorithms:

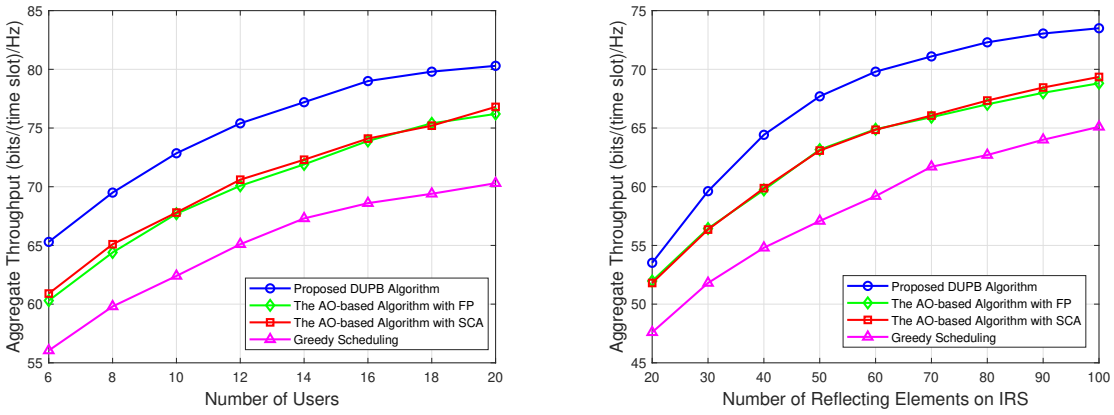
Table 3.1: Simulation Parameters for the Proposed DUPB Algorithm

Parameter	Value
Dimensionalities of the DNN layers in NCO d_k, d_z, d_a, d_h	$\max\{256, 4d_i\}$
Number of training iterations in each episode of NCO N_e	25
Dimensionalities of the DNN layers in CL-DDPG d_{fc1}, d_{fc2}	$\max\{256, 2d_{\text{crt\_in}}\}$
Maximum number of decision steps in CL-DDPG $I_{\max}$	1000
Learning rate	0.005
Minibatch sizes B, $\widehat{B}$	64, 128
Constant α for clipping the compatibility	10
Number of channel realizations used for evaluation B_e	5000
Reward function parameters β_0, $\beta_{\max}$, ϵ	10, 10, 0.1
Results scaling factors θ, λ	0.01, 0.01
Number of episodes and decision steps for warm-up $T_{\text{warm-up}}$, $I_{\text{warm-up}}$	10, 1000

- **AO-based algorithm with FP** [104]: In this algorithm, we first relax the binary user scheduling variables, and then solve the user scheduling, phase shift control, and beamforming subproblems iteratively using FP and SDR.

- **AO-based algorithm with SCA** [47]: Compared with the AO-based algorithm with FP, in this algorithm, we solve the phase shift control subproblem using the SCA approach [47].

- **Greedy scheduling**: In this algorithm, the base station schedules those M users with the largest value of $\left|\boldsymbol{h}_{D,n}(t) + \boldsymbol{G}^H(t)\boldsymbol{h}_{R,n}(t)\right|^2$, $n \in \mathcal{N}$ and then employs the AO-based algorithm with FP to obtain the phase shift matrix and beamforming vectors.



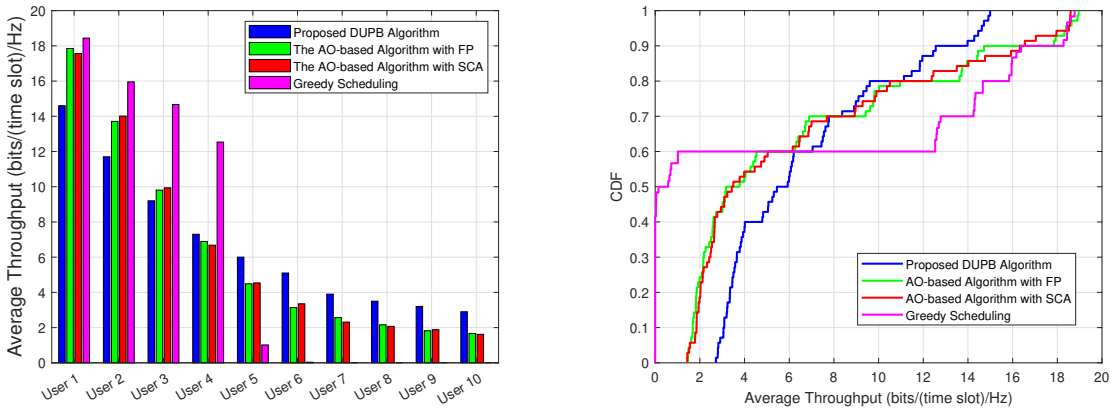
(a) (b)

Figure 3.7: Aggregate throughput versus (a) the number of users and (b) the number of reflecting elements when the objective is to maximize the aggregate throughput.

3.5.1 Aggregate Throughput

We vary the number of users and evaluate the aggregate throughput with the maximum aggregate throughput objective in Fig. 3.7(a). We set $M = K = 4$ and $L_R = 80$. We observe that the aggregate throughput of all considered algorithms increase with the number of users. When N is equal to 20, the proposed DUPB algorithm achieves an aggregate throughput that is 4.6%, 5.4% and 14.2% higher than the AO-based algorithm with SCA, the AO-based algorithm with FP, and the greedy scheduling algorithm, respectively.

Fig. 3.7(b) shows the aggregate throughput versus the number of reflecting elements on the IRS with the maximum aggregate throughput objective. We set $M = K = 4$ and $N = 10$. The results show that the aggregate throughput of all considered algorithms increase with the number of reflecting elements. The reason is that, having more reflecting elements on the IRS offers a higher DoF for controlling the phase shift of the reflecting channel. By properly controlling the phase shift of the IRS and beamforming at the base station, we can reap the benefits of the DoF offered by the IRS. When L_R is equal to 100, the proposed DUPB algorithm achieves an aggregate throughput that is 6.0%, 7.3%, and 12.9% higher than the AO-based algorithm with SCA, the AO-based algorithm with FP,



(a) (b)

Figure 3.8: (a) Average throughput per user and (b) the cumulative distribution function (CDF) of the average throughput per user for $M = K = 4$, $N = 10$, and $L_R = 80$.

and the greedy scheduling algorithm, respectively.

3.5.2 Fairness Among the Users

We evaluate the average throughput per user under the proportional fairness objective in Fig. 3.8(a). We sort the users in ascending order of their distances to the base station and index them accordingly. That is, the closest user to the base station is referred to as user 1, and the farthest user to the base station is referred to as user 10. All algorithms are evaluated after running 300 consecutive time slots. With proportional fairness objective, the users are allocated with an appropriate amount of network resources to maintain a balance between fairness and aggregate throughput. This is because the reward of scheduling a particular user is weighted based on its average throughput. Scheduling the users with low average throughput can sometimes lead to a higher reward. Hence, the network resources will not congregate at those users with better channel conditions. We also observe from Fig. 3.8(a) that the distribution of throughput per user varies between different algorithms. With the greedy scheduling algorithm, the users with good channel conditions (e.g., users 1 and 2) obtain a higher throughput, while the users with poor channel conditions (e.g.,

Table 3.2: Online Execution Runtime Comparison for Different Algorithms ($M = K = 4$)

Parameter Settings	$N = 10$, $L_R = 40$	$N = 10$, $L_R = 80$	$N = 10$, $L_R = 100$	$N = 20$, $L_R = 80$
Proposed DUPB Algorithm	79.4 min	116.9 min	159.6 min	124.7 min
The AO-based Algorithm with FP	68.5 min	175.6 min	282.7 min	332.8 min
The AO-based Algorithm with SCA	81.2 min	168.7 min	256.8 min	320.6 min
Greedy Scheduling Algorithm	27.6 min	56.9 min	84.5 min	57.1 min

users 9 and 10) are not scheduled by the base station. In the proposed DUPB algorithm, those users with poor channel conditions are scheduled more frequently and hence obtain a higher throughput.

In Fig. 3.8(b), we plot the cumulative distribution function (CDF) of the average throughput of the users. We observe that, for those users with relatively low average throughput, particularly lower than 4 bits/(time slot)/Hz, they are allocated with more resources under the proposed DUPB algorithm and hence achieve a higher average throughput. The results also show the difference in the tail distribution of the average throughput per user. With the proportional fairness objective, the users with preferable channel conditions may achieve the highest average throughput under the greedy scheduling algorithm. This is consistent with the results in Fig. 3.8(a). Moreover, the proposed DUPB algorithm reduces the standard deviation of the throughput distribution among the users by 29.7%, 33.4%, and 51.1% when compared with the AO-based algorithm with SCA, the AO-based algorithm with FP, and the greedy scheduling algorithm, respectively.

3.5.3 Runtime Comparison

We compare the online execution runtime of different algorithms for 10 consecutive time slots on the same computing server. The results are shown in Table 3.2. When L_R is equal to 100, the runtime of the proposed DUPB algorithm is 37.9% and 43.5% lower than the AO-based algorithm with SCA and the AO-based algorithm with FP, respectively. We also observe that the AO-based algorithm with FP incurs the longest runtime when L_R is

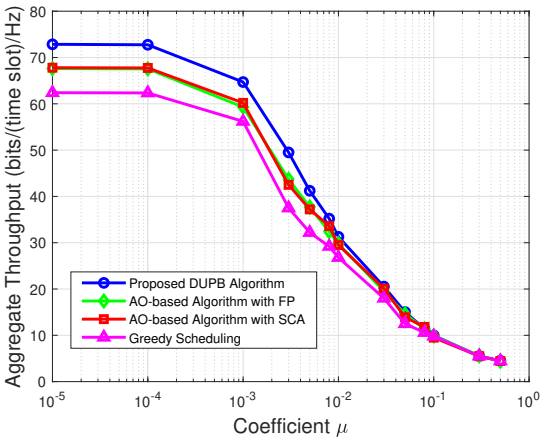


Figure 3.9: Aggregate throughput under imperfect channel estimation. We set $M = K = 4$, $L_R = 80$, and $N = 10$.

large, mainly due to the computational complexity of using SDR to obtain the rank-one matrix. In all considered settings, the greedy scheduling algorithm requires the shortest runtime as the user scheduling is determined by a pre-defined heuristic.

3.5.4 Effect of Imperfect Channel Estimation

We evaluate the impact of imperfect channel estimation on the performance of the considered algorithms. We consider the statistical channel estimation error [130, 131]. The estimated channel gain between the base station and the IRS is given by $\widehat{\boldsymbol{G}}(t) = \boldsymbol{G}(t) + \Delta\boldsymbol{G}(t)$, where $\Delta\boldsymbol{G}(t) \in \mathbb{C}^{L_R \times K}$ is channel estimation error of $\boldsymbol{G}(t)$. The elements in $\Delta\boldsymbol{G}(t)$ are assumed to follow complex Gaussian distribution with zero mean and variance $\mu^2 \|\boldsymbol{g}(t)\|_2^2$. The constant $\mu \in [0, 1)$ measures the significance of estimation error. The estimated channel between user n and the IRS is given by $\widehat{\boldsymbol{h}}_{R,n}(t) = \boldsymbol{h}_{R,n}(t) + \Delta\boldsymbol{h}_{R,n}(t)$. The elements in $\Delta\boldsymbol{h}_{R,n}(t)$ follow complex Gaussian distribution with zero mean and variance $\mu^2 \|\boldsymbol{h}_{R,n}(t)\|_2^2$. In Fig. 3.9, we evaluate the aggregate throughput under imperfect channel estimation with the maximum aggregate throughput as the objective. We observe performance degradations in all considered algorithms due to imperfect channel estimation. When μ is equal to 0.005, the proposed DUPB algorithm can retain 56.6% of the aggregate throughput under

perfect channel estimation. The performance degradation of the proposed DUPB algorithm and the other baseline algorithms due to imperfect channel estimation are similar.

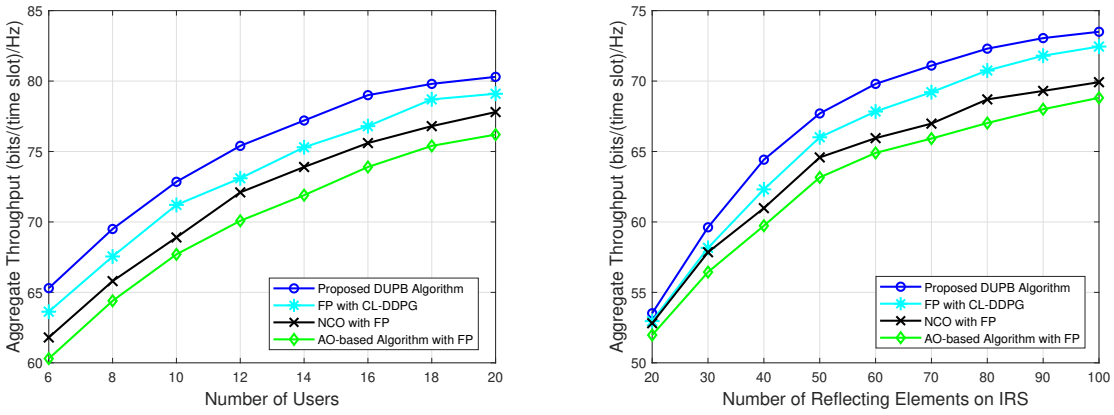
3.5.5 Performance Gain of Each Module

In order to provide additional insights on the throughput improvement obtained from the proposed DUPB algorithm, we evaluate the performance of the following two algorithms:

- **NCO-based user scheduling with FP-based phase shift control and beamforming optimization**: In this algorithm, we use the proposed NCO-based algorithm to determine the user scheduling, while the joint phase shift control and beamforming subproblem is solved using the FP and SDR-based approach. We refer to this algorithm as NCO with FP.

- **FP-based user scheduling with CL-DDPG-based phase shift control and beamforming optimization**: In this algorithm, we use the FP-based approach to solve the user scheduling subproblem. Given the user scheduling vector, the joint phase shift control and beamforming optimization subproblem is solved using the proposed CL-DDPG algorithm. We refer to this algorithm as FP with CL-DDPG.

We set $M = K = 4$ and $L_R = 80$. The performance comparison results are shown in Fig. 3.10. Results show that both the FP with CL-DDPG and NCO with FP algorithms achieve better performance than the AO-based algorithm. This indicates that both modules (i.e., the NCO algorithm for user scheduling and the CL-DDPG algorithm for phase shift control and beamforming optimization) contribute to the improved performance of our proposed DUPB algorithm. While the optimization-based approaches can obtain high quality suboptimal solutions, the following issues may still affect the optimality of these approaches due to the nonconvexity of the studied problem. First, the quality of the converged solution of the FP-based user scheduling algorithm can be highly sensitive to the initialization of the relaxed user scheduling variables. When the relaxed user scheduling variable of user n (i.e., $x_n \in [0, 1]$) is initialized to be close to 0, the local gradient will



(a) (b)

Figure 3.10: Aggregate throughput versus (a) the number of users and (b) the number of reflecting elements when the objective is to maximize the aggregate throughput.

encourage the algorithm to reduce the value of x_n further. Therefore, the FP-based user scheduling algorithm would converge to the stationary point in which user n is not being scheduled, i.e., $x_n = 0$. Thus, it can be difficult for the FP-based user scheduling algorithm to avoid being trapped by the local optimal solutions. When the user scheduling subproblem is solved using the NCO algorithm, we can explore the solution space better than the FP-based algorithm since (a) a large number of feasible actions are being explored by the NCO algorithm during the exploration phase and the training phase, (b) the NCO algorithm learns a stochastic user scheduling policy which tends to explore more feasible actions during the early phase of training, and (c) with the help of the REINFORCE algorithm, the policy learning in the NCO algorithm is designed to find the best solution within the explored solution space. The aforementioned features allow the NCO algorithm to efficiently explore the solution space and obtain a higher aggregate throughput.

We also observe from Fig. 3.10 that, when compared with the AO-based algorithm with FP, the CL-DDPG algorithm provides a higher performance gain than the NCO-based user scheduling algorithm. When the joint phase shift control and beamforming subproblem is solved using the AO-based algorithm with FP, since Gaussian randomization process is used in the SDR to obtain the phase shift control matrix, it may lead to performance

degradation. In our proposed CL-DDPG algorithm, since we can obtain the phase shift variables directly from the learned policy, the performance degradation due to Gaussian randomization can be mitigated.

3.6 Summary

In this chapter, we studied the joint user scheduling, phase shift control, and beamforming design in IRS-aided systems. We decomposed the formulated problem into two subproblems: a combinatorial optimization subproblem for user scheduling, and a nonconvex subproblem for joint phase shift control and beamforming optimization. We proposed a DUPB algorithm, in which the NCO technique is used to learn the stochastic policy for determining user scheduling. In addition, we proposed a CL-DDPG algorithm to jointly optimize the phase shift control and the beamforming vectors. Our CL-based approach facilitated the policy learning by exploiting the knowledge of the hidden convexity of the joint problem. Results showed that the proposed DUPB algorithm outperforms the AO-based algorithms and greedy scheduling algorithm under both maximum aggregate throughput and proportional fair objectives. The runtime of the proposed DUPB algorithm is lower than that of the proposed AO-based algorithms when the number of reflecting elements is large.

Chapter 4

Rate-Splitting for Intelligent Reflecting Surface-Aided Multiuser VR Streaming

4.1 Introduction

In the previous chapter, we investigated the IRS-aided multiuser systems in which the data streams of different users are assumed to be independent and uncorrelated. That is, each user decodes its signal by treating the signals of the remaining users in the systems as interference. However, there also exist some applications in B5G wireless systems where the data streams of different users can be correlated. In this chapter, we investigate whether such correlation can be exploited by RS and IRS to achieve additional multiplexing gain in B5G wireless systems. In particular, we focus on the multiuser VR streaming application in this chapter, in which the data requested by different users may be correlated due to their shared interests of the 360-degree video tiles. We propose an IRS-aided RS VR streaming system, in which the 360-degree video tiles requested by the users are transmitted using RS. We design a novel RS scheme, in which the common and private messages are constructed based on the video tile requests as well as the CSI of the users. We optimize the RS parameters in the proposed RS scheme to achieve the maximum multiplexing gain by exploiting the shared interests of the users. Moreover, we investigate the performance improvement that can be obtained by deploying an IRS to support the RS-based video tile transmission. To jointly optimize the IRS phase shifts, RS parameters, beamforming

vectors, and individual bitrates to maximize the achievable bitrate of the 360-degree video, we propose a Deep-GRAIL algorithm, in which we improve the efficacy of the vanilla DDPG algorithm by using imitation learning [31, 32]. The contributions of this chapter are as follows:

- We propose an IRS-aided RS VR streaming system, and formulate a joint optimization problem for maximization of the achievable bitrate of the 360-degree video. Our problem formulation comprises the joint optimization of the beamforming vectors at the base station, IRS phase shifts, RS parameters, and bitrates of the 360-degree video tiles requested by the users.

- We propose the Deep-GRAIL algorithm, in which imitation learning, actor-critic method, and DDPG are exploited to learn a policy for solving the formulated problem. Apart from the experience replay that maintains the exploration history of the learning agent, we introduce a demonstration replay that stores the solutions obtained by conventional optimization methods. Using imitation learning, the proposed algorithm can effectively improve the learned policy by exploiting both the experience replay and demonstration replay.

- We propose RavNet, which is a DNN designed for policy learning in the proposed Deep-GRAIL algorithm. In particular, one of the neural network layers in RavNet is the differentiable convex optimization (DCO) layer [132], which tackles the convex constraints of the formulated problem during the learning process.

- We evaluate the performance of the proposed Deep-GRAIL algorithm using a real-world VR streaming dataset [133]. Simulation results show that the proposed algorithm outperforms several baseline schemes, including the IRS-aided RS-NOUM system using an AO algorithm [62], the conventional IRS-aided multiuser system using an AO algorithm [63], the IRS-aided RS VR streaming system using an AO algorithm, and the IRS-aided RS VR streaming system using a supervised learning (SL) algorithm, in terms of the system sum-rate, achievable bitrate, and runtime.

The remainder of this chapter is organized as follows. The system model and problem formulation for IRS-aided RS VR streaming systems are presented in Section 4.2. In Section 4.3, we develop the Deep-GRAIL algorithm. In Section 4.4, we introduce the DNN structure and functionality of the proposed RavNet. Simulation results are presented in Section 4.5. A summary is given in Section 4.6.

Notations: In this chapter, we use upper-case and lower-case boldface letters to denote matrices and column vectors, respectively. $\mathbb{C}^{M\times N}$ denotes the set of $M \times N$ complex-valued matrices. $\boldsymbol{A}^T$ and $\boldsymbol{A}^H$ denote the transpose and conjugate transpose of matrix $\boldsymbol{A}$, respectively. vec($\boldsymbol{A}$) returns a vector obtained by stacking the columns of matrix $\boldsymbol{A}$. diag($\boldsymbol{x}$) returns a diagonal matrix where the diagonal elements are given by the elements of vector $\boldsymbol{x}$. $\Re(\boldsymbol{x})$ and $\Im(\boldsymbol{x})$ return the vectors that include the real and imaginary parts of the complex-valued elements of vector $\boldsymbol{x}$, respectively. $\sim$ means "distributed as". $\mathbb{E}[\cdot]$ represents statistical expectation. $\mathbb{1}(\cdot)$ denotes the indicator function, which is equal to 1 if its argument is true and is equal to 0 otherwise.

4.2 IRS-aided RS VR Streaming System and Problem Formulation

The considered IRS-aided RS VR streaming system is illustrated in Fig. 4.1. One base station and an IRS are deployed in an indoor facility to provide VR streaming service to N users. Let $\mathcal{N} = \{1, 2, \ldots, N\}$ denote the set of users. The base station has N_t antennas, while the IRS has L reflecting elements. The HMD of each user has one antenna. We denote $\phi_l \in [0, 2\pi)$, $l \in \{1, \ldots, L\}$, as the phase shift of the l-th reflecting element of the IRS. Time is slotted into intervals of equal duration. Let $\mathcal{T} = \{1, 2, \ldots\}$ denote the set of time slots. The time interval $[t, t+1)$ is referred to as time slot $t \in \mathcal{T}$. The direct channel between the base station and user $n \in \mathcal{N}$ in time slot t is denoted by $\boldsymbol{h}_{n,D}(t) \in \mathbb{C}^{N_t}$. The channel between the base station and the IRS in time slot t is denoted by $\boldsymbol{G}(t) \in \mathbb{C}^{L\times N_t}$. The phase shift matrix of the IRS in time slot t is an $L \times L$ diagonal matrix, denoted by

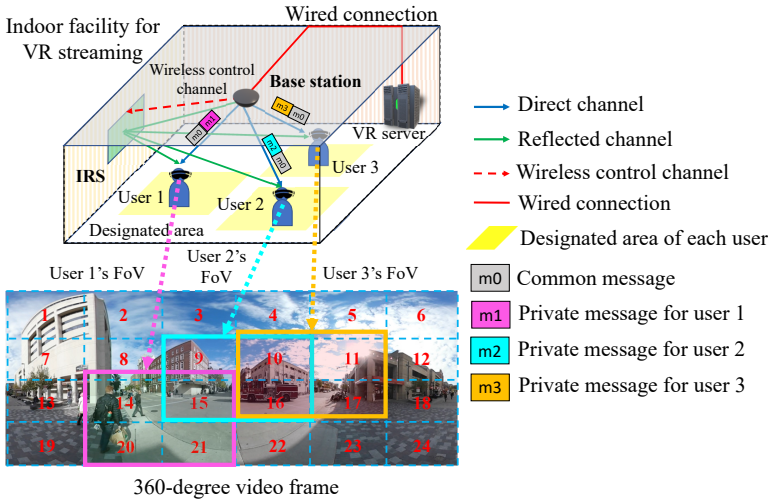


Figure 4.1: An IRS-aided RS VR streaming system. The upper part of the figure shows an indoor facility for VR streaming. The lower part of the figure illustrates a 360-degree video frame. The 2×2 boxes in the video frame represent the FoVs of the users, while the numbers are the indices of the corresponding 360-degree video tiles. Here, users 1 and 2 request the same video tile with index 15. Users 2 and 3 request the same video tile with indices 10 and 16.

$\boldsymbol{\Psi}(t) = \text{diag}(e^{j\phi_1(t)}, \ldots, e^{j\phi_L(t)})$. The channel between the IRS and user $n \in \mathcal{N}$ in time slot t is denoted as $\boldsymbol{h}_{n,R}(t) \in \mathbb{C}^L$.

We assume that the base station can obtain the global CSI by using existing channel estimation methods proposed for IRS-aided multiuser systems, such as [134–136]. To ensure proper functionality of the HMD, each user is designated a certain area (denoted by the yellow areas in Fig. 4.1) inside the indoor facility [137]. Each user can move freely within his/her designated area during the VR streaming session.

4.2.1 Video Tile Request of the Users

Each 360-degree video frame is divided into $I_{\max}$ video tiles with N_x rows and N_y columns, i.e., $I_{\max} = N_x N_y$. As an example, for the 360-degree video frame shown in Fig. 4.1, we have $N_x = 4$, $N_y = 6$, and $I_{\max} = 24$. We denote $\mathcal{I} = \{1, 2, \ldots, I_{\max}\}$ as the set of indices of the tiles. At the beginning of time slot $t \in \mathcal{T}$, user $n \in \mathcal{N}$ sends an uplink request to the base station that specifies the indices of the tiles it requested, which can be determined

based on the FoV of user n. The indices of tiles requested by user n in time slot t are collected in set $\mathcal{I}_n(t) \subseteq \mathcal{I}$. We denote the number of video tiles requested by user n in time slot t as $I_n(t)$. We have $|\mathcal{I}_n(t)| = I_n(t) \leq I_{\max}, n \in \mathcal{N}, t \in \mathcal{T}$, where $|\mathcal{I}_n(t)|$ denotes the cardinality of set $\mathcal{I}_n(t)$.

Remark 4.1: Although we assume user n informs the base station about $\mathcal{I}_n(t)$ via uplink signaling, the base station can also use existing prediction algorithms to predict $\mathcal{I}_n(t)$ based on historical information, see, e.g., [138]. The prediction of video tile requests is beyond the scope of this work but our proposed algorithm can also be straightforwardly applied to VR streaming systems with video tile request prediction.

4.2.2 Bitrate Selection, QoE, and User's Utility Model

The data of the 360-degree video is stored at the VR server, which is connected to the base station via a wired connection. We assume there are M bitrate selections of the 360-degree video available at the VR server. After receiving the video tile requests from the users, the base station needs to determine the bitrate of each requested video tile. Let $v_{n,i}(t)$ denote the bitrate of tile $i \in \mathcal{I}_n(t)$ requested by user n in time slot t. For bitrate selection, we have

$$\text{C1:} \quad v_{n,i}(t) \in \mathcal{S} = \{v_1, \ldots, v_M\},\ i \in \mathcal{I}_n(t),\ n \in \mathcal{N}, \tag{4.1}$$

where set $\mathcal{S}$ contains the M bitrate selections available at the VR server, and $v_1 < v_2 < \cdots < v_M$. We use vector $\boldsymbol{v}_n(t) = (v_{n,i}(t),\ i \in \mathcal{I}_n(t))$ to collect the bitrate selections of the tiles requested by user n in time slot t.

We model the QoE degradation caused by a bitrate switch among the tiles requested by a user in a particular time slot by the intra-frame quality switch loss [139, 140]. We denote the intra-frame quality switch loss of user n in time slot t as $\ell_n^{\text{intra}}(t)$. $\ell_n^{\text{intra}}(t)$ is determined by the variance of the bitrates of the tiles requested by user n in time slot t.

For user $n \in \mathcal{N}$ in time slot $t \in \mathcal{T}$, we have [139]

$$\ell_n^{\text{intra}}(t) = \frac{1}{I_n(t)} \sum_{i \in \mathcal{I}_n(t)} \left(v_{n,i}(t) - \frac{1}{I_n(t)} \sum_{j \in \mathcal{I}_n(t)} v_{n,j}(t) \right)^2. \tag{4.2}$$

The utility obtained by user n in time slot t is given by

$$u_n(t) = \sum_{i \in \mathcal{I}_n(t)} v_{n,i}(t) - \kappa^{\text{intra}} \ell_n^{\text{intra}}(t),\ n \in \mathcal{N}, \tag{4.3}$$

where $\kappa^{\text{intra}} > 0$ is a scaling factor for $\ell_n^{\text{intra}}(t)$. The utility in (4.3) captures the QoE improvement obtained for higher bitrates for the requested tiles and the QoE degradation caused by intra-frame quality switches.

4.2.3 RS-based Downlink VR Video Tile Transmission

The base station employs RS-based downlink transmission to exploit the tile requests shared by different users. In time slot t, the indices of the tiles requested by all users are collected in set $\mathcal{I}(t) = \bigcup_{n \in \mathcal{N}} \mathcal{I}_n(t)$, $t \in \mathcal{T}$. After receiving the video tile requests from the users, the base station constructs a common message taking the video tile requests and CSI of the users into account. When constructing the common message in the proposed IRS-aided RS VR streaming system, the base station needs to determine (a) the data of which tiles should be included in the common message, and (b) what is the proportion of the data of each tile in the common message. After construction, the common message is encoded into a data stream, which is denoted as $s_0(t) \in \mathbb{C}$, where $\mathbb{E}\left[|s_0(t)|^2\right] = 1$. The beamforming vector for the common message is denoted as $\boldsymbol{b}_0(t) \in \mathbb{C}^{N_t}$. In addition, the base station constructs a private message for user $n \in \mathcal{N}$ that includes the private part of the data of the tiles requested by user n in time slot t. The private message for user n is encoded as $s_n(t) \in \mathbb{C}$ with $\mathbb{E}\left[|s_n(t)|^2\right] = 1$. The beamforming vector for the private message of user n is denoted as $\boldsymbol{b}_n(t) \in \mathbb{C}^{N_t}$. We collect the beamforming vectors in vector

$\boldsymbol{b}(t) = \left[\boldsymbol{b}_0^T(t)\ \boldsymbol{b}_1^T(t)\ \cdots\ \boldsymbol{b}_N^T(t)\right]^T \in \mathbb{C}^{(N+1)N_t}$. We denote the maximum transmit power of the base station as $P_{\max}$. We have the following constraint on the beamforming vectors:

$$\text{C2:} \quad ||\boldsymbol{b}(t)||_2^2 \leq P_{\max}. \tag{4.4}$$

At the receiver side, the signal received by user n in time slot t is given by:

$$\begin{aligned} y_n(t) = &\left(\boldsymbol{h}_{n,D}^H(t) + \boldsymbol{h}_{n,R}^H(t)\,\boldsymbol{\Psi}(t)\,\boldsymbol{G}(t)\right)\boldsymbol{b}_0(t)s_0(t) \\ &+ \sum_{m\in\mathcal{N}} \left(\boldsymbol{h}_{n,D}^H(t) + \boldsymbol{h}_{n,R}^H(t)\,\boldsymbol{\Psi}(t)\,\boldsymbol{G}(t)\right)\boldsymbol{b}_m(t)s_m(t) + z_n(t),\ n\in\mathcal{N}, \end{aligned} \tag{4.5}$$

where $z_n(t)$ is the additive white Gaussian noise (AWGN) with variance σ^2 at user n in time slot t. User n first decodes the common message by treating the private messages of all users as interference. The SINR of the common message at user n in time slot t is given by:

$$\gamma_n^{\mathrm{c}}(t) = \frac{\left|\left(\boldsymbol{h}_{n,D}^H(t) + \boldsymbol{h}_{n,R}^H(t)\,\boldsymbol{\Psi}(t)\,\boldsymbol{G}(t)\right)\boldsymbol{b}_0(t)\right|^2}{\sum_{m\in\mathcal{N}}\left|\left(\boldsymbol{h}_{n,D}^H(t) + \boldsymbol{h}_{n,R}^H(t)\,\boldsymbol{\Psi}(t)\,\boldsymbol{G}(t)\right)\boldsymbol{b}_m(t)\right|^2 + \sigma^2}. \tag{4.6}$$

The achievable rate for the common message at user n in time slot t is $R_n^{\mathrm{c}}(t) = \log_2(1 + \gamma_n^{\mathrm{c}}(t))$. Let $R^{\mathrm{c}}(t)$ denote the transmission rate of the common message in time slot t. All users need to decode the common message first, and then remove it from their respective received signal to decode their private message. To ensure successful decoding of the common message at all users, we have the following constraint on $R^{\mathrm{c}}(t)$:

$$\text{C3:} \quad R^{\mathrm{c}}(t) = \min\{R_1^{\mathrm{c}}(t), \ldots, R_N^{\mathrm{c}}(t)\}. \tag{4.7}$$

We denote the proportion of $R^{\mathrm{c}}(t)$ that is dedicated to the data transmission of video tile $i \in \mathcal{I}(t)$ in time slot t as $c_i(t)$. We have

$$\text{C4:} \quad \sum_{i\in\mathcal{I}(t)} c_i(t) \leq 1, \tag{4.8}$$

$$\text{C5:} \quad c_i(t) \geq 0,\, i \in \mathcal{I}(t). \tag{4.9}$$

Remark 4.2: The shared interests of the users can be exploited in the proposed IRS-aided RS VR streaming system by properly choosing the values of $c_i(t)$, $i \in \mathcal{I}(t)$, based on the video tile requests and CSI of the users. Through the optimization of $c_i(t)$, the base station can determine which tiles should be included in the common message, and what are the corresponding proportions of the common rate that should be allocated to the transmission of these tiles. When multiple users request the same video tile, it may be beneficial to use a larger proportion of the common rate to transmit this tile since it can increase the individual utility of those users that requested this tile simultaneously. The proposed algorithm for optimizing $c_i(t)$ along with the other DoF of the system will be presented in Sections 4.3 and 4.4.

Remark 4.3: When $c_i = 0$, this means that no data from tile i is included in the common message. As an example, for the FoVs and the corresponding video tile requests of the users shown in Fig. 4.1, one possible construction of the common message is given as follows: $c_{10} = c_{15} = c_{16} = \frac{1}{3}$, and $c_9 = c_{11} = c_{14} = c_{17} = c_{20} = c_{21} = 0$. This means that the common message only includes data from tiles 10, 15, and 16, which are requested by multiple users, and the common rate is split equally between these three tiles.

After decoding the common message, user n removes the signal corresponding to the common message from $y_n(t)$ using SIC, and decodes its private message by treating the private messages of other users as interference. Thus, the SINR of the private message at user $n \in \mathcal{N}$ in time slot $t \in \mathcal{T}$ is given by [28]:

$$\gamma_n^{\mathrm{p}}(t) = \frac{\left| \left(\boldsymbol{h}_{n,D}^H(t) + \boldsymbol{h}_{n,R}^H(t)\, \boldsymbol{\Psi}(t)\, \boldsymbol{G}(t) \right) \boldsymbol{b}_n(t) \right|^2}{\sum\limits_{m \in \mathcal{N} \setminus \{n\}} \left| \left(\boldsymbol{h}_{n,D}^H(t) + \boldsymbol{h}_{n,R}^H(t)\, \boldsymbol{\Psi}(t)\, \boldsymbol{G}(t) \right) \boldsymbol{b}_m(t) \right|^2 + \sigma^2}. \tag{4.10}$$

The achievable rate of the private message of user n is given by $R_n^{\mathrm{p}}(t) = \log_2(1 + \gamma_n^{\mathrm{p}}(t))$. Let $p_{n,i}(t)$ denote the proportion of $R_n^{\mathrm{p}}(t)$ that is used to transmit the data of tile $i \in \mathcal{I}_n(t)$

in time slot t. We have

$$\text{C6:} \quad \sum_{i \in \mathcal{I}_n(t)} p_{n,i}(t) \leq 1, \, n \in \mathcal{N}, \tag{4.11}$$

$$\text{C7:} \quad p_{n,i}(t) \geq 0, \, i \in \mathcal{I}_n(t), \, n \in \mathcal{N}. \tag{4.12}$$

Remark 4.4: When $c_i(t) > 0$ and $p_{n,i}(t) > 0$, this means that portions of the data of tile $i \in \mathcal{I}_n(t)$ are included in both the common and private messages for user n in time slot t. However, when $c_i(t) = 0$ and $p_{n,i}(t) > 0$, the data of tile $i \in \mathcal{I}_n(t)$ is transmitted only via the private message to user n in time slot t.

4.2.4 Per-User Per-Tile QoE Requirement

After decoding the common and private messages, user n retrieves its requested video tiles by combining the decoded messages. Let $J^{\mathrm{c}}_{n,i}(t)$ denote the number of bits that user n obtained from the common message for tile $i \in \mathcal{I}_n(t)$ in time slot t. We have

$$J^{\mathrm{c}}_{n,i}(t) = W T_{\mathrm{DL}} c_i(t) R^{\mathrm{c}}(t), \, i \in \mathcal{I}_n(t), \, n \in \mathcal{N}, \tag{4.13}$$

where W and T_{DL} are the downlink transmission bandwidth and time duration, respectively. Let $J^{\mathrm{p}}_{n,i}(t)$ denote the number of bits that user n obtained from its private message for tile $i \in \mathcal{I}_n(t)$ in time slot t. We have

$$J^{\mathrm{p}}_{n,i}(t) = W T_{\mathrm{DL}} p_{n,i}(t) R^{\mathrm{p}}_n(t), \, i \in \mathcal{I}_n(t), \, n \in \mathcal{N}. \tag{4.14}$$

By combining the common and private messages, the total number of bits that user n received for tile $i \in \mathcal{I}_n(t)$ in time slot t is given by:

$$J_{n,i}(t) = J^{\mathrm{c}}_{n,i}(t) + J^{\mathrm{p}}_{n,i}(t), \, i \in \mathcal{I}_n(t), \, n \in \mathcal{N}. \tag{4.15}$$

The total number of bits required by user n to retrieve tile i with bitrate $v_{n,i}(t)$ is given

by:

$$J_{n,i}^{\min}(t) = T_v\, v_{n,i}(t),\ i \in \mathcal{I}_n(t),\ n \in \mathcal{N}, \tag{4.16}$$

where T_v denotes the time duration of a 360-degree video tile.

In order to ensure that all the video tiles requested by user n can be received with the chosen bitrate within the downlink transmission window, we have the following per-user per-tile QoE constraint:

$$\text{C8:}\quad J_{n,i}(t) \geq J_{n,i}^{\min}(t),\ i \in \mathcal{I}_n(t),\ n \in \mathcal{N}. \tag{4.17}$$

4.2.5 Problem Formulation

In time slot t, we tackle the following joint optimization problem for maximization of the summation of the utilities of the users in the IRS-aided RS VR streaming system:

$$\begin{aligned} \underset{\substack{\boldsymbol{b}(t),\boldsymbol{\Psi}(t),\\ \boldsymbol{v}_n(t),n\in\mathcal{N},\\ c_i(t),i\in\mathcal{I}(t),\\ p_{n,i}(t),i\in\mathcal{I}_n(t),n\in\mathcal{N}}}{\text{maximize}} \quad & u(t) \triangleq \sum_{n\in\mathcal{N}} u_n(t) \\ \text{subject to} \quad & \text{constraints C1}-\text{C8}, \\ & \text{C9: } \phi_l(t) \in [0,\, 2\pi),\ l \in \{1, 2, \ldots, L\}, \end{aligned} \tag{4.18}$$

where constraint C1 ensures that the bitrate of each tile can only be chosen from set $\mathcal{S}$. Constraint C2 is the maximum downlink transmission power constraint at the base station. Constraint C3 guarantees that the common message can be decoded by all users. Constraints C4−C7 are the constraints for the common and private rates allocated to the requested video tiles. Constraint C8 is the per-user per-tile QoE constraint. Constraint C9 is the IRS phase shift constraint. Problem (4.18) is a mixed-integer nonlinear programming problem. A suboptimal solution of problem (4.18) can be obtained by decomposing it into multiple subproblems, and solving them iteratively using AO. However, solving problem (4.18) using an AO algorithm can be computationally expensive and time-consuming for

VR streaming applications. To tackle this issue, in the next section, we propose the learning-based Deep-GRAIL algorithm to efficiently solve problem (4.18).

4.3 Deep-GRAIL: Deep Deterministic Policy Gradient with Imitation Learning Algorithm

Designing a learning-based algorithm for solving problem (4.18) is challenging due to the RS encoding and decoding procedure, and the constraints in problem (4.18). In the proposed Deep-GRAIL algorithm, we tackle these challenges using imitation learning [31, 32], and DCO [132].

4.3.1 MDP Formulation

We first model the sequential decision process for solving problem (4.18) in time slot $t \in \mathcal{T}$ as an MDP with $\tau^{\max}$ decision epochs. For notational simplicity, we drop time index t in this section. The state vector in the τ-th decision epoch of the MDP is defined as

$$\begin{aligned}\boldsymbol{s}(\tau) = \Big[&\boldsymbol{h}_{n,D},\, \mathrm{vec}(\mathrm{diag}(\boldsymbol{h}_{n,R}^{H})\boldsymbol{G}),\, n \in \mathcal{N},\\ &\boldsymbol{b}(\tau-1),\, \boldsymbol{c}(\tau-1),\, R_n^{\mathrm{c}}(\tau-1),\, R_n^{\mathrm{p}}(\tau-1),\\ &\boldsymbol{p}_n(\tau-1),\, \boldsymbol{v}_n(\tau-1),\, \boldsymbol{o}_n,\, n \in \mathcal{N}\Big],\end{aligned} \tag{4.19}$$

where $\boldsymbol{p}_n(\tau) = (p_{n,i}(\tau),\, i \in \mathcal{I})$https://www.overleaf.com/project/62ec4f2c34d663df8e8bb8e1 and $\boldsymbol{c}(\tau) = (c_i(\tau),\, i \in \mathcal{I})$. In addition, the binary vector $\boldsymbol{o}_n = (\mathbb{1}\left(i \in \mathcal{I}_n\right), i \in \mathcal{I}) \in \{0,1\}^{I_{\max}}$ in (4.19) contains the information about the video tile request of user n. Note that $\tau = 0$ corresponds to the initialization of the MDP.

Furthermore, the action vector in the τ-th decision epoch is defined as

$$\boldsymbol{a}(\tau) = (\boldsymbol{b}(\tau),\, \mathrm{vec}(\boldsymbol{\Psi}(\tau)), \boldsymbol{c}(\tau), \boldsymbol{p}_n(\tau), \boldsymbol{v}_n(\tau), n \in \mathcal{N})\,. \tag{4.20}$$

The action vector $\boldsymbol{a}(\tau)$ contains all optimization variables of problem (4.18).

For the reward function design, note that the objective function in problem (4.18), i.e., $u(\tau)$, can only take values from a finite set due to the discrete nature of bitrate selection. If $u(\tau)$ is used directly as the reward function, then the reward function becomes sparse and may prevent the learning agent from effectively improving the policy [141]. To tackle this issue, we first use the following inequality to establish the connection between $u(\tau)$ and the system sum-rate explicitly:

$$\begin{aligned} u(\tau) &= \sum_{n\in\mathcal{N}} \left(\sum_{i\in\mathcal{I}_n} v_{n,i}(\tau) - \kappa^{\text{intra}}\, \ell_n^{\text{intra}}(\tau) \right) \\ &\overset{(a)}{\leq} \sum_{n\in\mathcal{N}} \frac{WT_{\text{DL}}}{T_v} \Big(\sum_{i\in\mathcal{I}_n} p_{n,i}(\tau) R_n^{\text{p}}(\tau) + \sum_{i\in\mathcal{I}_n} c_i(\tau) R^{\text{c}}(\tau) \Big) \\ &\quad - \sum_{n\in\mathcal{N}} \kappa^{\text{intra}}\, \ell_n^{\text{intra}}(\tau) \triangleq r(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)), \end{aligned} \tag{4.21}$$

where inequality (a) follows from inequality (4.17). That is

$$\begin{aligned} & J_{n,i} \geq J_{n,i}^{\min}(\tau) \\ & \Longleftrightarrow WT_{\text{DL}} \Big(\sum_{i\in\mathcal{I}_n} p_{n,i}(\tau) R_n^{\text{p}}(\tau) + \sum_{i\in\mathcal{I}_n} c_i(\tau) R^{\text{c}}(\tau) \Big) \geq T_v \sum_{i\in\mathcal{I}_n} v_{n,i}(\tau) \\ & \Longrightarrow \sum_{n\in\mathcal{N}} \frac{WT_{\text{DL}}}{T_v} \Big(\sum_{i\in\mathcal{I}_n} p_{n,i}(\tau) R_n^{\text{p}}(\tau) + \sum_{i\in\mathcal{I}_n} c_i(\tau) R^{\text{c}}(\tau) \Big) \geq \sum_{n\in\mathcal{N}} \sum_{i\in\mathcal{I}_n} v_{n,i}(\tau). \end{aligned} \tag{4.22}$$

We use $r(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))$ as the reward function in the MDP to provide an informative feedback to the learning agent. In the reward function $r(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))$, we replace the discrete bitrate selections with the continuous achievable bitrates to overcome the sparsity of $u(\tau)$. We set the maximum achievable bitrate of a video tile in $r(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))$ to be equal to the maximum bitrate selection v_M. Since the difference between $r(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))$ and $u(\tau)$ cannot exceed $\sum_{n\in\mathcal{N}} |\mathcal{I}_n| \max_{i=1,\ldots,M-1} |v_{i+1} - v_i|$, using $r(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))$ as the reward function also leads to a policy that can achieve a high utility $u(\tau)$.

4.3.2 Actor-Critic Method with q-Step Return

Based on the MDP formulation, we use an actor network to learn a policy for solving problem (4.18). We denote the learnable parameters (i.e., the weights and biases) of the actor network as $\boldsymbol{\Phi}_{\text{act}}$. The policy learned by the actor network, which is denoted by $\pi_{\boldsymbol{\Phi}_{\text{act}}}(\boldsymbol{s}(\tau))$, defines a mapping from a state to an action. That is, $\boldsymbol{a}(\tau) = \pi_{\boldsymbol{\Phi}_{\text{act}}}(\boldsymbol{s}(\tau))$. The critic network learns a state-action value function $Q_{\boldsymbol{\Phi}_{\text{crt}}}$, which is parameterized by $\boldsymbol{\Phi}_{\text{crt}}$. The state-action value function $Q_{\boldsymbol{\Phi}_{\text{crt}}}(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))$ estimates the discounted total reward of selecting action $\boldsymbol{a}(\tau)$ in state $\boldsymbol{s}(\tau)$. That is,

$$Q_{\boldsymbol{\Phi}_{\text{crt}}}(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)) = \mathbb{E}_{\boldsymbol{s}\sim p_{\pi_{\boldsymbol{\Phi}_{\text{act}}}}, \boldsymbol{a}\sim\pi_{\boldsymbol{\Phi}_{\text{act}}}} \left[\sum_{\tau'=\tau}^{\tau^{\max}} \gamma^{\tau'-\tau} r(\boldsymbol{s}(\tau'), \boldsymbol{a}(\tau')) \right], \tag{4.23}$$

where $p_{\pi_{\boldsymbol{\Phi}_{\text{act}}}}$ denotes the distribution of the state transition as the result of taking actions based on policy $\pi_{\boldsymbol{\Phi}_{\text{act}}}$, and $\gamma \in [0, 1]$ is the discount factor.

The goal of the actor-critic method is to learn the policy which maximizes the discounted total reward [17]. We have

$$\underset{\boldsymbol{\Phi}_{\text{act}}}{\text{maximize}}\ \mathcal{L}(\boldsymbol{\Phi}_{\text{act}}) \triangleq \mathbb{E}_{\boldsymbol{s}\sim p_{\pi_{\boldsymbol{\Phi}_{\text{act}}}}, \boldsymbol{a}\sim\pi_{\boldsymbol{\Phi}_{\text{act}}}} \left[\sum_{\tau'=1}^{\tau^{\max}} \gamma^{\tau'-1} r(\boldsymbol{s}(\tau'), \boldsymbol{a}(\tau')) \right]. \tag{4.24}$$

The deterministic policy gradient [17] for solving problem (4.24) is given by:

$$\nabla\mathcal{L}_{\text{DPG}} = \mathbb{E}_{\boldsymbol{s}\sim p_{\pi_{\boldsymbol{\Phi}_{\text{act}}}}} \left[\nabla Q_{\boldsymbol{\Phi}_{\text{crt}}}(\boldsymbol{s}, \boldsymbol{a}) \,|_{\,\boldsymbol{a}=\pi_{\boldsymbol{\Phi}_{\text{act}}}(\boldsymbol{s})} \nabla \pi_{\boldsymbol{\Phi}_{\text{act}}}(\boldsymbol{s}) \right]. \tag{4.25}$$

The policy update based on the deterministic policy gradient may suffer from Q-value overestimation [56]. To tackle the overestimation issue in the proposed Deep-GRAIL algorithm, we use the following two techniques. First, while the vanilla DDPG algorithm [17] only uses one critic network, we use V critic networks to obtain V independent approximations of the Q-value. The target of Q-value approximation is determined by the minimum of these V approximations to alleviate overestimation. Second, we determine

the target of Q-value approximation using the q-step return [67, Section 7.1]. Compared with the single-step return, q-step return determines the discounted future reward over q consecutive decision epochs, which provides the learning agent with more information regarding future planning when compared with the single-step return (i.e., when $q = 1$). Hence, the target Q-value in the proposed Deep-GRAIL algorithm is given by:

$$\begin{aligned}\widehat{Q}_{\boldsymbol{\Phi}_{\text{crt}}}(\boldsymbol{s}(\tau), \pi_{\boldsymbol{\Phi}_{\text{act}}}(\boldsymbol{s}(\tau))) = &\sum_{\tau'=\tau}^{\tau+q-1} \gamma^{\tau'-\tau} r(\boldsymbol{s}(\tau'), \pi_{\boldsymbol{\Phi}_{\text{act}}}(\boldsymbol{s}(\tau'))) \\ &+ \gamma^q \min_{m=1,2,\ldots,V} Q_{\boldsymbol{\Phi}^{(m)}_{\text{crt}}}(\boldsymbol{s}(\tau+q), \pi_{\boldsymbol{\Phi}_{\text{act}}}(\boldsymbol{s}(\tau+q))),\end{aligned} \tag{4.26}$$

where $\boldsymbol{\Phi}^{(m)}_{\text{crt}}$ denotes the learnable parameters of the m-th critic network. Then, $\boldsymbol{\Phi}^{(m)}_{\text{crt}}, m = 1, \ldots, V$, is updated by minimizing the following TD error of Q-value approximation [67, Chapter 6]:

$$\underset{\boldsymbol{\Phi}^{(m)}_{\text{crt}}}{\text{minimize}}\ \mathcal{L}(\boldsymbol{\Phi}^{(m)}_{\text{crt}}) \triangleq \mathbb{E}_{\boldsymbol{s}\sim p_{\pi_{\boldsymbol{\Phi}_{\text{act}}}}, \boldsymbol{a}\sim\pi_{\boldsymbol{\Phi}_{\text{act}}}}\Big[\big(Q_{\boldsymbol{\Phi}^{(m)}_{\text{crt}}}(\boldsymbol{s}, \pi_{\boldsymbol{\Phi}_{\text{act}}}(\boldsymbol{s})) - \widehat{Q}_{\boldsymbol{\Phi}_{\text{crt}}}(\boldsymbol{s}, \pi_{\boldsymbol{\Phi}_{\text{act}}}(\boldsymbol{s}))\big)^2\Big]. \tag{4.27}$$

We update $\boldsymbol{\Phi}^{(m)}_{\text{crt}}, m = 1, \ldots, V$, using gradient descent with the following gradient:

$$\begin{aligned}\nabla\mathcal{L}(\boldsymbol{\Phi}^{(m)}_{\text{crt}}) = \mathbb{E}_{\boldsymbol{s}\sim p_{\pi_{\boldsymbol{\Phi}_{\text{act}}}}, \boldsymbol{a}\sim\pi_{\boldsymbol{\Phi}_{\text{act}}}}\Big[&2\nabla Q_{\boldsymbol{\Phi}^{(m)}_{\text{crt}}}(\boldsymbol{s}, \pi_{\boldsymbol{\Phi}_{\text{act}}}(\boldsymbol{s})) \\ &\big(Q_{\boldsymbol{\Phi}^{(m)}_{\text{crt}}}(\boldsymbol{s}, \pi_{\boldsymbol{\Phi}_{\text{act}}}(\boldsymbol{s})) - \widehat{Q}_{\boldsymbol{\Phi}_{\text{crt}}}(\boldsymbol{s}, \pi_{\boldsymbol{\Phi}_{\text{act}}}(\boldsymbol{s}))\big)\Big].\end{aligned} \tag{4.28}$$

The learning agent maintains an experience replay that stores the system transition history due to past decisions as a system transition tuple $(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau), r(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)), \boldsymbol{s}(\tau+1))$. To determine the gradients in (4.25) and (4.28), we first sample the system transition tuples of q consecutive decision epochs from the experience replay, and then determine the gradients for the sampled system transition. Note that q consecutive decision epochs are sampled in order to determine the q-step return in (4.26). We find the gradients in a minibatch-based manner and average the gradients over M_D different samples.

Note that while the aforementioned actor-critic method is designed for maximizing the

expected discounted total reward, we tackle the constraints in problem (4.18) during the learning process by using DCO layers in the DNN structure design of the proposed RavNet. The corresponding details will be presented in Section 4.4.2.

4.3.3 Policy Improvement using Imitation Learning and Demonstration Replay

In the vanilla DDPG algorithm [17] and its variant [56], the learning agent improves the learned policy only based on the system transition history obtained from its own exploration. However, the exploration of the learning agent can be inefficient when the dimensionality of the state and action space is large. To tackle this issue, we notice that problem (4.18) has a hidden convexity due to the fractional form of the SINR expressions in (4.6) and (4.10). Hence, a suboptimal solution of problem (4.18) can be obtained by using an AO algorithm exploiting convex optimization, FP, and SDR. The details of the AO algorithm are provided in Appendix B.

In the proposed Deep-GRAIL algorithm, we use imitation learning [31, 32] to allow the agent to learn from the AO algorithm and exploit the hidden convexity of problem (4.18). We introduce the *imitation loss* to characterize the difference between the action chosen by the learned policy and the solution obtained by the AO algorithm. By using imitation learning, the proposed Deep-GRAIL algorithm provides the actor-critic method with knowledge of the hidden convexity of the formulated problem, leading to an efficient policy learning.

In particular, the framework of the proposed Deep-GRAIL algorithm is illustrated in Fig. 4.2. Apart from the actor-critic method and the experience replay, we introduce a *demonstration replay* to store the solutions of problem (4.18) obtained by the AO algorithm over D time slots. Note that one time slot corresponds to one episode comprising $\tau^{\max}$ decision epochs in the MDP. We define set $\mathcal{D} = \{1, 2, \ldots, D\}$. In the d-th time slot, where $d \in \mathcal{D}$, the AO algorithm is invoked to solve problem (4.18). Since AO leads to an iterative algorithm, using it to solve problem (4.18) also results in a sequential system transition.

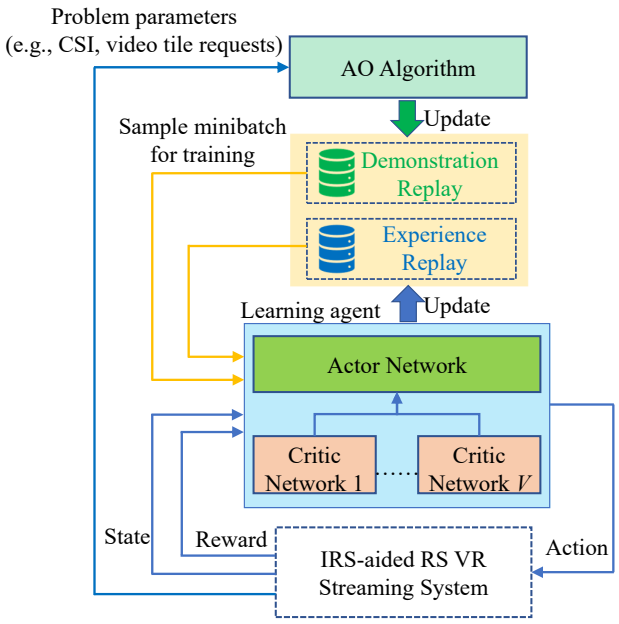


Figure 4.2: Overall framework of the proposed Deep-GRAIL algorithm. The learning agent comprises an actor network and V critic networks. An experience replay is employed to maintain the exploration history of the learning agent. Moreover, a demonstration replay is maintained by the learning agent to store the system transition tuples obtained from the execution of the AO algorithm.

In particular, in the τ-th decision epoch of the d-th time slot, we first initialize the AO algorithm with the variables in $\boldsymbol{s}^{(d)}(\tau)$, which is the state vector in the τ-th decision epoch of the d-th time slot. Here, we use the superscript (d) to denote the values of variables in the d-th time slot. We then execute the AO algorithm for one iteration and obtain the solution. We denote this solution as $\boldsymbol{a}_{\mathrm{AO}}^{(d)}(\tau)$. With $\boldsymbol{a}_{\mathrm{AO}}^{(d)}(\tau)$, we determine the reward and the next state of the MDP as $r(\boldsymbol{s}^{(d)}(\tau), \boldsymbol{a}_{\mathrm{AO}}^{(d)}(\tau))$ and $\boldsymbol{s}^{(d)}(\tau+1)$, respectively. Then, the system transition obtained from the execution of the AO algorithm in the τ-th decision epoch of the d-th time slot is denoted as the following system transition tuple:

$$\left(\boldsymbol{s}^{(d)}(\tau), \boldsymbol{a}_{\mathrm{AO}}^{(d)}(\tau), r(\boldsymbol{s}^{(d)}(\tau), \boldsymbol{a}_{\mathrm{AO}}^{(d)}(\tau)), \boldsymbol{s}^{(d)}(\tau+1)\right). \tag{4.29}$$

We index the system transition tuple in (4.29) with the tuple (d, τ). The system transition tuples obtained from the execution of the AO algorithm are stored in the demonstration

replay for imitation learning.

In each training iteration, we sample a minibatch of M_D different transition tuples from the demonstration replay. We denote the set that collects the indices of the system transition tuples within the minibatch as $\mathcal{M}_D$. Then, for each state $\boldsymbol{s}^{(d)}(\tau), (d,\tau) \in \mathcal{M}_D$, in the minibatch, we determine the imitation loss of the actor network based on the mean squared error between the action chosen by the actor network, i.e., $\pi_{\boldsymbol{\Phi}_{\text{act}}}(\boldsymbol{s}^{(d)}(\tau))$, and the solution obtained by the AO algorithm, i.e., $\boldsymbol{a}_{\text{AO}}^{(d)}(\tau)$. In particular, the imitation loss $\widehat{\mathcal{L}}_{\text{IMI}}$ is given by:

$$\widehat{\mathcal{L}}_{\text{IMI}} = \frac{1}{M_D} \sum_{(d,\tau)\in\mathcal{M}_D} ||\pi_{\boldsymbol{\Phi}_{\text{act}}}(\boldsymbol{s}^{(d)}(\tau)) - \boldsymbol{a}_{\text{AO}}^{(d)}(\tau)||^2. \tag{4.30}$$

We scale the elements in $\pi_{\boldsymbol{\Phi}_{\text{act}}}(\boldsymbol{s}^{(d)}(\tau))$ and $\boldsymbol{a}_{\text{AO}}^{(d)}(\tau)$ that correspond to beamforming and IRS phase shift variables to be between -1 and 1 to mitigate the potential impact of the different ranges of the variables.

Note that the solutions obtained by the AO algorithm are in general suboptimal due to the nonconvexity of the formulated problem. Hence, using the imitation loss in (4.30) may prevent the learning agent from finding better solutions than those obtained by the AO algorithm. To tackle this issue, for each sample with index $(d,\tau) \in \mathcal{M}_D$, we determine the discounted total reward that can be achieved by following the actor's policy in the remaining decision epochs (i.e., the Monte Carlo estimation of the value function [67, Section 7.1]) by

$$\widehat{Q}_{\boldsymbol{\Phi}_{\text{act}}}^{(d)}(\tau) = \sum_{\tau'=\tau}^{\tau^{\max}} \gamma^{\tau'-\tau} r(\boldsymbol{s}^{(d)}(\tau'), \pi_{\boldsymbol{\Phi}_{\text{act}}}(\boldsymbol{s}^{(d)}(\tau'))). \tag{4.31}$$

The discounted total reward obtained by using the AO algorithm in the remaining decision epochs is given by

$$\widehat{Q}_{\text{AO}}^{(d)}(\tau) = \sum_{\tau'=\tau}^{\tau^{\max}} \gamma^{\tau'-\tau} r(\boldsymbol{s}^{(d)}(\tau'), \boldsymbol{a}_{\text{AO}}^{(d)}(\tau')). \tag{4.32}$$

To overcome the potential suboptimality of the AO algorithm, we only calculate the imitation loss for those states and actions for which the AO algorithm achieves a higher discounted total reward than the actor's policy. This results in the following modified

imitation loss:

$$\mathcal{L}_{\text{IMI}} = \frac{1}{\widehat{M}_D} \sum_{(d,\tau)\in\mathcal{M}_D} \left(||\pi_{\mathbf{\Phi}_{\text{act}}}(\boldsymbol{s}^{(d)}(\tau)) - \boldsymbol{a}^{(d)}_{\text{AO}}(\tau)||^2 \mathbb{1}\left(\widehat{Q}^{(d)}_{\text{AO}}(\tau) > \widehat{Q}^{(d)}_{\mathbf{\Phi}_{\text{act}}}(\tau)\right) \right), \tag{4.33}$$

where $\widehat{M}_D = \sum_{(d,\tau)\in\mathcal{M}_D} \mathbb{1}\left(\widehat{Q}^{(d)}_{\text{AO}}(\tau) > \widehat{Q}^{(d)}_{\mathbf{\Phi}_{\text{act}}}(\tau)\right)$.

By combining with the deterministic policy gradient in (4.25), the overall gradient for updating the actor network in the proposed Deep-GRAIL algorithm is given by:

$$\nabla\mathcal{L}_{\text{act}} = \omega_1 \nabla\mathcal{L}_{\text{DPG}} + \omega_2 \nabla\mathcal{L}_{\text{IMI}}, \tag{4.34}$$

where ω_1 and ω_2 are positive parameters representing the weights of the deterministic policy gradient and the gradient of imitation loss, respectively.

4.3.4 Training Algorithm

The proposed training algorithm is illustrated in **Algorithm 6**. We first obtain the demonstration replay by executing the AO algorithm over D episodes. This results in $D\tau^{\max}$ system transition tuples being stored in the demonstration replay. Meanwhile, the learning agent explores the state and action space by taking actions based on the learned policy, and stores the resulting system transition tuples in the experience replay.

We train the learning agent for $T_{\max}$ episodes. In each training iteration, we first sample a minibatch from the demonstration replay and determine the imitation loss based on (4.33). Then, we sample another minibatch from the experience replay and determine the gradient for updating $\mathbf{\Phi}^{(m)}_{\text{crt}}, m = 1, \ldots, V$, based on (4.28). Moreover, we determine the gradient for updating the actor network based on (4.34). We use the Adam optimizer [124] with a learning rate of α to update the learnable parameters of the actor and critic networks based on the aforementioned gradients. In addition, the following techniques are employed in our training algorithm to improve the efficiency of policy learning:

- *Exploration noise*: We add an exploration noise $\boldsymbol{\varrho}_{\text{epl}}$ to the action determined by the

Algorithm 6 Deep-GRAIL: Training Algorithm

Set episode counter $t \leftarrow 0$.
Initialize $\boldsymbol{\Phi}_{\text{act}}$, $\boldsymbol{\Phi}_{\text{crt}}^{(m)}, m = 1, \ldots, V$.
Execute the AO algorithm for D episodes and store the system transition tuples in the demonstration replay.
Perform random exploration for $T_{\text{warm-up}}$ episodes and store the system transition tuples in the experience replay.
while $t \leq T_{\max}$ **do**
  Observe the CSI and $\mathcal{I}_n(t)$ of the users.
  Initialize $c_i(0) = \mathbb{1}(i \in \mathcal{I}(t))\frac{1}{|\mathcal{I}(t)|}$, $p_{n,i}(0) = \mathbb{1}(i \in \mathcal{I}_n(t))\frac{1}{|\mathcal{I}_n(t)|}$, $i \in \mathcal{I}, n \in \mathcal{N}$.
  Initialize $\boldsymbol{\Psi}(0)$ and $\boldsymbol{b}(0)$ based on random initialization.
  Initialize $\tau \leftarrow 1$.
  while $\tau \leq \tau^{\max}$ **do**
    Determine the action $\boldsymbol{a}(\tau) \leftarrow \pi_{\boldsymbol{\Phi}_{\text{act}}}(\boldsymbol{s}(\tau)) + \boldsymbol{\varrho}_{\text{epl}}$.
    Observe the reward $r(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))$.
    Obtain the next state $\boldsymbol{s}(\tau + 1)$ and store the tuple $(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau), r(\tau), \boldsymbol{s}(\tau + 1))$ in the experience replay.
    Sample M_D transition tuples from the demonstration replay and experience replay, respectively.
    Determine the gradients for updating the actor and critic networks based on (4.34) and (4.28), respectively.
    Update $\boldsymbol{\Phi}_{\text{act}}$, $\boldsymbol{\Phi}_{\text{crt}}^{(m)}, m = 1, \ldots, V$, using the Adam optimizer and the soft update in (4.35).
    $\tau \leftarrow \tau + 1$.
  end while
  $t \leftarrow t + 1$.
end while

actor network during the training phase to facilitate the exploration of the learning agent. The elements in $\boldsymbol{\varrho}_{\text{epl}}$ are generated from the Gaussian distribution with zero mean and variance σ_{epl}^2[5]. In addition, the learning agent randomly explores the state and action spaces for $T_{\text{warm-up}}$ episodes before updating the learnable parameters.

- *Delayed actor network update*: Delaying the update of the actor network can alleviate the impact of overestimation of the critic network on policy learning [56]. In the

[5]Note that we scale the values of the elements in $\boldsymbol{a}(\tau)$ that correspond to the beamforming and IRS phase shift variables to be between -1 and 1 in our implementation. Hence, we can generate the exploration noise for all elements in $\boldsymbol{a}(\tau)$ using the same Gaussian distribution since they all have the same range of magnitudes.

Algorithm 7 Deep-GRAIL: Online Execution Algorithm for Time Slot t

Obtain the CSI and video tile requests of the users.
Initialize $c_i(0) = \mathbb{1}(i \in \mathcal{I}(t))\frac{1}{|\mathcal{I}(t)|}$, $p_{n,i}(0) = \mathbb{1}(i \in \mathcal{I}_n(t))\frac{1}{|\mathcal{I}_n(t)|}$, $i \in \mathcal{I}, n \in \mathcal{N}$.
Initialize $\boldsymbol{\Psi}(0)$ and $\boldsymbol{b}(0)$ based on random initialization.
Initialize $\tau \leftarrow 1$.
while $\tau \leq \tau^{\max}$ **do**
    Determine the action $\boldsymbol{a}(\tau) \leftarrow \pi_{\boldsymbol{\Phi}_{\text{act}}}(\boldsymbol{s}(\tau))$.
    Obtain the next state $\boldsymbol{s}(\tau + 1)$.
    $\tau \leftarrow \tau + 1$.
end while
Retrieve $\boldsymbol{b}(t)$, $\boldsymbol{\Psi}(t)$, $\boldsymbol{c}(t)$, $\boldsymbol{p}(t)$, and $\boldsymbol{v}(t)$ from $\boldsymbol{a}(\tau^{\max})$.

proposed Deep-GRAIL algorithm, we update the actor network every δ training iterations ($\delta > 1$), while the critic networks are updated in each iteration.

- *Soft learnable parameter update*: The soft update technique can stabilize the training process and prevent divergence. Let $\boldsymbol{\Phi}_{\text{crt}}^{(m)'}, m = 1, \ldots, V$, and $\boldsymbol{\Phi}'_{\text{act}}$ denote the new parameters of the critic and actor networks that we obtain based on the gradients in (4.28) and (4.34), respectively. We update the learnable parameters with the following soft update rule:

$$\begin{aligned} \boldsymbol{\Phi}_{\text{act}} &\leftarrow \kappa \boldsymbol{\Phi}'_{\text{act}} + (1 - \kappa)\boldsymbol{\Phi}_{\text{act}}, \\ \boldsymbol{\Phi}_{\text{crt}}^{(m)} &\leftarrow \kappa \boldsymbol{\Phi}_{\text{crt}}^{(m)'} + (1 - \kappa)\boldsymbol{\Phi}_{\text{crt}}^{(m)}, \ m = 1, \ldots, V, \end{aligned} \tag{4.35}$$

  where κ is a constant and is between zero and one.

4.3.5 Online Execution Algorithm

The algorithm for online execution is illustrated in **Algorithm 7**. Given the CSI and video tile requests of the users, the proposed Deep-GRAIL algorithm is executed for $\tau^{\max}$ iterations to obtain the solutions. In the τ-th iteration, we feed state $\boldsymbol{s}(\tau)$ into the actor network and determine the action $\boldsymbol{a}(\tau)$. Then, the next state $\boldsymbol{s}(\tau + 1)$ is observed. We repeat this process iteratively until the maximum decision epoch $\tau^{\max}$ is reached. Compared

with the training algorithm, the online execution algorithm has a lower computational complexity since the update of the learnable parameters and the demonstration replay (i.e., the execution of the AO algorithm) are not required during online execution.

4.4 RavNet: Proposed DNN for IRS-aided RS VR Streaming Systems

In this section, we propose RavNet, a DNN that we design for policy learning in the considered IRS-aided RS VR streaming system. With the help of the DCO layer, we are able to integrate convex optimization as one of the DNN layers in RavNet. When combined with the proposed Deep-GRAIL algorithm, RavNet is capable of learning the policy efficiently, and meanwhile satisfying the constraints in problem (4.18).

4.4.1 Input Pre-processing

We first construct two three-dimensional (3-D) matrices, namely $\boldsymbol{S}^{(1)}(\tau)$ and $\boldsymbol{S}^{(2)}(\tau)$, from state $\boldsymbol{s}(\tau)$. $\boldsymbol{S}^{(1)}(\tau)$ collects the information about the channel, beamforming vectors, and IRS phase shifts. $\boldsymbol{S}^{(2)}(\tau)$ collects the information about the video tile requests, RS parameters, and bitrate selections.

Construction of $\boldsymbol{S}^{(1)}(\tau)$

$\boldsymbol{S}^{(1)}(\tau)$ is a 3-D matrix of size $(L+1)N_t \times N \times 7$. For ease of presentation, we refer to the first, second, and third dimensions of the 3-D matrix as row, column, and depth, respectively. $\boldsymbol{S}^{(1)}[x, y, z]$ returns the element in the x-th row, y-th column, and z-th depth of matrix $\boldsymbol{S}^{(1)}(\tau)$. $\boldsymbol{S}^{(1)}[:, y, z], z = 1, \ldots, 7$, returns the y-th column vector in the z-th depth of $\boldsymbol{S}(\tau)$. $\boldsymbol{S}^{(1)}[:, :, z], z = 1, \ldots, 7$, returns the two-dimensional (2-D) matrix of size $(L+1)N_t \times N$ in the z-th depth of $\boldsymbol{S}^{(1)}$. The elements in the first and second depths of $\boldsymbol{S}^{(1)}(\tau)$ are constructed from the real and imaginary parts of the CSI of the users,

respectively. We have

$$\boldsymbol{S}^{(1)}[:,n,1](\tau) = \Big(\Re\left\{\boldsymbol{h}_{n,D}\right\}, \Re\left\{\text{vec}(\text{diag}(\boldsymbol{h}_{n,R}^{H})\boldsymbol{G})\right\}\Big),\ n \in \mathcal{N}. \tag{4.36}$$

and

$$\boldsymbol{S}^{(1)}[:,n,2](\tau) = \Big(\Im\left\{\boldsymbol{h}_{n,D}\right\}, \Im\left\{\text{vec}(\text{diag}(\boldsymbol{h}_{n,R}^{H})\boldsymbol{G})\right\}\Big),\ n \in \mathcal{N}. \tag{4.37}$$

The remainder of $\boldsymbol{S}^{(1)}(\tau)$ is constructed from the beamforming vectors and IRS phase shifts. We align the beamforming vectors and IRS phase shifts with the CSI of their corresponding subchannels in the depth dimension of matrix $\boldsymbol{S}^{(1)}(\tau)$. By doing this, we allow the DNN to learn from the *positional information* (e.g., each beamforming or IRS phase shift variable is linked to a particular subchannel) in matrix $\boldsymbol{S}^{(1)}(\tau)$.

Based on the aforementioned rationale, the elements in the 3rd and 4th depth of $\boldsymbol{S}^{(1)}(\tau)$ are obtained from the beamforming vector of the common message chosen in the previous decision epoch, i.e., $\boldsymbol{b}_0(\tau-1)$. For $n \in \mathcal{N}$, we have

$$\boldsymbol{S}^{(1)}[:,n,3](\tau) = \Big(\underbrace{\Re\left\{\boldsymbol{b}_0(\tau-1)\right\}, \ldots, \Re\left\{\boldsymbol{b}_0(\tau-1)\right\}}_{L+1}\Big), \tag{4.38}$$

and

$$\boldsymbol{S}^{(1)}[:,n,4](\tau) = \Big(\underbrace{\Im\left\{\boldsymbol{b}_0(\tau-1)\right\}, \ldots, \Im\left\{\boldsymbol{b}_0(\tau-1)\right\}}_{L+1}\Big). \tag{4.39}$$

The elements in the 5th and 6th depth of $\boldsymbol{S}^{(1)}(\tau)$ are obtained from the beamforming vectors of the private messages chosen in the previous decision epoch, i.e., $\boldsymbol{b}_n(\tau-1)$. For $n \in \mathcal{N}$, we have

$$\boldsymbol{S}^{(1)}[:,n,5](\tau) = \Big(\underbrace{\Re\left\{\boldsymbol{b}_n(\tau-1)\right\}, \ldots, \Re\left\{\boldsymbol{b}_n(\tau-1)\right\}}_{L+1}\Big), \tag{4.40}$$

and

$$\boldsymbol{S}^{(1)}[:,n,6](\tau)=\Big(\underbrace{\Im\left\{\boldsymbol{b}_n(\tau-1)\right\},\ldots,\Im\left\{\boldsymbol{b}_n(\tau-1)\right\}}_{L+1}\Big). \tag{4.41}$$

The elements in the last depth of $\boldsymbol{S}^{(1)}(\tau)$ are determined by the IRS phase shifts chosen in the previous decision epoch. For $n\in\mathcal{N}$, we have

$$\boldsymbol{S}^{(1)}[:,n,7](\tau)=\Big(\underbrace{0,\ldots,0}_{N_t},\underbrace{\psi_1(\tau-1),\ldots,\psi_1(\tau-1)}_{N_t},\ldots,\underbrace{\psi_L(\tau-1),\ldots,\psi_L(\tau-1)}_{N_t}\Big). \tag{4.42}$$

Construction of $\boldsymbol{S}^{(2)}(\tau)$

$\boldsymbol{S}^{(2)}(\tau)$ is a 3-D matrix of size $N_x\times N_y\times 2N$. We construct $\boldsymbol{S}^{(2)}(\tau)$ in such a way that the 2-D matrices $\boldsymbol{S}^{(2)}[:,:,n]$ and $\boldsymbol{S}^{(2)}[:,:,n+N]$ show the achievable bitrates of the video tiles requested by user $n\in\mathcal{N}$ obtained from the common and private messages, respectively, based on the control variables determined in the previous decision epoch. We have

$$\boldsymbol{S}^{(2)}[x,y,z](\tau)=\begin{cases}\mathbb{1}\left\{i\in\mathcal{I}_z\right\}c_i(\tau-1)R^{\mathrm{c}}(\tau-1)), & z=1,\ldots,N,\\ \mathbb{1}\left\{i\in\mathcal{I}_{z-N}\right\}p_{z-N,i}(\tau-1)R^{\mathrm{p}}_{z-N}(\tau-1), & z=N+1,\ldots,2N,\end{cases} \tag{4.43}$$

where $i=y+(x-1)N_x$ for $x=1,\ldots N_x$, and $y=1,\ldots,N_y$. In fact, the value of i corresponds to the index of the video tile located in the x-th row and the y-th column of the 360-degree video frame.

4.4.2 Actor Network Structure

The actor network takes both $\boldsymbol{S}^{(1)}(\tau)$ and $\boldsymbol{S}^{(2)}(\tau)$ as inputs to determine the control variables. The proposed actor network structure tackles the constraints of problem (4.18) during the policy learning process. As shown in Fig. 4.3, we use the following DNN modules in the actor network:

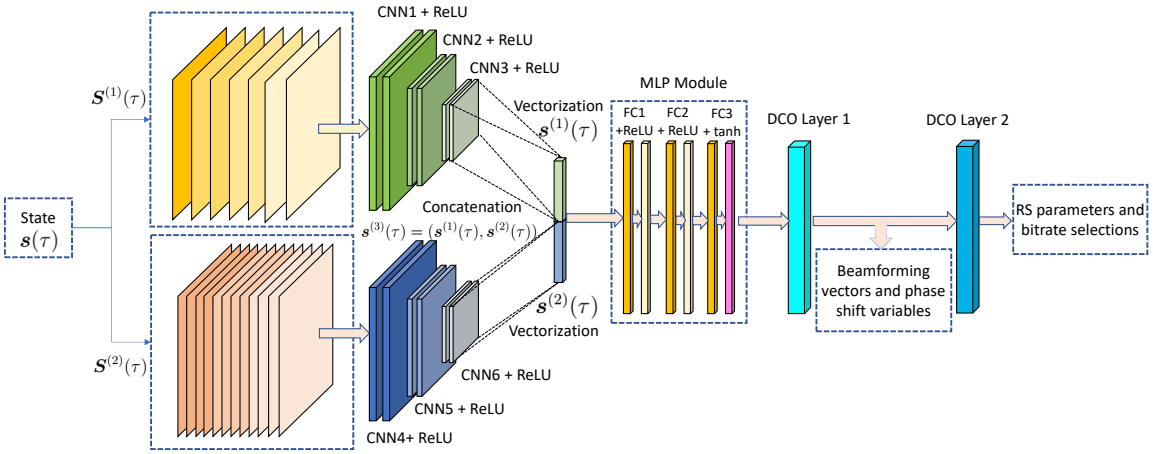


Figure 4.3: The network architecture of the actor network in RavNet. The actor network takes $\boldsymbol{S}^{(1)}(\tau)$ and $\boldsymbol{S}^{(2)}(\tau)$ as input, and determines the action $\boldsymbol{a}(\tau)$.

Convolutional Neural Network (CNN) Module

First, $\boldsymbol{S}^{(1)}(\tau)$ is fed into three CNN layers with kernel sizes $k_1 \times k_1$, $k_2 \times k_2$, $k_3 \times k_3$, and channel numbers ch_1, ch_2, and ch_3, respectively. Each CNN layer is followed by an ReLU activation layer. The output of the last CNN layer is vectorized into a vector, which is denoted by $\boldsymbol{s}^{(1)}(\tau)$.

We employ another network module comprising three CNN layers to process $\boldsymbol{S}^{(2)}(\tau)$. The kernel sizes of these three CNN layers are given by $k_4 \times k_4$, $k_5 \times k_5$, and $k_6 \times k_6$, while their channel numbers are ch_4, ch_5, and ch_6, respectively. We also apply an ReLU activation layer after each of the CNN layers. The output of the last CNN layer is reshaped into a vector, which we denote as $\boldsymbol{s}^{(2)}(\tau)$.

The CNN module is an important component in the proposed actor network because (a) it can learn from the positional information encoded in the depth dimension of $\boldsymbol{S}^{(1)}(\tau)$, and (b) it can capture the spatial correlation between the video tile requests of the users in $\boldsymbol{S}^{(2)}(\tau)$ and obtain the knowledge of the shared interests.

MLP Module

We concatenate $\boldsymbol{s}^{(1)}(\tau)$ and $\boldsymbol{s}^{(2)}(\tau)$ together to obtain a new vector $\boldsymbol{s}^{(3)}(\tau)$. That is, $\boldsymbol{s}^{(3)}(\tau) = (\boldsymbol{s}^{(1)}(\tau), \boldsymbol{s}^{(2)}(\tau))$. We feed $\boldsymbol{s}^{(3)}(\tau)$ into an MLP module with three FC layers, two ReLU activation layers, and one `tanh` activation layer to obtain the beamforming variables $\boldsymbol{b}'(\tau)$ and IRS phase shifts $\boldsymbol{\psi}'(\tau)$. We define $\boldsymbol{a}'(\tau) = (\boldsymbol{b}'(\tau), \boldsymbol{\psi}'(\tau))$.

DCO Layers

In order to satisfy the constraints in problem (4.18), we determine the projection of $\boldsymbol{a}'(\tau)$ onto the feasible set of problem (4.18) by solving the following optimization problem with the RS parameters and bitrate selection given by $\boldsymbol{c}(\tau-1)$, $\boldsymbol{p}_n(\tau-1)$, and $\boldsymbol{v}_n(\tau-1)$, respectively:

$$\begin{aligned} \underset{\boldsymbol{a}(\tau)}{\text{minimize}} \quad & ||\boldsymbol{a}(\tau) - \boldsymbol{a}'(\tau)||^2 \\ \text{subject to} \quad & \text{constraints C2, C8, C9.} \end{aligned} \tag{4.44}$$

Note that constraint C9 can be satisfied by using the outputs of the DNN modules as the phase shift values. Problem (4.44) can be transformed into a convex optimization problem by applying quadratic transform [10] to the common and private rate expressions. In the proposed RavNet, we solve problem (4.44) using the DCO layer [132]. Compared with conventional convex solvers (e.g., CVX [142]), the DCO layer can be integrated as a layer in RavNet. In addition, it can solve problem (4.44) efficiently in a batch-wise manner, which significantly facilitates the training process. We denote the feasible beamforming and IRS phase shift solutions obtained by solving problem (4.44) as $\boldsymbol{b}(\tau)$ and $\boldsymbol{\psi}(\tau)$, respectively.

We then feed $\boldsymbol{b}(\tau)$ and $\boldsymbol{\psi}(\tau)$ into a second DCO layer which solves the following optimization problem to obtain the RS parameters and bitrate selections:

$$\begin{aligned} \underset{\substack{\boldsymbol{v}_n, n\in\mathcal{N}, \\ c_i, i\in\mathcal{I}, \\ p_{n,i}, i\in\mathcal{I}_n, n\in\mathcal{N}}}{\text{maximize}} \quad & \sum_{n\in\mathcal{N}} \left(\sum_{i\in\mathcal{I}_n} v_{n,i} - \kappa^{\text{intra}} \ell_n^{\text{intra}} \right) \\ \text{subject to} \quad & \text{constraints C1, C3}-\text{C8.} \end{aligned} \tag{4.45}$$

Note that the intra-frame quality switch loss ℓ_n^{intra} is a convex function with respect to the bitrate selection variables in vector $\boldsymbol{v}_n$. Moreover, $R^{\text{c}}(\tau)$ and $R_n^{\text{p}}(\tau), n \in \mathcal{N}$, in problem (4.45) can be determined given $\boldsymbol{b}(\tau)$ and $\boldsymbol{\psi}(\tau)$. All constraints except constraint C1 are affine constraints. We relax constraint C1 as

$$v_1 \leq v_{n,i} \leq v_M,\ i \in \mathcal{I}_n,\ n \in \mathcal{N}. \tag{4.46}$$

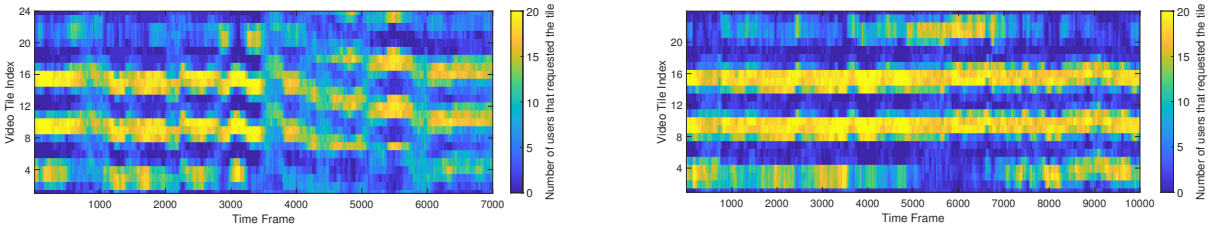
The relaxed problem is a convex optimization problem and can be solved using the DCO layer. We round down the solution of $v_{n,i}, i \in \mathcal{I}_n$, $n \in \mathcal{N}$, to the nearest feasible solution.

4.4.3 Critic Network Structure

The proposed critic network has a similar structure as the actor network. Since the critic network approximates the Q-value, the layers for obtaining the feasible actions in the actor network are not required in the critic network. This leads to the following two modifications: (a) the DCO layers are not present in the critic network, and (b) the `tanh` activation layer in the MLP module is replaced by the ReLU activation layer to generate the Q-value.

4.5 Performance Evaluation

We consider a 10 m × 10 m × 3.5 m indoor facility for VR streaming as illustrated in Fig. 4.1. Each user is designated a 2.7 m × 2.7 m area [137]. The base station is installed at the center of the ceiling, and the IRS is installed on one side of the wall at the midpoint between the ceiling and the floor. We assume all channels, including the channels between the base station and the users, are LoS based on the aforementioned deployments of the base station and the IRS. We consider the presence of LoS channels in our simulations to investigate the full potential of the proposed IRS-aided RS VR streaming system. We assume a carrier frequency of 60 GHz as this value is used in several commercial wireless



(a) 360-degree video streaming session 1

(b) 360-degree video streaming session 2

Figure 4.4: Visualization of the video tile requests obtained from two VR streaming sessions in the real-world dataset. Each video frame is divided into 24 video tiles. The x-axis indicates the video frame, while the time duration of each frame is 1 sec. The y-axis shows the indices of the video tiles (i.e., from 1 to 24). The color of the grid centered at (x, y) indicates the number of users that requested the y-th tile in the x-th video frame.

VR systems, see, e.g., [143, 144]. Let $d_{n,D}$, $d_{n,R}$, and d_0 denote the distance between the base station and user n, the distance between the IRS and user n, and the distance between the base station and the IRS, respectively. We determine the CSI of the direct and reflected channels by $\boldsymbol{h}_{n,D} = (\frac{\nu}{4\pi d_{n,D}})^{\zeta}\widehat{\boldsymbol{h}}_{n,D}$, $\boldsymbol{h}_{n,R} = (\frac{\nu}{4\pi d_{n,R}})^{\zeta}\widehat{\boldsymbol{h}}_{n,R}$, and $\boldsymbol{G} = (\frac{\nu}{4\pi d_0})^{\zeta}\widehat{\boldsymbol{G}}$, where ν is the wavelength of the carrier signal and ζ is the pathloss exponent. The elements in $\widehat{\boldsymbol{h}}_{n,D}$, $\widehat{\boldsymbol{h}}_{n,R}$, and $\widehat{\boldsymbol{G}}$ are complex Gaussian distributed with zero mean and unit variance. The other simulation parameter settings are given in Table 4.1.

To properly model the pattern of the video tiles requested by the users during the VR streaming session, we use the real-world dataset from [133] to determine the video tile requests in our simulations. The dataset from [133] includes the head movements of 20 users during multiple real-world VR streaming sessions. The head movement record of a particular user is used to determine the FoV and the video tiles requested by this user. We divide each 360-degree video frame into 24 tiles, with $N_x = 4$ and $N_y = 6$. The FoV of each user covers 110 degrees in horizontal direction and 90 degrees in vertical direction of the video frame. In Fig. 4.4, we visualize the video tile requests that we determined based on two VR streaming sessions from the real-world dataset [133]. We use FFmpeg [145] to encode the 360-degree video into different bitrates as given by set $\mathcal{S} = \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$ Mbps.

Table 4.1: Simulation Parameters for Performance Evaluation

Parameter	Value
Bandwidth for downlink W	1 GHz
Path loss exponent ζ	2.29 [146]
Maximum transmit power $P^{\max}$	1 Watt
Noise power	-174 dBm/Hz
Time duration of each video tile T_v	1 sec
Downlink transmission window T_{DL}	10 ms
Coefficient for inter-frame quality switch loss κ^{intra}	10
Number of decision epochs per time slot (i.e., per episode) $\tau^{\max}$	50
Learning rate α	5×10^{-4}
Minibatch size M_D	512
Number of critic networks V	6
Value of q for q-step return	5
Kernel size of the CNN layers	2×2
Number of channels of the CNN layers ch_1, ch_2, ch_3, ch_4, ch_5, ch_6	16, 16, 32, 16, 16, 32
Hidden dimensionality of the MLP module	1024
Number of FC layers in the MLP module	3
Coefficients for training loss ω_1, ω_2	10^{-3} , 1
Coefficient for soft update κ	5×10^{-3}
Discount factor γ	0.95

We conduct the simulation using a computing server with an Intel Core i5-9500 @ 3.0 GHz CPU, and an NVIDIA GeForce RTX 2070 GPU with 8 GB memory. We compare the performance of the following baseline systems and algorithms:

- **IRS-aided RS VR streaming system with AO algorithm**: We use the AO algorithm presented in Appendix B to solve problem (4.18).

- **IRS-aided RS VR streaming system with SL algorithm**: In this algorithm, we train a DNN module using SL to minimize the mean squared error between its output and the solution of the AO algorithm presented in Appendix B. This DNN module uses the same network structure as the proposed actor network.

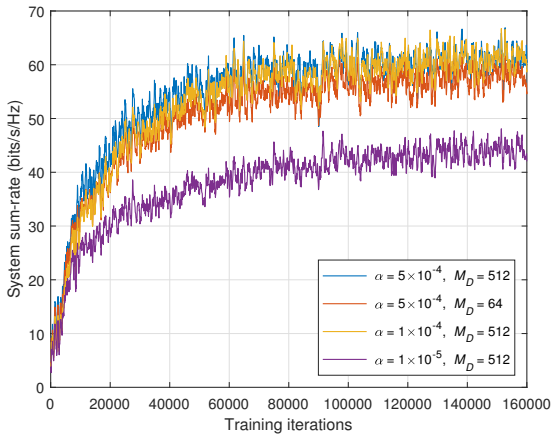


Figure 4.5: Convergence of the proposed Deep-GRAIL algorithm. We set $N_t = 6$, $N = 6$, and $L = 100$.

- **IRS-aided RS-NOUM system [62]**: We extend the RS-NOUM system proposed in [62] by including an IRS. In this system, the information of the requested tiles are sent to all users via multicast. Each user also receives a dedicated unicast message regarding its requested tiles. The multicast and unicast messages are combined using RS. We solve the sum-rate maximization problem for the resulting system with the constraints of problem (4.18) using an AO-based algorithm with weighted minimum mean square error (WMMSE), SDR, and convex optimization.

- **IRS-aided multiuser system without RS (IRS-aided MU system) [63]**: In this system, the requested video tiles are sent to the users via unicast without RS. We solve the sum-rate maximization problem for the resulting system with the constraints of problem (4.18) using an AO-based algorithm with FP, SDR, and convex optimization.

4.5.1 Convergence of the Deep-GRAIL Algorithm

We first investigate the convergence of the proposed Deep-GRAIL algorithm. We show the achievable system sum-rate of the Deep-GRAIL algorithm versus the number of training iterations in Fig. 4.5. We observe that with a properly chosen learning rate α, e.g.,

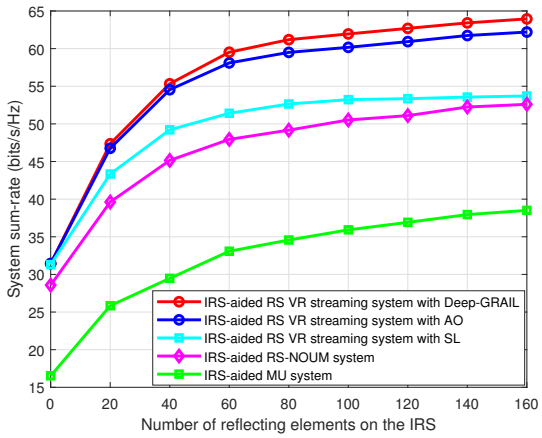


Figure 4.6: System sum-rate versus the number of reflecting elements L. We set $N_t = N = 6$. Note that $L = 0$ represents the system without an IRS.

5×10^{-4}, the proposed Deep-GRAIL algorithm can efficiently improve the learned policy. The results in Fig. 4.5 also show that setting α to be too small, e.g., 10^{-5}, can lead to slow convergence and inefficient policy learning. In addition, we observe that increasing the minibatch size M_D from 64 to 512 leads to a higher system sum-rate.

4.5.2 Achievable System Sum-Rate

In Fig. 4.6, we vary the number of reflecting elements L and investigate the system sum-rate. We observe that the performance improvement of the proposed IRS-aided RS VR streaming system with Deep-GRAIL algorithm over the same system with AO algorithm increases with the value of L. This is because when the IRS phase shift subproblem is solved using FP and SDR, Gaussian randomization is needed to obtain IRS phase shift matrix, which may incur significant performance degradation [11]. The proposed Deep-GRAIL algorithm avoids such performance loss since (a) SDR is not required for the Deep-GRAIL algorithm, and (b) the imitation loss in (4.33) prevents the learning agent from being affected by the suboptimality of the AO algorithm with SDR. In particular, when $L = 160$, the IRS-aided RS VR streaming system with the proposed Deep-GRAIL algorithm achieves a system sum-rate that is 2.8%, 19.1%, 21.6%, and 66.1% higher than

that of the IRS-aided RS VR streaming system with AO algorithm, IRS-aided RS VR streaming system with SL algorithm, IRS-aided RS-NOUM system, and IRS-aided MU system, respectively.

In addition, in Fig. 4.6, $L = 0$ implies a system without IRS. We observe that all considered systems benefit significantly from having an IRS present for improving the system sum-rate. For the IRS-aided RS VR streaming system with Deep-GRAIL algorithm, deploying an IRS with $L = 100$ reflecting elements results in a system sum-rate improvement of 91.44% compared to the same system without IRS. This is due to the SINR improvement achieved with the additional propagation channels created by the IRS. Furthermore, for the proposed IRS-aided RS VR streaming system, since the common rate is determined by the user experiencing the minimum SINR (as shown in (4.7)), the additional DoF introduced by the IRS are implicitly exploited to increase the rate of the common message. In particular, our results show that the average of the achievable rate of the common message of the users, i.e., R^{c} in (4.7), in the proposed IRS-aided RS VR streaming system with $L = 100$ reflecting elements is 12.81 bits/s/Hz, while only an average of 5.61 bits/s/Hz is achieved in the same system without IRS. Therefore, using an IRS with $L = 100$ reflecting elements increases the common rate by 128.3%, which allows more data to be transmitted via the common message to exploit the shared interests and improve the QoE of the users. Our results demonstrate the benefits of IRS for mitigating the performance bottleneck of RS caused by the user experiencing the minimum SINR.

In Fig. 4.7, we show the system sum-rate versus the number of users N. We set $N_t = 6$ and $L = 100$. We observe that the performance gains of the IRS-aided RS VR streaming system over the IRS-aided RS-NOUM and IRS-aided MU systems become more pronounced with more users. With more users, a particular tile is more likely to be requested by multiple users, and therefore there are more shared tile requests of the users to be exploited by the IRS-aided RS VR streaming system. When $N = 8$, the IRS-aided RS VR streaming system with the proposed Deep-GRAIL algorithm achieves a system sum-rate that is 2.6%, 25.2%, 26.7%, and 90.8% higher than that of the IRS-aided RS

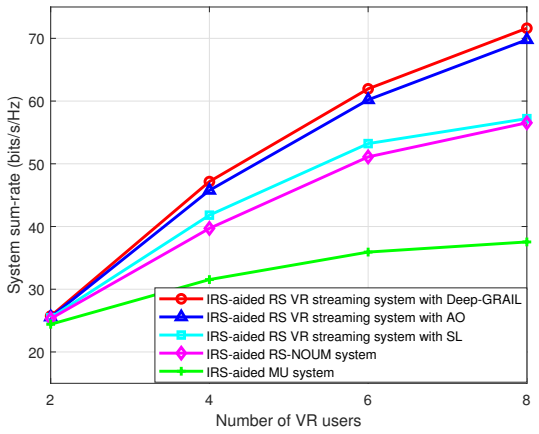


Figure 4.7: System sum-rate versus the number of VR users N. We set $N_t = 6$ and $L = 100$.

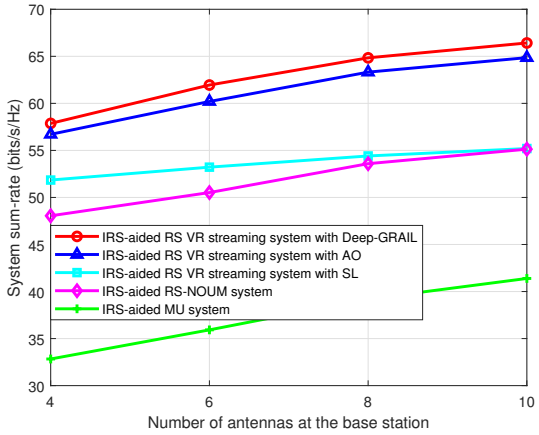


Figure 4.8: System sum-rate versus the number of antennas N_t at the base station. We set $N = 6$ and $L = 100$.

VR streaming system with AO algorithm, IRS-aided RS VR streaming system with SL algorithm, IRS-aided RS-NOUM system, and IRS-aided MU system, respectively.

In Fig. 4.8, we show the system sum-rate versus the number of antennas N_t at the base station. All algorithms except the IRS-aided RS VR streaming system with SL algorithm exhibit a similar performance gain as the number of antennas increases. The IRS-aided RS VR streaming system with SL algorithm suffers a larger sum-rate degradation when N_t becomes larger. This is caused by the increase in the mean squared error between

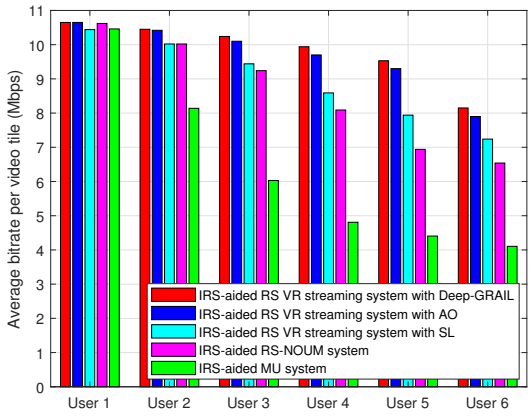


Figure 4.9: Average achievable bitrate for each user. We set $N_t = N = 6$ and $L = 100$.

the beamforming vectors determined by the SL algorithm and the beamforming vectors obtained by the AO algorithm. When $N_t = 10$, the IRS-aided RS VR streaming system with the proposed Deep-GRAIL algorithm achieves a 2.3%, 20.3%, 20.5%, and 60.5% higher system sum-rate than the IRS-aided RS VR streaming system with AO algorithm, IRS-aided RS VR streaming system with SL algorithm, IRS-aided RS-NOUM system, and IRS-aided MU system, respectively. Compared with the baseline schemes, the performance gain of the IRS-aided RS VR streaming system is due to the optimization of the RS parameters, i.e., $c_i, i \in \mathcal{I}$, given the video tile requests of the users. Through the optimization of c_i, the base station can properly determine the video tiles that should be included in the common message, as well as the proportions of the common message allocated to them, such that the utility is maximized.

4.5.3 Bitrate Allocation per User

In Fig. 4.9, we show the average bitrate per video tile for each user. We set $N_t = N = 6$ and $L = 100$. We sort the users in descending order of their average bitrates. That is, the user with the highest average bitrate is referred to as user 1, while the user with the lowest average bitrate is referred to as user 6. The results in Fig. 4.9 show that the users achieve higher bitrates in the IRS-aided RS-enabled systems compared to the IRS-aided

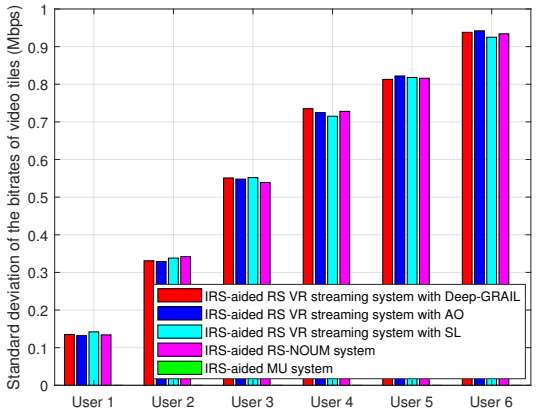


Figure 4.10: Standard deviation of the bitrates for the video tiles. We set $N_t = N = 6$ and $L = 100$.

MU system. This is because, with RS, the common message can be exploited to improve the QoE of multiple users simultaneously when those users have shared video tile requests.

In Fig. 4.10, we show the standard deviation of the bitrates for the video tiles received by the users. For the IRS-aided MU system, the standard deviation is zero. Due to the absence of a common message and the consideration of the intra-frame quality switch loss in the objective function $u(t)$, solving the bitrate selection subproblem in the IRS-aided MU system causes the bitrates for all tiles requested by a particular user to be identical. For the RS-enabled systems, those users with lower average bitrates (e.g., users 5 and 6) experience higher standard deviations of the bitrates of the received video tiles. This is because by exploiting the common message, those users can obtain higher bitrates for video tiles that are requested by multiple users than for video tiles that are requested only by an individual user.

4.5.4 Runtime Comparison

In Table 4.2, we compare the average runtime of different algorithms per time slot. We observe that the average runtimes of the learning-based algorithms, i.e., the Deep-GRAIL

Table 4.2: Average Runtime Comparison for Different Schemes

Parameter Settings	$N_t = 6$, $N = 4$, $L = 100$	$N_t = 6$, $N = 6$, $L = 100$	$N_t = 6$, $N = 6$, $L = 160$	$N_t = 10$, $N = 6$, $L = 100$
IRS-aided RS VR streaming system with Deep-GRAIL algorithm	8.52 sec	10.38 sec	13.71 sec	8.65 sec
IRS-aided RS VR streaming system with SL algorithm	0.75 sec	1.15 sec	1.64 sec	0.85 sec
IRS-aided RS VR streaming system with AO algorithm	840.43 sec	981.70 sec	3799.18 sec	1167.50 sec
IRS-aided RS-NOUM system	778.72 sec	895.94 sec	3569.42 sec	1094.63 sec
IRS-aided MU system	561.48 sec	612.18 sec	2501.98 sec	792.84 sec

and SL algorithms, are lower than that of the AO algorithms. Moreover, the increases in runtime with respect to the value of N_t, N, and L for the learning-based algorithms are less significant than the AO algorithms. This is because the computationally expensive processes needed for solving the beamforming and IRS phase shift subproblems using FP, WMMSE, and SDR are not needed in the learning-based algorithms. In particular, when $N_t = 6$, $N = 6$, and $L = 160$, the average runtime of the Deep-GRAIL algorithm is only 0.36%, 0.38%, and 0.55% of the average runtimes of the IRS-aided RS VR streaming system with AO algorithm, IRS-aided RS-NOUM system, and IRS-aided MU system, respectively. The average runtime of the SL algorithm is lower than that of the Deep-GRAIL algorithm since the Deep-GRAIL algorithm needs to be executed for $\tau^{\max}$ decision epochs per time slot, while the SL algorithm is not an iterative algorithm and only needs to be executed once per time slot.

4.6 Summary

In this chapter, we proposed a novel IRS-aided RS VR streaming system, in which the shared interests of the VR users were exploited via RS to improve the QoE of 360-degree

video streaming. We used IRS to create additional propagation channels, and improve the performance of RS by increasing the minimum SINR experienced by the common message at different users. We formulated the joint optimization of the RS parameters, IRS phase shifts, beamforming vectors, and bitrate selection as a mixed-integer nonlinear programming problem, in which the intra-frame quality switch loss and per-user per-tile QoE requirement were taken into consideration. We proposed the Deep-GRAIL algorithm and RavNet, in which imitation learning, actor-critic method, DDPG, and DCO layers were employed to efficiently solve the formulated problem. Simulation results based on a real-world dataset showed that the DoF introduced by RS and IRS can be efficiently exploited by the proposed Deep-GRAIL algorithm to achieve a higher system sum-rate compared to that of the IRS-aided RS-NOUM and IRS-aided MU systems. The performance improvement of the proposed IRS-aided RS VR streaming system became more pronounced in the presence of more shared video tile requests. Our simulation results also revealed the respective contribution of RS and IRS to the performance gain achieved with the proposed IRS-aided RS VR streaming system. Through a runtime comparison with existing AO algorithms, we demonstrated the advantages of the proposed learning-based Deep-GRAIL algorithm in terms of runtime reduction, and its suitability for potential deployment in practical VR streaming systems.

Chapter 5

Conclusions and Future Work

In Section 5.1 of this chapter, we conclude this thesis by summarizing the key results and contributions of our work. The limitations and future work of this thesis are presented in Section 5.2.

5.1 Conclusions

During the past two decades, conventional optimization methods, such as convex optimization and AO, have been extensively applied to design resource allocation algorithms for wireless systems. While the conventional optimization methods solve the resource allocation problems by exploiting the (hidden) convexity, the resulting computational complexity can increase significantly with respect to the DoF in wireless systems. For the B5G wireless systems with high DoF, the computational complexity of conventional optimization methods can be prohibitively high, making the algorithms difficult to implement in practical systems.

In this thesis, we investigated the DRL-based resource allocation algorithms design for B5G wireless systems, with the objective of providing both IoT devices and mobile users with satisfactory QoS. In the proposed algorithms, the policies for solving the resource allocation problems in B5G wireless systems are learned by exploiting various model-free DRL techniques, with the help of the DNN structures that we customized specifically for each resource allocation problem. Using the proposed algorithms, we were able to explore the potential of three specific physical layer and medium access control techniques, namely, GFMA, IRS, and RS, to improve the spectral efficiency of B5G wireless systems.

Designing a distributed pilot sequence selection scheme is crucial for the B5G systems to support heterogeneous IoT applications using GFMA. The lack of global information in the IoT devices has to be overcome to accommodate the user-specific QoS requirements and avoid pilot sequence selection collisions. In Chapter 2, we tackled this challenge by using the MA-DRL technique with a CTDE framework. Our approach addressed the lack of global information from the following three perspectives: (a) we exploited the global information available at the base station during the centralized training phase to facilitate policy learning, (b) we applied the factorization technique to obtain the policies that can be executed by the users distributively without requiring excessive information exchange with other users in GFMA systems, and (c) we trained the LSTM layers such that the historical transitions of the underlying MDP can be stored in the hidden states of the LSTM layers to help the policy learning under partial information. We demonstrated the effectiveness of the proposed DRL-based scheme through the comparison to the existing ACB-based and ACK-based schemes. Our work in Chapter 2 offered a DRL framework that can be applied to solve various resource allocation problems in B5G wireless systems when distributed solutions are required.

The other major contribution of this thesis was the DRL-based algorithms design we proposed in Chapters 3 and 4 for solving the mixed-integer nonconvex resource allocation problems in IRS-aided and RS systems. While AO has been applied in various existing research to optimize the DoF in IRS-aided and RS systems, the AO-based approaches may have high computational complexity, and their performance can suffer from the nonconvexity of the optimization problems. In Chapter 3, we tackled such challenges by proposing an end-to-end learning framework, in which two different DRL modules, namely, NCO and CL-DDPG, were designed to optimize the discrete user scheduling variables, and the continuous beamforming and IRS phase shift variables, respectively. These two DRL modules were integrated by the joint training in the proposed DUPB algorithm. The mixed-integer nonconvex optimization problem we formulated in Chapter 4 was more challenging due to the constraints required for the multiuser VR video streaming in RS systems. In Chapter

4, we designed the RavNet, which is a DNN module that can effectively learn the policy for improving the QoE of the users while accommodating the constraints during the learning process. What made it possible for us to tackle the constraints during policy learning was DCO, a technique that can integrate convex optimization as a layer in the DNN module. The DRL-based algorithms we proposed in Chapters 3 and 4 also offered computationally efficient solutions to other mixed-integer nonconvex resource allocation problems in B5G wireless systems.

Besides, as we discussed in Chapter 1, the learning efficiency of the DRL-based algorithms can be affected by the curse of dimensionality in B5G wireless systems. Another major contribution of Chapters 3 and 4 was to combine CL and imitation learning with the DRL techniques to overcome the curse of dimensionality. In our algorithms design, we used CL and imitation learning to extract the knowledge of the hidden convexity of the resource allocation problems based on the solutions of the conventional optimization methods. Such knowledge was exploited by the learning agent via the reward and loss function design in the proposed DRL-based algorithms. Our results demonstrated the effectiveness of using CL and imitation learning to combat the curse of dimensionality in the DRL-based algorithms design. Our work in Chapters 3 and 4 also shed light on the promising research direction of combining the DRL techniques with the human expert knowledge to improve learning efficiency in B5G wireless systems.

5.2 Limitations and Future Work

In the following, we discuss the limitations of the proposed DRL-based algorithms design, and present the potential directions for future research work.

In this thesis, we assumed that the learning agent in DRL can obtain perfect information of the state of the underlying MDP. In particular, in Chapters 3 and 4, we proposed DRL-based algorithms for B5G wireless systems with perfect CSI. However, perfect CSI may be difficult to obtain in practical wireless systems. As shown by the results in Chapter 3,

the proposed DRL-based algorithms suffer from performance degradations when perfect CSI is not available. Hence, it is worthy to study the potential methods to tackle the lack of perfect CSI in DRL-based algorithms design. Recently, several robust DRL algorithms (e.g., [147, 148]) have been proposed for policy learning under noisy state observation. Existing works on channel estimation have shown that the channel estimation error in practical systems can be modeled using complex Gaussian distributions. Therefore, it is interesting to investigate how to apply the robust DRL algorithms [147, 148] to tackle the channel estimation error in B5G wireless systems. In addition, the training algorithm and DNN structure design may need to be revisited to accommodate the imperfect CSI during the policy learning process.

Moreover, the scalability of the proposed DRL-based algorithm designs can be improved. In Chapter 2, we proposed a CTDE framework for policy learning in GFMA systems. During the centralized training phase, both the time-frequency and computational resources required for collecting information from the users and training DNN modules may increase with the number of users in GFMA systems. Besides, for the DNN modules proposed in this thesis, the number of learnable parameters needs to be increased to accommodate more users in the B5G systems. This may result in an increase in the computational complexity during the training phase in a wireless system with a large number of users. To further improve the scalability of the proposed DRL-based algorithms, it may be possible to exploit mean-field theory [149, 150] to reduce the amount of information that is required to be collected during the training phase. In addition, it is also interesting to study the feasibility of allowing the users to share a part of the learnable parameters of their DNN modules, which can efficiently reduce the computational complexity of the training phase [151, 152].

For the proposed DRL-based algorithm designs, the DNN modules need to be retrained when the settings of the wireless systems change. The changes on settings may include the changes in the number of users, the number of antennas at the base station, and the number of reflecting elements on the IRS. The retraining process can be computationally intensive

in wireless systems where the system settings are changing frequently. One approach to mitigate such issue is to apply meta-reinforcement learning [153–155]. Using meta-reinforcement learning, the learning agent learns a meta policy during the training process based on the explorations in wireless systems with different settings. The meta policy can adapt to a new system setting faster, i.e., requiring fewer iterations of retraining, than the policy learned without using meta-reinforcement learning. Therefore, it is a promising research direction to extend the proposed DRL-based algorithms by applying meta-reinforcement learning.

We now present the following potential directions for extending the system model studied in this thesis:

1. **GFMA systems with delay and energy consumption constraints**: In Chapter 2, we investigated the application of MA-DRL to improve the aggregate throughput of GFMA systems with user-specific average throughput requirements. Apart from the average throughput, there are several additional QoS requirements that may be considered in GFMA systems, including data transmission delay and energy consumption. Age of information (AoI) [156, 157] can be used to evaluate the delay performance in GFMA systems. In addition, since many IoT devices in GFMA may rely on batteries, optimizing the energy efficiency [158–160] in GFMA systems is an interesting research topic. Hence, extending the MA-DRL based algorithm proposed in Chapter 2 to optimize the transmission delay and energy efficiency in GFMA systems is a promising research direction.

2. **Multicell IRS-aided systems**: To extend the work in Chapter 3, we can consider the multicell systems in which multiple IRSs are deployed to facilitate data transmission. Compared with the single-cell scenario, the beamforming vectors of base stations, as well as the phase shifts of IRSs, need to be jointly optimized to mitigate both the intra-cell and inter-cell interference in multicell IRS-aided systems [161, 162]. In addition, each base station may only obtain partial CSI of the users in multicell

IRS-aided systems. Therefore, it is worth investigating how to extend the proposed DRL-based algorithms to tackle the lack of global CSI in multicell IRS-aided systems.

3. **FoV prediction and caching in IRS-aided RS VR streaming systems**: To extend the work in Chapter 4, incorporating the FoV prediction [138] and caching [163] with the proposed Deep-GRAIL algorithm is an interesting topic for future research. The data of 360-degree video tiles can be cached at the HMDs based on the FoV prediction of the users. This allows the data of some video tiles to be transmitted to the users beforehand, and therefore those cached video tiles can be rendered immediately upon request. To accommodate these features, new DNN modules can be designed to predict the FoVs based on historical information. Moreover, the DRL-based algorithms design for joint FoV prediction and DoF optimization in multiuser IRS-aided RS VR streaming systems requires further investigation.

Bibliography

[1] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, D. Niyato, O. Dobre, and H. V. Poor, "6G Internet of things: A comprehensive survey," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 359–383, Jan. 2022.

[2] Ericsson, "Ericsson mobility report data and forecasts," Jun. 2022. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/mobility-report

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

[4] R. Ali, Y. B. Zikria, A. K. Bashir, S. Garg, and H. S. Kim, "URLLC for 5G and beyond: Requirements, enabling incumbent technologies and network intelligence," *IEEE Access*, vol. 9, pp. 67 064–67 095, Apr. 2021.

[5] T. Qiu, N. Chen, K. Li, M. Atiquzzaman, and W. Zhao, "How can heterogeneous Internet of things build our future: A survey," *IEEE Commun. Surveys & Tuts.*, vol. 20, no. 3, pp. 2011–2027, Third Quarter 2018.

[6] GoPro, "Video settings and resolutions," Feb. 2022. [Online]. Available: https://community.gopro.com/s/article/MAX-Video-Settings-and-Resolutions

[7] M. S. Elbamby, C. Perfecto, M. Bennis, and K. Doppler, "Toward low-latency and ultra-reliable virtual reality," *IEEE Network*, vol. 32, no. 2, pp. 78–84, Apr. 2018.

[8] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[9] J. C. Bezdek and R. J. Hathaway, "Convergence of alternating optimization," *Neural, Parallel Sci. Comput.*, vol. 11, pp. 351–368, Dec. 2003.

[10] K. Shen and W. Yu, "Fractional programming for communication systems − Part I: Power control and beamforming," *IEEE Trans. Signal Process.*, vol. 66, no. 10, pp. 2616–2630, May 2018.

[11] Z. Luo, W. Ma, A. M. So, Y. Ye, and S. Zhang, "Semidefinite relaxation of quadratic optimization problems," *IEEE Signal Process. Mag.*, vol. 27, no. 3, pp. 20–34, May 2010.

[12] Q. Wu and R. Zhang, "Intelligent reflecting surface enhanced wireless network: Joint active and passive beamforming design," in *Proc. of IEEE Global Commun. Conf. (GLOBECOM)*, Abu Dhabi, UAE, Dec. 2018.

[13] C. Huang, A. Zappone, G. C. Alexandropoulos, M. Debbah, and C. Yuen, "Reconfigurable intelligent surfaces for energy efficiency in wireless communication," *IEEE Trans. Wireless Commun.*, vol. 18, no. 8, pp. 4157–4170, Aug. 2019.

[14] B. Zheng and R. Zhang, "IRS meets relaying: Joint resource allocation and passive beamforming optimization," *IEEE Wireless Commun. Lett.*, vol. 10, no. 9, pp. 2080–2084, Sept. 2021.

[15] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[16] O. Vinyals *et al.*, "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, pp. 350–354, Oct. 2019.

[17] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *Proc. of Int'l Conf. on Learning Representations (ICLR)*, San Juan, Puerto Rico, May 2016.

[18] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. of Int'l Conf. on Machine Learning (ICML)*, Stockholm, Sweden, Jun. 2018.

[19] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[20] L. Lei, Y. Tan, K. Zheng, S. Liu, K. Zhang, and X. Shen, "Deep reinforcement learning for autonomous Internet of things: Model, applications and challenges," *IEEE Commun. Surveys & Tuts.*, vol. 22, no. 3, pp. 1722–1760, Third Quarter 2020.

[21] A. Feriani and E. Hossain, "Single and multi-agent deep reinforcement learning for AI-enabled wireless networks: A tutorial," *IEEE Commun. Surveys & Tuts.*, vol. 23, no. 2, pp. 1226–1252, Second Quarter 2021.

[22] 3GPP TS 38.213 V16.8.0, "Technical specification group radio access network; NR; Physical layer procedures for control (Release 16)," Jun. 2022.

[23] V. W. S. Wong, R. Schober, D. W. K. Ng, and L. Wang, *Key Technologies for 5G Wireless Systems*. Cambridge University Press, 2017.

[24] Q. Wu, S. Zhang, B. Zheng, C. You, and R. Zhang, "Intelligent reflecting surface-aided wireless communications: A tutorial," *IEEE Trans. Commun.*, vol. 69, no. 5, pp. 3313–3351, May 2021.

[25] S. Gong, X. Lu, D. T. Hoang, D. Niyato, L. Shu, D. I. Kim, and Y.-C. Liang, "Toward smart wireless communications via intelligent reflecting surfaces: A contemporary survey," *IEEE Commun. Surveys & Tuts.*, vol. 22, no. 4, pp. 2283–2314, Fourth Quarter 2020.

[26] B. Clerckx, Y. Mao, R. Schober, E. A. Jorswieck, D. J. Love, J. Yuan, L. Hanzo, G. Y. Li, E. G. Larsson, and G. Caire, "Is NOMA efficient in multi-antenna networks? A critical look at next generation multiple access techniques," *IEEE Open J. Commun. Soc.*, vol. 2, pp. 1310–1343, Jun. 2021.

[27] H. Joudeh and B. Clerckx, "Sum-rate maximization for linearly precoded downlink multiuser MISO systems with partial CSIT: A rate-splitting approach," *IEEE Trans. Commun.*, vol. 64, no. 11, pp. 4847–4861, Nov. 2016.

[28] Y. Mao, E. Piovano, and B. Clerckx, "Rate-splitting multiple access for overloaded cellular Internet of things," *IEEE Trans. Commun.*, vol. 69, no. 7, pp. 4504–4519, Jul. 2021.

[29] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. of Int'l Conf. on Machine Learning (ICML)*, Montreal, Canada, Jun. 2009.

[30] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey," *J. Mach. Learn. Res.*, vol. 21, pp. 1–50, Jul. 2020.

[31] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *Proc. of IEEE Int'l Conf. on Robot. Autom. (ICRA)*, Brisbane, Australia, May 2018.

[32] P. Rashidinejad, B. Zhu, C. Ma, J. Jiao, and S. Russell, "Bridging offline reinforcement learning and imitation learning: A tale of pessimism," in *Proc. of Conf. on Neural Information Processing Systems (NeurIPS)*, Dec. 2021.

[33] J. Zhang, L. Lu, Y. Sun, Y. Chen, J. Liang, J. Liu, H. Yang, S. Xing, Y. Wu, J. Ma, I. B. F. Murias, and F. J. L. Hernando, "PoC of SCMA-based uplink grant-free transmission in UCNC for 5G," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 6, pp. 1353–1362, Jun. 2017.

[34] S. Han, X. Tai, W. Meng, and C. Li, "A resource scheduling scheme based on feedback for SCMA grant-free uplink transmission," in *Proc. of IEEE Int'l Conf. on Commun. (ICC)*, Paris, France, May 2017.

[35] J. Shen, W. Chen, F. Wei, and Y. Wu, "ACK feedback based UE-to-CTU mapping rule for SCMA uplink grant-free transmission," in *Proc. of Int'l Conf. on Wireless Commun. Signal Process.*, Nanjing, China, Oct. 2017.

[36] J. Sun, W. Wu, and X. Wu, "A contention transmission unit allocation scheme for uplink grant-free SCMA systems," in *Proc. of IEEE Int'l Conf. on Commun. (ICC)*, Kansas City, MO, May 2018.

[37] S. Wang, H. Liu, P. H. Gomes, and B. Krishnamachari, "Deep reinforcement learning for dynamic multichannel access in wireless networks," *IEEE Trans. on Cogn. Commun. Netw.*, vol. 4, no. 2, pp. 257–265, Jun. 2018.

[38] O. Naparstek and K. Cohen, "Deep multi-user reinforcement learning for distributed dynamic spectrum access," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 310–323, Jan. 2019.

[39] Y. Yu, T. Wang, and S. C. Liew, "Deep-reinforcement learning multiple access for heterogeneous wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1277–1290, Jun. 2019.

[40] Q. Wu, X. Guan, and R. Zhang, "Intelligent reflecting surface-aided wireless energy and information transmission: An overview," *Proc. of IEEE*, vol. 110, no. 1, pp. 150–170, Jan. 2022.

[41] C. Huang, S. Hu, G. C. Alexandropoulos, A. Zappone, C. Yuen, R. Zhang, M. D. Renzo, and M. Debbah, "Holographic MIMO surfaces for 6G wireless networks: Opportunities, challenges, and trends," *IEEE Wireless Commun.*, vol. 27, no. 5, pp. 118–125, Oct. 2020.

[42] S. Gong, X. Lu, D. T. Hoang, D. Niyato, L. Shu, D. I. Kim, and Y. Liang, "Toward smart wireless communications via intelligent reflecting surfaces: A contemporary survey," *IEEE Commun. Surveys & Tuts.*, vol. 22, no. 4, pp. 2283–2314, Fourth Quarter 2020.

[43] C. Chaccour, M. N. Soorki, W. Saad, M. Bennis, and P. Popovski, "Risk-based optimization of virtual reality over terahertz reconfigurable intelligent surfaces," in *Proc. of IEEE Int'l Conf. on Commun. (ICC)*, Dublin, Ireland, Jun. 2020.

[44] X. Yu, D. Xu, and R. Schober, "MISO wireless communication systems via intelligent reflecting surfaces," in *Proc. of IEEE/CIC Int'l Conf. on Commun. in China (ICCC)*, Chengdu, China, Aug. 2019.

[45] Y. Jia, C. Ye, and Y. Cui, "Analysis and optimization of an intelligent reflecting surface-assisted system with interference," in *Proc. of IEEE Int'l Conf. on Commun. (ICC)*, Jun. 2020.

[46] S. Abeywickrama, R. Zhang, Q. Wu, and C. Yuen, "Intelligent reflecting surface: Practical phase shift model and beamforming optimization," *IEEE Trans. Commun.*, vol. 68, no. 9, pp. 5849–5863, Sept. 2020.

[47] D. Xu, X. Yu, Y. Sun, D. W. K. Ng, and R. Schober, "Resource allocation for IRS-assisted full-duplex cognitive radio systems," *IEEE Trans. Commun.*, vol. 68, no. 12, pp. 7376–7394, Dec. 2020.

[48] X. Ma, S. Guo, H. Zhang, Y. Fang, and D. Yuan, "Joint beamforming and reflecting design in reconfigurable intelligent surface-aided multi-user communication systems," *IEEE Trans. Wireless Commun.*, vol. 20, no. 5, pp. 3269–3283, May 2021.

[49] Z. Wan, Z. Gao, F. Gao, M. D. Renzo, and M.-S. Alouini, "Terahertz massive MIMO with holographic reconfigurable intelligent surfaces," *IEEE Trans. Commun.*, vol. 69, no. 7, pp. 4732–4750, Jul. 2021.

[50] K. Feng, X. Li, Y. Han, S. Jin, and Y. Chen, "Physical layer security enhancement exploiting intelligent reflecting surface," *IEEE Commun. Lett.*, vol. 25, no. 3, pp. 734–738, Mar. 2021.

[51] R. Huang and V. W.S. Wong, "Towards reliable communications in intelligent reflecting surface-aided cell-free MIMO systems," in *Proc. of IEEE Global Commun. Conf. (GLOBECOM)*, Madrid, Spain, Dec. 2021.

[52] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," *CoRR*, vol. abs/1611.09940, Jan. 2017. [Online]. Available: http://arxiv.org/abs/1611.09940

[53] W. Kool, H. Van Hoof, and M. Welling, "Attention, learn to solve routing problems!" in *Proc. of Int'l Conf. on Learning Representations (ICLR)*, New Orleans, LA, May 2019.

[54] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. of Conf. on Neural Information Processing Systems (NeurIPS)*, Long Beach, CA, Dec. 2017.

[55] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3-4, pp. 229–256, May 1992.

[56] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. of Int'l Conf. on Machine Learning (ICML)*, Stockholm, Sweden, Jul. 2018.

[57] Fortune Business Insights, "Virtual reality market size," Jan. 2022. [Online]. Available: https://www.fortunebusinessinsights.com/industry-reports/virtual-reality-market-101378

[58] G. Zhou, Y. Mao, and B. Clerckx, "Rate-splitting multiple access for multi-antenna downlink communication systems: Spectral and energy efficiency tradeoff," *IEEE Trans. Wireless Commun.*, vol. 21, no. 7, pp. 4816–4828, Jul. 2021.

[59] N. Q. Hieu, D. T. Hoang, D. Niyato, and D. I. Kim, "Optimal power allocation for rate splitting communications with deep reinforcement learning," *IEEE Wireless Commun. Lett.*, vol. 10, no. 12, pp. 2820–2823, Dec. 2021.

[60] A. Bansal, K. Singh, B. Clerckx, C.-P. Li, and M.-S. Alouini, "Rate-splitting multiple access for intelligent reflecting surface aided multi-user communications," *IEEE Trans. Veh. Technol.*, vol. 70, no. 9, pp. 9217–9229, Sept. 2021.

[61] H. Joudeh and B. Clerckx, "Rate-splitting for max-min fair multigroup multicast beamforming in overloaded systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 11, pp. 7276–7289, Nov. 2017.

[62] Y. Mao, B. Clerckx, and V. O. K. Li, "Rate-splitting for multi-antenna non-orthogonal unicast and multicast transmission: Spectral and energy efficiency analysis," *IEEE Trans. Commun.*, vol. 67, no. 12, pp. 8754–8770, Dec. 2019.

[63] Q. Wu and R. Zhang, "Intelligent reflecting surface enhanced wireless network via joint active and passive beamforming," *IEEE Trans. Wireless Commun.*, vol. 18, no. 11, pp. 5394–5409, Nov. 2019.

[64] M. Najafi, V. Jamali, R. Schober, and H. V. Poor, "Physics-based modeling and scalable optimization of large intelligent reflecting surfaces," *IEEE Trans. Commun.*, vol. 69, no. 4, pp. 2673–2691, Apr. 2021.

[65] K.-L. Besser and E. A. Jorswieck, "Reconfigurable intelligent surface phase hopping for ultra-reliable communications," *IEEE Trans. Wireless Commun.*, vol. 21, no. 11, pp. 9082–9095, Nov. 2022.

[66] H. Fu, S. Feng, and D. W. K. Ng, "Resource allocation design for IRS-aided downlink MU-MISO RSMA systems," in *Proc. of IEEE Int'l Conf. on Commun. (ICC) Workshop*, Jun. 2021.

[67] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018.

[68] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia, "Incorporating functional knowledge in neural networks," *J. Mach. Learn. Res.*, vol. 10, pp. 1239–1262, Jun. 2009.

[69] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. of AAAI Conf. on Artificial Intelligence*, Phoenix, AZ, Feb. 2016.

[70] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. of Int'l Conf. on Machine Learning (ICML)*, Stockholm, Sweden, Jun. 2018.

[71] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, Aug. 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[72] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," in *Proc. of Int'l Conf. on Machine Learning (ICML)*, Lille, France, Jul. 2015.

[73] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, p. 484–489, Jan. 2016.

[74] C. Harold Liu, Z. Chen, J. Tang, J. Xu, and C. Piao, "Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 2059–2070, Sept. 2018.

[75] S. Zhu, L. Gui, N. Cheng, F. Sun, and Q. Zhang, "Joint design of access point selection and path planning for UAV-assisted cellular networks," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 220–233, Jan. 2020.

[76] Y. Hu, M. Chen, W. Saad, H. V. Poor, and S. Cui, "Distributed multi-agent meta learning for trajectory design in wireless drone networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 10, pp. 3177–3192, Oct. 2021.

[77] A. Khalili, E. M. Monfared, S. Zargari, M. R. Javan, N. M. Yamchi, and E. A. Jorswieck, "Resource management for transmit power minimization in UAV-assisted RIS HetNets supported by dual connectivity," *IEEE Trans. Wireless Commun.*, vol. 21, no. 3, pp. 1806–1822, Mar. 2022.

[78] H. Yang, X. Xie, and M. Kadoch, "Intelligent resource management based on reinforcement learning for ultra-reliable and low-latency IoV communication networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4157–4169, May 2019.

[79] H. Ye, G. Y. Li, and B.-H. F. Juang, "Deep reinforcement learning based resource allocation for V2V communications," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3163–3173, Apr. 2019.

[80] X. Zhang, M. Peng, S. Yan, and Y. Sun, "Deep-reinforcement-learning-based mode selection and resource allocation for cellular V2X communications," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6380–6391, Jul. 2020.

[81] J. Jang and H. J. Yang, "Deep reinforcement learning-based resource allocation and power control in small cells with limited information exchange," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 13 768–13 783, Nov. 2020.

[82] X. Huang, S. Leng, S. Maharjan, and Y. Zhang, "Multi-agent deep reinforcement learning for computation offloading and interference coordination in small cell networks," *IEEE Trans. Veh. Technol.*, vol. 70, no. 9, pp. 9282–9293, Sept. 2021.

[83] Y. Wei, F. R. Yu, M. Song, and Z. Han, "Joint optimization of caching, computing, and radio resources for fog-enabled IoT using natural actor–critic deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2061–2073, Apr. 2019.

[84] C. Xu, Y. Xie, X. Wang, H. H. Yang, D. Niyato, and T. Q. S. Quek, "Optimal status update for caching enabled IoT networks: A dueling deep R-network approach," *IEEE Trans. Wireless Commun.*, vol. 20, no. 12, pp. 8438–8454, Dec. 2021.

[85] J. Yao and N. Ansari, "Caching in dynamic IoT networks by deep reinforcement learning," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3268–3275, Mar. 2021.

[86] X. Wang, C. Wang, X. Li, V. C. M. Leung, and T. Taleb, "Federated deep reinforcement learning for Internet of things with decentralized cooperative edge caching," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9441–9455, Oct. 2020.

[87] M. Samir, D. Ebrahimi, C. Assi, S. Sharafeddine, and A. Ghrayeb, "Leveraging UAVs for coverage in cell-free vehicular networks: A deep reinforcement learning approach," *IEEE Trans. Mobile Comput.*, vol. 20, no. 9, pp. 2835–2847, Sept. 2021.

[88] Y. Al-Eryani, M. Akrout, and E. Hossain, "Multiple access in cell-free networks: Outage performance, dynamic clustering, and deep reinforcement learning-based design," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 4, pp. 1028–1042, Apr. 2021.

[89] Y. Zhao, J. Hu, K. Yang, and S. Cui, "Deep reinforcement learning aided intelligent access control in energy harvesting based WLAN," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 14 078–14 082, Nov. 2020.

[90] Y. Li, X. Zhao, and H. Liang, "Throughput maximization by deep reinforcement learning with energy cooperation for renewable ultradense IoT networks," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 9091–9102, Sept. 2020.

[91] T. Jaakkola, S. Singh, and M. Jordan, "Reinforcement learning algorithm for partially observable Markov decision problems," in *Proc. of Conf. on Neural Information Processing Systems (NeurIPS)*, Denver, CO, Dec. 1994.

[92] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," in *Proc. of Int'l Conf. on Machine Learning (ICML)*, Sydney, Australia, Aug. 2017.

[93] J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. S. Torr, P. Kohli, and S. Whiteson, "Stabilising experience replay for deep multi-agent reinforcement learning," in *Proc. of Int'l Conf. on Machine Learning (ICML)*, Sydney, Australia, 2017.

[94] T. Rashid, M. Samvelyan, C. S. Witt, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. of Int'l Conf. on Machine Learning (ICML)*, Stockholm, Sweden, Jul. 2018.

[95] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proc. of Int'l Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, Stockholm, Sweden, Jul. 2018.

[96] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, Jul. 1966.

[97] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. of Int'l Conf. on Learning Representations (ICLR)*, San Juan, Puerto Rico, May 2016.

[98] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade*, 2nd ed., G. Montavon, G. B. Orr, and K.-R. Müller, Eds. Springer, 2012.

[99] N. Roux, M. Schmidt, and F. Bach, "A stochastic gradient method with an exponential convergence rate for finite training sets," in *Proc. of Conf. on Neural Information Processing Systems (NeurIPS)*, Lake Tahoe, NV, Dec. 2012.

[100] E. Björnson, Ö. Özdogan, and E. G. Larsson, "Intelligent reflecting surface versus decode-and-forward: How large surfaces are needed to beat relaying?" *IEEE Wireless Commun. Lett.*, vol. 9, no. 2, pp. 244–248, Feb. 2020.

[101] S. Duan, V. Shah-Mansouri, Z. Wang, and V. W. S. Wong, "D-ACB: Adaptive congestion control algorithm for bursty M2M traffic in LTE networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 12, pp. 9847–9861, Dec. 2016.

[102] R. Huang, V. W. S. Wong, and R. Schober, "Throughput optimization in grant-free NOMA with deep reinforcement learning," in *Proc. of IEEE Global Commun. Conf. (GLOBECOM)*, Waikoloa, HI, Dec. 2019.

[103] R. Huang, V. W.S. Wong, and R. Schober, "Throughput optimization for grant-free multiple access with multiagent deep reinforcement learning," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 228–242, Jan. 2021.

[104] R. Huang and V. W. S. Wong, "Neural combinatorial optimization for throughput maximization in IRS-aided systems," in *Proc. of IEEE Global Commun. Conf. (GLOBECOM)*, Taipei, Taiwan, Dec. 2020.

[105] R. Huang and V. W.S. Wong, "Joint user scheduling, phase shift control, and beamforming optimization in intelligent reflecting surface-aided systems," *IEEE Trans. Wireless Commun.*, vol. 21, no. 9, pp. 7521–7535, Sept. 2022.

[106] R. Huang, V. W.S. Wong, and R. Schober, "Rate-splitting for intelligent reflecting surface-aided multiuser VR streaming," accepted for publication in *IEEE J. Sel. Areas Commun.*, Dec. 2022.

[107] J. R. Kok and N. Vlassis, "Sparse cooperative Q-learning," in *Proc. of Int'l Conf. on Machine Learning (ICML)*, Banff, Alberta, Canada, Jul. 2004.

[108] M. J. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan and Claypool Publishers, 2010.

[109] J. Ahn, B. Shim, and K. B. Lee, "EP-based joint active user detection and channel estimation for massive machine-type communications," *IEEE Trans. Commun.*, vol. 67, no. 7, pp. 5178–5189, Jul. 2019.

[110] 3GPP TS 38.214 V16.1.0, "Technical specification group radio access network; NR; Physical layer procedures for data (Release 16)," Apr. 2020.

[111] 3GPP TS 23.501 V16.13.0, "Technical specification group services and system aspects; System architecture for the 5G system (5GS); Stage 2 (Release 16)," Jun. 2022.

[112] M. L. Littman, "Friend-or-foe Q-learning in general-sum games," in *Proc. of Int'l Conf. on Machine Learning (ICML)*, Williamstown, MA, Jun. 2001.

[113] J. Hu and M. P. Wellman, "Nash Q-learning for general-sum stochastic games," *J. Mach. Learn. Res.*, vol. 4, pp. 1039–1069, Nov. 2003.

[114] K. Tumer and D. H. Wolpert, *Collectives and the Design of Complex Systems*. Springer, 2004.

[115] G. Weiss, *Multiagent Systems, Second Edition*. MIT Press, 2013.

[116] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. van Hasselt, and D. Silver, "Distributed prioritized experience replay," *arXiv preprint arXiv:1803.00933*, Mar. 2018.

[117] S. Kapturowski, G. Ostrovski, W. Dabney, J. Quan, and R. Munos, "Recurrent experience replay in distributed reinforcement learning," in *Proc. of Int'l Conf. on Learning Representations (ICLR)*, New Orleans, LA, May 2019.

[118] V. K. N. Lau, "Proportional fair space-time scheduling for wireless communications," *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1353–1360, Aug. 2005.

[119] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Proc. of Conf. on Neural Information Processing Systems (NeurIPS)*, Long Beach, CA, Dec. 2017.

[120] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh, "Set transformer: A framework for attention-based permutation-invariant neural networks," in *Proc. of Int'l Conf. on Machine Learning (ICML)*, Long Beach, CA, Jun. 2019.

[121] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, Jun. 2016.

[122] A. E. Orhan and X. Pitkow, "Skip connections eliminate singularities," in *Proc. of Int'l Conf. on Learning Representations (ICLR)*, Vancouver, Canada, May 2018.

[123] M. Ilse, J. Tomczak, and M. Welling, "Attention-based deep multiple instance learning," in *Proc. of Int'l Conf. on Machine Learning (ICML)*, Stockholm, Sweden, Jul. 2018.

[124] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. of Int'l Conf. on Learning Representations (ICLR)*, San Diego, CA, May 2015.

[125] G. Hacohen and D. Weinshall, "On the power of curriculum learning in training deep networks," in *Proc. of Int'l Conf. on Machine Learning (ICML)*, Long Beach, CA, Jun. 2019.

[126] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. of Int'l Conf. on Machine Learning (ICML)*, Beijing, China, Jun. 2014.

[127] S. Zhang and R. Zhang, "Intelligent reflecting surface aided multiple access: Capacity region and deployment strategy," in *Proc. of IEEE Int'l Workshop on Signal Process. Advances Wireless Commun.*, May 2020.

[128] 3GPP TR 38.901 V16.1.0, "Technical specification group radio access network; Study on channel model for frequencies from 0.5 to 100 GHz (Release 16)," Dec. 2019.

[129] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. of Conf. on Neural Information Processing Systems (NeurIPS)*, Vancouver, Canada, Dec. 2019.

[130] G. Zhou, C. Pan, H. Ren, K. Wang, and A. Nallanathan, "A framework of robust transmission design for IRS-aided MISO communications with imperfect cascaded channels," *IEEE Trans. Signal Process.*, vol. 68, pp. 5092–5106, Aug. 2020.

[131] J. Zhang, M. Kountouris, J. G. Andrews, and R. W. Heath, "Multi-mode transmission for the MIMO broadcast channel with imperfect channel state information," *IEEE Trans. Commun.*, vol. 59, no. 3, pp. 803–814, Mar. 2011.

[132] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter, "Differentiable convex optimization layers," in *Proc. of Conf. on Neural Information Processing Systems (NeurIPS)*, Vancouver, Canada, Dec. 2019.

[133] S. Knorr, C. Ozcinar, C. O. Fearghail, and A. Smolic, "Director's cut: A combined dataset for visual attention analysis in cinematic VR content," in *Proc. of ACM SIGGRAPH European Conf. on Visual Media Production*, London, United Kingdom, Dec. 2018.

[134] Z. Wang, L. Liu, and S. Cui, "Channel estimation for intelligent reflecting surface assisted multiuser communications: Framework, algorithms, and analysis," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6607–6620, Oct. 2020.

[135] L. Wei, C. Huang, G. C. Alexandropoulos, C. Yuen, Z. Zhang, and M. Debbah, "Channel estimation for RIS-empowered multi-user MISO wireless communications," *IEEE Trans. Commun.*, vol. 69, no. 6, pp. 4144–4157, Jun. 2021.

[136] X. Guan, Q. Wu, and R. Zhang, "Anchor-assisted channel estimation for intelligent reflecting surface aided multiuser communication," *IEEE Trans. Wireless Commun.*, vol. 21, no. 6, pp. 3764–3778, Jun. 2022.

[137] Meta Quest, "Setting up your play area and guardian," Dec. 2021. [Online]. Available: https://support.oculus.com/guardian/

[138] A. Nguyen, Z. Yan, and K. Nahrstedt, "Your attention is unique: Detecting 360-degree video saliency in head-mounted display for head movement prediction," in *Proc. of ACM Int'l Conf. on Multimedia*, Seoul, Republic of Korea, Oct. 2018.

[139] M. Tang and V. W. S. Wong, "Online bitrate selection for viewport adaptive 360-degree video streaming," *IEEE Trans. Mobile Comput.*, vol. 21, no. 7, pp. 2506–2517, Jul. 2022.

[140] S. Tavakoli, S. Egger, M. Seufert, R. Schatz, K. Brunnström, and N. García, "Perceptual quality of HTTP adaptive streaming strategies: Cross-experimental analysis of multi-laboratory and crowdsourced subjective studies," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 8, pp. 2141–2153, Aug. 2016.

[141] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Proc. of Conf. on Neural Information Processing Systems (NeurIPS)*, Long Beach, CA, Dec. 2017.

[142] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming," http://cvxr.com/cvx, Mar. 2014.

[143] Steam, "Valve Index base station," Jun. 2022. [Online]. Available: https://store.steampowered.com/app/1059570/Valve_Index_Base_Station/

[144] VIVE, "VIVE wireless adapter," Apr. 2022. [Online]. Available: https://www.vive.com/ca/accessory/wireless-adapter/

[145] FFmpeg, https://ffmpeg.org/, Jun. 2022.

[146] S. Ju, Y. Xing, O. Kanhere, and T. S. Rappaport, "Millimeter wave and sub-terahertz spatial statistical channel model for an indoor office building," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 6, pp. 1561–1575, Jun. 2021.

[147] H. Zhang, H. Chen, C. Xiao, B. Li, M. Liu, D. Boning, and C.-J. Hsieh, "Robust deep reinforcement learning against adversarial perturbations on state observations," in *Proc. of Conf. on Neural Information Processing Systems (NeurIPS)*, Dec. 2020.

[148] T. Oikarinen, W. Zhang, A. Megretski, L. Daniel, and T.-W. Weng, "Robust deep reinforcement learning through adversarial loss," in *Proc. of Conf. on Neural Information Processing Systems (NeurIPS)*, Dec. 2021.

[149] H. Kim, J. Park, M. Bennis, S. Kim, and M. Debbah, "Mean-field game theoretic edge caching in ultra-dense networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 1, pp. 935–947, Jan. 2020.

[150] B. Zhou and W. Saad, "Age of information in ultra-dense IoT systems: Performance and mean-field game analysis," *arXiv preprint arXiv:2006.15756*, Jun. 2020.

[151] J. Foerster, I. A. Assael, N. De Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Proc. of Conf. on Neural Information Processing Systems (NeurIPS)*, Barcelona, Spain, Dec. 2016.

[152] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *Proc. of Int'l Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, Auckland, NZ, May 2017.

[153] K. Li, A. Gupta, A. Reddy, V. H. Pong, A. Zhou, J. Yu, and S. Levine, "MURAL: Meta-learning uncertainty-aware rewards for outcome-driven reinforcement learning," in *Proc. of Int'l Conf. on Machine Learning (ICML)*, Jul. 2021.

[154] Y. Tang, T. Kozuno, M. Rowland, R. Munos, and M. Valko, "Unifying gradient estimators for meta-reinforcement learning via off-policy evaluation," in *Proc. of Conf. on Neural Information Processing Systems (NeurIPS)*, Dec. 2021.

[155] A. Gupta, R. Mendonca, Y. Liu, P. Abbeel, and S. Levine, "Meta-reinforcement learning of structured exploration strategies," in *Proc. of Conf. on Neural Information Processing Systems (NeurIPS)*, Montreal, Canada, Dec. 2018.

[156] M. Emara, H. Elsawy, and G. Bauch, "A spatiotemporal model for peak AoI in uplink IoT networks: Time versus event-triggered traffic," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6762–6777, Jan. 2020.

[157] H. Hu, K. Xiong, G. Qu, Q. Ni, P. Fan, and K. B. Letaief, "AoI-minimal trajectory planning and data collection in UAV-assisted wireless powered IoT networks," *IEEE Internet Things J.*, vol. 8, no. 2, pp. 1211–1223, Jan. 2021.

[158] C. You, K. Huang, and H. Chae, "Energy efficient mobile cloud computing powered by wireless energy transfer," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1757–1771, 2016.

[159] X. Liu and N. Ansari, "Green relay assisted D2D communications with dual batteries in heterogeneous cellular networks for IoT," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1707–1715, Oct. 2017.

[160] ——, "Toward green IoT: Energy solutions and key challenges," *IEEE Commun. Mag.*, vol. 57, no. 3, pp. 104–110, Mar. 2019.

[161] C. Pan, H. Ren, K. Wang, W. Xu, M. Elkashlan, A. Nallanathan, and L. Hanzo, "Multicell MIMO communications relying on intelligent reflecting surfaces," *IEEE Trans. Wireless Commun.*, vol. 19, no. 8, pp. 5218–5233, Aug. 2020.

[162] H. Xie, J. Xu, and Y.-F. Liu, "Max-min fairness in IRS-aided multi-cell MISO systems with joint transmit and reflective beamforming," *IEEE Trans. Wireless Commun.*, vol. 20, no. 2, pp. 1379–1393, Feb. 2021.

[163] S. Liao, J. Wu, J. Li, and K. Konstantin, "Information-centric massive IoT-based ubiquitous connected VR/AR in 6G: A proposed caching consensus approach," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5172–5184, Apr. 2021.

Appendix A

Appendix for Chapter 2

We apply stochastic network optimization to transform problem (2.7) into an optimization problem that can be solved in each time slot. We use a *virtual queue* [108] to take into account the throughput requirement of user n. The dynamic of the virtual queue of user n is given by

$$q_n(t+1) \triangleq \max\left[q_n(t) - r_n(t) + \mu_n^{\text{req}}, 0\right]. \tag{A.1}$$

We define $\mathbf{q}(t) \triangleq (q_1(t), \ldots, q_N(t))$ as the vector of the backlogs of the queues of all users. We use the following Lyapunov function to measure the backlogs:

$$L(\boldsymbol{q}(t)) \triangleq \frac{1}{2}\sum_{n\in\mathcal{N}} q_n(t)^2. \tag{A.2}$$

We have the following upper bound on the conditional Lyapunov drift [108]

$$\begin{aligned}
\Delta L(\boldsymbol{q}(t)) &= \mathbb{E}[L(\boldsymbol{q}(t+1)) - L(\boldsymbol{q}(t)) \mid \boldsymbol{q}(t)] \\
&\leq \mathbb{E}\left[\sum_{n\in\mathcal{N}} \frac{(\mu_n^{\text{req}})^2 + r_n(t)^2}{2} \;\middle|\; \boldsymbol{q}(t)\right] + \sum_{n\in\mathcal{N}} q_n(t)\mu_n^{\text{req}} - \mathbb{E}\left[\sum_{n\in\mathcal{N}} q_n(t) r_n(t) \;\middle|\; \boldsymbol{q}(t)\right] \\
&\leq B + \sum_{n\in\mathcal{N}} q_n(t)\mu_n^{\text{req}} - \mathbb{E}\left[\sum_{n\in\mathcal{N}} q_n(t) r_n(t) \;\middle|\; \boldsymbol{q}(t)\right],
\end{aligned} \tag{A.3}$$

where B is a constant that bounds the first term on the right-hand side of the above inequality. We add $V\mathbb{E}\big[-\sum_{n\in\mathcal{N}} r_n(t) \;\big|\; \boldsymbol{q}(t)\big]$ to both sides of the inequality, where V is a positive constant representing the importance of aggregate reward maximization. The following bound on the Lyapunov drift-plus-penalty equation can be derived:

$$\Delta L(\boldsymbol{q}(t)) + V\mathbb{E}\left[-\sum_{n\in\mathcal{N}} r_n(t) \;\middle|\; \boldsymbol{q}(t)\right]$$
$$\leq B - \mathbb{E}\left[\sum_{n\in\mathcal{N}} q_n(t) r_n(t) \;\middle|\; \boldsymbol{q}(t)\right] + \sum_{n\in\mathcal{N}} q_n(t)\mu_n^{\text{req}} + V\mathbb{E}\left[-\sum_{n\in\mathcal{N}} r_n(t) \;\middle|\; \boldsymbol{q}(t)\right]. \tag{A.4}$$

Given the observed $\boldsymbol{q}(t)$, as μ_n^{req} and B are constants in time slot t, minimizing the right-hand side of (A.4) can be accomplished by solving the following problem:

$$\begin{aligned} \underset{\boldsymbol{g}_n(t),\, n\in\mathcal{N}}{\text{maximize}} \quad & \sum_{n\in\mathcal{N}} \left(q_n(t) + V\right) r_n(t) \\ \text{subject to} \quad & \sum_{k\in\mathcal{K}} g_{nk}(t) \leq 1, \;\; n \in \mathcal{N}. \end{aligned} \tag{A.5}$$

The optimal solution of problem (A.5) can be obtained in each time slot using the *Lyapunov drift-plus-penalty algorithm* [108]. In particular, the objective function of problem (A.5) can be regarded as a weighted summation of the rewards $r_n(t)$ of all users, where the weight is determined by the virtual queue of the user and the value of V. To obtain the optimal solution, we sort the users in descending order of their backlog, i.e., $\hat{\boldsymbol{q}}(t) \triangleq (\hat{q}_1(t), \hat{q}_2(t), \cdots, \hat{q}_N(t))$. By selecting the top K users in descending order of the backlog and assigning one unique pilot sequence to each of the K users, the optimum of problem (A.5) can be obtained, which is $VK + \sum_{k=1}^{K} \hat{q}_k(t)$. Note that this requires global information and centralized scheduling.

Appendix B

Appendix for Chapter 4

We decompose problem (4.18) into three subproblems where each of the subproblems can be solved by exploiting its hidden convexity. For notational simplicity, we drop time index t in this section. We define $\boldsymbol{p} = (p_{n,i},\, i \in \mathcal{I}, n \in \mathcal{N})$, $\boldsymbol{c} = (c_i,\, i \in \mathcal{I})$, and $\boldsymbol{v} = (v_{n,i},\, i \in \mathcal{I}, n \in \mathcal{N})$. While the objective function in problem (4.18) only depends on bitrate selection $\boldsymbol{v}$, motivated by the inequality in (4.21), we use the following function to take the effects of $\boldsymbol{b}$, $\boldsymbol{\Psi}$, $\boldsymbol{c}$, $\boldsymbol{p}$, and $\boldsymbol{v}$ on the objective function into consideration:

$$r(\boldsymbol{b}, \boldsymbol{\Psi}, \boldsymbol{c}, \boldsymbol{p}, \boldsymbol{v}) = \sum_{n \in \mathcal{N}} \frac{W T_{\mathrm{DL}}}{T_v} \Big(\sum_{i \in \mathcal{I}_n} p_{n,i} R_n^{\mathrm{p}}(\boldsymbol{b}, \boldsymbol{\Psi}) + \sum_{i \in \mathcal{I}_n} c_i R^{\mathrm{c}}(\boldsymbol{b}, \boldsymbol{\Psi}) \Big) - \sum_{n \in \mathcal{N}} \kappa^{\mathrm{intra}}\, \ell_n^{\mathrm{intra}}(\boldsymbol{v}) \tag{B.1}$$

For the beamforming subproblem, we optimize the beamforming vectors for maximization of $r(\boldsymbol{b})$ subject to the maximum transmit power constraint C2 and the per-user per-tile QoE constraint C8. We have

$$\begin{aligned} \underset{\boldsymbol{b}}{\text{maximize}} \quad & r(\boldsymbol{b}) \\ \text{subject to} \quad & \text{constraints C2 and C8.} \end{aligned} \tag{B.2}$$

Subproblem (B.2) can be solved using FP [10] and WMMSE [28] by introducing auxiliary variables, and updating the beamforming vectors and the auxiliary variables iteratively. Both WMMSE and FP are guaranteed to converge to a stationary solution of problem (B.2).

The IRS phase shift subproblem is given by

$$\begin{aligned} \underset{\boldsymbol{\Psi}}{\text{maximize}} \quad & r(\boldsymbol{\Psi}) \\ \text{subject to} \quad & \text{constraints C8 and C9.} \end{aligned} \tag{B.3}$$

For the IRS phase shift subproblem, we define vector $\boldsymbol{\lambda} = (e^{-j\psi_1}, \ldots, e^{-j\psi_L}, \rho) \in \mathbb{C}^{L+1}$, where $\rho \in \mathbb{C}$ and $|\rho|^2 = 1$. We further define matrix $\boldsymbol{\Lambda} = \boldsymbol{\lambda}\boldsymbol{\lambda}^H \in \mathbb{C}^{(L+1)\times(L+1)}$ to replace the IRS phase shift constraint C9. This leads to the following equality constraints:

$$\text{C10:} \quad \text{Diag}(\boldsymbol{\Lambda}) = \boldsymbol{I}_{L+1}, \tag{B.4}$$

$$\text{C11:} \quad \text{rank}(\boldsymbol{\Lambda}) = 1, \tag{B.5}$$

where $\boldsymbol{I}_{L+1}$ denotes the $(L+1) \times (L+1)$ identity matrix. For user $n \in \mathcal{N}$, we define the following matrix

$$\boldsymbol{\Theta}_n = \begin{bmatrix} \text{diag}(\boldsymbol{h}_{R,n}^H)\, \boldsymbol{G} \\ \boldsymbol{h}_{D,n}^H \end{bmatrix} \in \mathbb{C}^{(L+1)\times N_t}. \tag{B.6}$$

To solve subproblem (B.3) in a tractable manner, we rewrite the SINR of the common message of user n in (4.6) as follows:

$$\Gamma_n^{\text{c}} = \frac{\text{Tr}(\boldsymbol{\Lambda}^T \boldsymbol{\Theta}_n \boldsymbol{b}_0 \boldsymbol{b}_0^H \boldsymbol{\Theta}_n^H)}{\sum_{j\in\mathcal{N}} \text{Tr}(\boldsymbol{\Lambda}^T \boldsymbol{\Theta}_n \boldsymbol{b}_j \boldsymbol{b}_j^H \boldsymbol{\Theta}_n^H) + \sigma^2}. \tag{B.7}$$

The SINR of the private message of user n in (4.10) can be rewritten as

$$\Gamma_n^{\text{p}} = \frac{\text{Tr}(\boldsymbol{\Lambda}^T \boldsymbol{\Theta}_n \boldsymbol{b}_n \boldsymbol{b}_n^H \boldsymbol{\Theta}_n^H)}{\sum_{j\in\mathcal{N}\setminus\{n\}} \text{Tr}(\boldsymbol{\Lambda}^T \boldsymbol{\Theta}_n \boldsymbol{b}_j \boldsymbol{b}_j^H \boldsymbol{\Theta}_n^H) + \sigma^2}. \tag{B.8}$$

Similar to the beamforming subproblem, we use FP [10] to tackle the multi-ratio fractional objective function in subproblem (B.3). We apply quadratic transform [10] to the common

rate and the private rate of user $n \in \mathcal{N}$ as follows:

$$\widetilde{R}_n^{\mathrm{c}} = \log_2\Big(1 + 2y_n\sqrt{\mathrm{Tr}(\boldsymbol{\Lambda}^T\boldsymbol{\Theta}_n\boldsymbol{b}_0\boldsymbol{b}_0^H\boldsymbol{\Theta}_n^H)} - y_n^2\big(\sum_{j\in\mathcal{N}}\mathrm{Tr}(\boldsymbol{\Lambda}^T\boldsymbol{\Theta}_n\boldsymbol{b}_j\boldsymbol{b}_j^H\boldsymbol{\Theta}_n^H) + \sigma^2\big)\Big), \tag{B.9}$$

and

$$\widetilde{R}_n^{\mathrm{p}} = \log_2\Big(1 + 2z_n\sqrt{\mathrm{Tr}(\boldsymbol{\Lambda}^T\boldsymbol{\Theta}_n\boldsymbol{b}_n\boldsymbol{b}_n^H\boldsymbol{\Theta}_n^H)} - z_n^2\big(\sum_{j\in\mathcal{N}\setminus\{n\}}\mathrm{Tr}(\boldsymbol{\Lambda}^T\boldsymbol{\Theta}_n\boldsymbol{b}_j\boldsymbol{b}_j^H\boldsymbol{\Theta}_n^H) + \sigma^2\big)\Big), \tag{B.10}$$

where y_n and z_n are the auxiliary variables. This leads to the following optimization problem

$$\begin{aligned}
\underset{\boldsymbol{\Lambda},\boldsymbol{y},\boldsymbol{z},\widetilde{R}^{\mathrm{c}}}{\text{maximize}}\ & f^{\mathrm{PS}}(\boldsymbol{\Lambda},\widetilde{R}^{\mathrm{c}},\boldsymbol{y},\boldsymbol{z}) \triangleq \sum_{n\in\mathcal{N}}\frac{WT_{\mathrm{DL}}}{T_v}\Big(\widetilde{R}_n^{\mathrm{p}} + \sum_{i\in\mathcal{I}_n}c_i\widetilde{R}^{\mathrm{c}}\Big)\\
\text{subject to}\ & \widetilde{R}^{\mathrm{c}} \geq 0\\
& \widetilde{R}^{\mathrm{c}} \leq \widetilde{R}_n^{\mathrm{c}},\ n\in\mathcal{N}\\
& WT_{\mathrm{DL}}(p_{n,i}\widetilde{R}^{\mathrm{p}} + c_i\widetilde{R}^{\mathrm{c}}) \geq T_v\, v_{n,i},\ i\in\mathcal{I}_n,\ n\in\mathcal{N}\\
& \text{constraints C10 and C11,}
\end{aligned} \tag{B.11}$$

where $\boldsymbol{y} = (y_1,\ldots,y_N)$ and $\boldsymbol{z} = (z_1,\ldots,z_N)$. For user $n \in \mathcal{N}$, the optimal y_n and z_n for fixed $\boldsymbol{\Lambda}$ are given by

$$y_n^\star = \frac{\sqrt{\mathrm{Tr}(\boldsymbol{\Lambda}^T\boldsymbol{\Theta}_n\boldsymbol{b}_0\boldsymbol{b}_0^H\boldsymbol{\Theta}_n^H)}}{\sum_{j\in\mathcal{N}}\mathrm{Tr}(\boldsymbol{\Lambda}^T\boldsymbol{\Theta}_n\boldsymbol{b}_j\boldsymbol{b}_j^H\boldsymbol{\Theta}_n^H)+\sigma^2},\ n\in\mathcal{N}, \tag{B.12}$$

and

$$z_n^\star = \frac{\sqrt{\mathrm{Tr}(\boldsymbol{\Lambda}^T\boldsymbol{\Theta}_n\boldsymbol{b}_n\boldsymbol{b}_n^H\boldsymbol{\Theta}_n^H)}}{\sum_{j\in\mathcal{N}\setminus\{n\}}\mathrm{Tr}(\boldsymbol{\Lambda}^T\boldsymbol{\Theta}_n\boldsymbol{b}_j\boldsymbol{b}_j^H\boldsymbol{\Theta}_n^H)+\sigma^2},\ n\in\mathcal{N}. \tag{B.13}$$

For fixed y_n and z_n, $n \in \mathcal{N}$, we use SDR [11] to tackle constraint C11, and after relaxation the problem can be solved using convex optimization. A suboptimal solution of subproblem (B.11) can be obtained by iteratively optimizing y_n, z_n, $n \in \mathcal{N}$, and $\boldsymbol{\Lambda}$ [10]. The FP-based algorithm for solving subproblem (B.11) is provided in **Algorithm 8**.

Algorithm 8 Algorithm for Phase Shift Subproblem (B.11)

Initialize $\boldsymbol{\lambda}$ to a feasible value $\boldsymbol{\lambda}^{(0)}$ and obtain $\boldsymbol{\Lambda}^{(0)}$.
Initialize the FP termination threshold $\varepsilon_{\mathrm{FP}}$.
Initialize the iteration counter $\tau \leftarrow 0$.
Initialize $f^{\mathrm{PS}}(\boldsymbol{\Lambda}^{(\tau)}, (\widetilde{R}^{\mathrm{c}})^{(\tau)}, \boldsymbol{y}, \boldsymbol{z}) \leftarrow 0$.
repeat
    Determine the values of $y_n^\star$, $z_n^\star$, $n \in \mathcal{N}$ based on (B.12) and (B.13), respectively.
    $y_n \leftarrow y_n^\star, z_n \leftarrow z_n^\star$, $n \in \mathcal{N}$.
    Solve subproblem (B.11) for fixed $\boldsymbol{y}$ and $\boldsymbol{z}$, and obtain the optimal $\boldsymbol{\Lambda}^{(\tau+1)}$ and $(\widetilde{R}^{\mathrm{c}})^{(\tau+1)}$.
    $\tau \leftarrow \tau + 1$.
until $|f^{\mathrm{PS}}(\boldsymbol{\Lambda}^{(\tau)}, (\widetilde{R}^{\mathrm{c}})^{(\tau)}, \boldsymbol{y}, \boldsymbol{z}) - f^{\mathrm{PS}}(\boldsymbol{\Lambda}^{(\tau-1)}, (\widetilde{R}^{\mathrm{c}})^{(\tau-1)}, \boldsymbol{y}, \boldsymbol{z})| \leq \varepsilon_{\mathrm{FP}}$.
Decompose $\boldsymbol{\Lambda}^{(\tau)}$ to obtain the phase shift matrix $\boldsymbol{\Psi}^\star$.

After $\boldsymbol{\Psi}$ and $\boldsymbol{b}$ have been determined, the RS parameters and bitrate selection can be obtained using the same approach as for solving problem (4.45). We omit the details here for brevity. We solve the aforementioned three subproblems iteratively until the objective function converges.

Since a feasible solution is required for the initialization of the AO algorithm, we propose the following method for obtaining a feasible solution. We first initialize the bitrate selection to the minimum value, i.e., v_1. That is,

$$v_{n,i}(t) = \begin{cases} v_1, & i \in \mathcal{I}_n, \\ 0, & \text{otherwise.} \end{cases} \tag{B.14}$$

We initialize vector $\boldsymbol{c}$ by splitting the common rate equally between the tiles in $\mathcal{I}$. That is,

$$c_i = \begin{cases} \frac{1}{|\mathcal{I}|}, & i \in \mathcal{I}, \\ 0, & \text{otherwise.} \end{cases} \tag{B.15}$$

Similarly, we initialize $p_{n,i}$ as follows:

$$p_{n,i}(t) = \begin{cases} \frac{1}{|\mathcal{I}_n|}, & i \in \mathcal{I}_n, \\ 0, & \text{otherwise.} \end{cases} \tag{B.16}$$

Then, we find a feasible beamforming solution by solving the following problem:

$$\begin{aligned} \underset{\boldsymbol{b}}{\text{maximize}} \quad & r(\boldsymbol{b}) + \sum_{n\in\mathcal{N}} \sum_{i\in\mathcal{I}_n} \eta \min\{R^{\text{c}} c_i + R_n^{\text{p}} p_{n,i} - v_{n,i}, 0\} \\ \text{subject to} \quad & \text{constraint C2,} \end{aligned} \tag{B.17}$$

where $\eta > 0$ is the scaling factor of the penalty from violating constraint C8. Similar to the beamforming subproblem in (B.2), problem (B.17) becomes a convex optimization problem after applying quadratic transform to R^{c} and R_n^{p}.

After solving problem (B.17), we solve the following optimization problem to obtain a feasible IRS phase shift matrix:

$$\begin{aligned} \underset{\boldsymbol{\Psi}}{\text{maximize}} \quad & r(\boldsymbol{\Psi}) + \sum_{n\in\mathcal{N}} \sum_{i\in\mathcal{I}_n} \eta \min\{R^{\text{c}} c_i + R_n^{\text{p}} p_{n,i} - v_{n,i}, 0\} \\ \text{subject to} \quad & \text{constraint C9.} \end{aligned} \tag{B.18}$$

Problem (B.18) can be solved using FP and SDR. The proposed iterative algorithm for obtaining an initial solution is shown in **Algorithm 9**. We increase the value of η by multiplying it with a scale factor $\beta > 1$ after each iteration to increase the penalty for violating the per-user per-tile QoE constraints. Typical values of β range from 2 to 10. We solve problems (B.17) and (B.18) iteratively until a feasible solution is found or the maximum number of iterations is reached.

Algorithm 9 Algorithm for Obtaining an Initial Solution

Initialize $\boldsymbol{v}$ based on (B.14).
Initialize $\boldsymbol{c}$ and $\boldsymbol{p}$ based on (B.15) and (B.16), respectively.
Initialize $\boldsymbol{b}^{(0)}$ and $\boldsymbol{\Psi}^{(0)}$ based on random initialization.
Initialize $\eta \leftarrow \eta_0$.
Initialize iteration counter $\tau \leftarrow 1$.
for $\tau \leq \tau^{\max}$ **do**
  Solve problem (B.17) and obtain solution $\boldsymbol{b}^{(\tau)}$.
  Solve problem (B.18) and obtain solution $\boldsymbol{\Psi}^{(\tau)}$.
  if $(\boldsymbol{b}^{(\tau)}, \boldsymbol{\Psi}^{(\tau)})$ is a feasible solution **then**
    break.
  end if
  $\tau \leftarrow \tau + 1$.
  $\eta \leftarrow \beta\eta$.
end for