

# Scheduler Designs in Wireless Networks

Zhouyou Gu

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

Centre for IoT and Telecommunications  
School of Electrical and Information Engineering  
The University of Sydney

April 2023


*To my wife, Wenli, and my parents, Xiaofen and Liming.*

# Declaration

This is to certify that to the best of my knowledge, the content of this thesis is my own work. This thesis has not been submitted for any degree or other purposes. I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources has been acknowledged. I conducted the studies in this thesis under the supervision of Dr. Wibowo Hardjawana and Prof. Branka Vucetic, while enrolled in the School of Electrical and Information Engineering at the University of Sydney as a Ph.D. candidate. The primary contributions of this thesis are presented in Chapter 3, 4 and 5, where

- Chapter 3 includes the content from paper [J1] that has been published. I designed the study, performed simulations, implemented the prototype, collected and analysed the data, and wrote the manuscript drafts. I also collaborated with co-authors in addition to my supervisors for research discussions.
- Chapter 4 includes the content from paper [J2] that is under review. I designed the study, performed simulations, collected and analysed the data, and wrote the manuscript drafts.
- Chapter 5 includes the content from paper [J3] that is under review. I designed the study, performed simulations, collected and analysed the data, and wrote the manuscript drafts. I also collaborated with co-authors in addition to my supervisors for research discussions.

In addition to the statements above, in cases where I am not the corresponding author of a published item, permission to include the published material has been granted by the corresponding author.

  
\_\_\_\_\_ **Zhouyou Gu** 4 April 2023

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

  
\_\_\_\_\_ **Wibowo Hardjawana** 4 April 2023

# Acknowledgements

Foremost, I would like to express my sincere appreciation to my supervisor, Dr. Wibowo Hardjawana, who encouraged me to embark on my Ph.D. journey. His support and guidance have been instrumental in my progress. He has been a source of inspiration and motivation. His patience and approachable manners have been crucial in helping me overcome the challenges of this incredible journey. Also, I would like to express my great gratitude to my co-supervisor, Prof. Branka Vucetic, who has been a role model to me. I thank her for the considerable experience and high standards she demonstrated to me and for her valuable advice on my research and career.

I would like to acknowledge the help from Dr. Changyang She. His valuable feedback helped me greatly and his extensive knowledge contributed to my research in the first paper published during my Ph.D. candidature. I would like to thank my co-authors from Telstra Corporation Ltd., especially Simon Lumb, and co-authors from Morse Micro, especially Kishore Chikkam, for their valuable insights and suggestions on my research from an industrial perspective. I would also like to thank Dr. Vera Miloslavskaya for her valuable knowledge of coding theory that she has taught me, though the knowledge was not utilised in this thesis. I am thankful to all my colleagues at the Centre for IoT and Telecommunications for creating a positive working environment and making my journey pleasant. Thanks go to Dr. Floriana Badalotti for carefully proofreading the thesis content.

I appreciate my various sources of financial support: the Australian Government Research Training Program Scholarship, Postgraduate Research Supplementary and Completion Scholarships of the University of Sydney, the University of Sydney's research assistant employment arranged by Dr. Wibowo Hardjawana, Prof. Branka Vucetic's Australian Research Council Laureate Fellowship, the project funding from Telstra Corporation Ltd., and the University of Sydney's External Research Collaboration Seed Funding (Morse Micro), DVC Research, 2022.

I would like to express my deepest gratitude to my parents, who have been a constant source of love and support. Finally, I would like to express my heartfelt gratitude to my wife. She has been my constant companion, sounding board, and source of strength during difficult times in my Ph.D. journey. Her patience and understanding have been crucial in helping me balance my academic and personal life. Without her love and support, I could not have achieved this milestone in my life.

# Publications

The following is a list of submitted and published papers produced during my Ph.D. candidature and included as thesis chapters.

## Journal Papers

- [J1] Z. Gu, C. She, W. Hardjawana, S. Lumb, D. McKechnie, T. Essery, and B. Vucetic, “Knowledge-assisted deep reinforcement learning in 5G scheduler design: From theoretical framework to implementation,” *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 2014–2028, 2021.
- [J2] Z. Gu, W. Hardjawana, and B. Vucetic, “Opportunistic scheduling using statistical information of wireless channels,” *Submitted to IEEE Transactions on Wireless Communications*, 2022.
- [J3] Z. Gu, B. Vucetic, K. Chikkam, P. Aliberti, and W. Hardjawana, “Graph representation learning for contention and interference management in wireless networks,” *Submitted to IEEE/ACM Transactions on Networking*, 2023.

The following papers, which are not included as thesis chapters, have also been published during my Ph.D. candidature.

## Journal Papers

- [J4] C. She, C. Sun, Z. Gu, Y. Li, C. Yang, H. V. Poor, and B. Vucetic, “A tutorial on ultrareliable and low-latency communications in 6G: Integrating domain knowledge into deep learning,” *Proceedings of the IEEE*, vol. 109, no. 3, pp. 204–246, 2021.
- [J5] C. She, R. Dong, Z. Gu, Z. Hou, Y. Li, W. Hardjawana, C. Yang, L. Song, and B. Vucetic, “Deep learning for ultra-reliable and low-latency communications in 6G networks,” *IEEE Network*, vol. 34, no. 5, pp. 219–225, 2020.

# Abstract

Wireless networks have undergone significant development in recent years, driven by the increasing demand for wireless connectivity and data services. Radio resource schedulers are developed to assign network users available resources, such as frequency and time, based on network conditions to handle the growing user demands, providing transmission opportunities for each user. Well-designed schedulers optimise wireless resource allocation to ensure that all users receive a fair and high quality of service (QoS) and that the network operates at its maximum performance. However, as new types of wireless network services emerge, the existing schedulers can no longer satisfy their QoS requirements and maximise the network performance objective. Thus, new schedulers are urgently needed in wireless networks. In this thesis, we study scheduler designs in cellular and Wi-Fi networks. We discuss the limitations of the existing scheduler design methods and propose new methods to address these limitations.

We first develop a deep reinforcement learning (DRL) algorithm to flexibly design wireless schedulers, where we consider the QoS requirements of the time-sensitive traffic in 5G cellular networks. Since the scheduling policy is a deterministic mapping from channel and queue states to scheduling actions, it can be optimised as a neural network (NN) using the deep deterministic policy gradient (DDPG) algorithm. We show that a straightforward implementation of DDPG converges slowly, has poor QoS performance, and cannot be implemented in real-world 5G systems. To address these issues, we propose a theoretical DRL framework, where theoretical models from wireless communications are used to formulate a Markov decision process in DRL. To reduce the convergence time and improve the QoS of each user, we design a knowledge-assisted DDPG (K-DDPG) that exploits expert knowledge of the scheduler design problem, such as the knowledge of the QoS, the target scheduling policy, and the importance of each training sample, determined by the approximation error of the value function and the number of packet losses. Furthermore, we develop an architecture for online training and inference, where K-DDPG initialises the scheduler off-line and then fine-tunes the scheduler online to handle the mismatch between off-line simulations and real-world systems. Simulation results show that our approach reduces the convergence time of DDPG significantly and achieves better QoS than existing schedulers (reducing 30% ~ 50% packet losses). Experimental results show that with off-line initialisation, our approach achieves better initial QoS than random initialisation, and the online fine-tuning converges in a few minutes.

As scheduler designs are usually formulated as stochastic optimisation processes, including the DRL algorithm developed above, they require considerable time to interact with the network and optimise scheduler parameters. We then study the acceleration of the scheduler design’s convergence using statistical channel state information (CSI). Particularly, we consider the design of a class of widely used schedulers in cellular networks, namely max-weight schedulers (MWSs), which is applied to maximise a utility function of users’ average data rates. MWSs schedule the user with the highest weighted instantaneous data rate in each time slot. Existing stochastic optimisation methods require hundreds of time slots to adjust the MWS’s weights according to the instantaneous CSI before finding the optimal weights that maximise the utility function. In contrast, our MWS design requires few slots for estimating the statistical CSI. Specifically, we formulate a weight optimisation problem using the mean and variance of users’ signal-to-noise ratios (SNRs) to construct constraints bounding users’ feasible average rates. The formulated objective and optimisation variables are the utility function and the MWS’s weights, respectively. We develop an iterative solver for the problem and prove that it finds the optimal weights. We also design an online architecture where the solver adaptively generates optimal weights for networks with varying mean and variance of the SNRs. Simulation results show that our methods effectively require 4 ~ 10 times fewer slots to find the optimal weights and achieve 5% ~ 15% better average rates than the existing methods.

The above methods are developed for scheduler designs where users are associated with a single base station. We finally extend the research to the network-wise coordinated design of schedulers across multiple base stations. We study how base stations (or access points) can schedule the time slots based on the restricted access window (RAW) mechanism in Wi-Fi 802.11ah networks. RAW can manage contention and interference by grouping users and allocating periodic time slots for each user group’s transmissions. We will find the optimal user grouping decisions in RAW to maximise the worst-case user throughput in the network. We review existing user grouping approaches and highlight their performance limitations when applied to the above problem. We then propose to formulate user grouping as a graph construction problem where vertices represent users and edge weights indicate the contention and interference. This formulation applies the graph’s max cut to group users, and it optimises the edge weights to construct the optimal graph whose max cut results in the optimal grouping decisions. We develop an actor-critic graph representation learning (AC-GRL) algorithm to construct the optimal graph. Specifically, the actor NN is trained to estimate the optimal graph’s edge weights using path losses between users and access points. A graph cut procedure uses semidefinite programming to efficiently solve the max cut and return the grouping decisions for the given weights. The critic NN approximates users’ throughput achieved by the above decisions and is used to improve the actor. We also design an architecture that uses the online-measured throughput to fine-tune the decisions in response to changes in the number of users. Simulations show that our methods achieve 30% ~ 80% higher worst-case user throughput than the existing approaches and that the proposed architecture can further improve the worst-case user throughput by 15% ~ 35%.

# Contents

<b>Declaration</b>	<b>ii</b>
<b>Acknowledgements</b>	<b>iii</b>
<b>Publications</b>	<b>iv</b>
<b>Abstract</b>	<b>v</b>
<b>Contents</b>	<b>xii</b>
<b>List of Figures</b>	<b>xv</b>
<b>List of Tables</b>	<b>xvi</b>
<b>List of Algorithms</b>	<b>xvii</b>
<b>List of Acronyms</b>	<b>xviii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Schedulers in Wireless Networks . . . . .	1
1.2 Research Problems and Contributions . . . . .	3
1.2.1 Flexible Scheduler Design Using Deep Reinforcement Learning	5
1.2.2 Acceleration of Convergence in Scheduler Design Using Statis- tical Information of Wireless Channels . . . . .	6



1.2.3	Network-Wise Coordination in Scheduler Design Using Graph Representation Learning . . . . .	8
1.3	Thesis Outline . . . . .	10
1.4	Notations . . . . .	10
<b>2</b>	<b>Background</b>	<b>12</b>
2.1	Channel Capacity . . . . .	12
2.1.1	Shannon Capacity . . . . .	13
2.1.2	Channel Capacity in Finite Block Length Regime . . . . .	13
2.2	Radio Resource Scheduling in Wireless Networks . . . . .	14
2.2.1	Radio Resource Scheduling in Cellular Networks . . . . .	14
2.2.2	Radio Resource Scheduling in Wi-Fi Networks . . . . .	16
2.3	Deep Learning . . . . .	17
2.3.1	Fully-connected Neural Networks . . . . .	17
2.3.2	Graph Neural Networks . . . . .	18
2.3.3	Stochastic Gradient Descent . . . . .	19
2.4	Deep Reinforcement Learning . . . . .	20
2.4.1	Markov Decision Process . . . . .	20
2.4.2	Reinforcement Learning . . . . .	22
2.5	Convex Optimisation . . . . .	23
2.5.1	Semidefinite Programming . . . . .	25
<b>3</b>	<b>Knowledge-Assisted Deep Reinforcement Learning for Flexible Scheduler Design</b>	<b>28</b>
3.1	Introduction . . . . .	29
3.1.1	Related Works . . . . .	30
3.1.2	Our Methods . . . . .	32
3.2	Scheduler for Time-Sensitive Traffic in 5G . . . . .	32
3.2.1	Wireless Scheduler in 5G NR . . . . .	32
3.2.2	QoS Requirements of Time-Sensitive Traffic . . . . .	34
3.3	Straightforward Implementation of DDPG for Scheduler Design . . . . .	34

## Contents

3.3.1	Preliminaries of DDPG . . . . .	34
3.3.2	Problem Formulation . . . . .	37
3.4	Issues in the Straightforward Implementation of DDPG . . . . .	38
3.4.1	Issues in Problem Formulation . . . . .	39
3.4.2	Issues in Training Algorithm . . . . .	40
3.4.3	Issues in Online Implementation . . . . .	41
3.5	Theoretical DRL Framework . . . . .	41
3.5.1	Theoretical Models and Results . . . . .	42
3.5.2	Action Space Reduction . . . . .	43
3.5.3	Generalisation of State . . . . .	43
3.5.4	Reliability Evaluation . . . . .	43
3.5.5	Markov Property . . . . .	44
3.6	Knowledge-assisted DDPG . . . . .	44
3.6.1	Multi-head Critic for Individual QoS Evaluation . . . . .	45
3.6.2	Reward Shaping for Instantaneous Feedback . . . . .	46
3.6.3	Importance Sampling . . . . .	47
3.7	Online DDPG Architecture . . . . .	49
3.7.1	Off-line Initialisation . . . . .	50
3.7.2	Scheduler at the BS . . . . .	51
3.7.3	Online Training in the Edge Server . . . . .	52
3.8	Simulation Results . . . . .	55
3.8.1	Simulation Platform . . . . .	55
3.8.2	Performance of the T-DRL Framework and K-DDPG Algorithm . . . . .	56
3.9	Prototype of Online Architecture and Experimental Results . . . . .	60
3.9.1	Prototype . . . . .	60
3.9.2	Mismatch in Simulated and Real-world Networks . . . . .	62
3.9.3	Tests in a Real-world Network . . . . .	62
3.10	Summary . . . . .	65

## 4 Opportunistic Scheduling Using Statistical Information of Wireless

<b>Channels</b>	<b>66</b>
4.1 Introduction . . . . .	67
4.1.1 Related Works . . . . .	67
4.1.2 Our Methods . . . . .	68
4.2 System Model and Problem Formulation of Opportunistic Scheduling	69
4.2.1 System Model . . . . .	70
4.2.2 MWS Design Problem Formulation . . . . .	71
4.2.3 Rate Estimation Using Full Knowledge of Statistical CSI . . . .	72
4.3 Rate Estimation Using Mean and Variance of SNRs . . . . .	74
4.3.1 Bounding the Feasible Rate Region . . . . .	74
4.3.2 The Proposed Rate Estimation for MWSs . . . . .	78
4.4 Proposed MVWO for MWS Design . . . . .	79
4.4.1 Issue of Existing Iterative Solvers for the BLOP . . . . .	79
4.5 Proposed Iterative Solver for MVWO . . . . .	80
4.5.1 Initialisation of the Proposed Iterative Process . . . . .	80
4.5.2 Low-Complexity Iterative Updates of Weights . . . . .	81
4.5.3 Termination Condition of the Proposed Iterative Process . . . .	82
4.6 Convergence of Proposed Iterative Solver . . . . .	82
4.6.1 Convergence of Algorithm 2 . . . . .	83
4.6.2 Computational Complexity of the Proposed Iterative Solver . .	84
4.7 Online MVWO Architecture for Varying Mean and Variance of SNRs	85
4.8 Evaluation of Proposed Methods . . . . .	87
4.8.1 Simulation Configurations . . . . .	87
4.8.2 Other OS Approaches Compared in Simulation . . . . .	87
4.8.3 Performance of Proposed Rate Estimation Method . . . . .	89
4.8.4 Evaluation on the Convergence of Algorithm 2 . . . . .	91
4.8.5 Performance and Time Complexity of MVWO . . . . .	94
4.8.6 Performance of Online MVWO Architecture for Varying Mean and Variance of SNRs . . . . .	98
4.9 Summary . . . . .	100

<b>5</b>	<b>Scheduler Design Using Graph Representation Learning for Con-</b>	<b>101</b>
	<b>tention and Interference Management</b>	
5.1	Introduction . . . . .	102
5.1.1	Related Works . . . . .	103
5.1.2	Our Methods . . . . .	105
5.2	System Model and Problem Formulation . . . . .	106
5.2.1	Configurations of the Wireless Network . . . . .	106
5.2.2	Network States . . . . .	108
5.2.3	User Grouping Problem in RAW . . . . .	108
5.3	Proposed User Grouping Framework for Wireless Networks . . . . .	110
5.4	Proposed Actor-Critic Graph Representation Learning Algorithm . . . . .	112
5.4.1	Design of the Graph-Constructing Actor . . . . .	114
5.4.2	Design of the Graph Cut Procedure . . . . .	115
5.4.3	Design of the Graph-Evaluating Critic . . . . .	119
5.4.4	The Flow of the AC-GRL Algorithm . . . . .	122
5.5	Proposed Online Fine-Tuning Architecture . . . . .	124
5.6	Evaluation of Proposed Methods . . . . .	126
5.6.1	Simulation Configurations . . . . .	127
5.6.2	Compared Methods in Simulations . . . . .	128
5.6.3	Performance of Proposed AC-GRL Algorithms . . . . .	130
5.6.4	Performance of Designed User Grouping Decisions . . . . .	131
5.6.5	Performance of Proposed Online Architecture . . . . .	135
5.7	Summary . . . . .	138
<b>6</b>	<b>Conclusion</b>	<b>139</b>
6.1	Summary of Content and Results . . . . .	139
6.2	Future Directions . . . . .	141
6.2.1	Extension to DRL-Based Scheduler Design . . . . .	141
6.2.2	Extension to MVWO-Based Scheduler Design . . . . .	142
6.2.3	Extension to GRL-Based Scheduler Design . . . . .	143

*Contents*

<b>A Proofs of Chapter 3</b>	<b>144</b>
<b>B Proofs of Chapter 4</b>	<b>146</b>
<b>C Proofs of Chapter 5</b>	<b>151</b>

# List of Figures

2.1	Illustration of resource blocks in cellular networks. . . . .	15
2.2	Illustration of RAW time slots in Wi-Fi 802.11 ah networks. . . . .	17
3.1	Illustration of a downlink scheduler. . . . .	32
3.2	Illustration of DDPG. . . . .	35
3.3	Processing in the BS: (a) low-complexity scheduler, (b) high-complexity scheduler. . . . .	41
3.4	Illustration of (a) single-head critic and (b) multi-head critic. . . . .	45
3.5	Illustration of the potential function for reward shaping. . . . .	47
3.6	Proposed online DDPG architecture. . . . .	51
3.7	Parallel processing in the BS. . . . .	51
3.8	Packet loss probabilities, where $K = 5$ and $N = 50$ . (a) Straightforward implementation of DDPG, (b) DDPG in T-DRL framework, (c) K-DDPG in T-DRL framework. . . . .	54
3.9	Packet loss probabilities, where $K = 15$ and $N = 50$ . (a) Straightforward implementation of DDPG, (b) DDPG in T-DRL framework, (c) K-DDPG in T-DRL framework. . . . .	54
3.10	Rewards of different DDPG in the T-DRL framework, where $K = 15$ and $N = 50$ . . . . .	57
3.11	Rewards of DDPG with the assistance of different kinds of knowledge in the T-DRL framework, where $K = 15$ and $N = 50$ . . . . .	58

*List of Figures*

3.12	Reliability of scheduler for different total numbers of RBs, where $K = 15$ . (a) Average packet loss probability of all users; (b) Packet loss probability of the worst-case user. . . . .	59
3.13	Diagram of the experiment setup. . . . .	61
3.14	Results of online training in the proposed architecture. . . . .	63
4.1	Illustration of a wireless scheduler. . . . .	69
4.2	The proposed online MVWO architecture. . . . .	86
4.3	The average rates achieved by $\mu(\cdot \mathbf{w})$ and their estimated values in (4.26) for different $\mathbf{w}$ . . . . .	90
4.4	Evaluation of the convergence of Algorithm 2 when $K = 5$ or $10$ and $\hat{\epsilon} = 10^{-4}$ . . . . .	92
4.5	The probability that the convergence of Algorithm 2 occurs in given iterations. . . . .	93
4.6	Values of the utility function achieved by MWSs optimised by the proposed MVWO method and the SUWO methods when $\tilde{T}$ time slots are used. . . . .	94
4.7	Difference in the utility function achieved by MWSs optimised by the proposed MVWO method and the SUWO methods when $\tilde{T}^*$ and $\tilde{T}$ time slots are used, respectively. . . . .	95
4.8	Values of the utility function achieved by the MWS optimised by our MVWO method, the HFS in (4.45), and the MDP-based OS optimised by DRL [92] when $K = 3$ . . . . .	97
4.9	The difference in the performance of MWSs optimised by the online MVWO architecture and the SUWO methods [19], [20], where $K = 5$ . . . . .	99
5.1	Illustration of a wireless network. . . . .	106
5.2	Illustration of user grouping in RAW, where $K = 3$ , $Z = 2$ , $z_1 = 1$ and $z_2 = z_3 = 2$ . . . . .	109
5.3	The overall structure of the AC-GRL algorithm. . . . .	113
5.4	The structure of the actor. . . . .	114

*List of Figures*

5.5	Tree diagram illustrating the recursive graph cut when $Z = 4$ and $\beta = 1, \dots, \log_2(Z)$ . . . . .	116
5.6	The structure of the critic. . . . .	119
5.7	The proposed online fine-tuning architecture. . . . .	125
5.8	The training information of the inference NN. . . . .	132
5.9	The training information of the actor and the critic. . . . .	133
5.10	Comparison of the proposed method with other methods. . . . .	134
5.11	Locations of low-throughput users in the simulated area. . . . .	136
5.12	Ratios of worst-case and average user throughput to their initial values during fine-tuning for different numbers of users, $K$ . . . . .	137
A.1	Illustration of the queuing model. . . . .	144



# List of Tables

3.1	Issues of Straightforward Implementation of DDPG and Solutions to Address Them . . . . .	33
3.2	5G Scheduler Design Simulation Setup . . . . .	53
5.1	Configurations of FNNs in the Actor and the Critic . . . . .	128

# List of Algorithms

1	Knowledge-assisted DDPG . . . . .	50
2	Proposed Iterative Solver for the MVWO . . . . .	82
3	Recursive Graph Cut Procedure . . . . .	118
4	Proposed Actor-Critic Graph Representation Learning Algorithm . .	125

# List of Acronyms

3GPP	3rd Generation Partnership Project
4G	the 4th generation cellular network
5G	the 5th generation cellular network
6G	the 6th generation cellular network
AC-GRL	actor-critic graph representation learning
AP	access point
BLOP	bi-level optimisation problem
BS	base station
CDF	cumulative distribution function
CPU	central processing unit
CQI	channel quality indicator
CSI	channel state information
CSMA	carrier-sense multiple access
CSMA/CA	carrier-sense multiple access with collision avoidance
DCF	distributed coordination function
DDPG	deep deterministic policy gradient
DRL	deep reinforcement learning
E2E	end-to-end
EDF	earliest-deadline-first scheduler
eMBB	enhanced mobile broadband
eNodeB	evolved NodeB
FDD	frequency-division duplex
FIFO	first-in-first-out
FNN	fully-connected neural network
FPGA	field programmable gate array
GM	geometric mean
GNN	graph neural network
gNodeB	next generation NodeB
GPU	graphics processing unit
GRL	graph representation learning
HFS	heuristic fairness scheduler
HoL	head-of-line
IEEE	Institute of Electrical and Electronics Engineers

## *List of Acronyms*

IoT	Internet of Things
LHS	left-hand side
LLP	lower-level problem
LTE	long-term evolution
MAC	medium access control
MCS	modulation and coding scheme
ML	machine learning
mMTC	massive machine type communications
MPGNN	message-passing graph neural network
MT	maximum throughput scheduler
MVWO	mean-variance-based weight optimisation
MWS	max-weight scheduler
NN	neural network
NOMA	non-orthogonal multiple access
NP-hard	non-deterministic polynomial-time hardness
NR	New Radio
NS-3	Network Simulator 3
OAF	output activation function
OFDM	orthogonal frequency-division multiplexing
OFDMA	orthogonal frequency-division multiple access
OS	opportunistic scheduler
PDF	probability density function
PF	proportional fairness scheduler
PHY	physical
PoC	proof of concept
QoS	quality of service
RAN	radio access network
RAW	restricted access window
RB	resource block
RBG	resource block group
RE	resource element
REGNN	random-edge graph neural network
ReLU	rectified linear activation function
RF	radio frequency
RHS	right-hand side
RLC	radio link control
RR	round robin scheduler
RSSI	received signal strength indicator
SDP	semidefinite programming
SDR	software defined radio
SINR	signal-to-interference-plus-noise ratio
SISO	single-input and single-output
SNR	signal-to-noise ratio

## *List of Acronyms*

STA	station
SUWO	statistics-unaware weight optimisation
Tanh	hyperbolic tangent function
TB	transport block
TSN	time-sensitive networking
TTI	transmission time interval
UE	user equipment
ULP	upper-level problem
uRLLC	ultra reliable low latency communications
USB	universal serial bus
USRP	universal software radio peripheral
WLAN	wireless local area network

# Chapter 1

## Introduction

### 1.1 Schedulers in Wireless Networks

Wireless networks are critical in connecting people and devices in our modern world [1], [2]. Wireless networks have evolved in the past decades [3]–[6] by adopting recent theories and technologies to meet the growing demand for wireless connectivity and data services. Specifically, past wireless networks provide human-oriented communication services with information exchange among people, such as text messages, audio calls, and image/video sharing. Meanwhile, the recent development of wireless networks has also focused on communication services to connect machines, enabling a wide range of new application scenarios with diverse quality of service (QoS) requirements, e.g., service categories of the 5G use-cases and network setups, including ultra-reliable low latency communications (uRLLC), enhanced mobile broadband (eMBB) and massive machine type communications (mMTC) [7]. The key performance measures of these services are explained below.

- *Low latency and jitter*: The development of wireless networks leads to a fundamental change in industrial automation [8], [9] by enabling real-time interaction among sensors, controllers and actuators. Specifically, the data packets for these automation devices must be delivered within a short time and at a specific time

### 1.1. Schedulers in Wireless Networks

window. In other words, the wireless links for this service should have a low end-to-end (E2E) latency (less than 1 millisecond) with a small variance (or a low jitter). Such networks are referred to as time-sensitive networking (TSN).

- *High reliability and connectivity*: The above applications are also sensitive to network service failures, which impose the necessary minimum time interval between two contiguous failures, e.g., one month to one year. Such requirements can be translated into the decoding error probability of wireless transmissions [9]. Since the required interval of service failures is large, the decoding error probability is supposed to be extremely low; e.g., a typical reliability target of packet transmissions is to have at least 99.999% decoding success rate [10].
- *High throughput*: Besides machine-to-machine services, emerging applications demand human-to-machine interaction, such as extended reality services. These services encompass augmented and virtual reality [11] to provide remote manual control of machines. They require the delivery of a high-quality video stream from the machine's (real-world or virtual) cameras to headsets that project the video to human eyes over wireless networks. To eliminate eye fatigue or visual discomfort, the transmitted video needs to preserve depth perception information as well as high resolution, which significantly increases the required bandwidth to a range from 100 Gbps to 1 Tbps [12].

Two popular types of wireless networks can be chosen to provide communication services with the above QoS. The first is the cellular network, standardised by the 3rd Generation Partnership Project (3GPP) [13]. The fifth-generation (5G) cellular network has recently been released and attracted significant interest from industry and academia [3]. 5G networks use orthogonal frequency-division multiple access (OFDMA), where users share the wireless channel in frequency and time. Specifically, users' data are modulated and coded as orthogonal frequency-division multiplexing (OFDM) symbols. Further, every twelve subcarriers in one transmission time interval (fourteen contiguous OFDM symbols) form one resource block (RB). A scheduler allocates RBs among users to control the transmission opportunities of users. Another

## 1.2. Research Problems and Contributions

widely used wireless technology is Wi-Fi networks [14], [15], standardised by the Institute of Electrical and Electronics Engineers (IEEE) 802.11 groups. In a Wi-Fi network, each user contends for channel access based on carrier-sense multiple access with collision avoidance (CSMA/CA). Specifically, when a user detects other users are transmitting on the channel, it will wait until the channel is free and will further backoff for a random time before transmitting its packet. IEEE 802.11ah (Wi-Fi HaLow) introduced a restricted access window (RAW) mechanism to manage the contention and interference in the network, where the scheduler assigns users RAW time slots for user transmissions. As a result, contention and interference do not occur across different RAW slots, significantly improving the network performance.

As the schedulers [16] are responsible for allocating the available resources in the wireless networks, they determine the transmissions' QoS and the networks' overall performance. In other words, optimal schedulers maximise the network's efficiency and ensure a high-quality user experience. Therefore, it is essential to continue research and innovation in methods of optimising scheduler design.

## 1.2 Research Problems and Contributions

Undoubtedly, the scheduler design in wireless networks is an active area of research, with well-developed algorithms to meet the evolving demands of the users. This thesis will focus on the following aspects of the scheduler design methods.

- *Flexibility*: With the development of wireless networks, new services with more strict QoS requirements can be supported over the network. Since the QoS requirements of these services are new, the existing schedulers are no longer optimal for them. Thus, new schedulers need to be designed. Further, as these services have diverse QoS requirements, it is inefficient to design and optimise the scheduler for them manually. A flexible method that can automatically generate the schedulers for these services is much needed.
- *Convergence rate*: When the scheduler is required to maximise a long-term



## 1.2. Research Problems and Contributions

network performance objective, the scheduler design needs to consider the long-term network variations, e.g., the variations of the channel quality. However, this requires the scheduler design algorithm to constantly interact with the network and optimise the scheduler according to the instant channel state over time, e.g., using stochastic optimisation. As a result, a significant length of time is needed before the algorithm converges to the optimal scheduler, during which the network operates in sub-optimal performance. How to improve the convergence speed needs to be studied.

- *Network-wise coordination*: Most existing schedulers are designed for one base station (BS) or access point (AP). Meanwhile, with the dense deployment of the network, users associated with the different BSs (or APs) share the channel resources. Thus, they interfere with each other's transmissions, negatively impacting network performance. Clearly, coordinating the schedulers across multiple BSs (or APs) in the network enables network-wise management of the interference. It requires further investigation on the best way to design network-wise coordinated scheduling schemes.
- *Online implementation*: Practical scheduler design should consider real-world system constraints. For example, the limited computational resources at the BSs (or APs) restrict the complexity of the scheduling algorithm. To address this issue, edge servers can be connected with the BSs (or APs) to provide the additional resources that run the algorithm. This raises the challenge of designing the online architecture for scheduling algorithms in the edge server to interact with wireless networks.

To address the above challenges, we propose three new scheduler design methods that improve flexibility, convergence rate, and efficient network-wise coordination compared with the existing methods. Also, we design the online architectures to deploy the proposed methods in real-world networks. The contributions of this thesis are listed as follows.

### 1.2.1 Flexible Scheduler Design Using Deep Reinforcement Learning

We enable the flexible scheduler design by training a neural network (NN) as the scheduler in Chapter 3. Specifically, we consider the scheduler for the emerging TSN service. We formulate the scheduler design problem as a Markov decision process (MDP) by defining the network states, scheduling actions, and QoS-indicating rewards, where the optimal scheduler corresponds to the optimal MDP policy. Then, we can apply a deep reinforcement learning (DRL) algorithm, namely deep deterministic policy gradient (DDPG), to optimise the scheduler. As such, the scheduler can be automatically trained as the NN by the DRL algorithm. We find that straightforward implementation of the existing DRL algorithm to scheduler design does not achieve a good performance. Thus, we provide a comprehensive study on improving the DRL algorithm's efficiency in scheduler design problems by exploiting expert knowledge in wireless communications. *To the best of our knowledge, this work [J1] is the first to provide an end-to-end solution of a DRL-based scheduler design from the theoretical formulation to a real-time prototype.* The main contributions of this chapter are summarised as follows.

- We establish a theoretical DRL (T-DRL) framework for wireless scheduler design with the time-sensitive traffic in 5G systems. The framework uses existing theoretical models and results in wireless communications to formulate the optimal control problem. Based on the formulation, we prove that the problem is Markovian, and hence we can apply DRL algorithms to solve it [17]. Simulation results show that the DRL algorithm with the T-DRL framework enables the convergence of the scheduler, providing a satisfying QoS. Meanwhile, the straightforward implementation fails to converge into a useful scheduler.
- We design a knowledge-assisted DDPG (K-DDPG) algorithm that integrates DDPG with expert knowledge to improve the QoS of users and reduce the convergence time. We use the multi-head critic, reward shaping and importance

## 1.2. Research Problems and Contributions

sampling in K-DDPG to exploit the knowledge of the QoS of each user, the target scheduling policy, and the importance of training samples. As a result, the K-DDPG algorithm reduces the convergence time significantly, and the trained scheduler achieves 30% to 50% fewer packet losses than existing schedulers.

- We develop an architecture that enables online training and inference of K-DDPG to fine-tune the scheduler in real-world networks. An edge server in the architecture first initialises the scheduler offline in a simulation platform built upon the T-DRL framework. Then, it keeps fine-tuning the scheduler according to feedback from real-world networks. Meanwhile, the BS executes the scheduling policy at every TTI and shares the feedback from real-world networks with the edge server.
- We build a prototype of the proposed architecture using a standard-compliant cellular network software suite that can communicate with commercial devices [18]. In the prototype, the online training converges in a few minutes, and the online inference can be executed within each TTI (1 millisecond). Thus, our approach can be applied to scheduler design in 5G NR.

### 1.2.2 Acceleration of Convergence in Scheduler Design Using Statistical Information of Wireless Channels

Wireless networks are stochastic systems mostly due to the randomness of the channel conditions. The optimisation of the scheduler is usually formulated as a stochastic optimisation process, such as the DRL algorithm designed above. These types of processes require a long time for the algorithm to interact with the network and find the optimal scheduler parameters that maximise the network performance. In Chapter 4, we accelerate the scheduler design algorithm's convergence using the wireless channels' statistical information. Particularly, we consider the optimisation of a class of widely used schedulers in cellular networks, namely the max-weight schedulers (MWSs), that schedule the user with the highest weighted instantaneous data rate in every time slot.

## 1.2. Research Problems and Contributions

We formulate an optimisation problem to find the optimal weights in MWSs based on limited statistical information, namely the mean and variance of users' signal-to-noise ratios (SNRs). *To the best of our knowledge, this work [J2] is the first that proposes to design MWSs based on limited prior knowledge of statistical CSI, such as the mean and variance of users' SNRs, that costs few samples to estimate.* We summarise the main contributions of this chapter as follows.

- We formulate the weight optimisation problem for MWSs using the mean and variance of users' SNRs to maximise a utility function of users' average data rates. This enables us to find the optimal MWS's weights by measuring the mean and variance of users' SNRs and then solving the formulated weight optimisation problem to obtain the optimal weights. As a result, the proposed method reduces the time complexity (i.e., the number of time slots required) in optimising the MWS's weights 4 ~ 10 times compared to existing methods [19], [20], which use no prior knowledge of statistical CSI and directly perform online adjustment of weights, as shown by simulations.
- We design a new iterative solver for the weight optimisation problem, which has less computational complexity than existing iterative solvers designed for our problem structure [21]–[24]. Specifically, the designed solver updates weights in each iteration via normalisation and a linear combination of vectors rather than by solving an optimisation problem in each iteration as the existing iterative solvers do. It reduces the complexity of weight updates, e.g., from polynomial computational complexity in the existing solvers to the linear one in ours.
- We mathematically prove that the weights in the proposed solver converge to their optimal value within  $O(K \log K)$  iterations, where  $K$  is the number of users. The optimal value of weights maximises the utility function in the weight optimisation problem. The simulation results show that the designed iterative solver converges to the optimal weights in tens of iterations at a high probability, e.g., 90 ~ 100%, when there are up to ten users.

## 1.2. Research Problems and Contributions

- We design an online architecture where the proposed solver continuously adjusts the MWS's weights based on the time-varying mean and variance of the SNRs measured online. The simulation results show that the designed architecture achieves 5 ~ 15% better performance than the existing MWS approaches [19], [20] in terms of the geometrical mean of users' average rates (an equivalent expression of the studied utility function).

### 1.2.3 Network-Wise Coordination in Scheduler Design Using Graph Representation Learning

We investigate the coordination of the schedulers across multiple BSs (or APs) in the whole wireless network in Chapter 5. We study how to schedule the RAW slots for users for contention and interference management in the Wi-Fi HaLow networks. The objective here is to avoid user throughput starvation by maximising the worst-case user throughput. We construct a graph where each user is a vertex of the graph, and weighted directed edges represent the contention and inter-user interference from one user to another. Based on this graph, users are divided into a given number of groups by the graph's max cut. Then, each group is assigned a periodic RAW slot for their transmissions, where the contention and interference between users in different groups are eliminated. Here, we propose to construct the optimal graph whose max cut results in the optimal grouping decisions maximising the worst-case user throughput. We then design a machine learning (ML) algorithm to optimise an actor NN to construct the above optimal graph. We efficiently solve the constructed graph's max cut using semidefinite programming [25]. A critic NN is trained to evaluate the graph constructed by the actor, whose gradient is used to optimise the actor. Furthermore, considering the mismatch between real-world networks and simulated networks used in offline training, we study how to improve the actor-constructed graph based on online measurements. *To the best of our knowledge, this work [J3] is the first to propose to formulate the user grouping problem in wireless networks as a graph construction problem.* Our contributions in this chapter are summarised as follows.

## 1.2. Research Problems and Contributions

- We propose an optimisation framework that formulates the user grouping problem in RAW as a graph construction problem in which the graph's edge weights are optimisation variables and the grouping decisions are computed from the constructed graph's max cut. Unlike the existing graph-based approach [26]–[28] using heuristically constructed edges, the framework can flexibly optimise edge weights in the network's graph representation according to the specific network performance objective, i.e., maximising the worst-case user throughput.
- We develop an actor-critic graph representation learning (AC-GRL) algorithm that trains NNs to construct the optimal graph representing the impact of interference in a wireless network. The interference in each individual user pair is represented as the edge weight in the pair. Note that the edge weight in the NN-constructed graph indicates how likely a pair of users belong to the same group when the graph's max cut is applied to obtain user grouping decisions. This is unlike the existing ML-based approach [29], [30] that directly uses NNs to generate user grouping decisions, which loses the above individual user-pair-wise interference information by aggregating all neighbouring users' features. Consequently, these methods fail to return useful decisions in user grouping.
- We design an architecture for fine-tuning the NN-constructed graph based on users' throughput measured online. Specifically, we first initialise the actor and critic using offline-trained NN parameters. Next, the offline-trained actor constructs the graph based on the states of a given network. We then use the offline-trained critic continuously updates the graph's edge weights based on the user index with the worst-case throughput measured from the network. Meanwhile, user grouping decisions are also re-computed after each edge weight update by using the graph cut procedure.
- We implement the proposed methods in a system-level simulation platform [31], NS-3, compliant with Wi-Fi standards, where we also study and apply the existing Markov-model-based [32], [33], graph-based [26]–[28] and ML-based [29], [30] approaches to the user grouping problem in RAW. Simulations show that our

### 1.3. Thesis Outline

grouping decisions achieve around 30% ~ 80% higher worst-case user throughput than the existing approaches. Simulations also show that the proposed architecture can improve the worst-case user throughput 15% ~ 35% by online fine-tuning the graph constructed by the offline-trained NNs.

## 1.3 Thesis Outline

The rest of this thesis is organised as follows. Chapter 2 briefly introduces the concepts and methods used in the proposed schemes in this thesis. The main contributions of this thesis can be found in Chapter 3-5. Chapter 3 focuses on the DRL-based scheduler design in 5G cellular networks. In Chapter 4, we study the acceleration of scheduler design using the statistical information of wireless channels. In Chapter 5, we present the coordinated scheduler design in wireless networks for contention and interference management. Finally, Chapter 6 summarises this thesis and its major findings, along with some concluding remarks and future directions.

## 1.4 Notations

The notations used in this thesis are summarised as follows. The  $i$ -th element of a vector,  $\mathbf{x}$ , is denoted as  $x_i$ . The  $j$ -th element of the  $i$ -th row of a matrix,  $\mathbf{X}$ , is denoted as  $X_{i,j}$ . We write the definition of elements in a matrix,  $\mathbf{X}$ , as  $\mathbf{X} \triangleq [X_{i,j} | X_{i,j} = (\dots)]$ , where  $(\dots)$  is the expression that defines the elements in  $\mathbf{X}$ .  $\mathbf{diag}\{\mathbf{X}\}$  denotes the elements in the diagonal of a matrix,  $\mathbf{X}$ .  $\langle \mathbf{x}, \mathbf{y} \rangle$  denotes the inner product of  $\mathbf{x}$  and  $\mathbf{y}$ .  $\|\mathbf{x}\|_2$  denotes the  $\ell_2$ -norm of  $\mathbf{x}$ .  $\mathbf{x} \odot \mathbf{y}$  and  $\mathbf{x} \oslash \mathbf{y}$  are element-wise multiplication and division between  $\mathbf{x}$  and  $\mathbf{y}$ , respectively. A  $K$ -dimension vector,  $\mathbf{x}$ , with non-negative or positive elements are denoted as  $\mathbf{x} \in \mathbb{R}_{\geq 0}^K$  and  $\mathbf{x} \in \mathbb{R}_{> 0}^K$ , respectively. The  $i$ -th element of a tuple,  $(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(i)}, \dots)$ , is obtained as  $\mathbf{x}^{(i)} = \text{proj}_i[(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(i)}, \dots)]$ . The geometric mean of elements in a  $K$ -dimension vector,  $\mathbf{x}$ , is expressed as  $\text{GM}(\mathbf{x}) = (\prod_{k=1}^K x_k)^{\frac{1}{K}}$ , where  $x_k$  is the  $k$ -th element of  $\mathbf{x}$ . A  $K \times K$  positive semidefinite matrix,  $\mathbf{X}$ , is denoted as  $\mathbf{X} \succeq 0$ , where positive semi-definiteness implies  $\mathbf{z}^T \mathbf{X} \mathbf{z} \geq 0, \forall \mathbf{z} \in \mathbb{R}^K$ .

#### 1.4. Notations

$\mathbf{svd}(\mathbf{X})$  is a singular value decomposition of  $\mathbf{X}$ .  $\mathbf{sgn}(\mathbf{x})$  is a vector of each element's sign (+1 or -1) in  $\mathbf{x}$ .  $\mathbf{1}_{\{\dots\}}$  is an indicator function that equals 1 if the expression in  $\{\dots\}$  is true or otherwise equals 0.



# Chapter 2

## Background

*In this chapter, we provide some background information on the mathematical tools used in the analysis and optimisation in the thesis.*

### 2.1 Channel Capacity

In communication systems, the data rate (or throughput) is one of the main characteristics of the system's performance. The mapping of the data payload into a continuous-time signal that is transmitted over a communication channel can be described by

$$N_{\text{BLK}} = B \cdot T, \quad (2.1)$$

where  $B$  is the bandwidth of the channel in Hertz,  $T$  is the duration of the transmission in seconds, and  $N_{\text{BLK}}$  is referred to as the blocklength of the transmission. Note that data transmissions over wireless channels suffer from uncontrollable ambient noise, imperfections of the physical signalling, and interference from other concurrent transmissions [16]. The channel capacity [34]–[36], expressed in bits per second, is a fundamental limit on the amount of information that can be reliably transmitted through the communication channel.

## 2.1. Channel Capacity

### 2.1.1 Shannon Capacity

Shannon capacity is a term named after Claude Shannon, the father of modern information theory, who introduced the concept in 1948 [34]. In general, the Shannon capacity determines the maximum amount of information per unit of time (i.e., data rate:  $R_{\max}(N_{\text{BLK}}, \epsilon)$ ) that can be achieved over a wireless channel when the transmission blocklength approaches infinity (i.e.,  $N_{\text{BLK}} \rightarrow \infty$ ) and the error rate of the transmissions,  $\epsilon$ , is arbitrarily small. The asymptotic achievable rate of the additive white Gaussian noise (AWGN) channel is given based on Shannon's capacity as

$$\begin{aligned} C &= \lim_{N_{\text{BLK}} \rightarrow \infty, \epsilon \rightarrow 0} R_{\max}(N_{\text{BLK}}, \epsilon) \\ &= \mathbb{E}[B \log_2(1 + |h|^2 \phi)] \quad (\text{bit/second}) , \end{aligned} \tag{2.2}$$

where  $\phi$  is the transmissions' SNR (or SINR when interference exists) and  $h$  is the random normalised fading gain [37]. Here, the expectation in (2.2) is computed with respect to the random variable  $h$ . If the channel has flat fading and is quasi-static (i.e., the channel coherent time is larger than the transmission duration), i.e., the channel fading gain remains a constant during each transmission, the capacity is computed as  $C = B \log_2(1 + \phi)$ .

### 2.1.2 Channel Capacity in Finite Block Length Regime

In the finite block length regime, i.e., when  $N_{\text{BLK}}$  is small, the channel capacity is modified from the theoretical limit described by the Shannon Capacity. In this regime, the amount of information transmitted over a channel is limited by the finite size of the data blocks, in addition to the channel's noise level (or noise and interference) or bandwidth. An accurate approximation of the achievable capacity in a short block-length regime is given by [35], [36], [38], [39] as

$$R \approx C - \sqrt{\frac{V}{N_{\text{BLK}}}} f_Q^{-1}(\epsilon) \quad (\text{bit/second}) , \tag{2.3}$$

## 2.2. Radio Resource Scheduling in Wireless Networks

where  $C$  is the Shannon capacity in (2.2) and  $f_Q^{-1}$  is the inversion function of the tail distribution function of the standard normal distribution, i.e.,  $f_Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-\frac{u^2}{2}} du$ . Here,  $V$  in (2.3) is the channel dispersion computed as

$$V = t' \mathbb{V}[\log_2(1 + |h|^2\phi)] + 1 - \mathbb{E}^2 \left[ \frac{1}{1 + |h|^2\phi} \right], \quad (2.4)$$

where  $h$  is the random normalised fading gain and  $t'$  is the channel coherent time. When the channel has flat fading and quasi-static, the channel dispersion is computed as  $V = 1 - 1/(1 + \phi)^2$ . In such a case, if we assume that each transmission block contains one packet with length  $L$  in bits, then the above approximation can be used to estimate the error probability of transmissions as

$$\epsilon \approx f_Q \left( \frac{-L \ln 2 + N_{\text{BLK}} \ln(1 + \phi)}{\sqrt{N_{\text{BLK}} V}} \right). \quad (2.5)$$

## 2.2 Radio Resource Scheduling in Wireless Networks

This section explains the radio resource scheduling mechanisms in two common wireless networks, e.g., cellular and Wi-Fi networks. We will start with the transmission scheme used in each type of network and how they utilise space-time resources for the transmissions. Then, we will explain how these resources are scheduled.

### 2.2.1 Radio Resource Scheduling in Cellular Networks

Modern cellular networks establish communications with user equipment and base stations (BSs) using standardised wireless communication protocols by the 3rd Generation Partnership Project (3GPP) [13]. For example, the recently developed 5th-generation cellular network (5G) uses orthogonal frequency-division multiple access (OFDMA). Specifically, the transmitter modulates the data as orthogonal frequency-division multiplexing (OFDM) symbols, where coded data bits are mapped to the

## 2.2. Radio Resource Scheduling in Wireless Networks

subcarriers. The number of subcarriers in one OFDM symbol depends on the bandwidth and the subcarrier spacing configured in the networks as

$$N_s \approx \frac{B}{f_s}, \quad (2.6)$$

where  $B$  is the bandwidth of the channel used in the transmissions in Hertz,  $f_s$  is the subcarrier spacing in Hertz, and  $N_s$  is the number of subcarriers used in the transmissions. Further, every fourteen contiguous OFDM symbols are transmitted as one transmission time interval (TTI). Every twelve subcarriers in one TTI (fourteen contiguous OFDM symbols) form one resource block (RB). An illustration of RBs is shown in Fig. 2.1. The scheduler in cellular networks assigns users RBs to carry

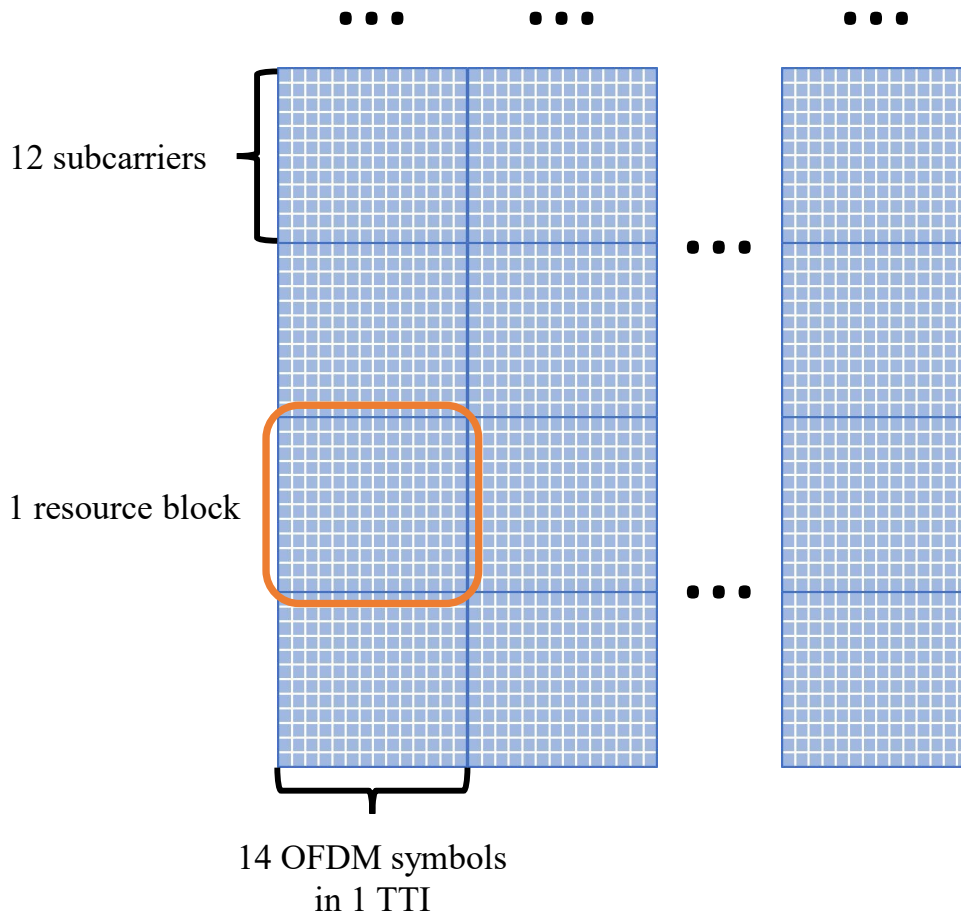


Figure 2.1: Illustration of resource blocks in cellular networks.

## 2.2. Radio Resource Scheduling in Wireless Networks

their transmission data. Specifically, for simple single-input single-output (SISO) transmissions (considered in this thesis), each RB is assigned to a single user and cannot be shared by multiple users. Also, one user can be assigned multiple RBs in one TTI. Consider a user's transmission with a given number of RBs. The number of information bits coded and modulated in scheduled RBs in this transmission is decided based on the modulation and coding scheme (MCS) selected by the scheduler to satisfy the decoding error rate requirement. The number of information bits in each transmission can be calculated as

$$L \approx 12 \times 14 \times N_{\text{RB}} \times \log_2(M) \times R , \quad (2.7)$$

where  $N_{\text{RB}}$  is the number of RBs assigned to the transmission,  $M$  is the modulation order (e.g., 2 for binary phase-shift keying and 4 for quadrature phase shift keying etc.) and  $R$  is the code rate.

### 2.2.2 Radio Resource Scheduling in Wi-Fi Networks

The Institute of Electrical and Electronics Engineers (IEEE) 802.11 task groups standardise the communication protocol in Wi-Fi networks. In Wi-Fi networks, the devices requiring network connection services are referred to as stations, and the devices providing these services are referred to as access points (APs). In this thesis, we refer to the stations as the users, consistent with the ones used in cellular networks. Users communicate with APs using a carrier sense multiple access/collision avoidance (CSMA/CA) method. Specifically, each device contends for channel access by listening to the channel and waiting until it is free before transmitting a packet. Random backoff is used to manage situations where multiple devices try to access the channel simultaneously. For example, each device randomly chooses a backoff time before transmitting the packet. This helps to distribute transmissions over time among the devices and avoid collisions. Note that when one device cannot sense another, they can make simultaneous packet transmissions that cause interference at the receiving devices, which increases packet decoding errors.

### 2.3. Deep Learning

The 802.11 ah standard [15] introduces a restricted access window (RAW) mechanism in the Wi-Fi network to manage contention and interference. Specifically, each user can be assigned a periodic RAW time slot in which the user transmits packets only. For example, as shown in Fig. 2.2, user 1 is assigned to slots 1, 3, 5, . . . , while user 2 is assigned to slots 2, 4, 6, . . . . In such a case, the contention and interference between user 1 and user 2 are eliminated. By scheduling RAW slots for users, we can manage the contention and interference in Wi-Fi networks. Scheduling of RAW slots will be further explained in Chapter 5.

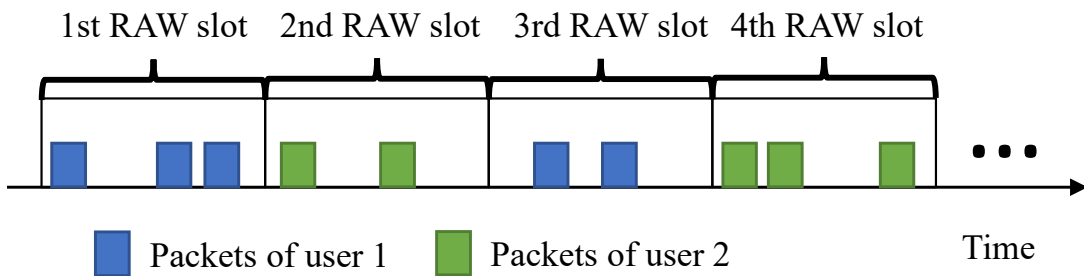


Figure 2.2: Illustration of RAW time slots in Wi-Fi 802.11 ah networks.

## 2.3 Deep Learning

Deep learning (DL) is a family of machine learning methods that train neural networks (NN) to approximate a desired function, where a NN,  $\mu(\cdot|\theta^\mu) : \mathcal{X} \rightarrow \mathcal{Y}$  is a function with adjustable parameters,  $\theta^\mu$ . Each NN requires a predefined functional structure to accommodate these parameters in the NN. For example, as explained below, two common structures of NNs are fully-connected NNs (FNN) and graph NNs (GNN).

### 2.3.1 Fully-connected Neural Networks

When we use a FNN as  $\mu(\cdot|\theta^\mu)$ , it consists of multiple fully-connected layers, where the mathematical expression from one layer to the next layer is

$$\mathbf{y}^{l+1} = \sigma^l(\mathbf{W}^l \mathbf{y}^l + \mathbf{b}^l), \quad l = 1, \dots, \chi - 1, \quad (2.8)$$

### 2.3. Deep Learning

where  $\chi$  is the number of layers,  $\mathbf{y}^1$  and  $\mathbf{y}^\chi$  are referred to as the input layer and output layer, respectively, and  $\mathbf{y}^l$ ,  $l \neq 1, l \neq \chi$  are referred to as the hidden layers. Further,  $\mathbf{W}^l$  and  $\mathbf{b}^l$  in the above are the adjustable matrix-shaped weights and vector-shaped biases in the  $l$ -th layer. These are the adjustable parameters of the NN, i.e.,  $\theta^\mu = \{\mathbf{W}^l, \mathbf{b}^l | l = 1, \dots, \chi\}$ . Also,  $\sigma^l$  is a non-linear activation function in the  $l$ -th layer, e.g.,  $\text{ReLU}(\cdot)$  or  $\text{Sigmoid}(\cdot)$  [40]. Since the input and output layers must have a corresponding dimension with the weights and biases in these two layers, FNNs have a predefined input and output dimension. Thus, FNN is not flexible in approximating the function with varying input and output dimensions.

#### 2.3.2 Graph Neural Networks

We can model the function's inputs as vertex or edge features on a graph whose size is scalable according to input and output dimensions. Then, GNNs can process the inputs and obtain the outputs by repeatedly aggregating the features on neighbouring vertices and edges. For example, this thesis considers a fully-connected directed graph,  $\mathcal{G}$ , with a set of vertices  $\mathcal{V}$  and a set of edges  $\mathcal{E}$ , where  $\mathcal{G} \triangleq (\mathcal{V}, \mathcal{E})$  and

$$\mathcal{V} \triangleq \{1, \dots, K\}, \quad \mathcal{E} \triangleq \{(i, j) | \forall i, j \in \mathcal{V}, i \neq j\}, \quad (2.9)$$

where  $K$  is the number of vertices in the graph and  $(i, j)$  is the edge from vertex  $i$  to vertex  $j$ . GNNs first map inputs to the features on the vertices and edges and then process them using multiple layers of GNN operators. For example, layers of GNN operators [41] can be expressed as

$$\mathbf{x}_i^{l+1} = g^l\left(\mathbf{x}_i^l, \oplus_{i \neq j} h^l(\mathbf{x}_i^l, \mathbf{x}_j^l, \mathbf{e}_{i,j} | \theta^{h^l}) | \theta^{g^l}\right), \quad l = 1, \dots, \chi - 1, \quad (2.10)$$

where  $\chi$  is the total number of layers,  $\mathbf{x}_i^l$  is the features on vertex  $i$  in the  $l$ -th layer, and  $\mathbf{e}_{i,j}$  is the features on the edge from vertex  $i$  to vertex  $j$ . Here,  $\oplus(\cdot)$  in the above is the differentiable function aggregating all neighbouring features (e.g., sum, max and mean functions) and  $h^l(\cdot | \theta^{h^l})$  and  $g^l(\cdot | \theta^{g^l})$  denote two differentiable parameterised functions

### 2.3. Deep Learning

such as FNNs in the  $l$ -th layer, where  $\theta^{h^l}$  and  $\theta^{g^l}$  are the adjustable parameters of the GNN, i.e.,  $\theta^\mu = \{\theta^{h^l}, \theta^{g^l} | l = 1, \dots, \chi\}$ .

#### 2.3.3 Stochastic Gradient Descent

The NN-based function approximation will serve two purposes in this thesis. The first purpose is to imitate an unknown function using the NN,  $\mu(\cdot|\theta^\mu)$  (either a FNN or a GNN). The training algorithm samples random input-output pairs of the functions and optimises the NN's parameters to minimise a loss function measuring the difference between the NN and the target function, e.g.,

$$L(\theta^\mu) = \mathbb{E}_{\mathbf{x} \sim p_{\mathcal{X}}} \left[ d(\mu(\mathbf{x}|\theta^\mu), f(\mathbf{x})) \right], \quad (2.11)$$

where  $f : \mathcal{X} \rightarrow \mathcal{Y}$  is the target function,  $p_{\mathcal{X}}$  is the distribution of the target function's input, and  $d(\cdot, \cdot)$  is the measure of the difference between the NN's and the target function's outputs, e.g.,  $\ell_2$ -norm or cross-entropy. The second purpose of the NN-based function approximation is to train the NN,  $\mu(\cdot|\theta^\mu)$ , as an optimiser minimising an objective function. Here, the loss function can be expressed mathematically as

$$L(\theta^\mu) = \mathbb{E}_{\mathbf{x} \sim p_{\mathcal{X}}} \left[ f(\mathbf{x}, \mu(\mathbf{x}|\theta^\mu)) \right], \quad (2.12)$$

where  $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$  is the objective function,  $\mathbf{x} \in \mathcal{X}$  is system states specifying the objective function, and  $p_{\mathcal{X}}$  is the distribution of  $\mathbf{x}$ . The parameters of the NN,  $\theta^\mu$ , are optimised for the above two purposes by performing the stochastic gradient descent method [42] as

$$\theta^\mu \leftarrow \theta^\mu - \eta \nabla_{\theta^\mu} L(\theta^\mu), \quad (2.13)$$

where  $\eta$  is the learning rate to control the update step size of the parameters.



## 2.4 Deep Reinforcement Learning

### 2.4.1 Markov Decision Process

A Markov decision process (MDP) is a mathematical framework for modelling decision-making problems where the consequence of the decision is uncertain. The MDP can be described by a 4-tuple,  $(\mathcal{S}, \mathcal{A}, \mathbf{P}, r)$ , explained as follows.

- State space  $\mathcal{S}$ : the set of all possible states that the system can be.
- Action space  $\mathcal{A}$ : the set of all possible actions in each state.
- Transition probabilities  $\mathbf{P}$ : the probabilities of moving from one state to another, given the state and the action taken.
- Reward function  $r$ : the reward received by the decision-maker for taking a particular action in a given state.

Note that the MDP is assumed to be stationary in time, i.e., the above properties of MDP do not change over time. In this thesis, we consider discrete-time MDPs where the system time is slotted as  $1, \dots, t, \dots$ . The state is measured at the beginning of each slot as  $s(t)$ , and then the action is taken in each slot as  $a(t)$ , where  $t = 1, 2, \dots$ . Note that  $s(t) \in \mathcal{S}$  and  $a(t) \in \mathcal{A}, \forall t$ , and we refer to  $s(t)$  and  $a(t)$  as the present state and the present action in the  $t$ -th slot, respectively. The reward received by taking the action in the  $t$ -th slot is  $r(t)$ ,  $t = 1, 2, \dots$ , where  $r(t)$  is a function on the present state and action. Then, the system transits to the next slot, and its state becomes  $s(t + 1)$ , which follows the transition probabilities in  $\mathbf{P}$ . The transitions in MDPs are memoryless, i.e., they only depend on the present state and action. They are not relevant to the states and actions in the previous slots, which can be mathematically written as

$$\begin{aligned} & \Pr[s(t + 1)|s(1), a(1), \dots, s(t), a(t)] \\ &= \Pr[s(t + 1)|s(t), a(t)] , \forall s(1), a(1), \dots, s(t), a(t) , \end{aligned} \tag{2.14}$$

where  $\Pr[s(t + 1)|s(t), a(t)]$  is one of the element in  $\mathbf{P}$  and can be written as  $p(s(t + 1), s(t), a(t))$ . The goal of the MDP is to find a stationary policy, i.e., a mapping

## 2.4. Deep Reinforcement Learning

from the state to the action in each slot, that maximises the long-term reward over time as

$$G(t) \triangleq r(t) + \gamma r(t+1) + \gamma^2 r(t+2) + \dots = \sum_{t'=t}^{\infty} \gamma^{(t'-t)} r(t'), \quad (2.15)$$

where  $\gamma \in [0, 1)$  is referred to as a discount factor that indicates how much future rewards contribute to the long-term rewards in present. Let  $\pi : \mathcal{S} \rightarrow \mathcal{A}$  denote any one of the feasible stationary policies, then a state-action value function of this policy can be defined as

$$Q^\pi(s(t), a(t)) \triangleq \mathbb{E} \left[ G(t) | s(t), a(t), a(t') = \pi(s(t')), \forall t' > t \right], \quad \forall t, \quad (2.16)$$

which shows the value of the long-term reward when taking  $a(t)$  given  $s(t)$  and then using  $\pi$  as the policy in all following states. A state value function can be also defined based on the given policy,  $\pi$ , as

$$V^\pi(s(t)) \triangleq \mathbb{E} \left[ G(t) | s(t), a(t') = \pi(s(t')), \forall t' \geq t \right] = Q^\pi(s(t), \pi(s(t))), \quad \forall t, \quad (2.17)$$

which represents the value of the long-term reward achieved by  $\pi$  given that the present state is  $s(t)$ . The above two functions can be decomposed by using the Bellman equation as

$$\begin{aligned} Q^\pi(s(t), a(t)) &= \mathbb{E} \left[ r(t) | s(t), a(t) \right] + \gamma \mathbb{E} \left[ Q^\pi(s(t+1), \pi(s(t+1))) | s(t), a(t) \right], \quad \forall t, \\ V^\pi(s(t)) &= \mathbb{E} \left[ r(t) | a(t) = \pi(s(t)) \right] + \gamma \mathbb{E} \left[ V^\pi(s(t+1)) | a(t) = \pi(s(t)) \right], \quad \forall t. \end{aligned} \quad (2.18)$$

The optimal policy satisfies the following equation as

$$V^{\pi^*}(s) = \max_{a \in \mathcal{A}} \left\{ \mathbb{E} \left[ r | s, a \right] + \gamma \mathbb{E}_{s' \sim \mathbf{P}} \left[ V^{\pi^*}(s') | a = \pi(s) \right] \right\}, \quad \forall s \in \mathcal{S}, \quad (2.19)$$

where  $s'$  is the next-slot state of  $s$  given action  $a$ . The above equality is referred to as the Bellman optimality equation and  $V^{\pi^*}(s)$  is referred to as the optimal state value function. Also, we can define the optimal state-action value function based on the

## 2.4. Deep Reinforcement Learning

optimal state value function as

$$Q^{\pi^*}(s, a) = \mathbb{E} [r|s, a] + \gamma \mathbb{E}_{s' \sim \mathbf{P}} [V^{\pi^*}(s')|a = \pi(s)], \quad \forall s \in \mathcal{S}, \quad \forall a \in \mathcal{A}, \quad (2.20)$$

The optimal value functions in the above can be solved by using dynamic programming methods [17], [43], such as value iteration or policy iteration, when the reward function and all transition probabilities are explicitly known. However, this can be difficult to achieve in a practical system, where the above information is unknown and requires extensive time and complexity to measure. To address this issue, reinforcement learning can be used to solve MDP and find the optimal policy without knowing the reward function and transition probabilities, as explained as follows.

### 2.4.2 Reinforcement Learning

Reinforcement learning solves MDPs, where an agent interacts with the process and learns the best action in each state. For example, Q-learning [44] is one of the well-known reinforcement learning algorithms. Q-learning uses a table to save all state-action values for all combinations of state-action pairs, e.g.,  $Q(s, a) \quad \forall s \in \mathcal{S}$  and  $\forall a \in \mathcal{A}$ . Then, the agent in Q-learning selects the best action in each state based on values saved in the table. Also, to explore all possible actions, the agent randomly selects one of the actions with a small probability. The above action generation can be written mathematically as

$$a(t) = \begin{cases} \arg \max_{a \in \mathcal{A}} Q(s(t), a) , & \text{with probability } 1 - \epsilon , \\ \text{randomly select an action in } \mathcal{A} , & \text{with probability } \epsilon , \end{cases} \quad \forall t , \quad (2.21)$$

where  $\epsilon$  is the exploration rate. After the action is taken, the process transits to the next state and the agent updates its table based on this transition as

$$Q(s(t), a(t)) \leftarrow (1 - \alpha)Q(s(t), a(t)) + \alpha [r(t) + \gamma \arg \max_{a \in \mathcal{A}} Q(s(t+1), a)] , \quad (2.22)$$

## 2.5. Convex Optimisation

where  $\alpha$  is the learning rate and  $Q(s(t), a(t))$  is the entry in the table that represents the learned state-action value function at  $(s(t), a(t))$ . It is shown that the above learning algorithm converges to the optimal state-action value function  $Q^{\pi^*}$  (assuming that  $\alpha$  follows time sequences that decrease to zero over time) [17].

Such a tabular approach suffers from the curse of dimensionality. Specifically, if the state space or the action is large or continuous, then we cannot efficiently save all state-action values in the table in practice due to the memory and storage limitations of a real-world computing system. To address this issue, one can train a NN to approximate the state-action value function for a given policy  $\pi$  [45] as

$$Q(s, a|\theta^Q) \approx \mathbb{E}[r|s, a] + \gamma \mathbb{E}_{s' \sim \mathbf{P}}[V^\pi(s')|a = \pi(s)], \quad \forall s \in \mathcal{S}, \quad \forall a \in \mathcal{A}, \quad (2.23)$$

which is referred to as the critic. Also, particularly when the action space is large, solving  $\max_a Q(s(t), a)$  in each slot and finding the action as (2.21) is difficult. Thus, another NN can be trained to approximate the optimal action that maximises the value function in each state as

$$\pi(s|\theta^\pi) \approx \arg \max_a Q(s, a), \quad \forall s \in \mathcal{S}, \quad (2.24)$$

which is referred to as the actor. Then, the actor and critic can be optimised by using the deep deterministic policy gradient (DDPG) [46], which will be further explained in Chapter 3 when we apply it to our scheduler design problem.

## 2.5 Convex Optimisation

Convex optimisation [47] is a branch of optimisation theory that deals with the minimisation of a convex function (or the maximisation of a concave function) over a convex set of optimisation variables. A general form of the convex optimisation prob-

## 2.5. Convex Optimisation

lem can be written as

$$\begin{aligned} \min_{\mathbf{x}} \quad & f_0(\mathbf{x}) \\ \text{s.t.} \quad & f_i(\mathbf{x}) \leq 0, \quad i = 1, \dots, m, \\ & g_i(\mathbf{x}) = 0, \quad i = 1, \dots, p, \end{aligned} \tag{2.25}$$

where  $m$  and  $p$  are the numbers of inequality constraints and equality constraints, respectively. There are three requirements for the above problem (2.25) to be a convex optimisation problem:

- the objective function,  $f_0(\mathbf{x})$ , is a convex function.
- the inequality constraints,  $\{f_1(\mathbf{x}), \dots, f_p(\mathbf{x})\}$ , are convex functions.
- the equality constraints,  $\{g_1(\mathbf{x}), \dots, g_p(\mathbf{x})\}$ , are affine, i.e., they have the structure as  $\mathbf{a}^T \mathbf{x} - b = 0$  with a vector  $\mathbf{a}$  and a real number  $b$ .

Additionally, generalised inequality constraints can be added to the convex problem formulation to specify that the optimisation variables belong to a convex cone. For example, this thesis considers the generalised inequality defined on the cone of positive semidefinite matrices, e.g.,  $\succeq_{\mathcal{S}_+^K}$ , where  $\mathcal{S}_+^K$  is the cone of  $K \times K$  positive semidefinite matrices. We write the matrix  $\mathbf{X}$  belonging to this cone as

$$\mathbf{X} \succeq_{\mathcal{S}_+^K} 0, \quad \text{or in short, } \mathbf{X} \succeq 0. \tag{2.26}$$

The convex optimisation problem, including generalised inequality constraints, can be written as

$$\begin{aligned} \min_{\mathbf{X}} \quad & f_0(\mathbf{X}) \\ \text{s.t.} \quad & f_i(\mathbf{X}) \leq 0, \quad i = 1, \dots, m, \\ & g_i(\mathbf{X}) = 0, \quad i = 1, \dots, p, \\ & \mathbf{X} \succeq 0, \end{aligned} \tag{2.27}$$

where  $\mathbf{X}$  is a  $K \times K$  matrix and  $f_0(\mathbf{X})$ ,  $\{f_1(\mathbf{X}), \dots, f_p(\mathbf{X})\}$  and  $\{g_1(\mathbf{X}), \dots, g_p(\mathbf{X})\}$  are convex objective function, convex constraints and affine constraints on the elements of  $\mathbf{X}$ , respectively. It is well-known that a convex optimisation problem can be solved in

## 2.5. Convex Optimisation

the polynomial time of the problem's descriptive size, e.g., the dimension of  $\mathbf{X}$ .

### 2.5.1 Semidefinite Programming

Semidefinite programming (SDP) is a type of convex optimisation that involves optimising a linear function over the set of positive semidefinite matrices with only affine constraints as

$$\begin{aligned} \min_{\mathbf{X}} \quad & \mathbf{tr}(\mathbf{C}^T \mathbf{X}) \\ \text{s.t.} \quad & \mathbf{tr}(\mathbf{A}_i^T \mathbf{X}) = 0, \quad i = 1, \dots, m, \\ & \mathbf{X} \succeq 0, \end{aligned} \tag{2.28}$$

where  $\mathbf{tr}(\cdot)$  denotes the trace of a matrix, e.g.,  $\mathbf{tr}(\mathbf{Y}^T \mathbf{X}) = \sum_{i,j} Y_{i,j} X_{i,j}$  for two square matrices  $\mathbf{Y}$  and  $\mathbf{X}$ . One of the important applications of SDPs is to relax combinatorial optimisation problems that are typically at non-deterministic polynomial-time hardness (NP-hard), and then to approximate optimal solutions of the original problems in polynomial time.

#### The Goemans-Williamson Algorithm

For example, works in [25] use the SDP relaxation to solve a graph cut problem known as NP-hard. Consider a directed graph,  $\mathcal{G}$ , with a set of  $K$  vertices,  $\mathcal{V}$  and a set of edges  $\mathbf{E}$ , where

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}), \quad \mathcal{V} = \{1, \dots, K\}, \quad \mathcal{E} \subseteq \hat{\mathcal{E}} \triangleq \{(i, j) | \forall i, j \in \mathcal{V}, i \neq j\}, \tag{2.29}$$

where  $\hat{\mathcal{E}}$  is the set of all possible edges in a fully-connected directed graph with  $K$  vertices. Each edge is associated with a weight as  $W_{i,j}, \forall (i, j) \in \mathcal{E}$ , assuming  $W_{i,j} \geq 0$ . The graph cut problem is to divide the vertices into two subsets to maximise the summation of weights between these two subsets. Let  $y_k = 1$  if the  $k$ -th vertex is divided in the first subset and  $y_k = -1$  if divided in the second subset,  $k = 1, \dots, K$ .

## 2.5. Convex Optimisation

We can formulate

$$\begin{aligned} \max_{\mathbf{y}} \quad & \sum_{(i,j) \in \mathcal{E}} W_{i,j} \left( \frac{1 - y_i y_j}{2} \right) \\ \text{s.t.} \quad & y_k \in \{-1, 1\}, \forall k, \end{aligned} \tag{2.30}$$

where  $(1 - y_i y_j)/2$  is equal to 0 if vertices  $i$  and  $j$  are in the same subset, or 1 otherwise. The problem in (2.30) is referred to as the max-cut problem of  $\mathcal{G}$ . Define a  $K \times K$  matrix  $\mathbf{X}$  as

$$\mathbf{X} \triangleq \mathbf{y}\mathbf{y}^T, \tag{2.31}$$

One can verify that [48]

$$y_k \in \{-1, 1\}, \forall k \iff \mathbf{X} \succeq 0, \mathbf{diag}\{\mathbf{X}\} = \mathbf{1}, \text{Rank}(\mathbf{X}) = 1, \tag{2.32}$$

where  $\mathbf{X}$  is defined as (2.31) and  $\text{Rank}(\cdot)$  is the rank of a matrix. Then, the right-hand side of in (2.32) can be used to substitute the constraint and the optimisation variables in (2.30), which reformulates the max-cut problem as

$$\begin{aligned} \max_{\mathbf{X}} \quad & \sum_{(i,j) \in \mathcal{E}} W_{i,j} \left( \frac{1 - X_{i,j}}{2} \right) \\ \text{s.t.} \quad & \mathbf{X} \succeq 0, \mathbf{diag}\{\mathbf{X}\} = \mathbf{1}, \text{Rank}(\mathbf{X}) = 1. \end{aligned} \tag{2.33}$$

Next, by removing the constraint on the rank of  $\mathbf{X}$ , the above problem in (2.33) can be relaxed as

$$\begin{aligned} \max_{\mathbf{X}} \quad & \sum_{(i,j) \in \mathcal{E}} W_{i,j} \left( \frac{1 - X_{i,j}}{2} \right) \\ \text{s.t.} \quad & \mathbf{X} \succeq 0, \mathbf{diag}\{\mathbf{X}\} = \mathbf{1}. \end{aligned} \tag{2.34}$$

This is a SDP problem that can be converted into the standard form in (2.28). It is referred to as a max-cut SDP problem that can be solved by the convex optimisation solver [49].

Once the solver finds the optimal  $\mathbf{X}$ ,  $\mathbf{X}^*$ , that maximises the objective in (2.34), a rounding method is used to convert  $\mathbf{X}^*$  to the cutting decisions  $\mathbf{y}$ . Specifically, the singular value decomposition (SVD) is performed on  $\mathbf{X}^*$ , and because  $\mathbf{X}^*$  is a positive

## 2.5. Convex Optimisation

semi-definite matrix, a SVD can be found in the structure as [48]

$$\mathbf{U}\mathbf{\Sigma}(\mathbf{U})^T = \mathbf{svd}(\mathbf{X}^*) , \quad (2.35)$$

where  $\mathbf{U}$  and  $\mathbf{\Sigma}$  are both  $K \times K$  square matrices and  $\mathbf{\Sigma}$  is also a diagonal matrix with non-negative eigenvalues of  $\mathbf{X}^*$  on its diagonal. The cutting decisions are rounded as

$$\mathbf{y} = [y_1, \dots, y_2] = \mathbf{sgn}(\mathbf{U}(\mathbf{\Sigma})^{\frac{1}{2}}\delta) , \quad (2.36)$$

where  $\delta$  is a random vector in  $\mathbb{R}^K$ . It has been proved [25] that this method achieves at least 0.87854 of optimal achievable max cut objective in (2.30), which provides a near-optimal solution for max cut over given edge weights.



## Chapter 3

# Knowledge-Assisted Deep Reinforcement Learning for Flexible Scheduler Design

*A scheduler in wireless networks maps the network state to some scheduling decision (or action) according to a quality of service (QoS) indicator. It is simple to define the state and the action by checking what network information the scheduler has access to and what transmission configurations the scheduler can control, respectively. Also, the QoS indicator can be obtained by directly translating the service requirement. However, the optimal logic of the scheduler is somewhat hard to obtain, especially when the network system and the service requirement are complex. Thus, it is difficult to apply the classic analysis-based method. Furthermore, modern wireless networks can be flexibly configured for each individual service, where the optimal logic of schedulers for each case is different. Manually optimising the schedulers one by one for different cases seems impractical. Recently, machine learning (ML) methods have been widely applied in solving optimisation problems. The advantages of ML methods are twofold: 1) they do not require comprehensive system analysis; 2) they retain the same structure when addressing the same class of problems (though their detailed techniques differ).*

### 3.1. Introduction

*These features denote the possibility of enabling a flexible scheduler design using ML methods, which will be studied in this chapter.*

## 3.1 Introduction

5th generation (5G) cellular networks are expected to support emerging applications with time-sensitive traffic, such as autonomous vehicles, factory automation, tactile internet, and virtual/augmented reality [50]–[52]. Time-sensitive traffic has stringent QoS requirements, including reliability and latency requirements on ultra-reliable low latency communications (uRLLC) [7] as well as low jitter [9], which differ from the design goal of the previous generations of cellular networks, i.e., pursuing higher data rates. Existing schedulers like proportional fair [53], round-robin [54], earliest-deadline-first [55], and maximum throughput [56] were not developed for time-sensitive traffic. Thus, wireless schedulers should be re-designed to meet the QoS requirements of time-sensitive traffic in 5G.

A wireless scheduler in 5G is a multi-dimensional function that takes the queue state information (QSI) and the channel state information (CSI) as its input and outputs the amount of resources allocated to users. Such a problem can be formulated as an optimal control problem of a Markov decision process (MDP), which can be solved by reinforcement learning [J5]. Classic reinforcement learning algorithms and dynamic programming suffer from the curse of dimensionality and are only applicable to problems with small state-action spaces. One can apply deep Q-learning to overcome this difficulty, where the state-action value function is approximated by a neural network (NN) [45]. With deep Q-learning, the scheduler needs to find the optimal action that maximises the value function in each transmission time interval (TTI) in 5G New Radio (NR). Since the action space could be very large, the scheduler struggles to solve the optimisation problem in one TTI. More recently, actor-critic deep reinforcement learning (DRL) algorithms have been developed to handle this issue, where two NNs approximate the policy and the long-term rewards, respectively [17]. If the optimal

### 3.1. Introduction

policy is deterministic, which is the usual case in most optimal control problems [17], the actor-critic DRL algorithms become a deep deterministic policy gradient (DDPG) algorithm [46].

Since DDPG is a model-free algorithm that requires no transition probabilities of the MDP, it usually converges very slowly. However, communication environments in real-world networks are non-stationary. Once communication environments change, the pre-trained scheduling policy cannot achieve good QoS. Therefore, to apply DDPG in scheduler design, we must fine-tune the policy online and reduce its convergence time in real-world networks. On the other hand, the feed-forward inference of the scheduler, i.e., computing the output of the NN for a given input at the base station (BS), must be completed within each TTI ( $0.125 \sim 1$  ms). This poses a challenge for implementing learning-based schedulers in real-world 5G systems [53].

#### 3.1.1 Related Works

##### Schedulers for Time-sensitive Traffic

How to develop a scheduler for time-sensitive traffic in wired networks has been discussed in time-sensitive networking standardisation [57], [58]. For example, the authors of [58] developed an urgent-based scheduler and analysed the worst-case latency. Wireless schedulers were investigated to serve time-sensitive traffic with deterministic packet arrival processes in wireless communications in [59], [60]. Specifically, semi-persistent scheduling was adopted in 5G NR for periodic transmissions of control signals [59]. A system-level simulator for evaluating schedulers' end-to-end (E2E) performance when supporting deterministic traffic was used in [60]. In the above publications, either the channels (wired networks in [57], [58]) or the arrival processes (control signalling and data packets in [59], [60]) are assumed to be deterministic. However, both wireless channels and arrival processes are stochastic in various 5G applications, such as machine-type communications, vehicle safety applications, and ultra-reliable and low-latency communications [61]–[63]. For these applications, how to achieve low latency and low jitter with high reliability remains an open chal-

### 3.1. Introduction

lenge.

#### **DDPG in Wireless Networks**

DDPG has been widely applied to solve optimal control problems in wireless networks. The authors of [64] used DDPG to select a radio resource scheduling policy from existing schedulers and resource allocation policies. Considering that network slicing will be adopted in 5G to serve different kinds of services, DDPG was used to allocate resources among different slices in [65] and among different users in [66]. To further implement DDPG in wireless networks, an online architecture was implemented in virtual radio access networks for jointly controlling computing resources and the modulation and coding scheme, where a controller sends actions to virtual BSs every 20 seconds [67]. Another online DDPG architecture for network slicing was implemented in the 4G radio access networks in [68], where the controller allocates resources among different slices every 20 ~ 100 ms. None of the existing architectures can be used for 5G scheduler design, where the scheduler takes actions in every TTI at a time resolution of  $\sim 0.1$  ms. Thus, an online architecture that enables DDPG in 5G scheduler design is much needed.

#### **Knowledge-Assisted Learning in Communications**

How to improve the training efficiency with the assistance of expert knowledge in vertical industries remains an important issue [69]. This knowledge is referred to as the design principles and insights that human experts exploit to design learning algorithms rather than specific data or information. Based on the expert knowledge of physical layer communications, the authors in [70], [71] designed the structures of NNs to improve training efficiency. Since the experts in wireless communications have developed many optimisation algorithms and heuristic solutions, the existing policies can be used to generate training samples for DRL algorithms [72]. Nevertheless, further research is needed on how to establish a DRL framework for scheduler design, and how to improve the training efficiency and the QoS of each user by exploiting expert knowledge in wireless communications.

### 3.1.2 Our Methods

In the remainder of this chapter, we first investigate a straightforward implementation of DDPG in 5G scheduler design for time-sensitive traffic, where no communication model or expert knowledge of the wireless scheduler design problem is exploited. Straightforward implementation suffers from several issues related to problem formulation, training, and online implementation, which are listed with our solutions in Table 3.1. The contributions of this chapter have been outlined in Chapter 1.

## 3.2 Scheduler for Time-Sensitive Traffic in 5G

### 3.2.1 Wireless Scheduler in 5G NR

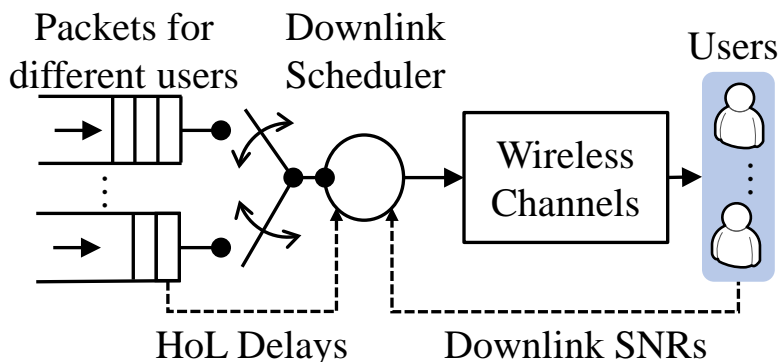


Figure 3.1: Illustration of a downlink scheduler.

We consider a downlink scheduler in 5G NR, where  $K$  users are served by one BS, as shown in Fig. 3.1. Packets of the  $k$ -th user are waiting in the  $k$ -th queue in the buffer of the BS, and each queue is served according to the first-in-first-out (FIFO) order. The duration of one slot is equal to the duration of one TTI in 5G NR, and is denoted by  $\Delta_0$ .

We leverage indicators  $x_k(t)$ ,  $k = 1, \dots, K$ , to represent whether users are scheduled in the  $t$ -th slot. If the  $k$ -th user is not scheduled in the  $t$ -th slot,  $x_k(t) = 0$ . Otherwise,  $x_k(t) = 1$  and one packet will be transmitted to the user. The packet size of the  $k$ -th user and the number of resource blocks (RBs) allocated to it in the  $t$ -th slot are denoted by  $L_k$  (bits) and  $n_k(t)$ , respectively. Since orthogonal frequency division

Table 3.1: Issues of Straightforward Implementation of DDPG and Solutions to Address Them

Issues	Our Solutions
Section 3.4.1 Issues in Problem Formulation	Section 3.5 Theoretical DRL Framework
1) Large action space	1) Action space reduction
2) Low flexibility of states	2) Generalisation of states
3) Poor reliability evaluation	3) Theoretical formulation for reliability evaluation
Section 3.4.2 Issues in Training Algorithm	Section 3.6 Knowledge-assisted DDPG
1) Unaware of individual QoS	1) Multi-head critic for individual QoS evaluation
2) Delayed reward/Sparse reward	2) Reward shaping for instantaneous non-zero feedback
3) Inaccurate critic at rarely visited state-action pairs	3) Importance sampling
Section 3.4.3 Issues in Online Implementation	Section 3.7 Online Architecture
1) Poor initial QoS	1) off-line initialisation & online fine-tuning
2) Long processing time in each TTI	2) Parallel processing in the BS

### 3.3. Straightforward Implementation of DDPG for Scheduler Design

multiplexing is adopted in 5G NR systems,  $n_k(t)$  can be adjusted in each slot by subcarrier allocation. As illustrated in Fig. 3.1, a scheduler determines  $x_k(t)$  and  $n_k(t)$  according to the QSI and CSI of all users, such as head-of-line (HoL) delays and downlink signal-to-noise ratios (SNRs) of users at the  $t$ -th slot. These are denoted by  $d_k(t)$  and  $\phi_k(t)$ , respectively, for  $k = 1, \dots, K$ .

#### 3.2.2 QoS Requirements of Time-Sensitive Traffic

The time-sensitive traffic has stringent QoS requirements, including delay, jitter, and reliability [9], [57], [73]. To satisfy the delay requirement, the delay experienced by packets should be larger than the minimum and smaller than the maximum delay bounds, which are denoted by  $D_{\min}$  and  $D_{\max}$ , respectively [9], [73]. Such a constraint also guarantees that the jitter does not exceed  $D_{\max} - D_{\min}$ . The reliability of a user is defined as packet loss probability. A packet is lost if  $d_k(t) \notin [D_{\min}, D_{\max}]$  or the decoding at the receiver fails. For typical time-sensitive traffic, as shown in 3GPP standards [9], the maximum delay bound is around 1 to 10 ms. Also, the jitter needs to be less than two TTIs, and the target reliability varies from 99.9% to 99.999%.

## 3.3 Straightforward Implementation of DDPG for Scheduler Design

In this section, we introduce a straightforward implementation of DDPG for scheduler design.

### 3.3.1 Preliminaries of DDPG

#### Training of DDPG

As shown in Fig. 3.2, DDPG is an actor-critic reinforcement learning algorithm [46], where the actor and the critic are two NNs that determine the action in a given state and evaluate the long-term reward of the state-action pair (taking an action in a

### 3.3. Straightforward Implementation of DDPG for Scheduler Design

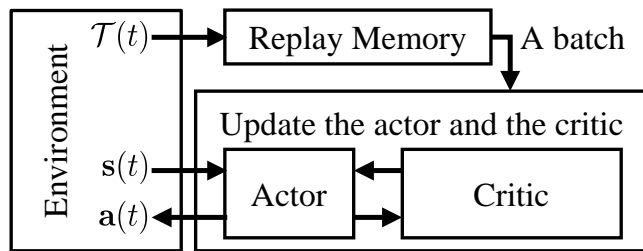


Figure 3.2: Illustration of DDPG.

state), respectively. Given a system state, the action to be executed is obtained from the actor as

$$\mathbf{a}(t) = \mu(\mathbf{s}(t)|\theta^\mu) , \quad (3.1)$$

where  $\mathbf{s}(t)$  and  $\mathbf{a}(t)$  are the state and action in the  $t$ -th slot,  $\mu(\cdot|\theta^\mu)$  represents the actor, and  $\theta^\mu$  represents the parameters of the actor, i.e., weights and biases.

Given  $\mathbf{s}(t)$  and  $\mathbf{a}(t)$ , the long-term reward is estimated by a state-action value function as

$$Q(\mathbf{s}(t), \mathbf{a}(t)) = \mathbb{E} \left[ \sum_{i=0}^{\infty} \gamma^i r(t+i) \mid \mathbf{s}(t), \mathbf{a}(t), \mathbf{a}(t') = \mu(\mathbf{s}(t')|\theta^\mu), \forall t' > t \right], \quad (3.2)$$

where  $r(t+i)$  is the instantaneous reward in the  $(t+i)$ -th slot and  $\gamma$  is the discount factor that measures the importance of future rewards. The state-action value function in (3.2) is approximated by the critic  $Q(\cdot|\theta^Q)$ , where  $\theta^Q$  represents its parameters.

DDPG initialises the parameters of the NNs,  $\theta^\mu$  and  $\theta^Q$ , as random values. In each time slot, the system observes the current state and generates an action according to  $\mathbf{a}(t) = \mu(\mathbf{s}(t)|\theta^\mu) + \mathcal{N}(t)$ , where  $\mathcal{N}(t)$  is an exploration noise. After taking a certain action at the  $t$ -th slot, the system observes the instantaneous reward and the state in the  $(t+1)$ -th slot. The transition between the two slots, i.e.,  $\mathcal{T}(t) \triangleq \langle \mathbf{s}(t), \mathbf{a}(t), r(t), \mathbf{s}(t+1) \rangle$ , is stored in a replay memory with the size of  $|\mathcal{I}|$ , where  $\mathcal{I}$  is the set of time indices of transitions saved in the memory. After that, a batch of transitions are selected from the memory and used as the training samples to optimise the parameters of the NNs. The  $i$ -th transition in the selected batch is



### 3.3. Straightforward Implementation of DDPG for Scheduler Design

$\mathcal{T}(t_i) = \langle \mathbf{s}(t_i), \mathbf{a}(t_i), r(t_i), \mathbf{s}(t_i + 1) \rangle$ ,  $t_i \in \mathcal{N}_{\text{tr}}$ , where  $\mathcal{N}_{\text{tr}} = \{t_1, \dots, t_{N_{\text{tr}}}\}$  is the set of indices of transitions in the batch and  $N_{\text{tr}}$  is the batch size.

To optimise the parameters of the critic, we use Bellman equation [17],

$$Q(\mathbf{s}(t_i), \mathbf{a}(t_i)) = \mathbb{E} \left[ r(t_i) + \gamma Q(\mathbf{s}(t_i + 1), \mu(\mathbf{s}(t_i + 1) | \theta^\mu)) \middle| \mathbf{s}(t_i), \mathbf{a}(t_i) \right]. \quad (3.3)$$

The realisation of the right-hand side of (3.3) in the  $t_i$ -th slot is defined as  $y(t_i) \triangleq r(t_i) + \gamma Q(\mathbf{s}(t_i + 1), \mu(\mathbf{s}(t_i + 1) | \theta^\mu) | \theta^Q)$ . To obtain an accurate approximation of the long-term reward, the DDPG algorithm minimises the difference between  $y(t_i)$  and  $Q(\mathbf{s}(t_i), \mathbf{a}(t_i) | \theta^Q)$ . Thus, the parameters of the critic are optimised by minimising the following loss function as

$$L(\theta^Q) = \frac{1}{N_{\text{tr}}} \sum_{i=1}^{N_{\text{tr}}} \left[ y(t_i) - Q(\mathbf{s}(t_i), \mathbf{a}(t_i) | \theta^Q) \right]^2. \quad (3.4)$$

Since the optimal policy maximises the state-action value function, the loss function of the actor is defined as

$$L(\theta^\mu) = -\frac{1}{N_{\text{tr}}} \sum_{i=1}^{N_{\text{tr}}} Q(\mathbf{s}(t_i), \mu(\mathbf{s}(t_i) | \theta^\mu) | \theta^Q), \quad (3.5)$$

which is minimised during training.

#### DDPG with discrete actions

The original DDPG requires the actions to be continuous variables. Thus, they can be directly obtained from the continuous output of the actor. If the action space,  $\mathcal{A}$ , is discrete, we need to map the continuous output of the actor to the discrete action. Using the method in [65], we can find the closest valid action as  $\arg \min_{\mathbf{a}(t) \in \mathcal{A}} \|\mathbf{a}(t) - \mu(\mathbf{s}(t) | \theta^\mu)\|_2$ , where  $\|\mathbf{x}\|_2$  is the  $\ell_2$  norm of a vector  $\mathbf{x}$ .

### 3.3.2 Problem Formulation

In the straightforward implementation, we define the state, action, and reward by using the control signalling or observations that are directly available from 5G NR systems.

#### Action of the scheduler

The scheduler determines the number of RBs that will be allocated to each user. Thus, the action of the scheduler in the  $t$ -th slot is given by

$$\mathbf{a}(t) \triangleq [n_1(t), \dots, n_K(t)]^T, \quad (3.6)$$

where  $n_k(t)$  is the nearest integer to the  $k$ th element of  $\mu(\mathbf{s}(t)|\theta^\mu)$ . If  $n_k(t) = 0$ , the  $k$ -th user will not be scheduled in the  $t$ -th slot.

#### State of the network

Since HoL delays and downlink SNRs are available at 5G BSs, we define the normalised state in the  $t$ -th slot as

$$\mathbf{s}(t) \triangleq \left[ \frac{d_1(t)}{D_{\max}}, \dots, \frac{d_K(t)}{D_{\max}}, \frac{\log \phi_1(t)}{\log \phi_{\max}}, \dots, \frac{\log \phi_K(t)}{\log \phi_{\max}} \right]^T, \quad (3.7)$$

where  $\phi_{\max}$  is the maximum SNR represented by the maximum channel quality indicator [74].

#### Reward of the scheduler

According to the QoS requirements in Section 3.2.2, the total instantaneous reward in the  $t$ -th slot of the system is defined as the total number of packets successfully received by users in this slot, which is mathematically written as

$$r(t) \triangleq \sum_{k=1}^K r_k(t), \quad (3.8)$$

### 3.4. Issues in the Straightforward Implementation of DDPG

where  $r_k(t)$  is the number of packets received by the  $k$ -th user in the  $t$ -th slot, i.e.,

$$r_k(t) = \begin{cases} \mathbf{1}_{\{D_{\min} \leq d_k(t) \leq D_{\max}\}} \cdot \mathbf{1}_k^{\text{dec}}(t), & \text{if } x_k(t) = 1, \\ 0, & \text{if } x_k(t) = 0, \end{cases} \quad (3.9)$$

where  $\mathbf{1}_{\{D_{\min} \leq d_k(t) \leq D_{\max}\}}$  and  $\mathbf{1}_k^{\text{dec}}(t)$  are two indicators. Specifically, if the packet is scheduled when  $d_k(t) \in [D_{\min}, D_{\max}]$ , then  $\mathbf{1}_{\{D_{\min} \leq d_k(t) \leq D_{\max}\}}$  equals 1; otherwise, it equals 0. In the case that the packet is successfully decoded,  $\mathbf{1}_k^{\text{dec}}(t) = 1$ , otherwise,  $\mathbf{1}_k^{\text{dec}}(t) = 0$ .

Finally, the optimal control problem that maximises the long-term reward of a scheduler can be formulated as

$$\max_{\mu(\cdot|\theta^\mu)} \mathbb{E} \left[ \sum_{i=0}^{\infty} \gamma^i r(t+i) \right]. \quad (3.10)$$

In this work, we only consider time-sensitive traffic. To extend the work to 5G networks with other network services, we can formulate different scheduler design problems for other network services by changing the reward function. For example, we can use the long-term average rate as the reward function to maximise the fairness and the throughput of the system. However, in this case, PF is the optimal scheduler. Note that there is no need to use DRL or DDPG in scenarios where the optimal schedulers are available. For time-sensitive traffic, the optimal scheduler is not available. Therefore, we use learning-based methods to design the scheduler.

## 3.4 Issues in the Straightforward Implementation of DDPG

It seems that DDPG can be directly applied in scheduler design. However, as shown in this section, there are some issues to be addressed in the straightforward implementation.

### 3.4.1 Issues in Problem Formulation

#### Large action space

Denote the total number of RBs assigned to time-sensitive traffic by  $N$ . Thus, the possible number of RBs allocated to a user varies from 0 to  $N$ . With  $K$  users, the size of the action space in the straightforward implementation,  $|\mathcal{A}|$ , is  $(N + 1)^K$  based on (3.6). In a 5G NR system, the value of  $N$  can be higher than 100 depending on the total bandwidth and the bandwidth of each RB. Thus, the action space could be extremely large. A reinforcement learning algorithm converges as the number of visits to each state-action pair approaches infinite [17]. When the action space is large, finding the optimal scheduler is nearly impossible in practice.

#### Low flexibility of states

Furthermore, the actor in the straightforward approach is a mapping from HoL delays and SNRs in (3.7) to the numbers of RBs in (3.6). The total number of RBs assigned to time-sensitive traffic,  $N$ , the bandwidth of each RB,  $W$ , and the duration of each TTI,  $\Delta_0$ , are hidden variables that are not included in the input of DDPG, and hence are assumed to be constant. According to the standard of 3GPP, these hidden variables are flexible in 5G NR [74]. When  $W$ ,  $\Delta_0$  and  $N$  become different, we need to train a new scheduler by using DDPG, which is inflexible and inefficient.

#### Poor reliability evaluation

In addition, for time-sensitive traffic, the required reliability can be up to 99.999%. The long-term reward in (3.10) is linear with the reliability. Thus, by increasing the reliability of all users from 99.99% to 99.999%, the long-term reward only increases 0.01%. As a result, the gradient of (3.5) will be very small. Since the gradient descent method is used to update the parameters of the actor, the convergence time of the actor will be very long.

### 3.4.2 Issues in Training Algorithm

#### Unaware of individual QoS

The output of the critic is a single scalar value estimating the long-term reward of all users. Thus, it is not aware of the QoS of each user, and some users may suffer from poor QoS.

#### Delayed reward/Sparse reward

Due to the requirement on jitter, the scheduler receives positive rewards if packets are scheduled with HoL delays in  $[D_{\min}, D_{\max}]$ . When  $d_k(t) < D_{\min}$ , the instantaneous reward is 0 no matter whether the packet is scheduled or not. In other words, the scheduler needs to take a series of actions to get a delayed non-zero reward. For example, only by taking the following actions,  $n_k(t) = 0$  when  $d_k(t) < D_{\min}$ , and  $n_k(t) > 0$  when  $d_k(t) \in [D_{\min}, D_{\max}]$ , the scheduler can receive a positive reward. Due to the fact that the non-zero rewards are sparse, such an issue is also referred to as sparse reward in [17]. Since the scheduler is not told which actions to take in order to get the non-zero reward, it is difficult for DDPG to learn the correct actions.

#### Inaccurate critic at rarely visited state-action pairs

To train the parameters of the critic, DDPG selects a batch of transitions from the replay memory with equal probabilities. The state-action pairs that are visited with high frequency are more likely to be selected than those that are rarely visited. When the packet loss probability is small, the state-action pairs with packet losses are rarely visited and the critic is inaccurate at these state-action pairs. As a result, it is difficult for DDPG to achieve high reliability.

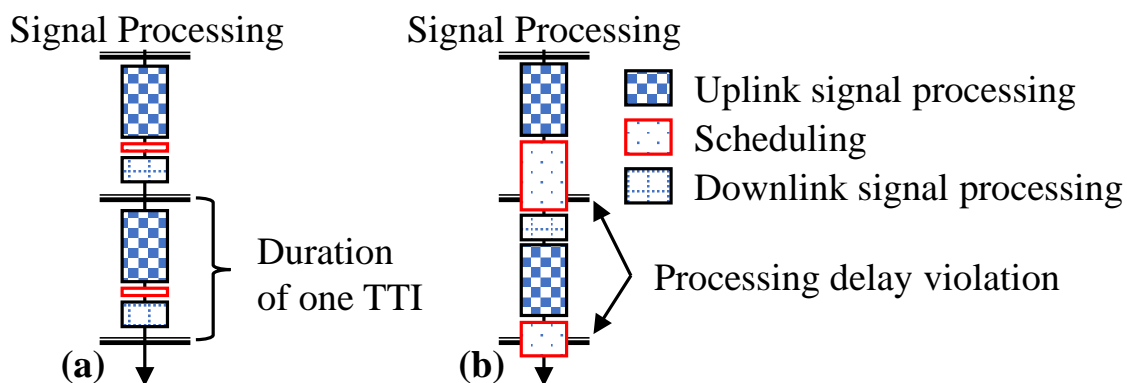


Figure 3.3: Processing in the BS: (a) low-complexity scheduler, (b) high-complexity scheduler.

### 3.4.3 Issues in Online Implementation

#### Poor initial QoS performance

Since the parameters of the actor and critic are randomly initialised in DDPG, the QoS performance is poor during the first a few slots, which leads to high packet loss probability. Since the algorithm interacts with real-world networks, random initialisation will cause severe QoS violations.

#### Processing delay violation in each TTI

The BS needs to perform scheduling and the baseband signal processing (i.e., decoding uplink packets and encoding downlink packets) within each TTI. As shown in Fig. 3.3, if the processing of the feed-forward inference and the baseband signal processing is not finished in one TTI, radio signals cannot be transmitted in the assigned time slots, leading to radio link failures. We refer to this issue as processing delay violation.

## 3.5 Theoretical DRL Framework

To address the issues in problem formulation, we propose a T-DRL framework to simplify the optimal control problem, where we exploit theoretical models and results to 1) reduce the action space, 2) generalise the state, and 3) evaluate the reliability.

### 3.5.1 Theoretical Models and Results

The number of packets that arrive at the queue of the  $k$ -th user in the  $t$ -th slot is denoted by  $b_k(t)$ . For typical time-sensitive traffic, such as mission-critical IoTs and vehicle networks, the packet arrival processes follow Bernoulli processes, i.e.,  $b_k(t) \in \{0, 1\}$  [61]–[63] and the packet size is small, e.g., 20 or 32 bytes [63]. We assume that with probability  $p_k$ , one packet arrives in the slot,  $b_k(t) = 1$ . With probability  $1 - p_k$ , no packet arrives in the slot,  $b_k(t) = 0$ .

When transmitting a small packet, the required bandwidth is assumed to be smaller than the coherence bandwidth. We assume that the duration of each TTI is smaller than the channel coherence time. Thus, the wireless channel is flat fading and quasi-static, and the blocklength of channel codes is short. To transmit  $L_k$  bits of data to the  $k$ -th user, the decoding error probability in the short blocklength regime can be accurately approximated by [38] as

$$\epsilon_k(t) \approx f_Q \left( \frac{-L_k \ln 2 + \Delta_0 W n_k(t) \ln [1 + \phi_k(t)]}{\sqrt{\Delta_0 W n_k(t) V_k(t)}} \right), \quad (3.11)$$

where  $f_Q$  is Q-function,  $W$  is the bandwidth of each RB,  $V_k(t)$  in (3.11) is the channel dispersion defined as  $V_k(t) = 1 - 1/[1 + \phi_k(t)]^2$  [38].

To avoid long transmission delays and large jitters, retransmission cannot be used to improve reliability. In this case, if a packet is not successfully decoded by the user, it is lost. To achieve high reliability, the target decoding error probability of the  $k$ -th user should not exceed a threshold, i.e.,

$$\epsilon_k(t) \leq \epsilon_{\max}. \quad (3.12)$$

Since the decoding error probability in (3.11) decreases with the number of RBs,  $n_k(t)$ , the minimum number of RBs required to satisfy the constraint in (3.12), denoted by  $n_k^*(t)$ , can be obtained via binary search [75].

### 3.5.2 Action Space Reduction

If the  $k$ -th user is scheduled, the number of RBs required to guarantee the reliability in (3.12),  $n_k^*(t)$ , can be obtained by substituting (3.11) into (3.12). Thus, the scheduler only needs to determine which users to be scheduled. We define the action of the scheduler as

$$\hat{\mathbf{a}}(t) = [x_1(t), \dots, x_K(t)]^T, \quad (3.13)$$

where  $x_k(t), k = 1, \dots, K$ , are binary variables that are obtained from the output of the actor according to Section 3.3.1. Thus, the number of possible actions is  $2^K$ , which is much smaller than that of the straightforward implementation, i.e.,  $(N+1)^K$ , according to the definition of actions in (3.6). Given  $\hat{\mathbf{a}}(t)$  in (3.13), the number of RBs allocated to the  $k$ -th user can be obtained from the following expression as

$$n_k(t) = \begin{cases} x_k(t)n_k^*(t), & \text{if } \sum_{k=1}^K x_k(t)n_k^*(t) \leq N, \\ \lfloor \frac{x_k(t)n_k^*(t)}{\sum_{k=1}^K x_k(t)n_k^*(t)} N \rfloor, & \text{if } \sum_{k=1}^K x_k(t)n_k^*(t) > N. \end{cases} \quad (3.14)$$

### 3.5.3 Generalisation of State

To enable DRL in 5G NR with flexible configurations, we replace  $\frac{\log \phi_k(t)}{\log \phi_{\max}(t)}$  in (3.7) with  $\frac{n_k^*(t)}{N}$ , which is obtained from the theoretical formulas in (3.11) and (3.12) and depends on  $\phi_k, N, W$  and  $\Delta_0$ . Based on the theoretical models and results, the generalised state of the system in the  $t$ -th slot is given by

$$\hat{\mathbf{s}}(t) \triangleq \left[ \frac{d_1(t)}{D_{\max}}, \dots, \frac{d_K(t)}{D_{\max}}, \frac{n_1^*(t)}{N}, \dots, \frac{n_K^*(t)}{N} \right]^T. \quad (3.15)$$

### 3.5.4 Reliability Evaluation

From the definition of the decoding error probability in (3.11), we have  $\mathbb{E}[\mathbf{1}_k^{\text{dec}}(t)] = 1 - \epsilon_k(t)$ . By replacing  $\mathbf{1}_k^{\text{dec}}(t)$  in (3.9) with  $1 - \epsilon_k(t)$ , the reward of the  $k$ -th user can



### 3.6. Knowledge-assisted DDPG

be expressed as

$$\tilde{r}_k(t) = \begin{cases} \mathbf{1}_{\{D_{\min} \leq d_k(t) \leq D_{\max}\}}(1 - \epsilon_k(t)), & \text{if } x_k(t) = 1, \\ 0, & \text{if } x_k(t) = 0. \end{cases} \quad (3.16)$$

As mentioned in Section 3.4.1, when  $\tilde{r}_k(t)$  is close to 1, the training efficiency of DDPG is low. To handle this issue, we define the reward of the  $k$ -th user in the  $t$ -th slot as

$$\hat{r}_k(t) = -\log [1 - \tilde{r}_k(t)] . \quad (3.17)$$

It is worth noting that the reward function in (3.17) is not well defined in the straightforward implementation. This is because the reward in (3.9),  $r_k(t)$ , is a binary number that may be equal to 1. With the definition in (3.17), the expectation of  $\hat{r}_k(t)$  is more sensitive to the scheduling policy than the expectation of  $\tilde{r}_k(t)$ . For example, by increasing the reliability from 99% to 99.99%,  $\mathbb{E}[\tilde{r}_k(t)]$  increases by 1%, but  $\mathbb{E}[\hat{r}_k(t)]$  is doubled.

The total reward is defined as the summation of the rewards of all users, i.e.,

$$\hat{r}(t) = \sum_{k=1}^K \hat{r}_k(t) . \quad (3.18)$$

#### 3.5.5 Markov Property

To apply DRL, we need to prove that transitions of the system follow an MDP. Otherwise, DRL algorithms may not converge [17]. By assuming that the wireless channel fading is Markovian [76], the Markov property holds for the scheduler design problem (see proof in Appendix).

## 3.6 Knowledge-assisted DDPG

To address issues in the training phase of the straightforward implementation, we propose K-DDPG by exploiting expert knowledge, which is formally defined as the

### 3.6. Knowledge-assisted DDPG

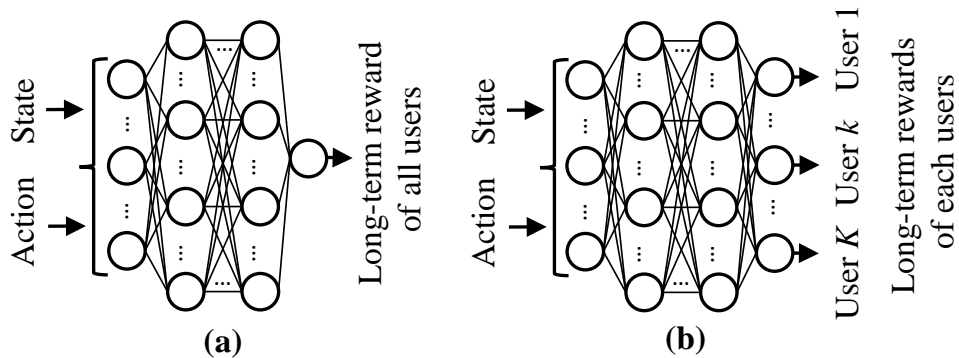


Figure 3.4: Illustration of (a) single-head critic and (b) multi-head critic.

design principles and insights from human experts. Specifically, for scheduler design, the expert knowledge includes 1) the rewards of multiple users, 2) the target scheduling policy, and 3) the importance of transitions. With the help of knowledge, K-DDPG improves the QoS of each user and reduces the convergence time.

#### 3.6.1 Multi-head Critic for Individual QoS Evaluation

The single-head critic in Fig. 3.4 is not aware of the reward of each component in the system, e.g., the QoS of each user,  $\hat{r}_k(t)$ ,  $k = 1, \dots, K$ . Since there are multiple users, the total long-term reward is the summation of the long-term reward of each user, according to (3.18). Based on the knowledge of the reward structure, we decompose the reward into  $K$  components for  $K$  users. We denote the rewards of all users at the  $t$ -th slot as

$$\hat{\mathbf{r}}(t) \triangleq [\hat{r}_1(t), \dots, \hat{r}_K(t)]. \quad (3.19)$$

The decomposed long-term rewards are approximated by the state-action value function denoted by  $Q_1(\mathbf{s}, \mathbf{a}), \dots, Q_K(\mathbf{s}, \mathbf{a})$ , where  $Q_k(\mathbf{s}, \mathbf{a})$  is the state-action value function of the  $k$ -th user, defined as follows,

$$Q_k(\hat{\mathbf{s}}(t), \hat{\mathbf{a}}(t)) = \mathbb{E} \left[ \sum_{i=0}^{\infty} \gamma^i \hat{r}_k(t+i) \mid \hat{\mathbf{s}}(t), \hat{\mathbf{a}}(t), \hat{\mathbf{a}}(t') = \mu(\hat{\mathbf{s}}(t')) | \theta^\mu, \forall t' > t \right]. \quad (3.20)$$

### 3.6.2 Reward Shaping for Instantaneous Feedback

Exploring the optimal policy with delayed rewards/sparse rewards is time-consuming. To handle this issue, we apply *reward shaping* [69] to generate non-zero instantaneous feedback in each slot. According to the requirement on jitter, a target scheduling policy should only schedule users with  $d_k(t) \in [D_{\min}, D_{\max}]$ . For users with  $d_k(t) < D_{\min}$ , they should not be scheduled. Based on the knowledge of the target scheduling policy, we define a potential function,  $\Psi(d_k(t))$ , which generates non-zero instantaneous reward according to

$$\hat{r}_k(t) = \hat{r}_k(t) - \Psi(d_k(t)) + \gamma\Psi(d_k(t+1)) , \quad (3.21)$$

which is the shaped reward of the  $k$ -th user. To illustrate the relation between the instantaneous feedback and the potential function, we considered an example in Fig. 3.5, where the potential function increases linearly with  $d_k(t)$  when  $d_k(t) < D_{\min}$ . When  $d_k(t) < D_{\min}$ ,  $\hat{r}_k(t) = 0$  no matter whether the user is scheduled or not. If the user is not scheduled,  $d_k(t+1) = d_k(t) + 1$ , then  $-\Psi(d_k(t)) + \gamma\Psi(d_k(t+1)) > 0$ , since  $\gamma$  is closed to 1. In other words, the scheduler will receive a positive instantaneous reward. On the other hand, if the user is scheduled, then  $d_k(t+1) < d_k(t)$  and  $-\Psi(d_k(t)) + \gamma\Psi(d_k(t+1)) < 0$ . In this case, the scheduler will receive a negative instantaneous reward.

When reward shaping is used in reinforcement learning, the state-action value function, denoted by  $\dot{Q}_k(\hat{\mathbf{s}}(t), \hat{\mathbf{a}}(t))$ , will be different from the original  $Q_k(\hat{\mathbf{s}}(t), \hat{\mathbf{a}}(t))$ . The Bellman equation can be re-expressed as

$$\dot{Q}_k(\hat{\mathbf{s}}(t), \hat{\mathbf{a}}(t)) = \mathbb{E} \left[ \hat{r}_k(t) + \gamma \dot{Q}_k(\hat{\mathbf{s}}(t+1), \mu(\hat{\mathbf{s}}(t+1)|\theta^\mu)) \middle| \hat{\mathbf{s}}(t), \hat{\mathbf{a}}(t) \right] , \quad (3.22)$$

where  $k = 1 \dots, K$ . By substituting (3.21) into (3.22), we can derive that (see the details in [69])

$$\dot{Q}_k(\hat{\mathbf{s}}(t), \hat{\mathbf{a}}(t)) = Q_k(\hat{\mathbf{s}}(t), \hat{\mathbf{a}}(t)) - \Psi(d_k(t)) . \quad (3.23)$$

### 3.6. Knowledge-assisted DDPG

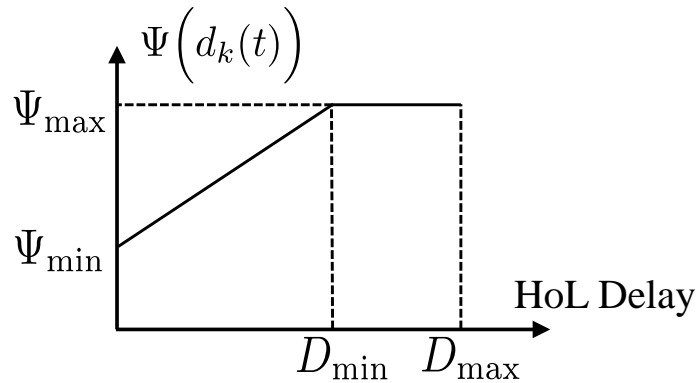


Figure 3.5: Illustration of the potential function for reward shaping.

Since  $\Psi(d_k(t))$  does not depend on the action to be taken, the actor that maximises  $\dot{Q}_k(\hat{\mathbf{s}}(t), \hat{\mathbf{a}}(t))$  is the same as the actor that maximises  $Q_k(\hat{\mathbf{s}}(t), \hat{\mathbf{a}}(t))$ , i.e.,

$$\begin{aligned} \arg \max_{\mu(\cdot|\theta^\mu)} \dot{Q}_k(\hat{\mathbf{s}}(t), \mu(\hat{\mathbf{s}}(t)|\theta^\mu)) &= \arg \max_{\mu(\cdot|\theta^\mu)} Q_k(\hat{\mathbf{s}}(t), \mu(\hat{\mathbf{s}}(t)|\theta^\mu)) - \Psi(d_k(t)) \\ &= \arg \max_{\mu(\cdot|\theta^\mu)} Q_k(\hat{\mathbf{s}}(t), \mu(\hat{\mathbf{s}}(t)|\theta^\mu)) . \end{aligned} \quad (3.24)$$

Therefore, the optimal actor does not change with the potential function [69].

After reward shaping, the multi-dimensional state-action value function of the  $K$  users is defined as  $\mathbf{Q}(\hat{\mathbf{s}}(t), \hat{\mathbf{a}}(t)) \triangleq [\dot{Q}_1(\hat{\mathbf{s}}(t), \hat{\mathbf{a}}(t)), \dots, \dot{Q}_K(\hat{\mathbf{s}}(t), \hat{\mathbf{a}}(t))]^T$  [77]. As illustrated in Fig. 3.4b,  $\mathbf{Q}(\hat{\mathbf{s}}(t), \hat{\mathbf{a}}(t))$  is approximated by a multi-head critic,

$$\mathbf{Q}(\hat{\mathbf{s}}(t), \hat{\mathbf{a}}(t)|\Theta^q) \triangleq [\dot{Q}_1(\hat{\mathbf{s}}(t), \hat{\mathbf{a}}(t)|\Theta^q), \dots, \dot{Q}_K(\hat{\mathbf{s}}(t), \hat{\mathbf{a}}(t)|\Theta^q)]^T , \quad (3.25)$$

which is a NN with parameters  $\Theta^q$ .

#### 3.6.3 Importance Sampling

To train the critic, the original DDPG selects a batch of training samples from  $|\mathcal{I}|$  transitions in the replay memory. All transitions will be selected with the same probability,  $1/|\mathcal{I}|$ . However, transitions are not with the same importance. In scheduler design, transitions with higher approximation errors of the state-action value function or with more packet losses are more important than the other transitions. Specifically,

### 3.6. Knowledge-assisted DDPG

we define a weight,  $w(t)$ , of transition  $\hat{\mathcal{T}}(t), t \in \mathcal{I}$ . The probability that transition  $\hat{\mathcal{T}}(t)$  will be selected is given by [78]

$$p_{\text{tr}}(t) = \frac{w(t)}{\sum_{i \in \mathcal{I}} w(i)}. \quad (3.26)$$

We set the initial weight of the transition generated in the  $t$ -th time slot,  $\hat{\mathcal{T}}(t)$ , as the maximum weight of all transitions that have been stored in the replay memory during the previous  $(t - 1)$  slots,  $w(t) = \max_{i \in \mathcal{I}} w(i)$ . For the transition generated in the first slot, the weight is set as a small positive number in order to avoid zero weight.

Based on the selected batch, we first update the weights of transitions in the batch based on the approximation error of the critic and the number of packet losses, i.e.,

$$w(t_i) \leftarrow \sum_{k=1}^K \left\{ \left[ \dot{y}_k(t_i) - \dot{Q}_k(\hat{\mathbf{s}}(t_i), \hat{\mathbf{a}}(t_i) | \Theta^q) \right]^2 \cdot \left[ 1 + (1 - x_k) \mathbf{1}_{\{d_k(t) = D_{\max}\}} + x_k \mathbf{1}_{\{d_k(t) \notin [D_{\min}, D_{\max}]\}} \right] \right\}, \quad (3.27)$$

where  $\dot{y}_k(t_i) = \dot{r}_k(t_i) + \gamma \dot{Q}_k(\hat{\mathbf{s}}(t_i + 1), \mu(\hat{\mathbf{s}}(t_i + 1) | \theta^\mu) | \Theta^q)$  is the realisation of the right-hand side of (3.22) in the  $t_i$ -th slot. The first part in (3.27) is the approximation error of the critic. The second part in (3.27) depends on the number of packet losses. Specifically, if the  $k$ -th user is not scheduled when  $d_k(t) = D_{\max}$  or is scheduled when  $d_k(t) \notin [D_{\min}, D_{\max}]$ , there is a packet loss. If there is a packet loss, the second part is equal to 2. Otherwise, it is equal to 1.

To optimise the multi-head critic, we minimise the following loss function as <sup>1</sup>

$$L(\Theta^q) = \frac{1}{N_{\text{tr}}} \sum_{i=1}^{N_{\text{tr}}} u(t_i) \sum_{k=1}^K \left[ \dot{y}_k(t_i) - \dot{Q}_k(\hat{\mathbf{s}}(t_i), \hat{\mathbf{a}}(t_i) | \Theta^q) \right]^2, \quad (3.28)$$

<sup>1</sup>Since transitions are not selected with the same probability from the replay memory,  $\frac{1}{N_{\text{tr}}} \sum_{i=1}^{N_{\text{tr}}} \sum_{k=1}^K \left[ \dot{y}_k(t_i) - \dot{Q}_k(\hat{\mathbf{s}}(t_i), \hat{\mathbf{a}}(t_i) | \Theta^q) \right]^2$  is no longer the average of the approximation error of the state-action value function.

### 3.7. Online DDPG Architecture

where the co-efficient  $u(t_i)$  is defined as

$$u(t_i) = \frac{1}{p_{\text{tr}}(t_i)|\mathcal{I}|}, \quad (3.29)$$

which corrects the bias caused by importance sampling.

To find the optimal actor maximising the total state-action value,  $\sum_{k=1}^K \dot{Q}_k(\hat{\mathbf{s}}(t), \hat{\mathbf{a}}(t))$  [77], we optimise the parameters of the actor to minimise the following loss function as

$$L(\theta^\mu) = \mathbb{E} \left[ - \sum_{k=1}^K \dot{Q}_k(\hat{\mathbf{s}}(t), \hat{\mathbf{a}}(t)) \right] = - \frac{1}{N_{\text{tr}}} \sum_{i=1}^{N_{\text{tr}}} u(t_i) \sum_{k=1}^K \dot{Q}_k(\hat{\mathbf{s}}(t_i), \mu(\hat{\mathbf{s}}(t_i)|\theta^\mu)|\Theta^q). \quad (3.30)$$

The temporal copies of the  $\Theta^q$  and  $\theta^\mu$  are denoted by  $\hat{\Theta}^q$  and  $\hat{\theta}^\mu$ , respectively. We first optimise  $\hat{\Theta}^q$  and  $\hat{\theta}^\mu$  and then update  $\Theta^q$  and  $\theta^\mu$  “softly”. From (3.28) and (3.30), we can derive the gradients of  $L(\hat{\Theta}^q)$  and  $L(\hat{\theta}^\mu)$ , respectively, i.e.,

$$\begin{aligned} & \nabla_{\hat{\Theta}^q} L(\hat{\Theta}^q) \\ &= \frac{1}{N_{\text{tr}}} \sum_{i=1}^{N_{\text{tr}}} u(t_i) \sum_{k=1}^K \left\{ 2 \left[ \dot{y}_k(t) - \dot{Q}_k(\hat{\mathbf{s}}(t_i), \hat{\mathbf{a}}(t_i)|\hat{\Theta}^q) \right] \times \nabla_{\hat{\Theta}^q} \dot{Q}_k(\hat{\mathbf{s}}(t_i), \hat{\mathbf{a}}(t_i)|\hat{\Theta}^q) \right\}, \end{aligned} \quad (3.31)$$

$$\nabla_{\hat{\theta}^\mu} L(\hat{\theta}^\mu) = - \frac{1}{N_{\text{tr}}} \sum_{i=1}^{N_{\text{tr}}} u(t_i) \sum_{k=1}^K \left[ \nabla_{\mathbf{a}} \dot{Q}_k(\hat{\mathbf{s}}(t_i), \mathbf{a}|\hat{\Theta}^q)|_{\mathbf{a}=\mu(\hat{\mathbf{s}}(t_i)|\hat{\theta}^\mu)} \times \nabla_{\hat{\theta}^\mu} \mu(\hat{\mathbf{s}}(t_i)|\hat{\theta}^\mu) \right]. \quad (3.32)$$

The pseudo-code of K-DDPG is provided in Algorithm 1.

## 3.7 Online DDPG Architecture

In this section, we address the issues in the real-world implementation of DDPG by proposing an architecture for online training and inference. As shown in Fig. 3.6, the

---

**Algorithm 1** Knowledge-assisted DDPG

---

- 1: Initialise the parameters of the NNs,  $\Theta^q$  and  $\theta^\mu$ .
  - 2: Initialise temporal copies of the parameters:  $\hat{\Theta}^q \leftarrow \Theta^q$  and  $\hat{\theta}^\mu \leftarrow \theta^\mu$ .
  - 3: Initialise a replay memory with a size of  $I$ .
  - 4: **for** episode  $m = 1, \dots, M$  **do**
  - 5:   Set the system to an initial state, e.g., set queues of users as empty.
  - 6:   **for**  $t = (m - 1)T + 1, \dots, mT$  **do**
  - 7:     Observe state  $\hat{\mathbf{s}}(t)$ .
  - 8:     Generate action from  $\hat{\mathbf{a}}(t) = \mu(\hat{\mathbf{s}}(t)|\theta^\mu) + \mathcal{N}(t)$ , and execute the action.
  - 9:     Evaluate reward  $\hat{\mathbf{r}}(t)$  from (3.11), (3.16) and (3.17), and observe the next state  $\hat{\mathbf{s}}(t + 1)$ .
  - 10:     Save transition  $\hat{\mathcal{T}}(t) = \langle \hat{\mathbf{s}}(t), \hat{\mathbf{a}}(t), \hat{\mathbf{r}}(t), \hat{\mathbf{s}}(t + 1) \rangle$  and its weight  $w(t) = \max_{i \in \mathcal{I}} w(i)$ .
  - 11:     Select  $N_{\text{tr}}$  transitions as a batch of training samples based on (3.26).
  - 12:     Update weights of selected transitions based on (3.27).
  - 13:     Compute  $\nabla_{\hat{\Theta}^q} L(\hat{\Theta}^q)$  and  $\nabla_{\hat{\theta}^\mu} L(\hat{\theta}^\mu)$  from (3.31) and (3.32), respectively.
  - 14:     Optimize  $\hat{\Theta}^q$  and  $\hat{\theta}^\mu$  with the SGD algorithm.
  - 15:     Update  $\Theta^q$  and  $\theta^\mu$  based on  $\hat{\Theta}^q$  and  $\hat{\theta}^\mu$ :  

$$\Theta^q \leftarrow (1 - \tau)\Theta^q + \tau\hat{\Theta}^q ; \theta^\mu \leftarrow (1 - \tau)\theta^\mu + \tau\hat{\theta}^\mu .$$
  - 16:   **end for**
  - 17: **end for**
  - 18: Return  $\Theta^q$  and  $\theta^\mu$  for online fine-tuning.
- 

online DDPG architecture includes the scheduler at the BS and an edge server.

### 3.7.1 Off-line Initialisation

Before executing DDPG in the online architecture, we need to initialise the actor and the critic off-line in a simulation platform, which is built upon the configurations of the real-world network and the theoretical models. The basic idea is to generate transitions from the simulation platform and train the actor and the critic by using Algorithm 1. Considering that the simulation is not exactly the same as the real-world network, the actor and the critic are fine-tuned in the online architecture, which is introduced below.

### 3.7. Online DDPG Architecture

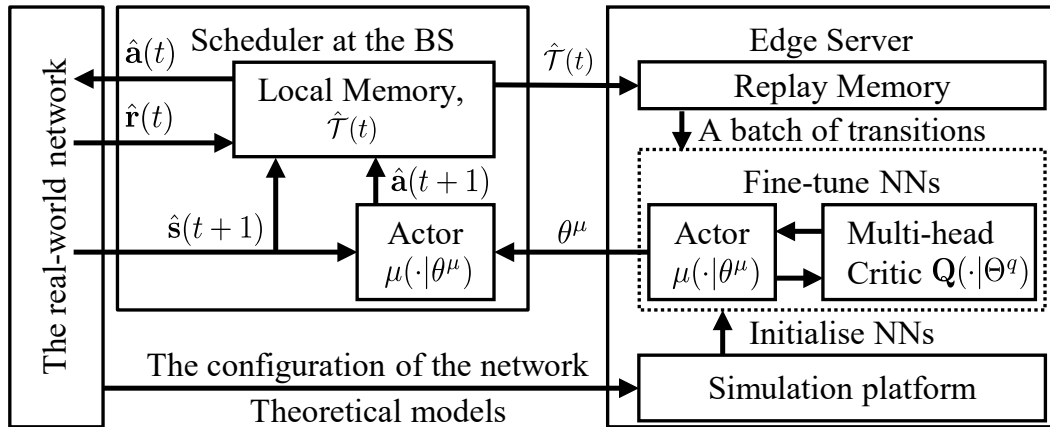


Figure 3.6: Proposed online DDPG architecture.

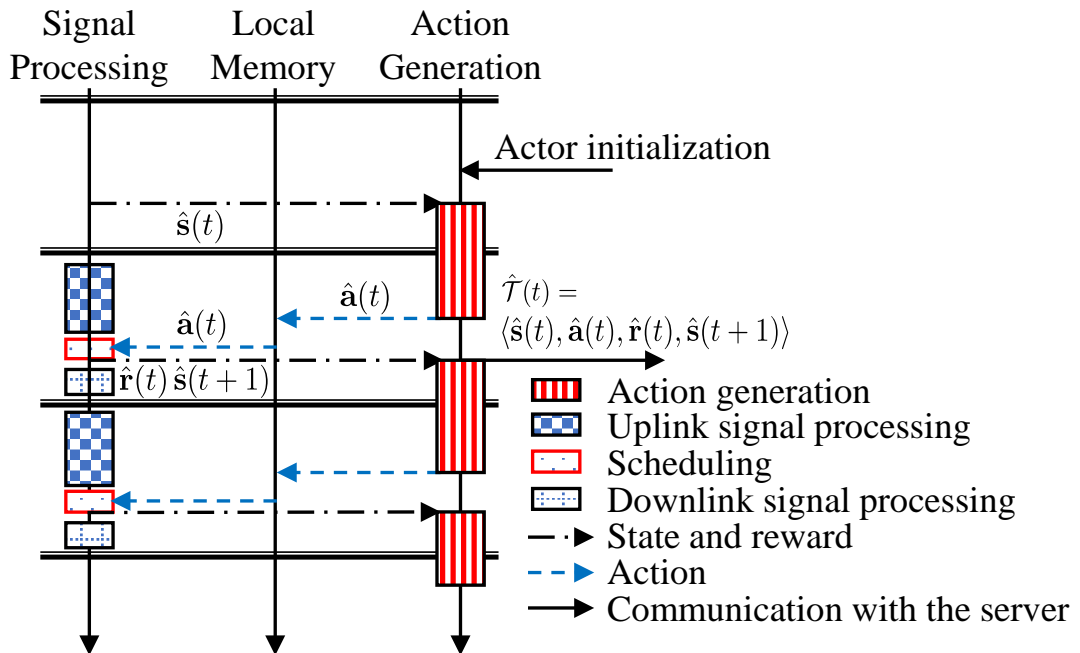


Figure 3.7: Parallel processing in the BS.

#### 3.7.2 Scheduler at the BS

After off-line initialisation, the BS fetches the parameters of the actor,  $\theta^\mu$ . In the  $t$ -th TTI, the BS observes the state  $\hat{\mathbf{s}}(t)$ , and generates an action according to the actor,  $\hat{\mathbf{a}}(t) = \mu(\hat{\mathbf{s}}(t)|\theta^\mu)$ . To avoid processing delay violation mentioned in Section 3.4.3, tasks for the action generation and the baseband signal processing are executed in parallel, as shown in Fig. 3.7. The generated action is saved in a local memory



### 3.8. Simulation Results

before it is executed in scheduling. The numbers of RBs allocated to the scheduled users are given by (3.14). After the action is executed, the BS computes the reward from (3.17) and observes the state in the next TTI,  $\hat{\mathbf{s}}(t+1)$ . Finally, the transition  $\hat{\mathcal{T}}(t) = \langle \hat{\mathbf{s}}(t), \hat{\mathbf{a}}(t), \hat{\mathbf{r}}(t), \hat{\mathbf{s}}(t+1) \rangle$ , is uploaded to the edge server and saved in the replay memory.

#### 3.7.3 Online Training in the Edge Server

In the edge server, the actor and the critic are initialised with the method in Section 3.7.1. Then, the server fine-tunes the actor and the critic by using transitions from the real-world scheduler at the BS. Specifically, this is achieved by executing lines 11-15 of Algorithm 1, iteratively. Once the actor and the critic are updated in each iteration, the parameters of the actor are sent to the scheduler. In order to enable the exploration in the real-world network, the server can add a noise in the *parameter space* of the actor according to  $\theta^\mu \leftarrow \theta^\mu \cdot (1 + \mathbb{N}(0, v^2) \cdot e^{-\lambda t \Delta_0})$ , where  $\mathbb{N}(0, v^2) \cdot e^{-\lambda t \Delta_0}$  are Gaussian noises that attenuate over time,  $v$  is the variance of the noise and  $\lambda$  is the attenuation rate [79].

As shown in the online architecture in Fig. 3.6, the rewards of users,  $\hat{\mathbf{r}}(t)$ , are uploaded to the edge server in each slot by the BS, where three kinds of knowledge are exploited in the K-DDPG algorithm. First, based on the knowledge that the QoS of the whole system depends on the QoS of each user, we use the multi-head critic in the edge server to approximate the long-term rewards of different users. Second, the shaped rewards,  $\hat{\mathbf{r}}(t)$ , are obtained from (3.21), where the form of the potential function  $\Psi(d_k(t))$  is designed by human experts based on their understanding of the knowledge of the target scheduling policy. Third, the knowledge of the importance of transitions is updated according to (3.27), where the weight of each transition depends on the approximation errors of the value function and the number of packet losses.

Table 3.2: 5G Scheduler Design Simulation Setup

System setup [63], [80]		Learning setup
BS transmit power spectrum density $\mathbf{P}_{\max}$	20 dBm/Hz	Exploration parameters $\sigma, \delta$
Noise power spectrum density $\mathbf{N}_0$	-90 dBm/Hz	Actor learning rate
Time slot duration (one TTI) $\Delta_0$	125 us	Critic learning rate
Bandwidth of a RB $W$	180 kHz	Soft-updating rate $\tau$
Packet size $L_k$	32 bytes	Replay memory size $ \mathcal{I} $
Packet arrival probability $p_k$	10%	Batch size $N_{\text{tr}}$
Required decoding error probability $\epsilon_{\max}$	$10^{-5}$	Time slots per episode $T$
Timeliness requirement $[D_{\min}, D_{\max}]$	[5, 7]	Potential function $\Psi_{\min}, \Psi_{\max}$
Maximum SNR $\log \phi_{\max}$	3.8	Discount factor $\gamma$
		1, 0.4
		$10^{-3}$
		$10^{-3}$
		$10^{-3}$
		10000
		20
		200
		0, 1
		0.9

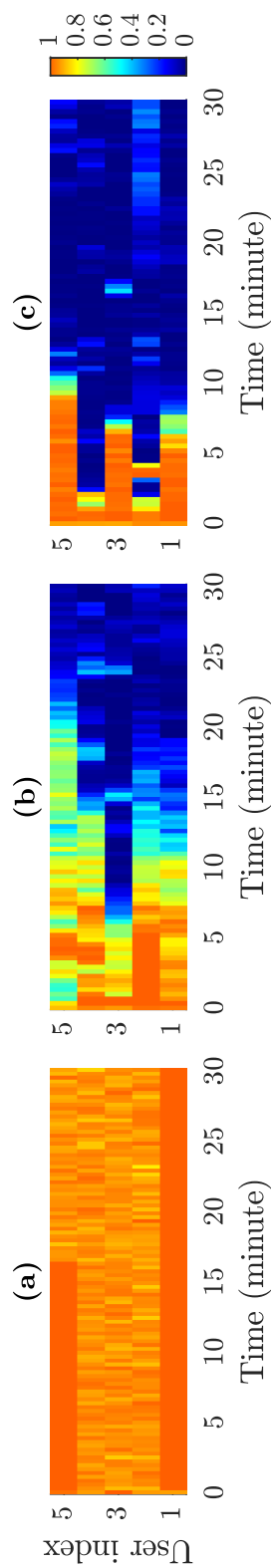


Figure 3.8: Packet loss probabilities, where  $K = 5$  and  $N = 50$ . (a) Straightforward implementation of DDPG, (b) DDPG in T-DRL framework, (c) K-DDPG in T-DRL framework.

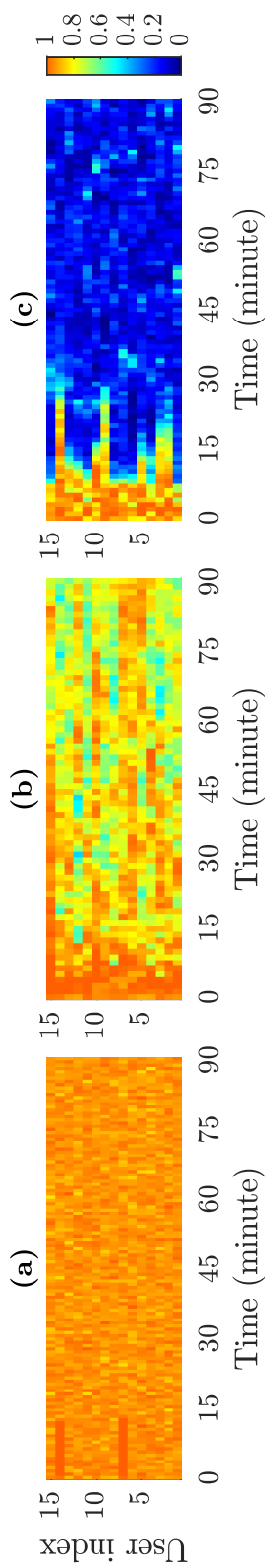


Figure 3.9: Packet loss probabilities, where  $K = 15$  and  $N = 50$ . (a) Straightforward implementation of DDPG, (b) DDPG in T-DRL framework, (c) K-DDPG in T-DRL framework.

## 3.8 Simulation Results

### 3.8.1 Simulation Platform

In the simulation platform, users randomly move with the velocity of 5 meters/second in a cell with a radius of 100 meters. The channel models are the same as that in Section 3.5.1. The path loss model is  $45 + 30 \log(l)$  dB, where  $l$  is the distance between a user and the BS in meters. At the beginning of each episode, we set each user at a random position in the cell. We assume that BS is in a factory and the small-scale channel gain follows a Rician distribution [76]. The ratio of the average power in the Line-of-Sight path to that in the Non-Line-of-Sight paths is set as 0.6. We consider a discrete-time channel model in the simulation. Given the small-scale channel gain in the current slot, with probability 80%, it remains the same in the next slot; with probability 20%, it varies according to Rician fading.

For hyper-parameters in DRL (i.e., exploration rates, learning rates, and soft-updating rate in Table 3.2), we tried different values and chose the best ones in this section. Both the actor and the critic have one input layer, one output layer, and two hidden layers. The number of neurons in each layer depends on the number of users. Specifically, the dimensions of the four layers of the actor are  $2K$ ,  $20K$ ,  $20K$  and  $K$ , respectively. The activation functions of the two hidden layers are ReLU functions. To ensure the output of the actor lies in  $[0, 1]$ , the activation function of the output layer is  $\frac{1}{2}\text{Tanh}(\cdot) + \frac{1}{2}$ . For the critic, the dimensions of the four layers are  $3K$ ,  $30K$ ,  $30K$ , and  $K$ , respectively. The ReLU function is used as activation functions of the two hidden layers, and no activation function is used in the output layer. In the simulation, the exploration noise,  $\mathcal{N}(t) \triangleq [\mathcal{N}_1(t), \dots, \mathcal{N}_K(t)]^T$ , is added to the output of the actor, where  $\mathcal{N}_k(t) = \mathcal{N}_k(t-1) + \delta \cdot \mathbb{N}(0, \sigma^2)$ .  $\mathbb{N}(0, \sigma^2)$  is a Gaussian variable with zero mean and variance  $\sigma^2$ . The parameter  $\delta$  is an adjustable exploration rate. The simulation setup is summarised in Table 3.2, unless mentioned otherwise.

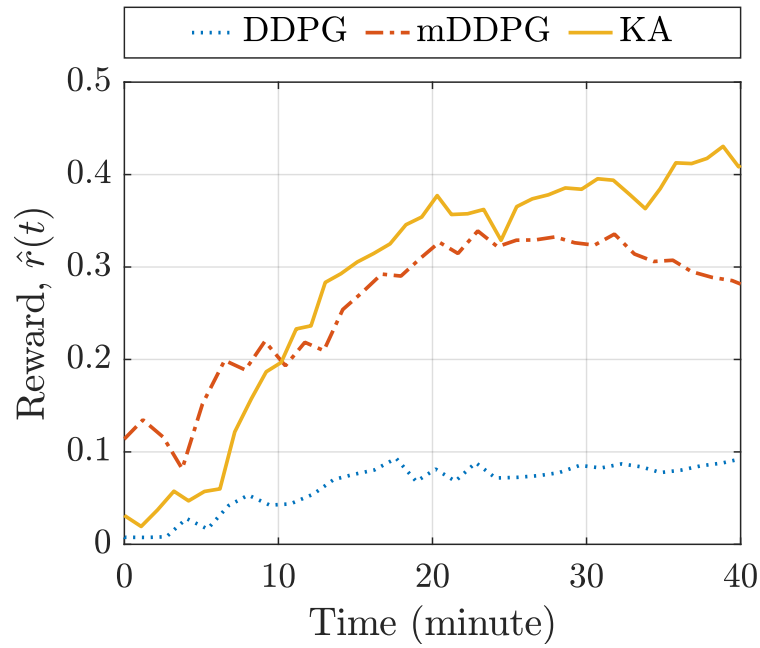
### 3.8.2 Performance of the T-DRL Framework and K-DDPG Algorithm

To illustrate the benefits of the T-DRL framework and the K-DDPG algorithms, Figs. 3.8 and 3.9 show the packet loss probabilities during off-line training in the simulation. The packet loss probabilities are measured every 5 episodes. When the number of users is small ( $K = 5$  in Fig. 3.8), DDPG converges after 25 minutes in the T-DRL framework, while the straightforward implementation of DDPG does not converge to a policy with low packet loss probabilities. With different types of expert knowledge of the scheduler design problem, K-DDPG can further reduce 50% of convergence time compared with DDPG (in the T-DRL framework). When the number of users is large ( $K = 15$  in Fig. 3.9), DDPG can hardly obtain a satisfactory scheduler without the assistance of knowledge. The results in Figs. 3.8 and 3.9 indicate that by applying K-DDPG in T-DRL framework, the scheduler learns faster than the cases without the knowledge or theoretical models.

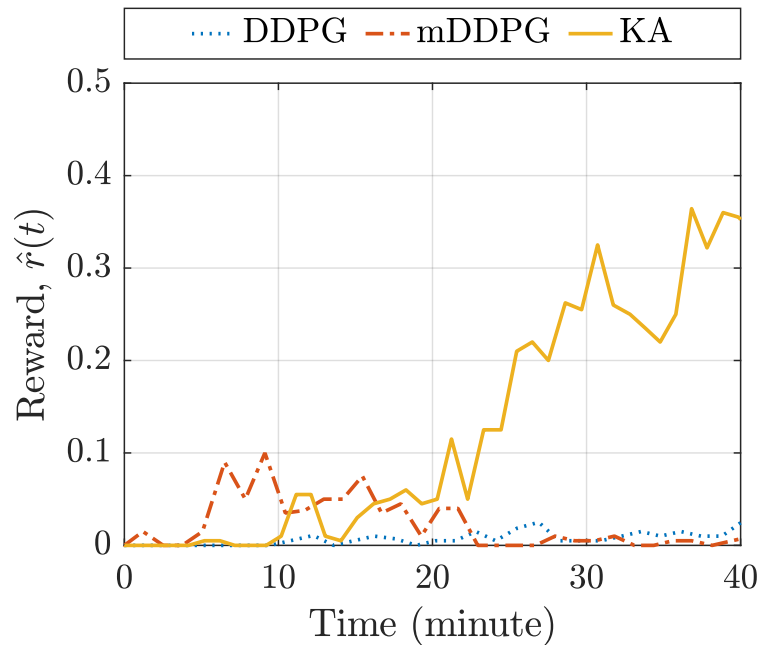
We then compare the reward of different DDPG algorithms in the T-DRL framework, including the original DDPG (with legend “DDPG”), an extension of DDPG in [72] (with legend “mDDPG”), and our K-DDPG (with legend “KA”). Fig. 3.10a shows the average reward of users achieved by these three schemes in a 40-minute training phase. The results show that the algorithm in [72] achieves a higher reward at the beginning of the training phase. This is because a human-written scheduler is used in exploration, which performs better than the randomly initialised actor in our scheme. However, our K-DDPG learns faster and achieves better performance than the other two algorithms by the end of the training phase. The reward of the worst-case user is shown in Fig. 3.10. The result indicates that K-DDPG is much better than the two other schemes.

To better illustrate the benefits of different kinds of knowledge, we illustrate the reward achieved by different algorithms: 1) original DDPG; 2) DDPG that exploits knowledge of the reward structure by using a multi-head critic (with legend “MH”); 3)

### 3.8. Simulation Results



(a) Average reward of users.

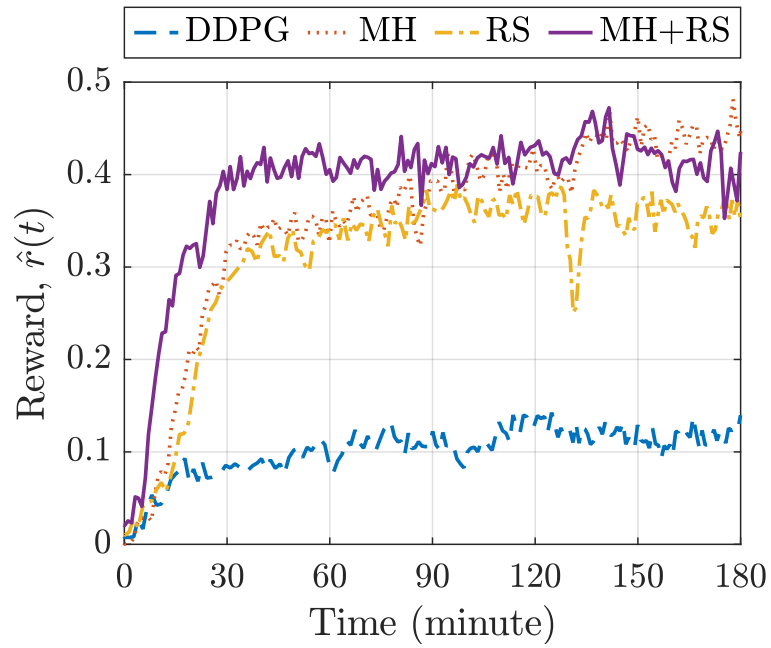


(b) Reward of the worst-case user.

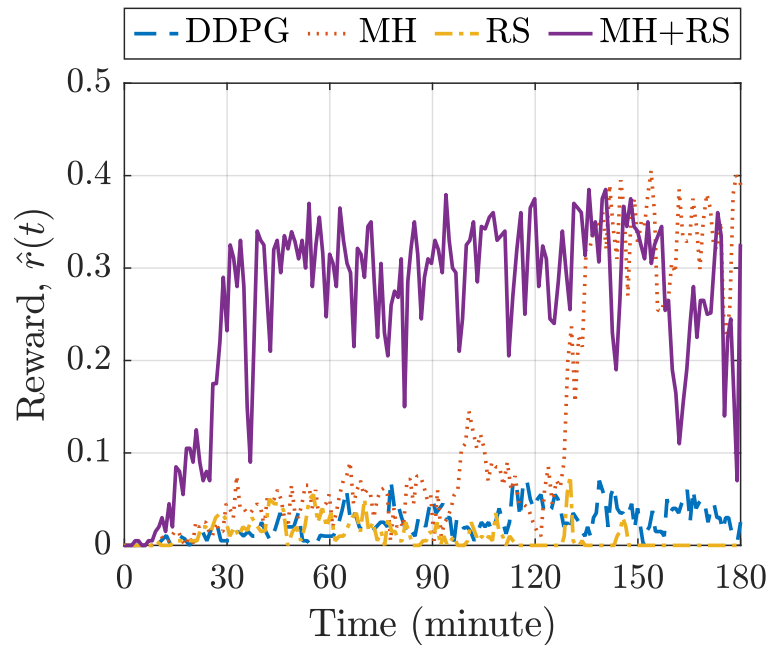
Figure 3.10: Rewards of different DDPG in the T-DRL framework, where  $K = 15$  and  $N = 50$ .

DDPG that exploits knowledge of the target scheduling policy by using reward shaping (with legend “RS”); 4) DDPG with both multi-head critic and reward shaping (with

### 3.8. Simulation Results



(a) Average reward of users.



(b) Reward of the worst-case user.

Figure 3.11: Rewards of DDPG with the assistance of different kinds of knowledge in the T-DRL framework, where  $K = 15$  and  $N = 50$ .

legend “MH+RS”). The result in Fig. 3.11 indicates that the multi-head critic helps improve the average reward and the reward of the worst-case user significantly. For

### 3.8. Simulation Results

example, the average reward of “MH” in Fig. 3.11a is 3 times higher than the average reward of DDPG. From the results in Fig. 3.11b, we can see that if reward shaping is further applied, the convergence time can be reduced by 80% from 140 minutes to 30 minutes. Note that the reward of “MH+RS” starts to decrease after 150 minutes of training. This is due to the overfitting of NNs. In practice, we only need to train the actor for 30 minutes with “MH+RS”.

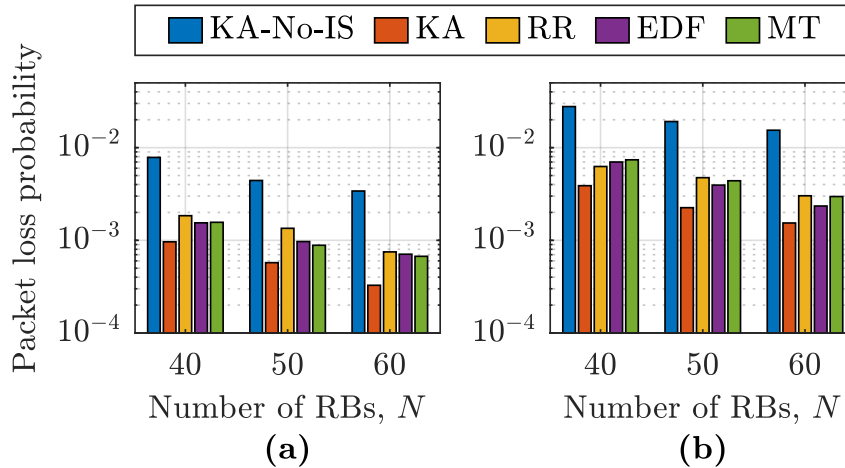


Figure 3.12: Reliability of scheduler for different total numbers of RBs, where  $K = 15$ . (a) Average packet loss probability of all users; (b) Packet loss probability of the worst-case user.

The reliability achieved by different schedulers is shown in Fig. 3.12. The packet loss probabilities are evaluated over 2000 episodes. To show the benefit of importance sampling, we evaluate the reliability of K-DDPG with and without it (with legends “KA” and “KA-No-IS”, respectively). In addition, we also evaluate the reliability of three existing schedulers: the round-robin scheduler (with legend “RR”), the earliest-deadline-first scheduler (with legend “EDF”) and the maximum throughput scheduler (with legend “MT”). The average packet loss probabilities of all users and the packet loss probabilities of the worst-case user are provided in Fig. 3.12a and Fig. 3.12b, respectively. The results indicate that without importance sampling, the scheduler can hardly achieve high reliability, while by using importance sampling, K-DDPG can reduce the packet loss probability by 30%  $\sim$  50% compared with the three existing schedulers.



## 3.9 Prototype of Online Architecture and Experimental Results

In this section, we show how to implement the proposed online DDPG architecture in a real-world network. Since 5G NR testbed [81] is still under development and is not available, we use an open-source Long Term Evolution (LTE) software suite [18] to build the prototype, in which we measure the processing time in both inference and training as well as the E2E latency and rewards experienced by users.

### 3.9.1 Prototype

#### Proposed architecture

The diagram of the prototype is shown in Fig. 3.13. We built K-DDPG based on the TDRL framework by using PyTorch in Python [40]. The algorithms run on a Dell 7820 workstation equipped with an RTX2080Ti graphics processing unit (GPU) and two Intel Xeon Gold 6134 central processing units (CPUs) with 8 cores each. The action generation process is developed based on libtorch in C++ [40]. We constructed the standard-compliant cellular network based on the open-source LTE software suite developed by Software Radio System (srsLTE) [18], which consists of eNodeB (srsENB, the BS), evolved packet core (srsEPC, the core network) and user equipment (srsUE, the user). We embedded the action generation process in the scheduler of srsENB. srsENB and srsEPC run on a Dell 7060 computer that has an Intel i7-8700 CPU with 6 cores, and srsUE run on Dell 7050 computers equipped with an Intel i7-6700 CPU with 4 cores. The radio transceivers for the BS and users are universal software radio peripheral (USRP) B210. We developed the communication protocol between the server and the BS in Google Protocol Buffers, which can automatically compile the protocol into Python and C++. We set the number of RBs,  $N$ , as 15. The duration of each slot,  $\Delta_0$ , is 1 ms and the bandwidth of each RB, including 12 subcarriers in LTE, is  $W = 180$  kHz.

### 3.9. Prototype of Online Architecture and Experimental Results

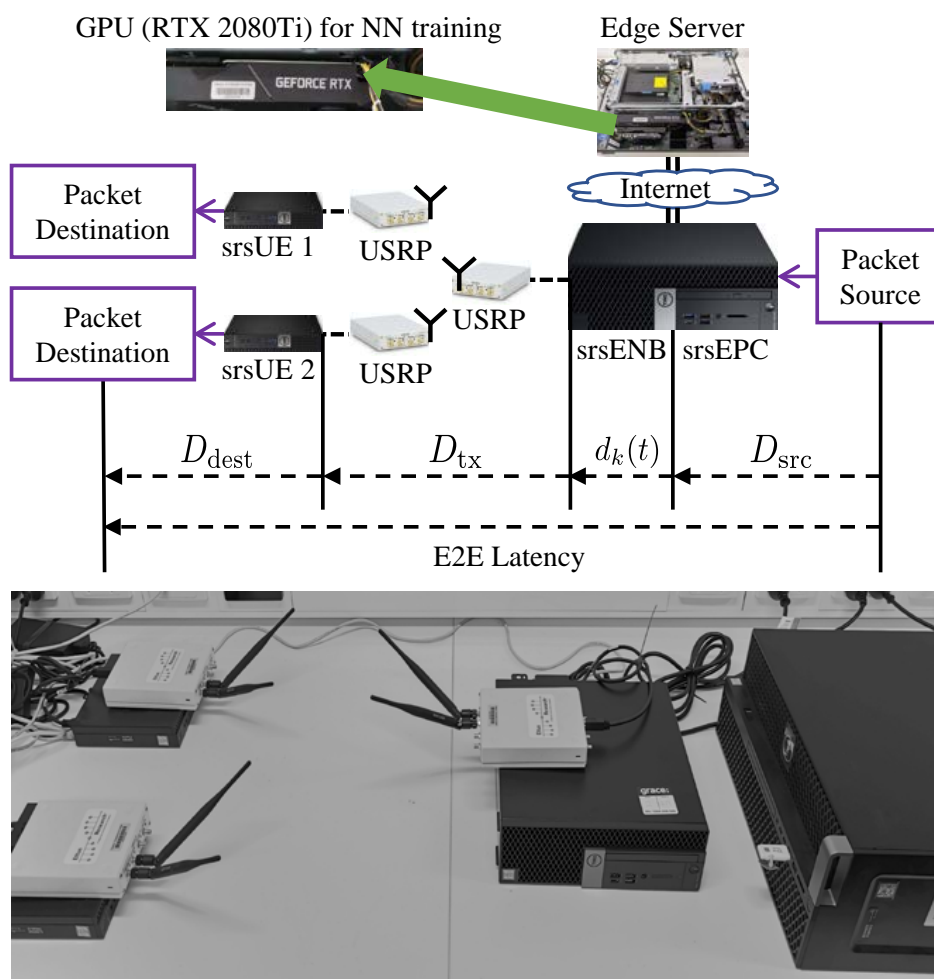


Figure 3.13: Diagram of the experiment setup.

#### Measurement platform

We developed a measurement platform to carry out experiments and measurements. In the platform, the packet source sends user datagram protocol (UDP) packets to the packet destinations. The packet size is 150 bytes and the arrival rate is 0.1 packet/ms. The E2E latency of a packet is measured at the packet destination by comparing the time it is sent by the source and the time the destination receives it. This requires the clocks of the computers to be highly synchronised. To achieve this goal, we implemented a time synchronisation system based on precision time protocol, which synchronises the clocks of the computers at a sub-microsecond level and allows accurate E2E latency measurements. Note that the clock synchronisation system is

not required to deploy T-DRL framework and K-DDPG in a commercial cellular network.

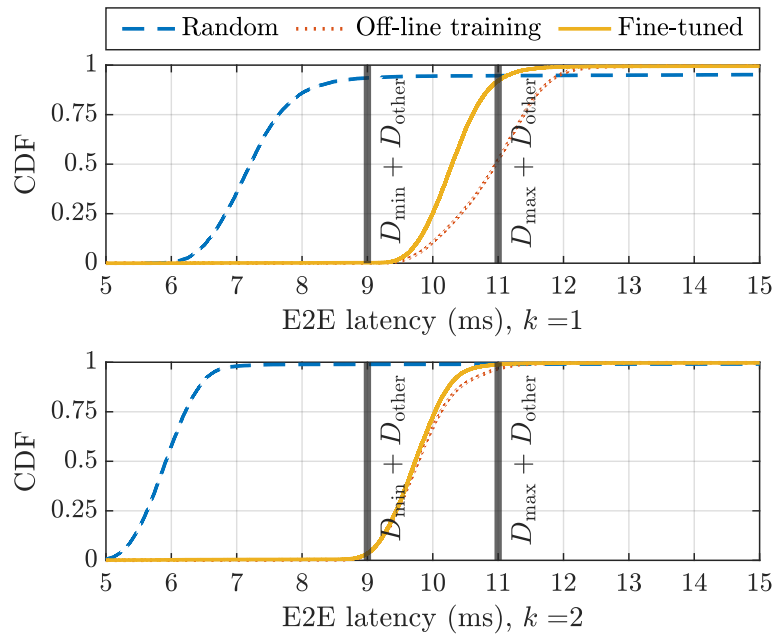
#### 3.9.2 Mismatch in Simulated and Real-world Networks

In the simulation, the time is discretised into slots (i.e., TTIs). Thus, the HoL delays are integers. In the real-world BS, the measured HoL delays are with nanosecond precision, denoted by  $\hat{d}_k$ , for  $k = 1, \dots, K$ . We convert it to the number of slots from  $d_k = \text{round}(\hat{d}_k/\Delta_0)$ , where  $\text{round}(x)$  is the closest integer to  $x$ . Furthermore, the CSI in the real-world BS is reported by users, i.e., a four-bit binary number referred to as the channel quality indicator. We can map this channel quality indicator to SNR based on the method in [80], [82]. E2E latency in the real-world network includes the delay from the packet source to srsENB,  $D_{\text{src}}$ , the queuing delay at the srsENB,  $d_k(t)$ , the transmission delay,  $D_{\text{tx}}$ , and the delay from srsUE to the packet destination,  $D_{\text{dest}}$ . We denote the total delays excluding the queuing delay as  $D_{\text{other}} \triangleq D_{\text{src}} + D_{\text{tx}} + D_{\text{dest}}$ . To meet the QoS requirements of time-sensitive traffic, the E2E latency should lie in  $[D_{\text{min}} + D_{\text{other}}, D_{\text{max}} + D_{\text{other}}]$ . We assume that  $D_{\text{src}} \ll D_{\text{tx}}$  and  $D_{\text{dest}} \ll D_{\text{tx}}$ . Then,  $D_{\text{other}} \approx D_{\text{tx}} = 4$  ms in the LTE system [83].

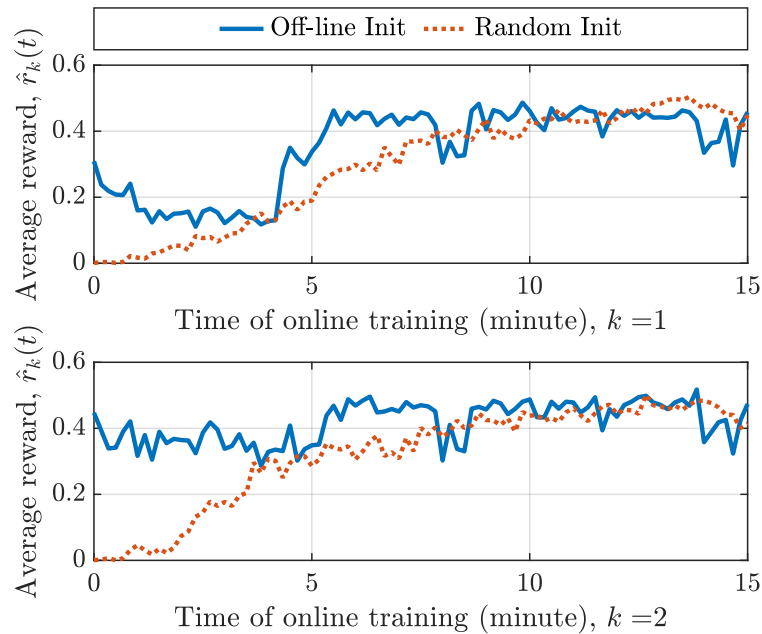
#### 3.9.3 Tests in a Real-world Network

Fig. 3.14a compares the cumulative distribution function (CDF) of the E2E latency experienced by two users, which is measured in the prototype for 2 minutes. The results show that with a high probability, the latency achieved by NNs initialised with random parameters (with legend ‘‘Random’’) does not lie in  $[D_{\text{min}} + D_{\text{other}}, D_{\text{max}} + D_{\text{other}}]$ . For the actor trained off-line in the simulation platform and directly applied in the real-world network without fine-tuning (with legend ‘‘Off-line training’’), with high probability  $d_2(t) \in [D_{\text{min}} + D_{\text{other}}, D_{\text{max}} + D_{\text{other}}]$ , but only around half of the packets are delivered to the first user with  $d_1(t) \in [D_{\text{min}} + D_{\text{other}}, D_{\text{max}} + D_{\text{other}}]$ . This is because the simulation platform is not exactly the same as the prototype. To handle this issue, the online DDPG architecture is applied to fine-tune the pre-trained

### 3.9. Prototype of Online Architecture and Experimental Results



(a) E2E latency in the prototype.



(b) Average rewards during online training.

Figure 3.14: Results of online training in the proposed architecture.

actor and critic in the prototype with a 15-minute online training phase, where we tried different configurations of the parameter space noise for online exploration as  $\theta^\mu \leftarrow \theta^\mu \cdot (1 + \mathcal{N}(0, v^2) \cdot e^{-\lambda t \Delta_0})$ , which is described in Section 3.7.3. We set the

### 3.9. Prototype of Online Architecture and Experimental Results

parameters as  $v = 0.1$  and  $\lambda = 5 \times 10^4$ , because these values can achieve the best performance according to our experience. With the fine-tuned actor (with legend “Fine-tuned”), the probability that  $d_1(t) \in [D_{\min} + D_{\text{other}}, D_{\max} + D_{\text{other}}]$  is improved remarkably, and the second user’s performance is also improved slightly. The average rewards of two users during online training are shown in Fig. 3.14b. The parameters of the actor and the critic are either initialised off-line in our simulation platform (with legend “Off-line Init”) or initialised with random variables (with legend “Random Init”). It shows that off-line initialisation not only significantly improves the initial performance but also reduces the convergence time of users by at least 40%. If the environment is highly dynamic (e.g., high mobility, burst traffic pattern, and frequent user list update), we might not be able to adjust the hyper-parameters of the actor and the critic in time, e.g., adjusting the number of hidden layers and the number of neurons in each layer. To handle this issue, one may consider applying few-shot learning methods [84] to further reduce the time needed for online fine-tuning. Also, one can use graph neural networks to transfer the trained NNs into scheduler design problems with different scales [29].

We measured the processing time of the feed-forward inference of the actor that runs on the Intel i7-8700 CPU at the BS. The average processing time of the inference is 0.036 ms and the maximum processing time is 0.067 ms, which is less than the duration of the shortest TTI in 5G NR, e.g., 0.125 ms. This result indicates that our scheduler can be operated at every TTI in real-world 5G systems. We also observed that the processing time grows as the sizes of the NNs increase. When the sizes of the NNs are large, we may need GPUs, field-programmable gate arrays or application-specific integrated circuits at the BS in order to avoid processing delay violation. Furthermore, we measured the average processing time of each training iteration in the edge server, i.e., around 5 ms. Thus, the online DDPG architecture can update the actor according to real-world networks every few milliseconds.

## 3.10 Summary

In this chapter, we developed the scheduler design method based on DDPG. We studied the straightforward implementation of DDPG and identified its limitations. To address these issues, we proposed the T-DRL framework to formulate the state, action and rewards of DDPG. Also, we designed the K-DDPG algorithm that exploits expert knowledge in communication systems to improve training efficiency. Finally, we developed the online architecture to fine-tune the NNs according to the real-time measurements from the BS. We conducted extensive simulations, showing our methods significantly improve the convergence of DDPG in scheduler design. Furthermore, we implement the online architecture prototype in a standard-compliant cellular network, which validates the practicality of our methods.

## Chapter 4

# Opportunistic Scheduling Using Statistical Information of Wireless Channels

*In the last chapter, we have developed a scheduler design algorithm based on deep reinforcement learning (DRL), where the algorithm interacts with the system and gradually optimises the scheduling policy based on the measurement and feedback from the network. Although we have designed several methods to improve the convergence time of the scheduler design algorithm, it still requires a long time to find the optimal scheduler. This is because the algorithm needs to interact with the network and “learn” the network’s stochastic behaviours. Assume that the complete statistical information of the network is known, then the above interaction is no longer needed. Meanwhile, measuring statistical information still requires time. This poses the questions of how to use the statistical information in the scheduler design and what statistical information should be used to reduce the overall time needed to find optimal schedulers eventually. In this chapter, we study these questions and provide a solution to accelerate the scheduler design’s convergence using the network’s statistical information.*

## 4.1 Introduction

Since time-varying channels limit the performance of multi-user wireless networks [85], scheduling policies for users' transmissions according to the stochastic variation of channel states is the key to optimising long-term system objectives of wireless networks. These objectives are framed as utility functions designed specifically to track networks' performance metrics [86], such as fairness or maximisation of data rates. Such scheduling strategies are referred to as opportunistic schedulers (OSs). Two OS classes have been reported in the open literature [86], [87], Markov decision process (MDP)-based OSs (including the one studied in the previous chapter) [J1], [88]–[93] and max-weight schedulers (MWSs) [19], [20], [94]–[97].

### 4.1.1 Related Works

#### MDP-based OSs

MDP-based OSs maximise the long-term utility function by calculating the optimal selection of users to be scheduled at each channel state, assuming a full prior knowledge of statistical channel state information (CSI), e.g., transition probabilities of channel states [88]–[91]. The calculated optimal user selections are saved in a lookup table that will be referred to for the channel state in each time slot [17]. Alternative to the above tabular approach, the MDP-based OS can also use a neural network (NN) to map the channel state into the optimal user selection in each slot, e.g., the DRL algorithm design in the last chapter. The NN's parameters can then be stochastically optimised by DRL based on the channel state, user selection and a well-designed reward signal in every slot [J1], [92], [93] without any knowledge of statistical CSI. As the NN contains many parameters that require optimisation, DRL methods take a long time to find the optimal MDP-based OS [J1].



## 4.1. Introduction

### Max-Weight Schedulers

The second OS class, namely MWSs [19], [20], [94]–[97], schedule a user with the highest weighted instantaneous utility, e.g., instantaneous rate, in each time slot, which is widely used in cellular networks. These methods continuously adjust the MWS’s weights for each slot based on every past channel state and user selection to maximise the long-term utility function. This approach requires no prior knowledge of statistical CSI. Since MWS designs only need to optimise a vector of weights, they have a much lower implementation complexity than the lookup table or the NN of MDP-based OSs mentioned above. Unfortunately, these MWS approaches still require hundreds of time slots in trials of adjusting weights before they find optimal ones, leading to sub-optimal system performance. Applying statistical CSI in the MWS design is a possible direction to save time slots in the online adjustment of weights [87]. However, time slots are still required to measure instantaneous CSI and further estimate the statistical CSI. Further research is needed on how to design MWSs based on the prior knowledge of statistical CSI that costs few slots to obtain.

### 4.1.2 Our Methods

In this chapter, we propose a new method that uses limited prior knowledge of statistical CSI to effectively reduce the number of time slots required in the MWS design. We find the optimal MWS’s weights to maximise the utility function as the sum of the logs of users’ average scheduled bit rates in the multi-user wireless network [19], [20], [86], [87]. This utility function is commonly used to formulate the service requirements of high data rate applications in enhanced mobile broadband (eMBB) [7], e.g., video streaming [98]. In this work, users’ signal-to-noise ratios (SNRs) are considered as CSI, which can be measured from radio signals, e.g., 5G cellular networks’ CSI reference signals [74]. We first derive each user’s average rate for given MWS’s weights from the full prior knowledge of statistical CSI, namely the probability density functions (PDFs) of users’ SNRs [99]–[101]. To use limited prior knowledge of statistical CSI instead, we re-derive the computation of users’ average rates for given MWS’s

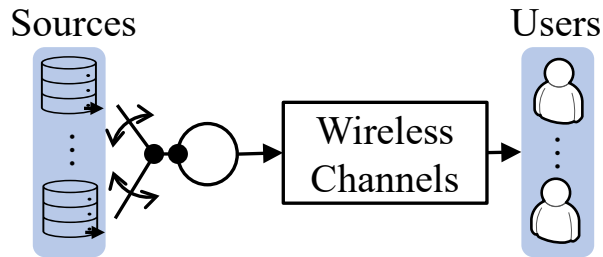


Figure 4.1: Illustration of a wireless scheduler.

weights as an optimisation problem based on only the mean and variance (i.e., the first and the second moment of the PDF [102]) of users' SNRs, referred to as a rate estimation problem. Here, we use the mean and variance of SNRs to construct constraints that bound users' feasible average data rates. Next, we formulate mean-variance-based weight optimisation (MVWO) that maximises the above utility function. We construct this problem as a bi-level optimisation problem (BLOP) [23], [24] with the MWS weights as optimisation variables. The rate estimation problem is embedded in the BLOP to specify the average rates at given weights. We design an iterative solver for the BLOP and mathematically prove that it returns the optimal MWS's weights in MVWO. Furthermore, since real-world networks have varying mean and variance of SNRs, e.g., due to the mobility of users, we study how to use the proposed MVWO method to adjust the MWS's weights based on online SNR measurements. This chapter's contributions have been summarised in Chapter 1.

## 4.2 System Model and Problem Formulation of Opportunistic Scheduling

In this section, we present the system model of the multi-user wireless network and formulate the optimisation problem of the MWS design. Further, we study the issues of using PDFs of users' SNRs to design MWSs.

### 4.2.1 System Model

We consider a wireless link of the BS shared by  $K$  users in time slots, as shown in Fig. 4.1. The duration of each slot in the system is  $\Delta_0$  in seconds. The bandwidth of the link is  $B$  in Hertz. A binary indicator of the user selection decision,  $x_k(t)$ , represents whether the BS transmits the data of the  $k$ -th user,  $k = 1, \dots, K$ , in the  $t$ -th slot or not,  $t = 1, 2, \dots$ , i.e.,

$$x_k(t) \in \{0, 1\}, \forall t, k. \quad (4.1)$$

For example,  $x_k(t) = 1$  if the  $k$ -th user occupies in the  $t$ -th slot and the BS transmits this user's data, otherwise  $x_k(t) = 0$ . We assume that only one user can access the channel in each slot, which is mathematically written as

$$\sum_{k=1}^K x_k(t) \leq 1, \forall t. \quad (4.2)$$

The user scheduling actions in the  $t$ -th slot are defined as

$$\mathbf{x}(t) \triangleq [x_1(t), \dots, x_K(t)]^T. \quad (4.3)$$

We assume that the channel of the  $k$ -th user has a stationary SNR, whose value is  $\phi_k(t)$  in the  $t$ -th slot and is assumed to be i.i.d. in each slot. We define the channel state in the  $t$ -th slot as

$$\mathbf{s}(t) \triangleq [\phi_1(t), \dots, \phi_K(t)]^T. \quad (4.4)$$

The mean and the variance of  $\phi_k(t)$  are denoted as

$$m_k^\phi \triangleq \mathbb{E}[\phi_k(t)], \quad v_k^\phi \triangleq \mathbb{V}[\phi_k(t)], \quad \forall k. \quad (4.5)$$

The SNRs of different users are assumed to be independent. The spectrum efficiency of the  $k$ -th user in the  $t$ -th slot is calculated by using Shannon capacity as  $\log_2(1 + \phi_k(t))$ . The amount of instantaneous bits scheduled for the  $k$ -th user in the  $t$ -th slot is then given as  $x_k(t)\Delta_0 B \log_2(1 + \phi_k(t))$ . We define a vector of scheduled instantaneous bit

## 4.2. System Model and Problem Formulation of Opportunistic Scheduling

rate for  $K$  users as

$$\mathbf{v}(\mathbf{x}(t), \mathbf{s}(t)) \triangleq \left[ x_1(t) \Delta_0 B \log_2 (1 + \phi_1(t)), \dots, x_K(t) \Delta_0 B \log_2 (1 + \phi_K(t)) \right]^T, \forall t, \quad (4.6)$$

where  $\mathbf{x}(t)$  and  $\mathbf{s}(t)$  are defined in (4.3) and (4.4), respectively.

At each slot  $t$ , any scheduler,  $\omega$ , decides the binary user scheduling actions based on the channel states as

$$\forall t, \mathbf{x}(t) = \omega(\mathbf{s}(t)), \text{ s.t. (4.1) (4.2)}. \quad (4.7)$$

The average rates scheduled by any scheduler for users  $1, \dots, K$ ,  $\mathbf{r} = [r_1, \dots, r_K]^T$  is given as

$$\mathbf{r} \triangleq \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^T \mathbf{v}(\mathbf{x}(t), \mathbf{s}(t)), \quad (4.8)$$

where  $\mathbf{x}(t)$ ,  $t = 1, 2, \dots$  can be binary user scheduling actions decided by any scheduler,  $\omega$ , as shown in (4.7). We define the feasible rate region,  $\mathcal{F}$ , that contains all possible values of the average rates achieved by any scheduler as follows,

$$\mathcal{F} \triangleq \{\mathbf{r} | (4.1) \text{ and } (4.2)\}, \quad (4.9)$$

which is a compact convex set based on the justification in [20].

### 4.2.2 MWS Design Problem Formulation

The utility function of the system is the sum of the logs (or proportional fairness [16], [103]) of users' average rates, expressed as

$$f(\mathbf{r}) \triangleq \sum_{k=1}^K \ln r_k, \quad (4.10)$$

and it is strictly increasing and concave. We use MWSs as OSs to schedule users at each time slot. We first define  $\mathbf{w} \triangleq [w_1, \dots, w_K]^T$ ,  $\|\mathbf{w}\|_2 = 1$  and  $\mathbf{w} \in \mathbb{R}_{>0}^K$ , as weights

## 4.2. System Model and Problem Formulation of Opportunistic Scheduling

of the MWS for  $K$  users, which are fixed for time  $T$ . The MWS,  $\mu$ , decides the user scheduling action,  $\mathbf{x}(t)$ , based on the channel state,  $\mathbf{s}(t)$ , at the  $t$ -th slot as

$$\forall t, \mathbf{x}(t) = \mu(\mathbf{s}(t)|\mathbf{w}) \triangleq \arg \max_{\mathbf{x}'(t)} \langle \mathbf{w}, \mathbf{v}(\mathbf{x}'(t), \mathbf{s}(t)) \rangle \text{ s.t. (4.1) (4.2)}. \quad (4.11)$$

Here, the values of the weights in MWSs control the probability that users are selected for every time slot. The average rates achieved by MWSs in (4.8) for users  $1, \dots, K$ ,  $\mathbf{r}^{\sim\mu(\cdot|\mathbf{w})} = [r_1^{\sim\mu(\cdot|\mathbf{w})}, \dots, r_K^{\sim\mu(\cdot|\mathbf{w})}]^T$ , can then be written as

$$\mathbf{r}^{\sim\mu(\cdot|\mathbf{w})} \triangleq \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^T \mathbf{v}(\mu(\mathbf{s}(t)|\mathbf{w}), \mathbf{s}(t)). \quad (4.12)$$

Note that  $\mathbf{r}^{\sim\mu(\cdot|\mathbf{w})}$  are feasible rates that belong to  $\mathcal{F}$ . The problem formulation of finding the optimal weights of MWSs that maximise (4.10) is

$$\max_{\mathbf{w}} f(\mathbf{r}^{\sim\mu(\cdot|\mathbf{w})}), \text{ s.t. } \mathbf{w} \in \mathbb{R}_{>0}^K, \|\mathbf{w}\|_2 = 1, \quad (4.12). \quad (\text{P1})$$

The numerical value of  $\mathbf{r}^{\sim\mu(\cdot|\mathbf{w})}$  at given  $\mathbf{w}$  is needed to solve (P1), which can be calculated based on the full knowledge of statistical CSI, namely PDFs of the SNRs [99]–[101], as follows.

### 4.2.3 Rate Estimation Using Full Knowledge of Statistical CSI

The  $k$ -th user's average rate achieved by MWSs,  $r_k^{\sim\mu(\cdot|\mathbf{w})}$ ,  $k = 1, \dots, K$ , in (4.12) can be further calculated as

$$\begin{aligned} r_k^{\sim\mu(\cdot|\mathbf{w})} &= \mathbb{E}[\Delta_0 B x_k(t) \log_2(1 + \phi_k(t))] \\ &\stackrel{(a)}{=} \Delta_0 B \mathbb{E}[x_k(t)] \mathbb{E}[\log_2(1 + \phi_k(t)) | x_k(t) = 1] \\ &= \Delta_0 B \Pr[x_k(t) = 1] \int_{\phi} q_k(\phi | x_k(t) = 1) \log_2(1 + \phi) d\phi \\ &\stackrel{(b)}{=} \Delta_0 B \int_{\phi} q_k(\phi) \Pr[x_k(t) = 1 | \phi_k(t) = \phi] \log_2(1 + \phi) d\phi, \quad \forall k, \end{aligned} \quad (4.13)$$

## 4.2. System Model and Problem Formulation of Opportunistic Scheduling

where (a) is because  $x_k(t)$  is binary and (b) uses Bayes' theorem [99]. Here,  $q_k(\phi)$  and  $q_k(\phi|x_k(t) = 1)$  in (4.13) are the PDFs of the  $k$ -th user's SNRs in all  $T$  time slots and in those time slots where this user is scheduled, respectively. Furthermore,  $\Pr[x_k(t) = 1|\phi_k(t) = \phi]$  in (4.13) is the probability that the  $k$ -th user is scheduled in the  $t$ -th slot for a given value of its SNR,  $\phi$ . Such probability can be calculated for the MWS,  $\mu(\cdot|\mathbf{w})$ , as

$$\begin{aligned} \Pr[x_k(t) = 1|\phi_k(t) = \phi] &\stackrel{(a)}{=} \prod_{j \neq k} \Pr[w_j \log_2(1 + \phi_j(t)) < w_k \log_2(1 + \phi)] \\ &= \prod_{j \neq k} \Pr[\phi_j(t) < (1 + \phi)^{\frac{w_k}{w_j}} - 1] = \prod_{j \neq k} \int_0^{(1+\phi)^{\frac{w_k}{w_j}} - 1} q_j(\psi) d\psi, \quad \forall k, t, \mathbf{w}, \end{aligned} \quad (4.14)$$

where (a) uses the definition of MWSs in (4.11) that a user occupies a slot when it has the highest weighted spectrum efficiency among other users in this slot. By substituting (4.14) into (4.13), the average rates of MWSs can be rewritten as

$$r_k^{\sim\mu(\cdot|\mathbf{w})} = \Delta_0 B \int_{\phi} q_k(\phi) \left( \prod_{j \neq k} \int_0^{(1+\phi)^{\frac{w_k}{w_j}} - 1} q_j(\psi) d\psi \right) \cdot \log_2(1 + \phi) d\phi, \quad \forall k, \mathbf{w}. \quad (4.15)$$

Here,  $r_k^{\sim\mu(\cdot|\mathbf{w})}$  can be calculated if PDFs of all users' SNRs are known. Such an approach for the calculation of  $\mathbf{r}^{\sim\mu(\cdot|\mathbf{w})}$  suffers from two issues. First, estimating PDFs of the SNRs has a high time complexity, or in other words, it requires a large number of time slots to collect samples of channel states such that each state is counted sufficient times. Second, even though the PDFs are obtained, it is difficult to numerically calculate the integrals in (4.15), and additional estimators of the integrals are required [99]. These issues in PDF-based rate estimation are difficult to address. Thus, we propose a new method to estimate the average rates in the sequel.

## 4.3 Rate Estimation Using Mean and Variance of SNRs

In this section, we propose a new method for calculating the average rates,  $\mathbf{r}^{\sim\mu(\cdot|\mathbf{w})}$ , achieved by MWSs,  $\mu(\cdot|\mathbf{w})$ , in order to solve (P1). The proposed method only uses the mean and variance of the SNRs defined in (4.5) to approximate the average rates of MWSs.

### 4.3.1 Bounding the Feasible Rate Region

As we cannot use (4.15) directly, we formulate a rate estimation problem that calculates  $\mathbf{r}^{\sim\mu(\cdot|\mathbf{w})}$  based on the feasible rate region,  $\mathcal{F}$ , defined in (4.9) as

**Corollary 1.**  $\forall \mathbf{w} \in \mathbb{R}_{>0}^K, \|\mathbf{w}\|_2 = 1,$

$$\mathbf{r}^{\sim\mu(\cdot|\mathbf{w})} = \arg \max_{\mathbf{r}} \langle \mathbf{w}, \mathbf{r} \rangle, \text{ s.t. } \mathbf{r} \in \mathcal{F}. \quad (4.16)$$

*Proof.* The proof is in the appendix. □

Next, we aim at using the mean and variance of the SNRs, defined in (4.5), to construct a convex set,  $\mathcal{G}$ , that represents  $\mathcal{F}$ . In this way, we can find an estimation of  $\mathbf{r}^{\sim\mu(\cdot|\mathbf{w})}$  by solving the optimisation problem at the right-hand side (RHS) of (4.16), in which  $\mathcal{F}$  is replaced by  $\mathcal{G}$ . To achieve this, we will construct convex expressions<sup>1</sup> as constraints based on the mean and variance of the SNRs to govern the randomness of  $\mathbf{x}(t)$ ,  $\mathbf{s}(t)$  and the correlation between them, which also specify the range of the value of users' feasible average rates in  $\mathcal{F}$ .

First, let us consider any feasible values of the average rate of the  $k$ -th user,  $r_k$ ,  $k = 1, \dots, K$ , including the average rates achieved by MWSs in (4.12). We note that

---

<sup>1</sup>We will focus on explaining our methods without explicit proof of the claim on the convexity of inequalities, equalities and optimisation problems. If interested, readers can use the checker implemented in [49] to validate the claimed convexity.

### 4.3. Rate Estimation Using Mean and Variance of SNRs

users' average rates are non-negative, which can be expressed in constraints as

$$r_k \geq 0, \forall k. \quad (\text{C1})$$

An upper bound on any feasible values of the  $k$ -th user's rate,  $r_k$ ,  $k = 1, \dots, K$ , is obtained as<sup>2</sup>

$$\begin{aligned} r_k &= \Delta_0 B \mathbb{E}[x_k(t) \log_2(1 + \phi_k(t))] = \Delta_0 B \mathbb{E}[x_k(t)] \mathbb{E}[\log_2(1 + \phi_k(t)) | x_k(t) = 1] \\ &\stackrel{(a)}{\leq} \Delta_0 B \mathbb{E}[x_k(t)] \log_2(1 + \mathbb{E}[\phi_k(t) | x_k(t) = 1]), \forall k, \end{aligned} \quad (\text{4.17})$$

where (a) uses Jensen's inequality on the expected value of the concave function,  $\log_2(1 + (\cdot))$ . Here,  $\mathbb{E}[\phi_k(t) | x_k(t) = 1]$  in (4.17) is each user's average SNR in scheduled slots, calculated by

$$\mathbb{E}[\phi_k(t) | x_k(t) = 1] = \frac{\mathbb{E}[x_k(t)\phi_k(t)]}{\mathbb{E}[x_k(t)]} = \frac{y_k}{p_k}, \forall k, \quad (\text{4.18})$$

where  $y_k$  and  $p_k$  are defined as

$$y_k \triangleq \mathbb{E}[x_k(t)\phi_k(t)], \quad p_k \triangleq \mathbb{E}[x_k(t)], \quad \forall k. \quad (\text{4.19})$$

Then, we substitute (4.18) and (4.19) into (4.17) as

$$r_k \leq \Delta_0 B \cdot p_k \cdot \log_2\left(1 + \frac{y_k}{p_k}\right), \forall k. \quad (\text{C2})$$

Note that  $p_k$  is the expected value of a binary number,  $x_k(t)$ , as defined in (4.1), which implies

$$0 \leq p_k \leq 1, \forall k. \quad (\text{C3})$$

---

<sup>2</sup>Note that the channel capacity in this chapter does not consider fading. We can extend the method to channel with fading by calculating the capacity as  $\mathbb{E}[B \log_2(1 + |h_k|^2 \phi_k(t))]$  in the  $t$ -th slot for user  $k$ , where  $h_k$  is the random normalised fading gain of user  $k$ . Regardless of the fading model, the amount of instantaneous bits scheduled for the  $k$ -th user in the  $t$ -th slot can be upper-bounded as  $x_k(t)\Delta_0 B \mathbb{E}[\log_2(1 + |h_k|^2 \phi_k(t))] \leq x_k(t)\Delta_0 B \log_2(1 + \mathbb{E}[|h_k|^2 \phi_k(t)]) = x_k(t)\Delta_0 B \log_2(1 + \phi_k(t))$ , using Jensen's inequality. This derivation will lead to the same upper bound of users' data rates. Thus, the MVWO method still applies.



### 4.3. Rate Estimation Using Mean and Variance of SNRs

The summation of  $x_k(t)$  for  $k \in \{1, \dots, K\}$  is less than 1 at each slot, as stated in (4.2), leading to a constraint on  $p_k$  as

$$\sum_{k=1}^K p_k \leq 1. \quad (\text{C4})$$

Next, we study the relationship between  $y_k$  and  $p_k$ ,  $k = 1, \dots, K$ , in the covariance matrix of binary indicators of user selection decisions,  $x_k(t)$ , and SNRs of users,  $\phi_k(t)$ ,  $k = 1, \dots, K$ , as

$$\mathbf{H} \triangleq \begin{bmatrix} \mathbf{H}^{xx} & \mathbf{H}^{x\phi} \\ (\mathbf{H}^{x\phi})^T & \mathbf{H}^{\phi\phi} \end{bmatrix}. \quad (\text{4.20})$$

$\mathbf{H}$  is a  $2K \times 2K$  matrix and each submatrix in  $\mathbf{H}$  has  $K \times K$  dimension.  $(\cdot)^T$  here is the transpose of a matrix. Elements in the diagonal  $\mathbf{H}^{x\phi}$  are the covariance between  $x_k(t)$  and  $\phi_k(t)$ ,  $k \in \{1, \dots, K\}$ , and is calculated based on  $y_k$  and  $p_k$  as

$$H_{k,k}^{x\phi} = y_k - p_k m_k^\phi, \forall k, \quad (\text{C5})$$

where  $m_k^\phi$  is the mean of  $\phi_k(t)$ , as defined in (4.5).

The lower-right part of  $\mathbf{H}$ ,  $\mathbf{H}^{\phi\phi}$ , is the covariance matrix of  $\phi_i(t)$  and  $\phi_j(t)$ ,  $i, j \in \{1, \dots, K\}$ , whose elements are

$$H_{i,j}^{\phi\phi} = \begin{cases} v_k^\phi, & \text{if } i = j, \\ 0, & \text{if } i \neq j, \end{cases} \quad \forall i, j \in \{1, \dots, K\}, \quad (\text{C6})$$

where  $v_k^\phi$  is the variance of  $\phi_k(t)$ , as defined in (4.5). Note that SNRs of users are independent. Thus, all off-diagonal elements in  $\mathbf{H}^{\phi\phi}$  are 0 in (C6).

$\mathbf{H}^{xx}$  is the covariance matrix of  $x_i(t)$  and  $x_j(t)$ ,  $i, j \in \{1, \dots, K\}$ . Note that  $x_i(t)$  and  $x_j(t)$  are binaries. The covariance of two binary random numbers is calculated

### 4.3. Rate Estimation Using Mean and Variance of SNRs

as

$$\begin{aligned} H_{i,j}^{xx} &= \mathbf{cov}(x_i(t), x_j(t)) = \mathbb{E}[x_i(t)x_j(t)] - \mathbb{E}[x_i(t)]\mathbb{E}[x_j(t)] \\ &= \Pr[x_i(t) = 1, x_j(t) = 1] - \mathbb{E}[x_i(t)]\mathbb{E}[x_j(t)] , \quad \forall i, j \in \{1, \dots, K\} . \end{aligned} \quad (4.21)$$

As stated in (4.2), if  $i \neq j$ ,  $x_i(t)$  and  $x_j(t)$  cannot be simultaneously equal to 1 for a given  $t$ . This means that  $\Pr[x_i(t) = 1, x_j(t) = 1] = 0$  if  $i \neq j$ . Also, when  $i = j$ , we have  $\Pr[x_i(t) = 1, x_j(t) = 1] = \Pr[x_i(t) = 1] = p_i$ . Therefore, the elements in  $\mathbf{H}^{xx}$  in (4.21) can be written as

$$H_{i,j}^{xx} = \begin{cases} p_i - p_i^2, & \text{if } i = j, \\ -p_i p_j, & \text{if } i \neq j, \end{cases} \quad \forall i, j \in \{1, \dots, K\} . \quad (4.22)$$

Note that the above equalities are not convex constraints. In order to construct convex constraints for  $\mathcal{G}$ , we can relax the above constraints in (4.22) into convex ones as

$$H_{k,k}^{xx} \leq p_k - p_k^2 , \quad \forall k , \quad (C7)$$

where constraints on the elements in the main diagonal and the off-diagonal of  $\mathbf{H}^{xx}$  are changed into inequalities and removed, respectively. Also, we observe that the summation of all elements in  $\mathbf{H}^{xx}$  in (4.22) is

$$\sum_{i=1}^K \sum_{j=1}^K H_{i,j}^{xx} = \sum_{i=1}^K p_i - \left( \sum_{i=1}^K p_i \right)^2 , \quad (4.23)$$

which can be relaxed into a convex constraint as

$$\sum_{i=1}^K \sum_{j=1}^K H_{i,j}^{xx} \leq \sum_{i=1}^K p_i - \left( \sum_{i=1}^K p_i \right)^2 . \quad (C8)$$

The positive semidefiniteness of covariance matrices is the constraint on all elements in  $\mathbf{H}$  as

$$\mathbf{H} \succeq 0 . \quad (C9)$$

### 4.3. Rate Estimation Using Mean and Variance of SNRs

We note that (C1), (C2),  $\dots$ , (C9) are all convex constraints.

Finally, the collection of (C1)-(C9) defines a convex set,  $\mathcal{E}$ , that contains all possible values of tuple  $(\mathbf{r}, \mathbf{p}, \mathbf{y}, \mathbf{H})$  as

$$\mathcal{E} \triangleq \{(\mathbf{r}, \mathbf{p}, \mathbf{y}, \mathbf{H}) | (\text{C1})\text{--}(\text{C9})\} , \quad (4.24)$$

where tuples' elements are defined as  $\mathbf{r} \triangleq \{r_1, \dots, r_K\}$ ,  $\mathbf{p} \triangleq \{p_1, \dots, p_K\}$ ,  $\mathbf{y} \triangleq \{y_1, \dots, y_K\}$  and (4.20). Also, we define the set of all possible values of  $\mathbf{r}$  in all tuples of  $\mathcal{E}$  as

$$\mathcal{G} \triangleq \{\mathbf{r} | \mathbf{r} = \text{proj}_1[(\mathbf{r}, \mathbf{p}, \mathbf{y}, \mathbf{H})], \forall (\mathbf{r}, \mathbf{p}, \mathbf{y}, \mathbf{H}) \in \mathcal{E}\} , \quad (4.25)$$

which can be interpreted as a projection of tuples in  $\mathcal{E}$  at their coordinates of  $\mathbf{r}$ . Note that (C1)-(C9) are all convex, which implies the set  $\mathcal{E}$  defined in (4.24) is convex and further implies the projection of  $\mathcal{E}$ ,  $\mathcal{G}$ , defined in (4.25) is also convex [47].

#### 4.3.2 The Proposed Rate Estimation for MWSs

By using  $\mathcal{G}$  as an approximation of  $\mathcal{F}$ , we then can rewrite the rate estimation problem in (4.16) that estimates the average rates of the MWS with weights,  $\mathbf{w}$ , as

$$\mathbf{r}^{\sim\mu(\cdot|\mathbf{w})} \approx \mathbf{r}_{\mathcal{G}}^{\sim\mu(\cdot|\mathbf{w})} \triangleq \arg \max_{\mathbf{r}} \langle \mathbf{w}, \mathbf{r} \rangle , \text{ s.t. } \mathbf{r} \in \mathcal{G} , \quad (4.26)$$

where  $\mathbf{r}_{\mathcal{G}}^{\sim\mu(\cdot|\mathbf{w})}$  is the estimated value of  $\mathbf{r}^{\sim\mu(\cdot|\mathbf{w})}$  in (4.12). As  $\mathcal{G}$  can be represented by (C1)-(C9) according to (4.24) and (4.25), we can further rewrite the problem in the RHS of (4.26) as

$$\max_{\mathbf{r}, \mathbf{p}, \mathbf{y}, \mathbf{H}} \langle \mathbf{w}, \mathbf{r} \rangle , \text{ s.t. } (\text{C1})\text{--}(\text{C9}) , \quad (4.27)$$

Note that the optimal  $\mathbf{r}$  from (4.27) is equal to the optimal  $\mathbf{r}$  from the RHS of (4.26), i.e.,  $\mathbf{r}_{\mathcal{G}}^{\sim\mu(\cdot|\mathbf{w})}$  in (4.26). Since (C1)-(C9) are all convex constraints and the objective function in (4.27),  $\langle \mathbf{w}, \mathbf{r} \rangle$ , is affine, the optimisation problem in (4.27) is a convex optimisation problem [47]. Note that  $\mathcal{G}$  must be a bounded set so that the estimated

#### 4.4. Proposed MVWO for MWS Design

rates in (4.26) or (4.27) have meaningful values; otherwise, they will be infinite and meaningless. We prove the boundedness of  $\mathcal{G}$  in the appendix. We refer to  $\mathcal{G}$  as the bounding set of  $\mathcal{F}$ .

### 4.4 Proposed MVWO for MWS Design

By replacing (4.12) with (4.27) as the constraint in (P1), we can then rewrite the optimisation of the utility function in (P1) as

$$\begin{aligned} & \max_{\mathbf{w}, \mathbf{r}, \mathbf{p}, \mathbf{y}, \mathbf{H}} f(\mathbf{r}) , \\ & \text{s.t. } \mathbf{w} \in \mathbb{R}_{>0}^K , \|\mathbf{w}\|_2 = 1 , \\ & \mathbf{r}, \mathbf{p}, \mathbf{y}, \mathbf{H} = \arg \max_{\mathbf{r}', \mathbf{p}', \mathbf{y}', \mathbf{H}'} \langle \mathbf{w}, \mathbf{r}' \rangle, \text{ s.t. (C1)-(C9) .} \end{aligned} \tag{P2}$$

We use the optimal  $\mathbf{w}$  of the above problem to approximate the weights of an optimal MWS. Note that (4.27) is embedded in (P2), resulting in (P2) having a standard form of BLOPs with known iterative solvers [23]. We refer to the embedded (4.27) as the lower-level problem (LLP) of (P2), and (P2) as the upper-level problem (ULP) of the embedded (4.27), respectively. We refer to (P2) as the MVWO since its formulation only uses the mean and variance of users' SNRs.

#### 4.4.1 Issue of Existing Iterative Solvers for the BLOP

We study how to solve (P2) by using iterative solvers. We first briefly explain the process of iterative solvers for BLOPs [23], [24]. Let  $\mathbf{w}^{(i)}$ ,  $i = 1, 2, \dots$ , denote the weights in the  $i$ -th iteration of an iterative solver. In each iteration, the LLP is first solved when the weights in its objective,  $\langle \mathbf{w}, \mathbf{r}' \rangle$ , are  $\mathbf{w}^{(i)}$ . Next, the process calculates the weights in the next iteration,  $\mathbf{w}^{(i+1)}$ . In the existing iterative solvers [23], the calculation of  $\mathbf{w}^{(i+1)}$  is done by solving an additional optimisation problem formulated based on the Lagrangian of the LLP, the utility function and the solution of the LLP (i.e., the value of  $\mathbf{r}$ ,  $\mathbf{p}$ ,  $\mathbf{y}$  and  $\mathbf{H}$  when the weights,  $\mathbf{w}$  in the LLP are  $\mathbf{w}^{(i)}$ ) in the  $i$ -th iteration. Such methods have a high computational complexity. Thus, a

#### 4.5. Proposed Iterative Solver for MVWO

low-complexity method is required to update the weights.

### 4.5 Proposed Iterative Solver for MVWO

In this section, we design an iteration solver to solve the MVWO problem in (P2). The designed solver iterates the weights,  $\mathbf{w}^{(i)}$ , until the rates in the solution of the LLP (i.e., the value of  $\mathbf{r}$  when  $\mathbf{w}^{(i)}$  is the weights in the LLP) maximise the objective of (P2),  $f(\mathbf{r})$ .

#### 4.5.1 Initialisation of the Proposed Iterative Process

To achieve the above, the designed solver first finds the optimal rates,  $\mathbf{r}^*$ , in the bounding set,  $\mathcal{G}$ , that maximise the objective of (P2),  $f(\mathbf{r})$ , which is defined as

$$\mathbf{r}^* \triangleq \arg \max_{\mathbf{r}'} f(\mathbf{r}') , \text{ s.t. } \mathbf{r}' \in \mathcal{G} , \quad (4.28)$$

and the value of  $\mathbf{r}^*$  can be obtained by solving the following problem as

$$\mathbf{r}^*, \mathbf{p}^*, \mathbf{y}^*, \mathbf{H}^* = \arg \max_{\mathbf{r}', \mathbf{p}', \mathbf{y}', \mathbf{H}'} f(\mathbf{r}') , \text{ s.t. (C1)-(C9)} . \quad (4.29)$$

Here, (C1)-(C9) in (4.29) are used to represent  $\mathcal{G}$  in (4.28), as defined in (4.24) and (4.25). Also, (4.29) is a convex optimisation problem because its constraints are convex and its objective is to maximise a concave function,  $f(\cdot)$ . Note that (4.29) is only solved once before the following iterative process starts, where we initialise the weights in the first iteration,  $\mathbf{w}^{(1)}$ , as

$$\mathbf{w}^{(1)} \triangleq \left[ \frac{1}{\sqrt{K}}, \dots, \frac{1}{\sqrt{K}} \right]^T . \quad (4.30)$$

### 4.5.2 Low-Complexity Iterative Updates of Weights

Next, the LLP for the given weights in the  $i$ -th iteration,  $\mathbf{w}^{(i)}$ , is solved as

$$\mathbf{r}^{(i)}, \mathbf{p}^{(i)}, \mathbf{y}^{(i)}, \mathbf{H}^{(i)} = \arg \max_{\mathbf{r}', \mathbf{p}', \mathbf{y}', \mathbf{H}'} \langle \mathbf{w}^{(i)}, \mathbf{r}' \rangle \text{ s.t. (C1)-(C9)}. \quad (4.31)$$

To calculate the weights in the  $i+1$ -th iteration, we first perform an intermediary step to calculate a vector,  $\mathbf{u}^{(i)}$ , that is a linear combination of  $\mathbf{w}^{(i)}$  and  $\mathbf{r}^* - \mathbf{r}^{(i)}$  as

$$\mathbf{u}^{(i)} \triangleq (a^{(i)} + b^{(i)})\mathbf{w}^{(i)} + (\mathbf{r}^* - \mathbf{r}^{(i)}), \quad (4.32)$$

where  $\mathbf{r}^*$  and  $\mathbf{r}^{(i)}$  are from (4.29) and (4.31), respectively.  $a^{(i)}$  and  $b^{(i)}$  in (4.32) are configured as

$$a^{(i)} \triangleq \frac{\|\mathbf{r}^* - \mathbf{r}^{(i)}\|_2^2}{\langle \mathbf{w}^{(i)}, \mathbf{r}^{(i)} - \mathbf{r}^* \rangle}, \quad b^{(i)} \triangleq -\min\{(\mathbf{r}^* - \mathbf{r}^{(i)}) \circlearrowleft \mathbf{w}^{(i)}\}. \quad (4.33)$$

By normalising  $\mathbf{u}^{(i)}$  in (4.32), we obtain the weights in the  $i+1$ -th iteration as

$$\mathbf{w}^{(i+1)} = \frac{\mathbf{u}^{(i)}}{\|\mathbf{u}^{(i)}\|_2}. \quad (4.34)$$

The update of weights in (4.32)-(4.34) has a linear computational complexity that is lower than the polynomial complexity of solving the additional optimisation problem in existing methods [23]. Furthermore, note that the weights in each iteration follow

**Corollary 2.** *If  $\mathbf{w}^{(i)} \in \mathbb{R}_{>0}^K$  and  $\|\mathbf{w}^{(i)}\|_2 = 1$ , then  $\mathbf{w}^{(i+1)} \in \mathbb{R}_{>0}^K$  and  $\|\mathbf{w}^{(i+1)}\|_2 = 1$ .*

*Proof.* The proof is in the appendix. □

Since the weights in the first iteration, as defined in (4.30), satisfy the constraints on weights in (P2), i.e.,  $\mathbf{w}^{(1)} \in \mathbb{R}_{>0}^K$  and  $\|\mathbf{w}^{(1)}\|_2 = 1$ , the weights in all following iterations satisfy those constraints based on Corollary 2, i.e., all iterated weights are feasible in the above process.

## 4.6. Convergence of Proposed Iterative Solver

---

**Algorithm 2** Proposed Iterative Solver for the MVWO

---

```
1: Find  $\mathbf{r}^*$  via solving (4.29).
2: Initialise  $\mathbf{w}^{(1)}$  as (4.30).
3: for  $i = 1, 2, \dots$  do
4:   Construct Problem (4.31) based on  $\mathbf{w}^{(i)}$ .
5:   Find the solution of Problem (4.31) as  $\mathbf{r}^{(i)}$ .
6:   if  $|\langle \mathbf{w}^{(i)}, \mathbf{r}^{(i)} - \mathbf{r}^* \rangle| < \hat{\epsilon}$  then
7:     break and terminate.
8:   else
9:     Calculate  $a^{(i)}$  and  $b^{(i)}$  as (4.33).
10:    Calculate  $\mathbf{u}^{(i)}$  and  $\mathbf{w}^{(i+1)}$  as (4.32) and (4.34).
11:   end if
12: end for
13: return  $\mathbf{w}^{(i)}$ .
```

---

### 4.5.3 Termination Condition of the Proposed Iterative Process

The iterated weights,  $\mathbf{w}^{(i)}$ , are optimal when  $\mathbf{r}^*$  is the optimal rates of the LLP in (4.31) because no other rates,  $\mathbf{r}$ , in  $\mathcal{G}$  can lead to higher  $f(\mathbf{r})$  than  $f(\mathbf{r}^*)$ , as defined in (4.28). We use the difference between the value of the LLP's objective function at  $\mathbf{r}^{(i)}$  and the one at  $\mathbf{r}^*$  to indicate whether  $\mathbf{r}^*$  is the optimal rates of the LLP or not in the  $i$ -th iteration. Such difference is expressed as  $|\langle \mathbf{w}^{(i)}, \mathbf{r}^{(i)} \rangle - \langle \mathbf{w}^{(i)}, \mathbf{r}^* \rangle|$ , whose minimum value is 0 and is achieved only if  $\mathbf{r}^*$  is the optimal rates of the LLP in the  $i$ -th iteration. In practice, if  $|\langle \mathbf{w}^{(i)}, \mathbf{r}^{(i)} \rangle - \langle \mathbf{w}^{(i)}, \mathbf{r}^* \rangle|$  is less than a small positive number,  $\hat{\epsilon}$ , then we stop the iteration and return  $\mathbf{w}^{(i)}$  as the optimal weights of (P2).

Algorithm 2 summarises the process of the proposed iterative solver. The analysis of the convergence of Algorithm 2 is presented in the following section.

## 4.6 Convergence of Proposed Iterative Solver

In this section, we prove that Algorithm 2 converges. Also, we study the computational complexity of the proposed solver.

## 4.6. Convergence of Proposed Iterative Solver

### 4.6.1 Convergence of Algorithm 2

We define the convergence of Algorithm 2 as that  $|\langle \mathbf{w}^{(i)}, \mathbf{r}^{(i)} - \mathbf{r}^* \rangle|$  converges to 0, i.e., for any positive number,  $\hat{\epsilon}$ ,  $|\langle \mathbf{w}^{(i)}, \mathbf{r}^{(i)} - \mathbf{r}^* \rangle|$  is less than  $\hat{\epsilon}$  after a number of iterations. Note that the above convergence defined for Algorithm 2 also implies that  $\mathbf{w}^{(i)}$  converges to the optimal weights that maximise  $f(\mathbf{r})$  in (P2), as discussed in Section 4.5.3. To prove the convergence of Algorithm 2, we study the relationship between the iterated weights,  $\mathbf{w}^{(i)}$  and the optimal weights of (P2). We denote the set of all optimal weights of (P2) as  $\mathcal{W}$ . Considering an arbitrary vector of weights  $\mathbf{w}^*$  from  $\mathcal{W}$ , the inner product between  $\mathbf{w}^{(i)}$  and  $\mathbf{w}^*$  has following properties.

**Lemma 1.** 1) If Algorithm 2 is not terminating in the  $i$ -th iteration, then  $\forall \mathbf{w}^* \in \mathcal{W}$ ,

$$\frac{\langle \mathbf{w}^{(i+1)}, \mathbf{w}^* \rangle}{\langle \mathbf{w}^{(i)}, \mathbf{w}^* \rangle} > \left[ 1 - \frac{(\langle \mathbf{w}^{(i)}, \mathbf{r}^{(i)} - \mathbf{r}^* \rangle)^2}{2\hat{R}^2} \right]^{-\frac{1}{2}} > 1 . \quad (4.35)$$

Here,  $\hat{R}$  in (4.35) is a finite number representing the maximum Euclidean distance between any two vectors in  $\mathcal{G}$ , defined as  $\hat{R} \triangleq \max_{\mathbf{r}^a, \mathbf{r}^b \in \mathcal{G}} \|\mathbf{r}^a - \mathbf{r}^b\|_2$ . 2)  $\frac{1}{\sqrt{K}} \leq \langle \mathbf{w}^{(i)}, \mathbf{w}^* \rangle \leq 1$ ,  $\forall i$  and  $\forall \mathbf{w}^* \in \mathcal{W}$ .

*Proof.* The proof can be found in the appendix. □

Lemma 1 shows that  $\langle \mathbf{w}^{(i)}, \mathbf{w}^* \rangle$  is strictly increasing as  $i$  increases. Its supremum is defined as

$$o(\mathbf{w}^*) \triangleq \sup_i \langle \mathbf{w}^{(i)}, \mathbf{w}^* \rangle , \quad \forall \mathbf{w}^* \in \mathcal{W} , \quad (4.36)$$

where  $o(\mathbf{w}^*)$  is a finite number less than or equal to 1 because  $\langle \mathbf{w}^{(i)}, \mathbf{w}^* \rangle$  is less than or equal to 1,  $\forall i$ , as shown in Lemma 1. We consider following inequalities in the  $i$ -th iteration that use the definition of  $o(\mathbf{w}^*)$  in (4.36),  $\forall \mathbf{w}^* \in \mathcal{W}$ ,

$$\langle \mathbf{w}^{(i+1)}, \mathbf{w}^* \rangle \leq o(\mathbf{w}^*) \Rightarrow \frac{\langle \mathbf{w}^{(i+1)}, \mathbf{w}^* \rangle}{\langle \mathbf{w}^{(i)}, \mathbf{w}^* \rangle} \leq \frac{o(\mathbf{w}^*)}{\langle \mathbf{w}^{(i)}, \mathbf{w}^* \rangle} . \quad (4.37)$$



#### 4.6. Convergence of Proposed Iterative Solver

By applying (4.35) of Lemma 1 to (4.37), we obtain

$$1 < \underbrace{\left[1 - \frac{(\langle \mathbf{w}^{(i)}, \mathbf{r}^{(i)} - \mathbf{r}^* \rangle)^2}{2\hat{R}^2}\right]^{-\frac{1}{2}}}_{(a)} < \frac{o(\mathbf{w}^*)}{\langle \mathbf{w}^{(i)}, \mathbf{w}^* \rangle}, \forall \mathbf{w}^* \in \mathcal{W}. \quad (4.38)$$

Note that  $\langle \mathbf{w}^{(i)}, \mathbf{w}^* \rangle$  converges to  $o(\mathbf{w}^*)$  based on the monotone convergence theorem of a sequence, which implies  $\frac{o(\mathbf{w}^*)}{\langle \mathbf{w}^{(i)}, \mathbf{w}^* \rangle}$  converges to 1. Thus, the value of (a) in (4.38) is between 1 and a real number converging to 1. This implies that the value of (a) in (4.38) also converges to 1 as

$$\left[1 - \frac{(\langle \mathbf{w}^{(i)}, \mathbf{r}^{(i)} - \mathbf{r}^* \rangle)^2}{2\hat{R}^2}\right]^{-\frac{1}{2}} \rightarrow 1, \Rightarrow |\langle \mathbf{w}^{(i)}, \mathbf{r}^{(i)} - \mathbf{r}^* \rangle| \rightarrow 0, \quad (4.39)$$

which proves the convergence of Algorithm 2.

#### 4.6.2 Computational Complexity of the Proposed Iterative Solver

Next, we study the computational complexity of Algorithm 2, specifically in terms of the number of iterations required for Algorithm 2 before it converges. Assuming that Algorithm 2 terminates at the  $I$ -th iteration, i.e.,  $|\langle \mathbf{w}^{(i)}, \mathbf{r}^{(i)} - \mathbf{r}^* \rangle|$  is less than  $\hat{\epsilon}$  at the  $I$ -th iteration (for simplicity, we assume that  $I$  is greater than 1), we obtain  $|\langle \mathbf{w}^{(i)}, \mathbf{r}^{(i)} - \mathbf{r}^* \rangle| \geq \hat{\epsilon}, \forall i$ , which can be applied into (4.35) as

$$\frac{\langle \mathbf{w}^{(i+1)}, \mathbf{w}^* \rangle}{\langle \mathbf{w}^{(i)}, \mathbf{w}^* \rangle} > \left[1 - \frac{\hat{\epsilon}^2}{2\hat{R}^2}\right]^{-\frac{1}{2}}, \forall i = 1, \dots, I-1. \quad (4.40)$$

By multiplying both sides of the above inequalities for  $i = 1, \dots, I-1$ , we obtain

$$\frac{\langle \mathbf{w}^{(I)}, \mathbf{w}^* \rangle}{\langle \mathbf{w}^{(1)}, \mathbf{w}^* \rangle} > \left[1 - \frac{\hat{\epsilon}^2}{2\hat{R}^2}\right]^{-\frac{1}{2}(I-1)}. \quad (4.41)$$

#### 4.7. Online MVWO Architecture for Varying Mean and Variance of SNRs

Note that  $\langle \mathbf{w}^{(1)}, \mathbf{w}^* \rangle$  is greater than or equal to  $\frac{1}{\sqrt{K}}$  and  $\langle \mathbf{w}^{(I)}, \mathbf{w}^* \rangle$  is less than or equal to 1 according to Lemma 1. By applying the above facts into (4.41), we obtain

$$\left[1 - \frac{\hat{\epsilon}^2}{2\hat{R}^2}\right]^{-\frac{1}{2}(I-1)} < \frac{\langle \mathbf{w}^{(I)}, \mathbf{w}^* \rangle}{\langle \mathbf{w}^{(1)}, \mathbf{w}^* \rangle} < \sqrt{K}, \Rightarrow I < \frac{\log K}{-\log \left[1 - \frac{\hat{\epsilon}^2}{2\hat{R}^2}\right]} + 1 \approx \frac{2\hat{R}^2 \log K}{\hat{\epsilon}^2}. \quad (4.42)$$

Note that  $\hat{\epsilon}$  is a small positive constant and  $\hat{R}$  is roughly in proportion to  $\sqrt{K}$ . Thus, the number of iterations of Algorithm 2 can be written as  $O(\frac{1}{\hat{\epsilon}^2} K \log K)$  (or  $O(K \log K)$  if we assume that  $\hat{\epsilon}$  is a constant). We note that this computational complexity describes the number of iterations required for updating the weights in Algorithm 2. Meanwhile, the computational complexity of solving the LLP in (4.31) in each iteration depends on the specific implementation of convex optimisation solver that typically has a polynomial complexity [104]. Also, the update of weights in each iteration has a linear computational complexity, as mentioned in Section 4.5.2.

## 4.7 Online MVWO Architecture for Varying Mean and Variance of SNRs

In this section, we propose an online architecture to apply our proposed MVWO method in networks with non-stationary wireless channels, e.g., users have mobility and distances between users and BSs change. In such cases, the statistics of SNRs, i.e., the mean and variance of users' SNRs, change over time. As shown in Fig. 4.2, the online MVWO architecture includes the scheduler at the BS and an edge server.

In each time slot, the scheduler at the BS observes the channel state,  $\mathbf{s}(t)$ , and generates a binary user scheduling action,  $\mathbf{x}(t)$  according to the MWS defined in (4.11). In real-world networks, e.g., 5G New Radio networks, SNRs can be measured based on the CSI reference signals transmitted with data signals in the wireless channel [74] and binary user scheduling actions can be mapped into radio resource configurations, including modulation-and-coding schemes and resource block allocations in

#### 4.7. Online MVWO Architecture for Varying Mean and Variance of SNRs

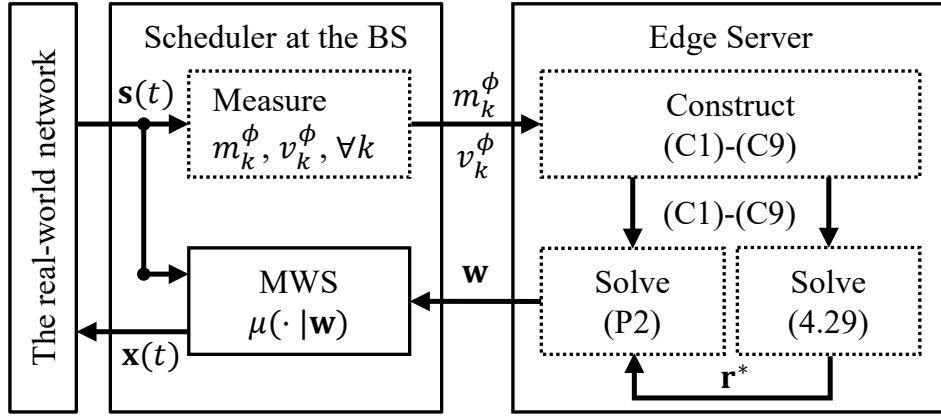


Figure 4.2: The proposed online MVWO architecture.

each transmission as shown in [J1]. Meanwhile, the scheduler continuously estimates the mean and variance of users' SNRs through a moving average over the observed values of the SNRs in the past slots in two following steps,

$$m_k^\phi \leftarrow \frac{1}{\beta} \sum_{\tau=0}^{\beta-1} \phi_k(t - \tau), \quad v_k^\phi \leftarrow \frac{1}{\beta} \sum_{\tau=0}^{\beta-1} (\phi_k(t - \tau))^2 - (m_k^\phi)^2, \quad \forall k, \quad (4.43)$$

where  $\beta$  is the number of the past time slots used in averaging. The measured mean and variance of the SNRs are uploaded to the edge server that calculates the weights of the MWS.

The edge server first constructs the constraints in (C1)-(C9) and constantly updates the constraints' parameters based on the latest sent mean and variance of the SNRs from the scheduler. Then, the convex optimisation problem in (4.29) is constructed and solved to find  $\mathbf{r}^*$  defined in (4.28) based on (C1)-(C9). Next, the iterations in lines 3-12 of Algorithm 2 are executed to solve (P2) based on (C1)-(C9) and  $\mathbf{r}^*$ , whose return value,  $\mathbf{w}$ , is sent to the scheduler as the weights of the MWS. The above process in the edge server is then repeated until the BS terminates.

## 4.8 Evaluation of Proposed Methods

In this section, we provide the simulation results that evaluate our proposed methods.

### 4.8.1 Simulation Configurations

We set  $\Delta_0 = 1$  (second) and  $B = 1$  (Hertz) for simplicity in simulations as they linearly scale the average rates while not affecting the performance of our methods. We vary the number of users,  $K$ , for different cases. Unless specifically stated, each user's mean of the SNRs,  $m_k^\phi$ ,  $k = 1, \dots, K$ , (i.e., the large-scale fading gain) in decibel (dB) is normally distributed with the mean of 10 dB and the standard deviation of 5 dB for different episodes [76], [105], and it remains constant within one episode. The variance of the SNRs depends on the small-scale fading gains of users, which are i.i.d. in each slot and follow the same normalised Rician distributions as the ratio of the average power in the line-of-sight path to that in the non-line-of-sight paths of 10 dB [76], [105]. With the above configurations, the SNRs of all users have the variance of  $0.17(m_k^\phi)^2$  or 4.00 in decimal or in decibel representation, respectively, where  $k = 1, \dots, K$ .

### 4.8.2 Other OS Approaches Compared in Simulation

#### MWS using no prior knowledge of statistical CSI

We compare our MVWO method with the MWS approaches in [19], [20] that can find the optimal MWS's weights to maximise the studied utility function, as defined in (4.10). Specifically, the weights in these approaches are tuned in every slot as

$$\lambda_k \leftarrow \left(1 - \frac{1}{\gamma}\right)\lambda_k + \frac{1}{\gamma}x_k(t) \log_2(1 + \phi_k(t)), \quad w_k \leftarrow \frac{1}{\lambda_k}, \quad \forall k, \quad (4.44)$$

where  $\lambda_k$  denotes an exponential average of the scheduled instantaneous bit rate of the  $k$ -th user (its initial value is set to a small positive number, e.g.,  $10^{-5}$ , to avoid

#### 4.8. Evaluation of Proposed Methods

division by zero).  $\gamma$  in (4.44) denotes the size of the exponential average time window, e.g, 100, 1000 or 10000. Note that the studied utility function is maximised by the above methods when  $\gamma$  approaches infinity and the MWS's weights are tuned after sufficient time [16]. Since these approaches use no statistical CSI, we refer to them as statistics-unaware weight optimisation (SUWO). We denote the weights tuned after  $\tilde{T}$  slots by using the SUWO methods with  $\gamma$  as  $\mathbf{w}_{\tilde{T}}^{\text{SUWO}\sim\gamma}$ , where  $\tilde{T}$  is varied for different cases and  $\mathbf{w}_{\tilde{T}}^{\text{SUWO}\sim\gamma}$  is normalised after tuning. We denote the average rates achieved by the MWS,  $\mu(\cdot|\mathbf{w}_{\tilde{T}}^{\text{SUWO}\sim\gamma})$ , as  $\mathbf{r}^{\sim\mu(\cdot|\mathbf{w}_{\tilde{T}}^{\text{SUWO}\sim\gamma})}$ .

#### MWS using prior knowledge of the mean of CSI

Also, a heuristic MWS will be compared, whose weights are designed based on only the mean of CSI. The studied utility function is a throughput fairness criterion. We can provide fair scheduling decisions for users by setting the weight of the  $k$ -th user as the inverse of the average spectrum efficiency, i.e.,

$$w_k \leftarrow \frac{1}{\mathbb{E}[\log_2(1 + \phi_k(t))]}, \quad \forall k, \quad (4.45)$$

which prevents the MWS from starving users with low spectrum efficiency. We refer to the MWS in (4.45) as a heuristic fairness scheduler (HFS). We denote weights in the HFS calculated based on the  $\tilde{T}$ -slot averaged spectrum efficiency as  $\mathbf{w}_{\tilde{T}}^{\text{HFS}}$ .

#### MDP-based OS using no prior knowledge of statistical CSI

Additionally, we will compare our method with the MDP-based OS optimised by DRL that uses no statistical CSI. The reward signal for the studied utility function in every slot is designed in [92] as

$$\delta_k(t) = x_k(t) \log_2(1 + \phi_k(t)) \cdot \left[ \frac{1}{t} \sum_{\tau=1}^t x_k(\tau) \log_2(1 + \phi_k(\tau)) \right]^{-1}, \quad \forall k. \quad (4.46)$$

The state and the action in the MDP are the channel state and the user scheduling actions in every slot,  $\mathbf{s}(t)$  and  $\mathbf{x}(t)$ , defined in (4.4) and (4.3), respectively. We use

#### 4.8. Evaluation of Proposed Methods

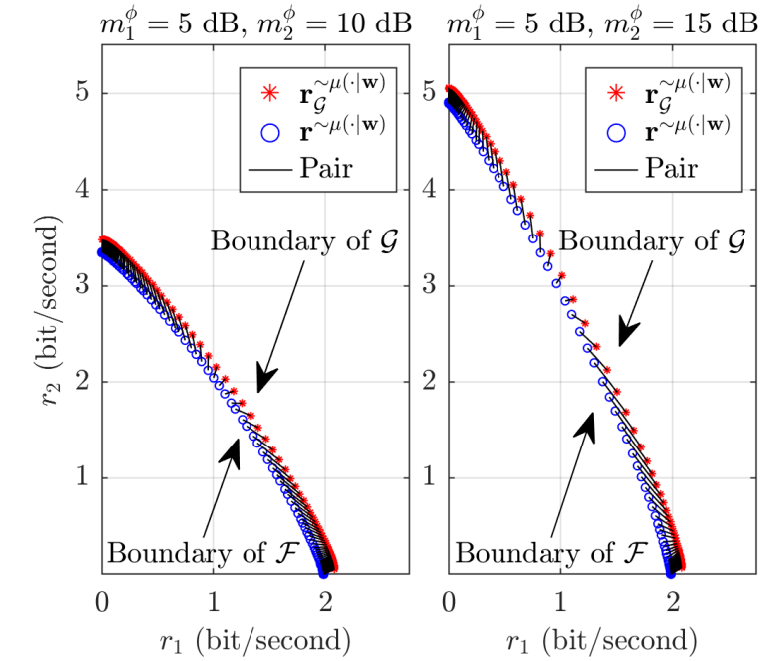
the actor-critic DRL algorithm [J1], [92] to train the NN,  $\pi(\cdot|\theta)$ , as the MDP-based OS, where  $\theta$  are the parameters of the NN. We denote the NN trained after  $\tilde{T}$  slots as  $\pi(\cdot|\theta_{\tilde{T}}^{\text{DRL}})$ , and its initial values are randomised.

##### 4.8.3 Performance of Proposed Rate Estimation Method

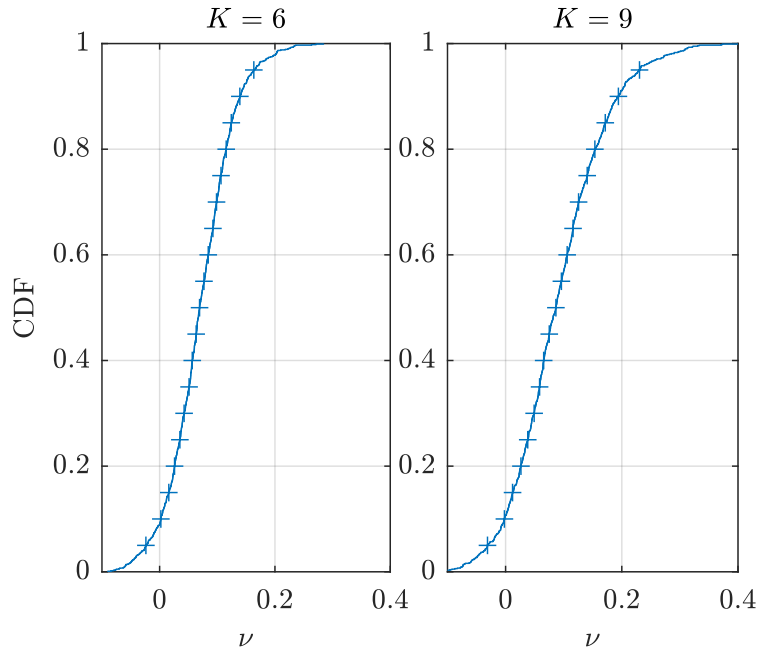
We first compare the estimated value of the average rates scheduled by  $\mu(\cdot|\mathbf{w})$  in (4.26),  $\mathbf{r}_{\mathcal{G}}^{\sim\mu(\cdot|\mathbf{w})}$ , and their measured value,  $\mathbf{r}^{\sim\mu(\cdot|\mathbf{w})}$ , for two users, i.e.,  $K = 2$ . Two users' weights are varied as  $w_1 = \sin(0.005i \cdot \frac{\pi}{2})$  and  $w_2 = \cos(0.005i \cdot \frac{\pi}{2})$  for  $i = 1, \dots, 199$ , which are all feasible weights, i.e.,  $\mathbf{w} \in \mathbb{R}_{>0}^K$  and  $\|\mathbf{w}\|_2 = 1$  for all  $i$ . In each case of  $\mathbf{w}$ ,  $\mathbf{r}_{\mathcal{G}}^{\sim\mu(\cdot|\mathbf{w})}$  is calculated by solving the rate estimation problem in (4.26), and  $\mathbf{r}^{\sim\mu(\cdot|\mathbf{w})}$  is measured by averaging the scheduled instantaneous rates in one episode with  $10^5$  slots. In Fig. 4.3a, the estimated average rates scheduled by the MWS and their actual value measured from the simulation with the same  $\mathbf{w}$  (with legends “ $\mathbf{r}_{\mathcal{G}}^{\sim\mu(\cdot|\mathbf{w})}$ ” and “ $\mathbf{r}^{\sim\mu(\cdot|\mathbf{w})}$ ”, respectively) are connected with a line (with legend “Pair”). Two users have the mean of the SNRs as  $m_1^\phi = 3.16$  and  $m_2^\phi = 10$  in decimal format (or  $m_1^\phi = 5$  and  $m_2^\phi = 10$  in dB), or  $m_1^\phi = 5$  dB and  $m_2^\phi = 15$  dB in Fig. 4.3a. The variance of the SNRs follows the configuration in Section 4.8.1. The results indicate that the estimated and measured average rates at given weights are close to each other. Also, note that they form two boundaries of  $\mathcal{F}$  and  $\mathcal{G}$ , respectively, due to the structure of (4.16) and (4.26) (e.g., maximisation of the weighted sum of a vector that belongs to a convex set) according to [106]. Since two boundaries show the same shape and are close to each other, this implies that  $\mathcal{G}$  is a close approximation of  $\mathcal{F}$ .

Next, we use the optimal rates in the bounding set,  $\mathbf{r}^*$  defined in (4.28), to estimate the average rates of MWSs,  $\mathbf{r}^{\sim\mu(\cdot|\mathbf{w}_{\tilde{T}}^{\text{SUWO}\sim\gamma})}$ , which is designed by the SUWO methods [19], [20]. We vary the number of users,  $K$ , as 6 and 9 and set  $\gamma = 10000$  and  $\tilde{T} = 1000K$ , which is sufficiently large for the SUWO methods to find the weights that achieve optimal rates. For each case of  $K$ , we run 1000 episodes where users' SNRs are configured as stated in Section 4.8.1 and  $\mathbf{r}^{\sim\mu(\cdot|\mathbf{w}_{\tilde{T}}^{\text{SUWO}\sim\gamma})}$  is measured in  $10^5$  slots of each

#### 4.8. Evaluation of Proposed Methods



(a) Sweeping  $\mathbf{w}$  when  $K = 2$ .



(b)  $\mathbf{w} = \mathbf{w}_{\hat{T}}^{\text{SUWO} \sim \gamma}$  when  $K = 6$  or  $9$ .

Figure 4.3: The average rates achieved by  $\mu(\cdot|\mathbf{w})$  and their estimated values in (4.26) for different  $\mathbf{w}$ .

episode. Note that the  $k$ -th element of  $\mathbf{r}^*$  and  $\mathbf{r}^{\sim\mu(\cdot|\mathbf{w}_{\hat{T}}^{\text{SUWO} \sim \gamma})}$ ,  $r_k^*$  and  $r_k^{\sim\mu(\cdot|\mathbf{w}_{\hat{T}}^{\text{SUWO} \sim \gamma})}$ , are the estimated value and the measured value of the  $k$ -th user's average rate achieved

#### 4.8. Evaluation of Proposed Methods

by  $\mu(\cdot|\mathbf{w}_{\tilde{T}}^{\text{SUWO}\sim\gamma})$ , respectively,  $k = 1, \dots, K$ , and all users are equivalent to each other. Thus, we only compare the first user's estimated and measured average rate, e.g.,  $r_1^*$  and  $r_1^{\sim\mu(\cdot|\mathbf{w}_{\tilde{T}}^{\text{SUWO}\sim\gamma})}$ , in terms of the ratio of the difference in the estimated and measured values to the measured value as  $\nu \triangleq (r_1^* - r_1^{\sim\mu(\cdot|\mathbf{w}_{\tilde{T}}^{\text{SUWO}\sim\gamma})})/r_1^{\sim\mu(\cdot|\mathbf{w}_{\tilde{T}}^{\text{SUWO}\sim\gamma})}$ , whose cumulative distribution function (CDF) is shown in Fig. 4.3b. The results indicate that over 90% of the estimated values of the average rates of MWSs are overshoot, or in other words, are bigger than the measured ones, e.g.,  $\nu > 0$ . The results also indicate that the estimated average rates differ from the measured ones by approximately 0 ~ 20%, which implies that the estimated average rates of MWSs in the proposed method are close to their measured values.

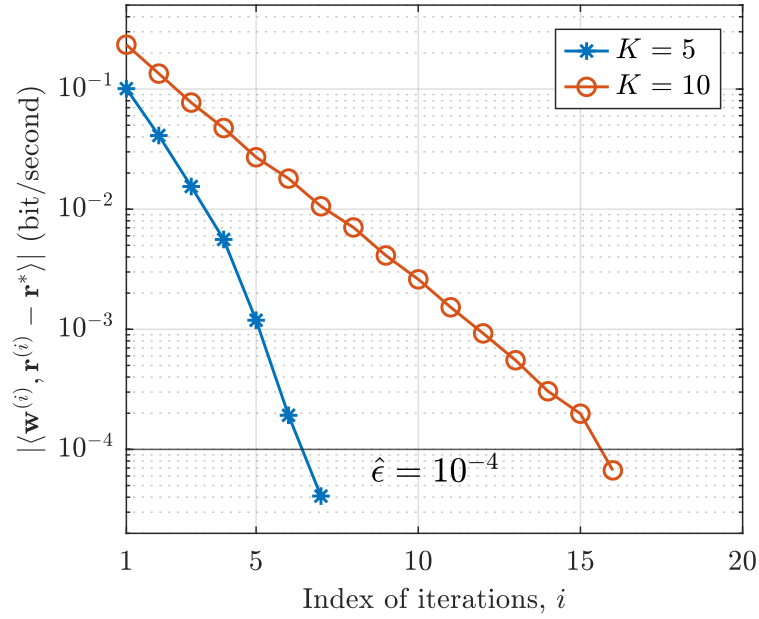
#### 4.8.4 Evaluation on the Convergence of Algorithm 2

Fig. 4.4a shows the value of  $|\langle \mathbf{w}^{(i)}, \mathbf{r}^{(i)} - \mathbf{r}^* \rangle|$  in each iteration of Algorithm 2, where the number of users are 5 and 10 (with legends “ $K = 5$ ” and “ $K = 10$ ”, respectively). The  $k$ -th user's mean of the SNRs is configured as  $m_k^\phi = k + 5$  dB,  $k = 1, \dots, K$ , and users' variance of the SNRs follows the same configuration as explained in Section 4.8.1. We set  $\hat{\epsilon} = 10^{-4}$ . With the above configurations, Algorithm 2 converges in 7 and 16 iterations, i.e.,  $I = 7$  and 16, for  $K = 5$  and 10, respectively, as shown in Fig. 4.4a. This validates the proof in Section 4.6.1.

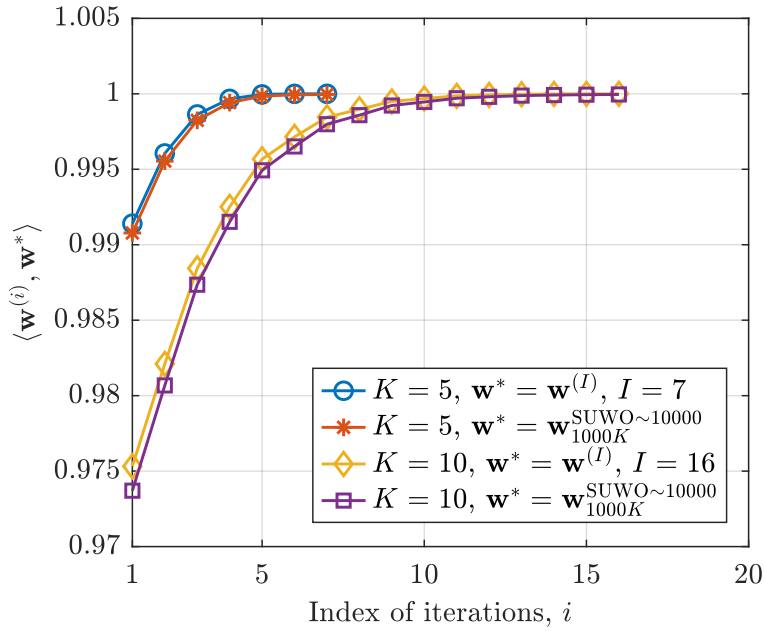
Additionally, we validate the monotone convergence of the sequence,  $\langle \mathbf{w}^{(i)}, \mathbf{w}^* \rangle$ , in Fig. 4.4b. Specifically, we show the values of this sequence when we take the weights in the last iteration of the solver,  $\mathbf{w}^{(I)}$ , as the optimal weights,  $\mathbf{w}^*$  (with legend “ $\mathbf{w}^* = \mathbf{w}^{(I)}$ ”), where  $I$  is 7 and 16 for  $K = 5$  and 10, respectively, as mentioned before. We also show the values of the sequence when the weights optimised by the SUWO methods [19], [20],  $\mathbf{w}_{\tilde{T}}^{\text{SUWO}\sim\gamma}$ , are considered as the optimal weights,  $\mathbf{w}^*$ , (with legend “ $\mathbf{w}^* = \mathbf{w}_{\tilde{T}}^{\text{SUWO}\sim\gamma}$ ”), where  $\gamma$  and  $\tilde{T}$  are set to sufficiently large values as 10000 and  $1000K$ , respectively. The results indicate that the values of  $\langle \mathbf{w}^{(i)}, \mathbf{w}^* \rangle$  monotonically increase to 1 during iterations, which is consistent with the proof in Lemma 1. This also implies that the Euclidean distance between  $\mathbf{w}^{(i)}$  and  $\mathbf{w}_{\tilde{T}}^{\text{SUWO}\sim\gamma}$



#### 4.8. Evaluation of Proposed Methods



(a) The convergence of  $|\langle \mathbf{w}^{(i)}, \mathbf{r}^{(i)} - \mathbf{r}^* \rangle|$ .



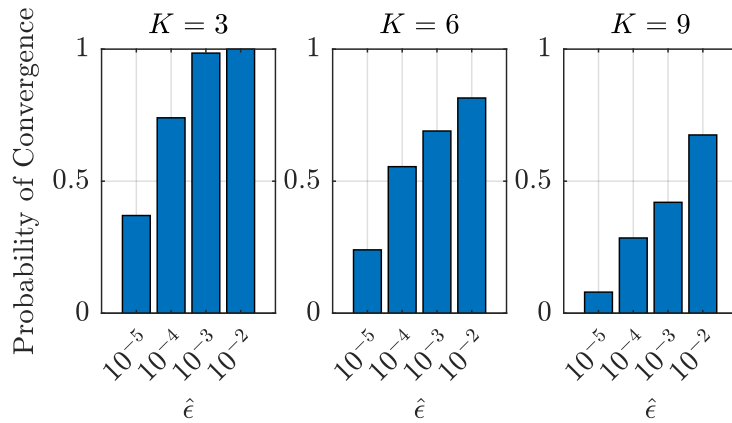
(b) The monotone convergence of  $\langle \mathbf{w}^{(i)}, \mathbf{w}^* \rangle$ .

Figure 4.4: Evaluation of the convergence of Algorithm 2 when  $K = 5$  or  $10$  and  $\hat{\epsilon} = 10^{-4}$ .

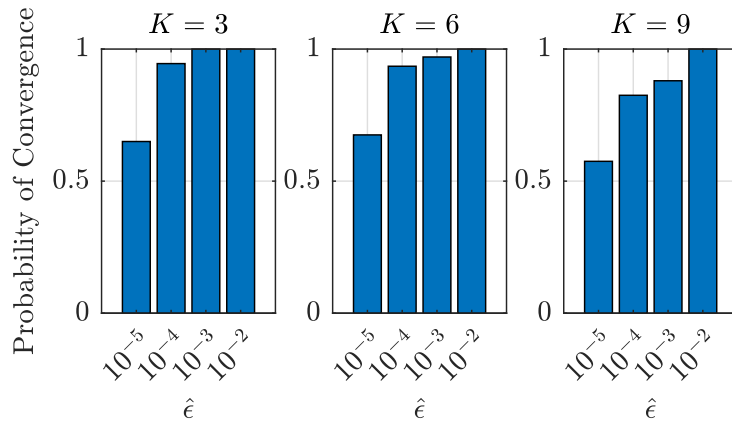
decreases to 0 because  $\|\mathbf{w}^{(i)} - \mathbf{w}^*\|_2^2 = 2 - 2\langle \mathbf{w}^{(i)}, \mathbf{w}^* \rangle$ .

Next, we measure the probability that the convergence of Algorithm 2 occurs within

#### 4.8. Evaluation of Proposed Methods



(a) Occurrence of convergence in 20 iterations.

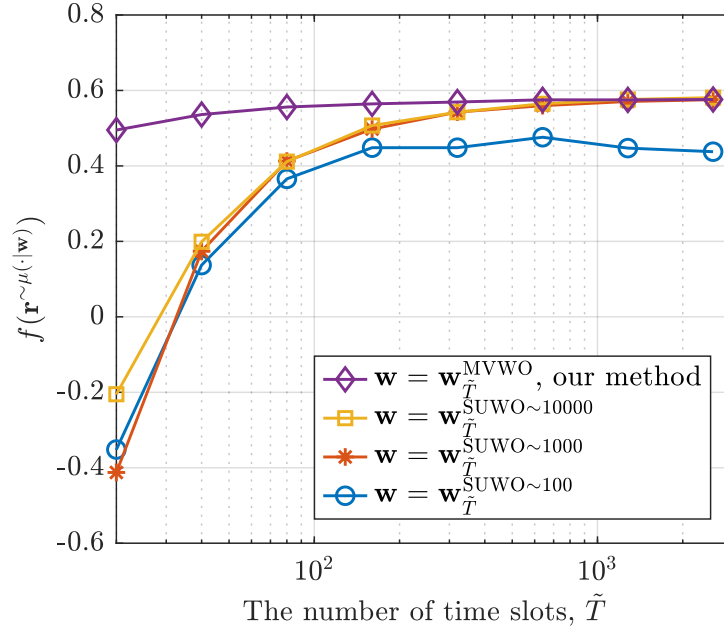


(b) Occurrence of convergence in 50 iterations.

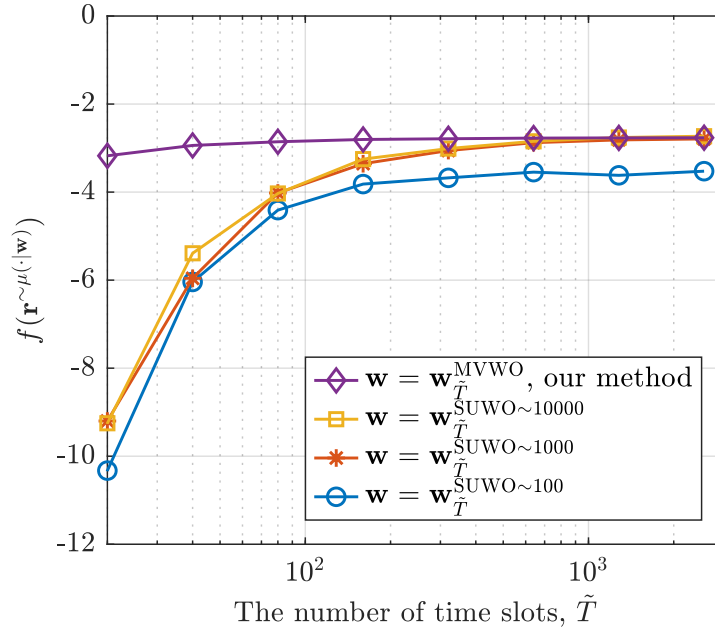
Figure 4.5: The probability that the convergence of Algorithm 2 occurs in given iterations.

a given number of iterations. We vary the number of users,  $K$ , as 3, 6 and 9. For each case of  $K$ , we run Algorithm 2 for 200 times where the mean and variance of users' SNRs in each run are randomised, as explained in Section 4.8.1. Figs. 4.5a and 4.5b show the probability that Algorithm 2 converges in 20 and 50 iterations, respectively. The results indicate that the algorithm is less likely to converge when  $K$  is larger or  $\hat{\epsilon}$  is smaller. When the allowed number of iterations increases from 20 to 50, the probability of convergence increases significantly. The algorithm converges approximately 90 ~ 100% in 50 iterations when  $\hat{\epsilon}$  is large. The above observation complies with the computational complexity analysis in Section 4.6.2.

### 4.8.5 Performance and Time Complexity of MVWO



(a)  $K = 3$ .

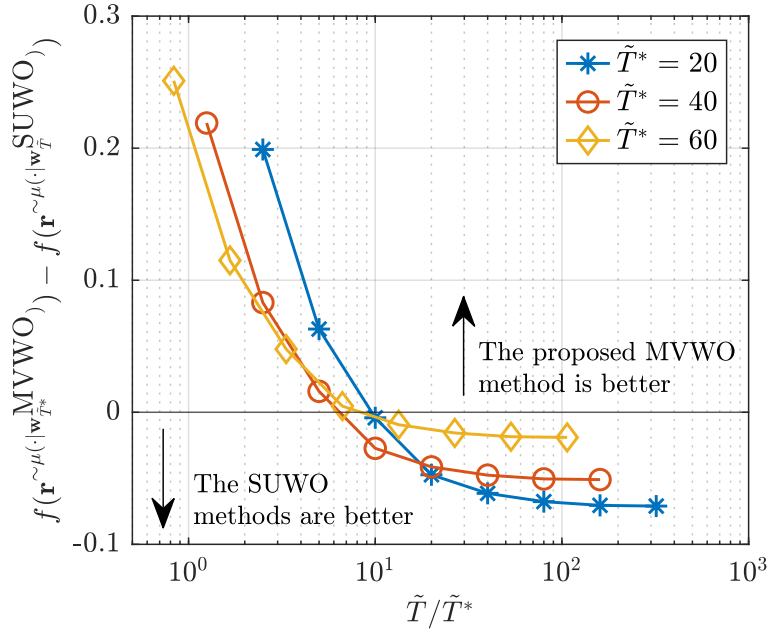


(b)  $K = 6$ .

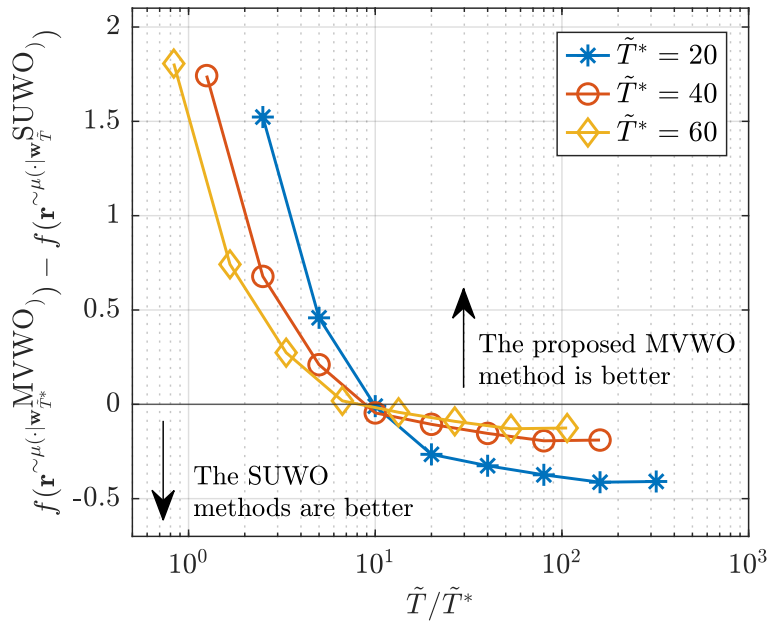
Figure 4.6: Values of the utility function achieved by MWSs optimised by the proposed MVWO method and the SUWO methods when  $\tilde{T}$  time slots are used.

Next, we compare the time complexity (i.e., the number of time slots required) to

#### 4.8. Evaluation of Proposed Methods



(a)  $K = 3$ .



(b)  $K = 6$ .

Figure 4.7: Difference in the utility function achieved by MWSs optimised by the proposed MVWO method and the SUWO methods when  $\tilde{T}^*$  and  $\tilde{T}$  time slots are used, respectively.

optimise the weights in the proposed MVWO method and the SUWO methods [19], [20]. We denote the optimal weights found by solving the MVWO in (P2) based on the

#### 4.8. Evaluation of Proposed Methods

mean and variance of the SNRs, estimated with  $\tilde{T}$  slots as  $\mathbf{w}_{\tilde{T}}^{\text{MVWO}}$ . Fig. 4.6 illustrates the difference in the value of the utility function,  $f(\mathbf{r}^{\sim\mu(\cdot|\mathbf{w})})$ , achieved by different MWSs,  $\mu(\cdot|\mathbf{w})$ , where the weights,  $\mathbf{w}$ , are found either by the SUWO methods [19], [20] and the proposed MVWO method (with legends “ $\mathbf{w} = \mathbf{w}_{\tilde{T}}^{\text{SUWO}\sim\gamma}$ ” and “ $\mathbf{w} = \mathbf{w}_{\tilde{T}}^{\text{MVWO}}$ , our method”, respectively). Here,  $\gamma$  are varied as 100, 1000 and 10000 in the SUWO methods.  $\hat{\epsilon}$  is set to  $10^{-4}$  in the proposed MVWO method. The number of users,  $K$ , is set as 3 and 6 in Figs. 4.6a and 4.6b. Each point in Fig. 4.6 is plotted based on the average value of  $f(\mathbf{r}^{\sim\mu(\cdot|\mathbf{w})})$  in 100 episodes. In each episode, we configure the mean and variance of users’ SNRs as explained in Section 4.8.1. The values of  $\mathbf{r}^{\sim\mu(\cdot|\mathbf{w})}$  are averaged over  $10^5$  slots for given weights in each episode. The results in Fig. 4.6 indicate that more slots spent in the estimation of the mean and variance of the SNRs help improve the performance of the weights found by the proposed MVWO method. This is because the estimated mean and variance are more accurate when more slots are used. Also, we observe that the performance of the proposed MVWO method reaches the highest value when it uses approximately 80 slots, while the performance of the SUWO methods reaches the same value in approximately 320  $\sim$  640 slots. This indicates that the proposed MVWO method costs 4  $\sim$  8 times fewer system time slots to find the optimal weights. Our method finds optimal weights because the proposed rate approximation method closely estimates the feasible rate region and the average rates for given weights. This estimation accurately represents the system’s behaviours and MWSs’ performance; consequently, no online weight adjustment is required.

We keep the same configuration as the above and fix the number of time slots used to estimate the mean and variance of SNRs in our MVWO method as  $\tilde{T}^*$ , while varying the number of time slots spent in weight tuning in the SUWO methods [19], [20] as  $\tilde{T}$ , where  $\gamma$  is set to 1000. We measure the difference between the averaged value of the utility function for various ratios of  $\tilde{T}$  to  $\tilde{T}^*$  when  $\tilde{T}^*$  is 20, 40 and 60. Figs. 4.7a and 4.7b indicate that the proposed MVWO method performs better than the existing SUWO methods when the ratio  $\tilde{T}/\tilde{T}^*$  is less than 10 and otherwise when the ratio is larger than 10. This implies that our methods spend 10 times fewer time slots than

#### 4.8. Evaluation of Proposed Methods

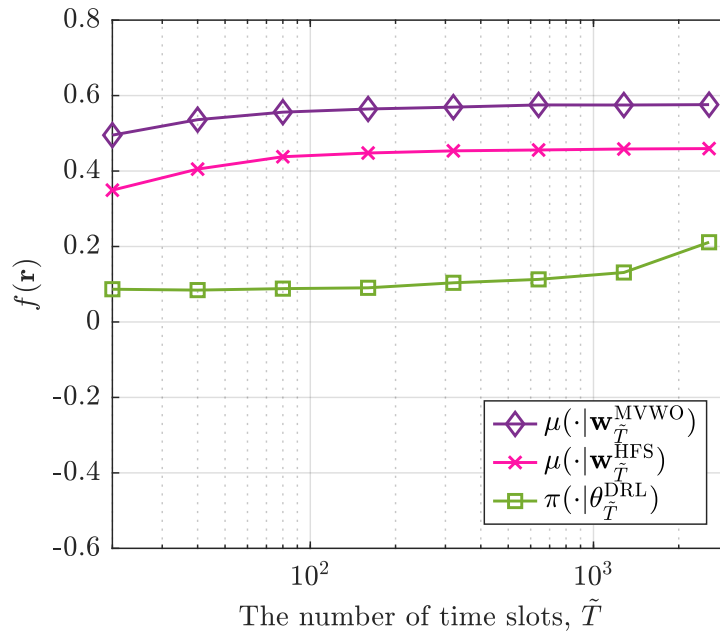


Figure 4.8: Values of the utility function achieved by the MWS optimised by our MVWO method, the HFS in (4.45), and the MDP-based OS optimised by DRL [92] when  $K = 3$ .

the SUWO methods to reach the same performance, and our method’s performance is better when the same number of time slots are used (i.e., when  $\tilde{T}/\tilde{T}^* = 1$ ).

Overall, the simulation results in Fig. 4.6 and Fig. 4.7 indicate that our method has a much lower time complexity than existing SUWO methods. This is because our method directly uses the measured statistical CSI, while it does not depend on user selection decisions of the MWS. In contrast, the existing SUWO methods require a time average of scheduled instantaneous bit rates, as shown in (4.44), which converges only after each user is scheduled sufficient times.

In Fig. 4.8, we further use the same configuration when  $K = 3$  and compare the values of the utility function achieved by the MWS optimised by our MVWO method (with legend “ $\mu(\cdot | \mathbf{w}_{\tilde{T}}^{\text{MVWO}})$ ”), the HFS in (4.45) (with legend “ $\mu(\cdot | \mathbf{w}_{\tilde{T}}^{\text{HFS}})$ ”), and the MDP-based OS optimised by the DRL method [92] (with legend “ $\pi(\cdot | \theta_{\tilde{T}}^{\text{DRL}})$ ”). The results show that the HFS has a close convergence speed to our method. This is because both methods configure their weights based on the estimation of the statistical CSI.

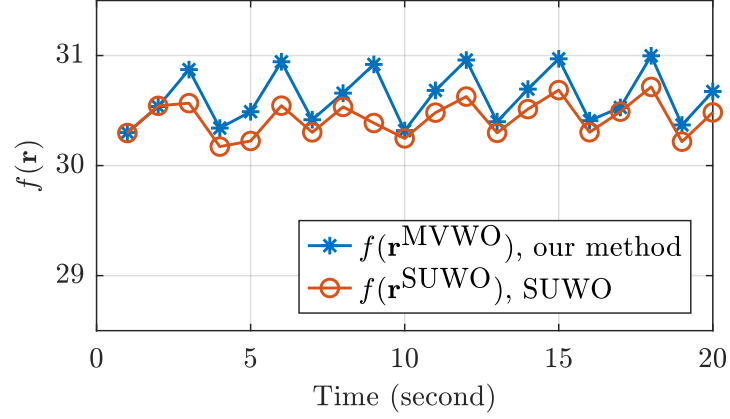
#### 4.8. Evaluation of Proposed Methods

However, the HFS performs worse than our method when both converge. This is because our MVWO method exploits the correlation between scheduling actions and channel states based on their first and second moments, which cannot be done by the HFS (which only uses the first moment of the channel's statistics). The results also show that the DRL method has much slower convergence and performs worse than ours. This is because the NN in the DRL method contains significantly more parameters than MWSs, which are difficult to train within a short time (e.g., within 1000 time slots). This observation on the time complexity of DRL is consistent with [92].

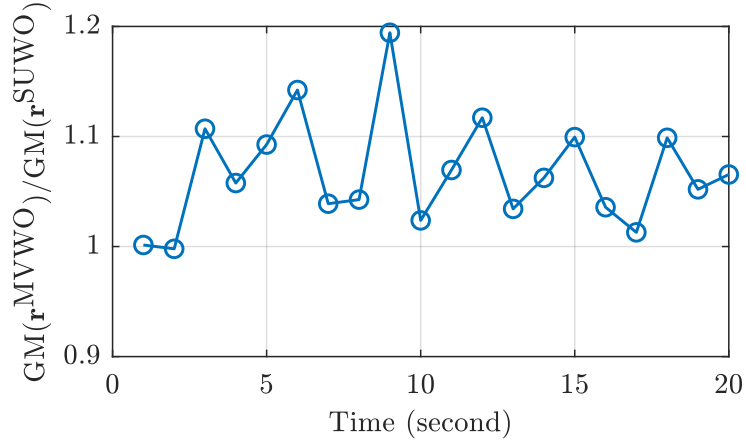
##### 4.8.6 Performance of Online MVWO Architecture for Varying Mean and Variance of SNRs

We then compare the performance of the MVWO method in the proposed online architecture in Section 4.7 to the SUWO methods [19], [20] in the network where the mean and variance of the SNRs are varying over time. Unlike in the previous case, the scenario in this simulation is closer to real-world networks as users have time-varying large-scale fading due to their mobility. We assume the number of users  $K$  is 3, and each user moves 5 meters per second backwards and forwards between two points on the ray line from the BS, which are 20 and 35 meters away from the BS. The initial position of the  $k$ -th user is at  $20 + 7.5k$  meters away from the BS,  $k = 1, \dots, K$ . The BS's transmit power spectrum density is 0 dBm/Hz and the noise spectrum density is  $-90$  dBm/Hz. The large-scale fading follows a path loss model as  $45 + 30 \log_{10}(l)$  dB, where  $l$  is the distance between a user and the BS in meters. The small-scale fading is the same as in Section 4.8.1. We use a typical periodicity of user's feedback on SNRs as the duration of a time slot, e.g.,  $\Delta_0$  is 10 milliseconds [74], and the bandwidth,  $B$ , is set as 5 MHz. We measure users' average rates every 1 second (or every 100 time slots) during 20 seconds (or 2000 time slots) and we denote the measured average rates achieved by our method and the SUWO methods [19], [20] as  $\mathbf{r}^{\text{MVWO}}$  and  $\mathbf{r}^{\text{SUWO}}$ , respectively. Since the users' rates are averaged every 100 time slots, the exponential

#### 4.8. Evaluation of Proposed Methods



(a) The utility functions achieved by the MVWO and SUWO methods.



(b) Difference between the utility functions achieved by the MVWO and SUWO methods

Figure 4.9: The difference in the performance of MWSs optimised by the online MVWO architecture and the SUWO methods [19], [20], where  $K = 5$ .

average window,  $\gamma$  of (4.44), in the compared SUWO methods is set to the same time scale, e.g.,  $\gamma = 100$  [16]. For our method, we set  $\beta = 20$  in (4.43) and  $\hat{\epsilon} = 10^{-4}$  in Algorithm 2. Note that the difference between the utility function in the above two methods can be written as

$$\begin{aligned}
 f(\mathbf{r}^{\text{MVWO}}) - f(\mathbf{r}^{\text{SUWO}}) &= \sum_{k=1}^K \ln r_k^{\text{MVWO}} - \sum_{k=1}^K \ln r_k^{\text{SUWO}} \\
 &= \ln \frac{\prod_{k=1}^K r_k^{\text{MVWO}}}{\prod_{k=1}^K r_k^{\text{SUWO}}} = K \ln \frac{\text{GM}(\mathbf{r}^{\text{MVWO}})}{\text{GM}(\mathbf{r}^{\text{SUWO}})},
 \end{aligned} \tag{4.47}$$



#### 4.9. Summary

where  $r_k^{\text{MVWO}}$  and  $r_k^{\text{SUWO}}$  represent the  $k$ -th user's rate in  $\mathbf{r}^{\text{MVWO}}$  and  $\mathbf{r}^{\text{SUWO}}$ , respectively. We quantitatively compare the performance of the two methods in terms of the ratio between the geometric mean of users' average rates,  $\text{GM}(\mathbf{r}^{\text{MVWO}})/\text{GM}(\mathbf{r}^{\text{SUWO}})$ , as shown in (4.47). Fig. 4.9a shows the value of the utility function every second in our method and the SUWO methods (with legends " $f(\mathbf{r}^{\text{MVWO}})$ , our method" and " $f(\mathbf{r}^{\text{SUWO}})$ , SUWO", respectively), which indicates that our method achieves a higher value of the utility function. Fig. 4.9b illustrates the ratio of the geometric mean of the average rates in two methods, where our method has a 5 ~ 15% improvement in geometrically averaged rates of users. This is because our MVWO method finds the optimal weights faster than other methods when the channel's statistics change as users move.

## 4.9 Summary

In this chapter, we investigated the methods that use CSI's mean and variance for the MWS scheduler design. We first estimated the feasible rate region of MWSs by using the mean and variance of users' SNR. This formulated the MVWO problem to find the optimal weights that maximise the utility function. We designed the solver for the MVWO problem and studied the solver's convergence. Since the mean and variance only require a few slots to estimate, our methods significantly reduce the number of time slots needed in optimising the MWSs, compared to other existing methods that optimise the weights via interaction with the network. We designed the online architecture to tune the weights according to the time-varying mean and variance of users' SNRs, which improves the average rates of users compared to other methods.

# Chapter 5

## Scheduler Design Using Graph Representation Learning for Contention and Interference Management

*In the previous two chapters, we have studied the methods to design a scheduler in a single base station (BS), where users are scheduled with orthogonal radio resources, and no interference exists in the network. Meanwhile, in most networks, the spectrum is reused at multiple BSs. Users associated with different BSs share the wireless channels and interfere with each other when transmitting simultaneously, especially in Wi-Fi networks, where users contend for transmission opportunities using a random channel access scheme. Since this scheme can lead to simultaneous transmissions that cause collisions and interference at receiving access points (APs), the network-wise coordination of transmissions is much needed. Note that the native Wi-Fi standards do not support the scheduling of the transmissions, while IEEE 802.11ah (or Wi-Fi HaLow) introduce a restricted access window (RAW) mechanism to allocate orthogonal time slots for users to schedule their transmissions. In this chapter, we investigate*

*the scheduling of RAW time slots to manage the contention and interference in Wi-Fi HaLow networks.*

## 5.1 Introduction

With the increasing demands for wireless connectivity, the IEEE task group ah has now developed a dedicated standard, namely IEEE 802.11ah (or Wi-Fi HaLow), which provides low-power and long-range wireless connections to network users [5], [15]. Like most 802.11 families, Wi-Fi HaLow users contend for channel access based on carrier-sense multiple access with collision avoidance (CSMA/CA). Specifically, when a user senses other users are transmitting on the channel, it will wait until the channel is free and further backoff for a random time before transmitting its packet to avoid packet collisions with other users.

Two issues exist in the Wi-Fi HaLow's channel access scheme [107]. First, when the user can sense many other users' transmissions, e.g., it is geographically surrounded by many users, it will experience a significant waiting and backoff time before each transmission, triggered by the surrounding users' transmissions. Hence, the user struggles to make its packet transmissions. Second, when two users cannot sense each other's transmissions due to large distance separation, i.e., hidden from each other, they can make concurrent transmissions causing inter-user interference at the receiving APs. Thus, the APs can barely decode transmissions due to low signal-to-interference-plus-noise ratios (SINR). The above two issues can cause users to suffer from throughput starvation.

The Wi-Fi HaLow network defines the restricted access window (RAW) mechanism [15], [108] to reduce the contention and interference among users. Specifically, users are divided into multiple groups in RAW. APs periodically schedule a RAW time slot for each user group's transmissions. Then, the users in each group contend for packet transmissions only during the assigned slots. Therefore, transmissions from users of other groups do not trigger the waiting time and backoffs in contention, which

## 5.1. Introduction

increases the time for transmissions. Also, the interference between two hidden users is eliminated if they are assigned to different RAW time slots. The open literature has not studied how to design a user grouping scheme in scheduling RAW slots to avoid throughput starvation. Several existing approaches can potentially be applied to design user grouping schemes in RAW as follows.

### 5.1.1 Related Works

#### Markov-model-based Approach

Research has been made to model each Wi-Fi user's channel access process as Markov chain [109]. Works in [32], [33] show that this model [109] can be used to formulate the optimisation problem of user grouping problem in RAW, e.g., to balance the channel utilisation [32] and the energy efficiency [33] for user groups. Note that the Markov chain in [109] only models the contention of users without considering the inter-user interference caused by hidden users in the network. The model developed in [110] can also be used, which extends the works in [109] by modelling the interference from hidden users. Thus, it achieves more accurate throughput estimation and can better capture the user throughput starvation effect in the network when applied to the user grouping problem. However, applying this model [110] requires prior knowledge of all hidden-user pairs whose measurements introduce significant overheads in the network. The overheads limit the performance of Wi-Fi HaLow networks since the networks' radio resources, e.g., bandwidth and power, are limited.

#### Graph-based Approach

Wireless network optimisation problems [26]–[28] can be formulated as graph theory problems [111]. Specifically, the graph's vertices represent users, and edges represent the contention or interference between vertices. Works in [26] construct an unweighted undirected graph by connecting two users if an AP can detect both users' transmissions. Then, a graph colouring problem assigns time slots as colours (or groups) to users such that no adjacent users occupy the same slot. Authors in [27], [28] con-

### 5.1. Introduction

struct weighted graphs with edges to indicate interference levels in user pairs. Here, undirected edges [27] assume the interference equally impacts both users' throughput, while directed edges [28] differentiate the interference power made by two users. Then, the graph's max cut divides users into a given number of groups by maximising the sum of edge weights between groups, and each group is assigned orthogonal frequencies to manage the interference. Note that the above works [26]–[28] heuristically construct the graph's edges using fixed rules predefined according to human intuition on how contention and interference affect network performance. Whether or not these graph constructions are optimal for the user grouping problem in RAW is unknown, and how to flexibly optimise the graph construction in wireless networks requires further investigation.

#### **Machine Learning Approach**

Machine learning (ML) methods have recently been widely applied to solve wireless network optimisation problems [112]. Typically, ML methods use network data to train a neural network (NN) that returns the optimal network control decisions for given network states [J1]. Thus, they do not require explicitly modelling the network behaviours such as contention and interference. Note that classic fully-connected NNs (FNNs) have pre-determined input and output dimensions, which is not flexible to varying dimensions of the problems, e.g., due to varying numbers of users in the network. Instead, graph neural networks (GNN) can be used with flexibility [113]. This is because GNNs can take network states as features on a graph whose size is adaptive to the dimensions of the problem. Then, GNNs obtain the control decisions by repeatedly aggregating features of neighbouring vertices (or edges) along with trainable parameters [29], [30]. Nevertheless, research has not been done on applying the ML-based approach and designing NN structures to solve the user group problem in RAW.

### 5.1.2 Our Methods

This chapter studies scheduling RAW slots in Wi-Fi HaLow networks to avoid user throughput starvation. This is achieved by finding optimal user grouping decisions in RAW that maximise the worst-case user throughput [114]. This objective avoids service outages and ensures service continuity in low-power and long-range wireless devices used for massive machine type communications (mMTC) [7], e.g., sensors used for continuous data collections [115]. We use path losses from users to APs as the network states to make the decisions, which are measurable at APs using signals sent by users [116] without additional measurement overheads. First, we construct a fully-connected weighted directed graph (or simply the graph onwards) to represent the network. Here, each user is a vertex. Each directed edge represents the asymmetric contention and interference between users, and the edge weight indicates how negatively one user’s transmissions impact another user’s throughput. We apply the graph’s max cut with the given edge weights to group users. Then, we propose adjusting the edge weights to construct the optimal graph whose max cut results in the optimal decisions. Next, we design an ML algorithm that trains a graph-constructing actor (a FNN) to approximate optimal mapping from each user pair’s states to the edge weights between them. The algorithm contains a graph cut procedure using semidefinite programming [25] to efficiently solve the max cut of the given actor-constructed graph, which returns the grouping decisions. Also, a graph-evaluating critic (a GNN) is trained to estimate users’ throughput for the given graph and states when using the above decisions, whose gradient optimises the actor. Here, we design the actor and the critic’s structures so they can abstract contention and interference information from states, e.g., a part of NNs is optimised to infer probabilities of hidden user pairs. Furthermore, since real-world networks can differ from those in offline training, we study how to adjust the graph based on online measurements. The contributions of this chapter have been listed in Chapter 1.

## 5.2 System Model and Problem Formulation

In this section, we present the system model of the wireless network, including the measured network configurations and network states. Then, we formulate the user grouping problem in RAW for contention and interference management.

### 5.2.1 Configurations of the Wireless Network

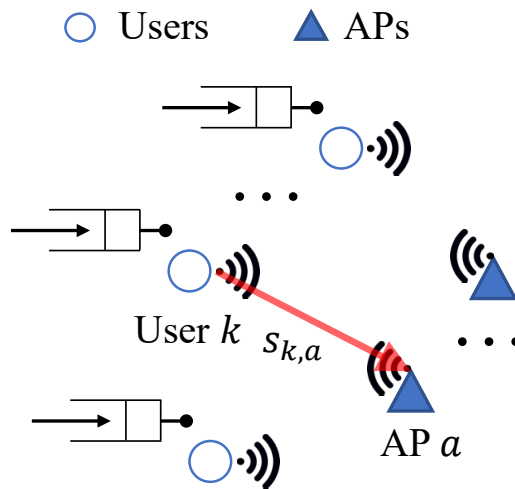


Figure 5.1: Illustration of a wireless network.

We consider a random-access-based wireless network, e.g., a Wi-Fi 802.11ah network, which consists of  $K$  users and  $A$  APs, as illustrated in Fig.5.1. We assume that all users and APs operate in the same channel with a bandwidth of  $B$  in Hertz (Hz). We denote the transmission power of users as  $\mathbf{P}_0$  in dBm and the noise power spectral density as  $\mathbb{N}_0$  in dBm/Hz. We denote the path loss from the  $k$ -th user to the  $a$ -th AP as  $\tilde{s}_{k,a}$  in dB,  $\forall k, a$ . We assume each user is associated with and transmits to the AP with the minimum path loss to the user. The AP that the  $k$ -th user is associated with is denoted as  $\hat{a}(k)$ , e.g.,

$$\hat{a}(k) \triangleq \arg \min_a \tilde{s}_{k,a}, \forall k. \quad (5.1)$$

We assume that each user has a stationary packet arrival process with the packet size  $L$  in bits. Packets arrived at each user are stored in a first-in-first-out queue with a

## 5.2. System Model and Problem Formulation

constant queue size, where the oldest packet is discarded when a new packet arrives and the queue is full. The duration to transmit each  $L$ -bit packet of user  $k$ ,  $d_k$  in seconds, depends on the modulation and coding scheme (MCS). We configure the MCS of each user to satisfy that the decoding error probability without interference,  $\epsilon_k$ , is lower than a threshold,  $\epsilon_{\max}$ . Here,  $\epsilon_k$  can be approximated as [38]

$$\epsilon_k \approx f_Q \left( \frac{-L \ln 2 + d_k B \ln(1 + \phi)}{\sqrt{d_k B V}} \right), \quad \forall k, \quad (5.2)$$

where  $\phi = (\mathbf{P}_0 / \tilde{s}_{k, \hat{a}(k)}) / (\mathbb{N}_0 B)$  is the signal-to-noise-ratio (SNR) of user  $k$ . Also,  $f_Q$  is the tail distribution function of the standard normal distribution, and  $V$  is the channel dispersion defined as  $V = 1 - 1/[1 + \phi]^2$  [38] in (5.2). Here, since  $\mathbf{P}_0$ ,  $\mathbb{N}_0$ ,  $B$  and  $L$  in (5.2) are all assumed to be a constant, the packet duration of each user only depends on the path loss to its associated AP,  $\tilde{s}_{k, \hat{a}(k)}$ ,  $\forall k$ .

Each user operates a CSMA/CA process to access the channel and transmit their packets [14], avoiding collisions with other users' packet transmissions. Specifically, each user first senses whether or not any other user is transmitting in the channel. If the above is true, the user continues to monitor the channel until the channel is sensed as idle. Then, the user backs off and stops transmitting for a random time, which helps reduce the probability of collision with other users also waiting for a transmission opportunity. If the channel is sensed as busy again during the backoff, the backoff countdown is paused and resumed again after the channel is sensed as idle. After the backoff ends, the user will transmit the oldest packet from its queue over the channel. If some users cannot sense this transmission, meaning they are hidden from that user, they will make simultaneous transmissions. These transmissions will cause interference at the receiving AP and raise the decoding error probability of the transmitted packet. In the event of a failed packet transmission, a negative acknowledgement from AP is received. Then, the user will retry transmitting the same packet upon failed packet transmission until successful delivery. Retransmissions will continue until the user receives an acknowledgement or reaches the maximum



## 5.2. System Model and Problem Formulation

attempts. The user will then transmit the next packet in the queue using the same process.

### 5.2.2 Network States

We assume that if the path loss between a user and an AP is greater than a threshold,  $\tilde{s}_{\max}$  in dB, then the user's transmissions cannot be detected and received at the AP. Thus, this path loss cannot be measured from the network. We write the measured path from the  $k$ -th user to the  $a$ -th AP as

$$s_{k,a} \triangleq \begin{cases} \tilde{s}_{k,a} , & \text{if } \tilde{s}_{k,a} \leq \tilde{s}_{\max} , \\ 2\tilde{s}_{\max} , & \text{if } \tilde{s}_{k,a} > \tilde{s}_{\max} , \end{cases} \quad \forall k, a , \quad (5.3)$$

where if  $\tilde{s}_{s,a} > \tilde{s}_{\max}$ , we set the value of  $s_{k,a}$  as  $2\tilde{s}_{\max}$ , indicating that it is immeasurable. Note that the path loss to each user's associated AP is always measurable because it is the user's minimum path loss to APs (otherwise, the user is not connected in the network), i.e.,  $\tilde{s}_{k,\hat{a}(k)} = s_{k,\hat{a}(k)}$ ,  $\forall k$ . Each user's states are its measured path losses to APs, defined as

$$\mathbf{s}_k \triangleq [s_{k,1}, \dots, s_{k,A}]^T , \quad \forall k , \quad (5.4)$$

and the whole network's states is a  $A \times K$  matrix that contains all users' measured path losses, defined as

$$\mathbf{S} \triangleq [\mathbf{s}_1, \dots, \mathbf{s}_K] . \quad (5.5)$$

### 5.2.3 User Grouping Problem in RAW

Fig. 5.2 illustrates a user grouping scheme in RAW of the Wi-Fi HaLow networks. We assume that all users and APs are synchronised in time, and the time is slotted in RAW with indices as  $1, \dots, t, \dots$ . The duration of each RAW slot is denoted as  $\Delta_0$  in seconds. We can simultaneously reduce the contention and interference in the network by dividing users into  $Z$  groups (for simplicity, we assume that  $Z$  is

## 5.2. System Model and Problem Formulation

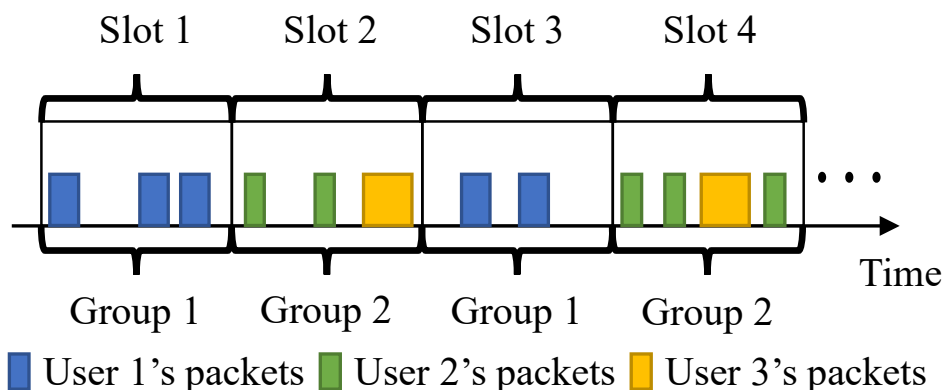


Figure 5.2: Illustration of user grouping in RAW, where  $K = 3$ ,  $Z = 2$ ,  $z_1 = 1$  and  $z_2 = z_3 = 2$ .

an exponent of 2). Then, different groups of users transmit their packets in separate periodical RAW slots. Specifically, We denote the group that user  $1, \dots, K$  is assigned to as  $\mathbf{z} \triangleq [z_1, \dots, z_K]^T$ , where  $z_k$  is user  $k$ 's group satisfying

$$z_k \in \{1, \dots, Z\}, \forall k, \quad (5.6)$$

Then, users in  $z$ -th group are  $\{k | z_k = z\}$ , and they transmit only in periodical time slots  $t = z, z + Z, z + 2Z, \dots$ . As a result, contention and interference only happen among users within the same group and are eliminated between any two groups. For example, in Fig. 5.2, three users ( $K = 3$ ) are grouped into two groups in RAW ( $Z = 2$ ), with user 1 in group 1 and user 2 and 3 in group 2. Then, group 1 (user 1) and group 2 (user 2 and 3) transmit in slots  $1, 3, 5 \dots$  and  $2, 4, 6 \dots$ , respectively. This user grouping scheme eliminates the contention and interference between user 1 and 2 and between user 1 and 3. We use  $u_k(t)$  to denote the number of packets successfully transmitted by the  $k$ -th user within RAW slot  $t$ . Note that  $u_k(t) = 0$  if  $t \neq z_k, z_k + Z, z_k + 2Z, \dots$  because users cannot transmit packets in slots other than the slot assigned to them. We denote the throughput of user  $1, \dots, K$  as  $\mathbf{r} \triangleq [r_1, \dots, r_K]^T$ , where  $r_k$  is defined as

$$r_k \triangleq \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^T u_k(t), \forall k, \quad (5.7)$$

which is the average number of packets transmitted over time.

### 5.3. Proposed User Grouping Framework for Wireless Networks

We write  $\mathbb{E}[r_k|\mathbf{S}, \mathbf{z}]$ ,  $\forall k$ , as the expected throughput of user  $k$  for given network states,  $\mathbf{S}$ , and grouping decisions,  $\mathbf{z}$ . We aim to maximise the network performance objective as the worst-case user throughput by controlling  $\mathbf{z}$  as

$$\max_{\mathbf{z}} \min_k \mathbb{E}[r_k|\mathbf{S}, \mathbf{z}] , \text{ s.t. (5.6) } , \quad (5.8)$$

where  $\min_k \mathbb{E}[r_k|\mathbf{S}, \mathbf{z}]$  is the worst-case user throughput for given  $\mathbf{S}$  and  $\mathbf{z}$ . The above straightforward formulation of the user grouping problem is hard to solve since no clear structure shows how to map the given network states,  $\mathbf{S}$ , to the grouping decisions,  $\mathbf{z}$ . We will propose a framework to reformulate the problem to address this issue in the sequel.

## 5.3 Proposed User Grouping Framework for Wireless Networks

This section presents the framework that formulates the user grouping problem as the graph construction optimisation that optimises a graph representing the wireless network.

We model the network as the fully-connected directed graph as  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , where  $\mathcal{V}$  and  $\mathcal{E}$  are the set of vertices representing users and the set of edges representing the contention and interference between users, respectively. Specifically,  $\mathcal{V}$  and  $\mathcal{E}$  are defined as

$$\mathcal{V} \triangleq \{1, \dots, K\} , \quad \mathcal{E} \triangleq \{(i, j) | \forall i, j \in \mathcal{V}, i \neq j\} , \quad (5.9)$$

where  $(i, j)$  is the edge from the  $i$ -th to the  $j$ -th vertex. The edge direction from  $i$  to  $j$  expresses the contention and interference that user  $i$ 's transmissions cause to user  $j$ . Each edge has a weight,  $W_{i,j}$ , representing how the contention and interference caused by user  $i$  negatively impact user  $j$ ' throughput. We assume that each edge weight is a bounded non-negative real number, i.e.,  $W_{i,j} \in [0, 1]$  (Without loss of generality, we assume that the upper bound of weights is 1),  $\forall i, j = 1, \dots, K$  and  $i \neq j$ . Here, the

### 5.3. Proposed User Grouping Framework for Wireless Networks

larger value of  $W_{i,j}$  implies that user  $j$ 's throughput is more negatively affected by user  $i$ 's transmissions. Note that  $W_{i,j}$  is generally not equal to  $W_{j,i}$  due to differences in two users' packet duration and path losses to APs. We collect all edge weights in the graph's adjacency matrix as

$$\mathbf{W} \triangleq [W_{i,j} | W_{i,j} \in [0, 1], \forall i \neq j; W_{k,k} = 0, \forall k] . \quad (5.10)$$

where  $\mathbf{W}$  is a  $K \times K$  matrix whose diagonal,  $W_{k,k}$ ,  $\forall k$ , is equal to 0, meaning that the graph has no self-loop edge on vertices.

Note that we can obtain the user grouping decisions by the max cut on the above graph as

$$\max_{\mathbf{z}} \sum_{i \neq j} W_{i,j} \mathbf{1}_{\{z_i \neq z_j\}}, \text{ s.t. } (5.6) , \quad (5.11)$$

which maximises the contention and interference, represented by edge weights, eliminated between different groups. Here,  $\mathbf{1}_{\{z_i \neq z_j\}}$  in (5.11) is the indicator function that equals to 1 if user  $i$  and  $j$  are assigned to different groups or otherwise equals to 0. In the above max cut problem, the higher the edge weights between two users, the more likely they are divided into two groups, leading to minimum contention and interference within each group. Based on this fact, we reformulate the user grouping problem in (5.8) as a bi-level optimisation problem as

$$\begin{aligned} & \max_{W_{i,j}, \forall i \neq j} \min_k \mathbb{E}[r_k | \mathbf{S}, \mathbf{z}] , \\ & \text{s.t. } W_{i,j} \in [0, 1], \forall i \neq j , \\ & \mathbf{z} = \arg \max_{\mathbf{z}'} \sum_{i \neq j} W_{i,j} \mathbf{1}_{\{z'_i \neq z'_j\}}, \text{ s.t. } (5.6). \end{aligned} \quad (5.12)$$

where the max cut problem in (5.11) is embedded as the lower-level problem (LLP), and  $\mathbf{z}$  is decided as the solution of the LLP. We can show that

**Lemma 2.** *Define the optimal solutions of edge weights as  $W_{i,j}^*$ ,  $\forall i \neq j$ , that maximises the network performance objective in (5.12), and define  $\mathbf{z}^*$  as the optimal solution of the LLP of (5.12) for the above optimal edge weights. Then,  $\mathbf{z}^*$  also maximises*

#### 5.4. Proposed Actor-Critic Graph Representation Learning Algorithm

the objective in (5.8).

*Proof.* The proof is in the appendix. □

The above statement implies that we can find the optimal grouping decisions in (5.8) by solving the problem (5.12) instead. Also, note that the grouping decisions in (5.12) only depend on the edge weights, as shown in the LLP of (5.12). As a result, the formulation in this section transforms the user grouping problem into the problem that finds the optimal construction of the graph representation of the network, e.g., by optimising edge weights in the graph.

Furthermore, we assume that each of the optimal edge weights in the graph,  $W_{i,j}^*$ , is a function of user  $i$  and user  $j$ ' states,  $\mathbf{s}_i$  and  $\mathbf{s}_j$ , as

$$W_{i,j}^* = \mu^*(\mathbf{s}_i, \mathbf{s}_j), \forall i \neq j, \quad (5.13)$$

where  $\mu^*(\cdot)$  is the optimal function that maps each user pair's states to the edge weights between them. Here, we note that  $\mathbf{s}_i$  and  $\mathbf{s}_j$  are parts of  $\mathbf{S}$  as defined in Section 5.2.2. Based on the formulation in (5.12) and the assumption in (5.13), the user grouping problem in (5.8) can be solved by finding the optimal graph-constructing function that uses network states to generate the edge weights in the graph. We then study how to use ML methods that train NN to approximate such a function in the next.

## 5.4 Proposed Actor-Critic Graph Representation Learning Algorithm

In this section, we develop the AC-GRL algorithm that trains NNs rather than uses fixed functions to generate optimal edge weights in the graph [117]. Note that optimal weights maximises the network performance objective in (5.12) for given network states,  $\mathbf{S} = [\mathbf{s}_1, \dots, \mathbf{s}_K]$  defined in Section 5.2.2. As shown in Fig. 5.3, the AC-GRL

#### 5.4. Proposed Actor-Critic Graph Representation Learning Algorithm

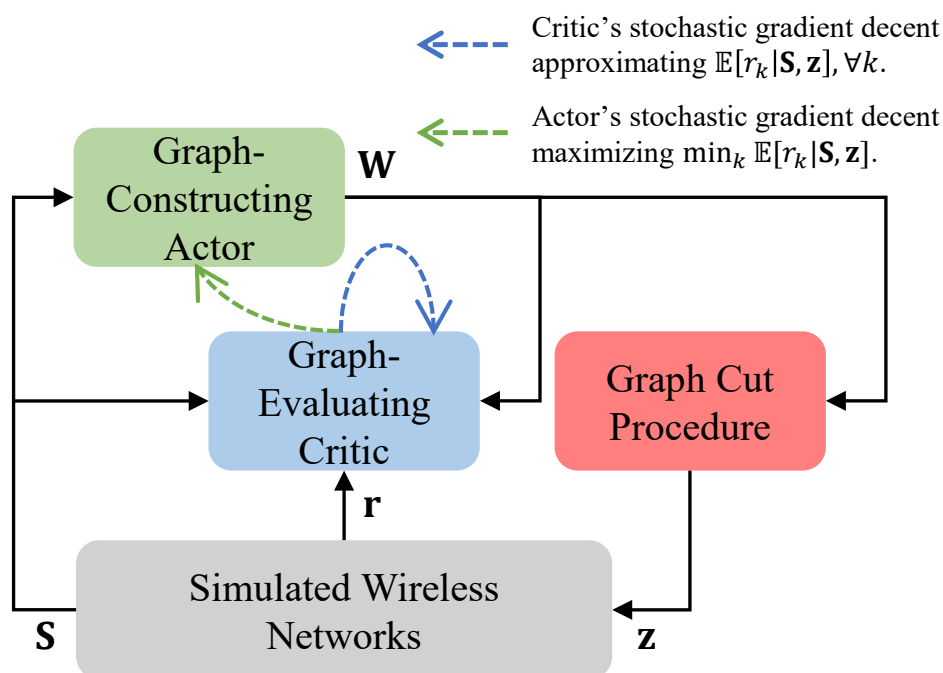


Figure 5.3: The overall structure of the AC-GRL algorithm.

algorithm consists of 1) the graph-constructing actor that uses  $\mathbf{S}$  to infer probabilities of hidden users and generate the graph's edge weights collected in the adjacency matrix,  $\mathbf{W}$  defined in Section 5.3. 2) the graph cut procedure that performs the graph's max cut to generate the user grouping decisions,  $\mathbf{z}$ , based on the LLP of (5.12) with given  $\mathbf{W}$ , and 3) the graph-evaluating critic that performs the same hidden user inference as the actor and further evaluates how good is the graph for the given  $\mathbf{W}$  and  $\mathbf{S}$  based on measured users' throughput,  $\mathbf{r}$  defined in Section 5.2.3.

The remaining section first presents the design of the above three components in Section 5.4.1, 5.4.2 and 5.4.3 and then explains the flow of the AC-GRL algorithm in Section 5.4.4.

#### 5.4. Proposed Actor-Critic Graph Representation Learning Algorithm

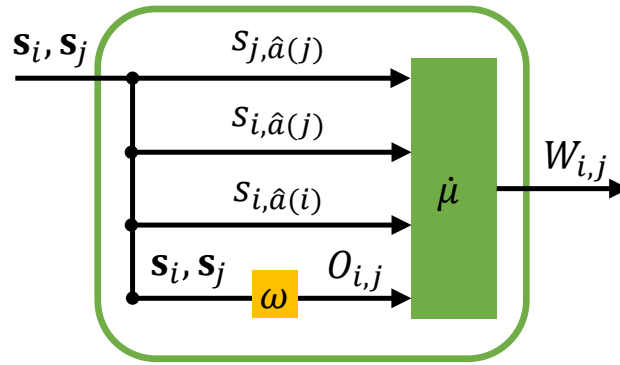


Figure 5.4: The structure of the actor.

##### 5.4.1 Design of the Graph-Constructing Actor

We use a NN,  $\mu(\cdot|\theta^\mu)$ , with trainable parameters  $\theta^\mu$ , to approximate the optimal mapping from the users' states to edge weights,  $\mu^*(\cdot)$  defined in (5.13), as

$$W_{i,j} \triangleq \mu(\mathbf{s}_i, \mathbf{s}_j|\theta^\mu) \approx \mu^*(\mathbf{s}_i, \mathbf{s}_j), \forall i \neq j. \quad (5.14)$$

where  $\mu(\cdot|\theta^\mu)$  is referred to as the graph-constructing actor. In the actor, we preprocess the network states of the user pair to help the actor abstract the information on the contention and interference. Specifically, for a given pair of users,  $i$  and  $j$ ,  $i \neq j$ , we can describe the contention and interference from user  $i$  to user  $j$  based on the following information contained in  $\mathbf{s}_i$  and  $\mathbf{s}_j$ :

- The path loss from user  $j$  to its associated AP,  $s_{j, \hat{a}(j)}$ , which determines receiving signal power at the AP.
- The path loss from user  $i$  to user  $j$ 's associated AP,  $s_{i, \hat{a}(j)}$ , which determines the interference power at the AP.
- The path loss from user  $i$  to its associated AP,  $s_{i, \hat{a}(i)}$ , which determines the duration of packets sent by user  $i$ , i.e., the duration of the interference made by user  $i$ .
- Whether or not user  $j$  can sense user  $i$ 's transmissions, or the opposite, whether user  $i$  is hidden from user  $j$ .

#### 5.4. Proposed Actor-Critic Graph Representation Learning Algorithm

Note that the network states do not measure whether or not two users can sense each other. Thus, we use a FNN,  $\omega(\cdot|\theta^\omega)$ , to infer this information from the network states as

$$\begin{aligned} O_{i,j} &\triangleq \omega(\mathbf{s}_i, \mathbf{s}_j|\theta^\omega) \\ &\approx \Pr [\text{user } j \text{ can sense user } i\text{'s transmissions}|\mathbf{s}_i, \mathbf{s}_j] , \end{aligned} \quad (5.15)$$

where we note that  $1 - O_{i,j}$  indicates how likely user  $i$  is hidden from user  $j$  for given  $\mathbf{s}_i$  and  $\mathbf{s}_j$ . Finally, we design the structure of the actor in (5.14) based on the above contention and interference information as

$$\begin{aligned} W_{i,j} &= \mu(\mathbf{s}_i, \mathbf{s}_j|\theta^\mu) \\ &= \dot{\mu}(s_{j,\hat{a}(j)}, s_{i,\hat{a}(j)}, s_{i,\hat{a}(i)}, O_{i,j}|\theta^\mu) \\ &= \dot{\mu}(s_{j,\hat{a}(j)}, s_{i,\hat{a}(j)}, s_{i,\hat{a}(i)}, \omega(\mathbf{s}_i, \mathbf{s}_j|\theta^\omega)|\theta^\mu) , \quad \forall i \neq j , \end{aligned} \quad (5.16)$$

where  $\omega(\cdot|\theta^\omega)$  is referred to as the inference NN. Here,  $\omega(\cdot|\theta^\omega)$  is considered as a part of the actor, and  $\dot{\mu}(\cdot|\theta^\mu)$  is also designed as a FNN. The above actor-generated edge weights are collected and returned as the graph's adjacency matrix,  $\mathbf{W}$  (note that diagonal elements of  $\mathbf{W}$  are 0, as defined in (5.10)).

#### 5.4.2 Design of the Graph Cut Procedure

The graph cut procedure solves the graph's max cut problem in (5.11) for given actor-generated edge weights,  $\mathbf{W}$ , which cuts the graph,  $\mathcal{G}$ , into  $Z$  parts and equivalently groups the graph's vertices (or the users),  $\mathcal{V}$ . As  $Z$  is assumed to be a power of 2, as mentioned in Section 5.2.3, we can cut the graph,  $\mathcal{G}$ , recursively by first dividing  $\mathcal{V}$  into two subsets and repeatedly dividing each subset into two until there are  $Z$  subsets, where each division of sets aims to disconnect the edges with higher weights. Fig. 5.5 illustrates a tree diagram of the graph cut process. There are  $\log_2(Z) + 1$  levels of graph cut. In the  $\beta$ -th level, the subsets of users are  $\mathcal{V}^{\beta,c}$ , where  $c = 1, \dots, 2^\beta$  and  $\beta = 0, \dots, \log_2(Z)$ , and the users in  $\mathcal{V}^{\beta,c}$  are

$$\mathcal{V}^{\beta,c} \triangleq \{k_1^{\beta,c}, \dots, k_{|\mathcal{V}^{\beta,c}|}^{\beta,c}\} , \quad (5.17)$$



#### 5.4. Proposed Actor-Critic Graph Representation Learning Algorithm

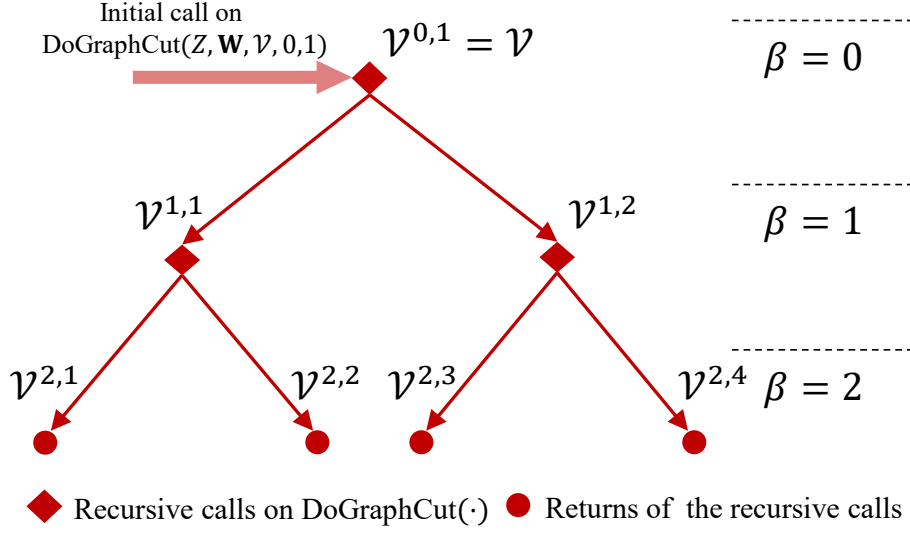


Figure 5.5: Tree diagram illustrating the recursive graph cut when  $Z = 4$  and  $\beta = 1, \dots, \log_2(Z)$ .

which are further divided into two subsets,  $\mathcal{V}^{\beta+1,2c-1}$  and  $\mathcal{V}^{\beta+1,2c}$ , in the next level. The edge weights between vertices in  $\mathcal{V}^{\beta,c}$  are collected as  $\mathbf{W}^{\beta,c}$ , a  $|\mathcal{V}^{\beta,c}| \times |\mathcal{V}^{\beta,c}|$  matrix, whose elements are determined by the original graph's adjacency matrix,  $\mathbf{W}$ , as

$$\mathbf{W}^{\beta,c} \triangleq [W_{i,j}^{\beta,c} | W_{i,j}^{\beta,c} = W_{k_i^{\beta,c}, k_j^{\beta,c}}, \forall k_i^{\beta,c}, k_j^{\beta,c} \in \mathcal{V}^{\beta,c}]. \quad (5.18)$$

We use the Goemans and Williamson's method [25] to divide a given set (or subset) of vertices. In detail, let us define an indicator  $y_i^{\beta,c}$  for each user  $k_i^{\beta,c}$  in  $\mathcal{V}^{\beta,c}$ , where  $y_i^{\beta,c} = -1$  if user  $k_i^{\beta,c}$  is divided into  $\mathcal{V}^{\beta+1,2c-1}$  and otherwise,  $y_i^{\beta,c} = +1$ , i.e., it is divided into  $\mathcal{V}^{\beta+1,2c}$ . Then, we can write the sub-problem to maximise the sum of weights on the disconnected edges between the vertices in  $\mathcal{V}^{\beta+1,2c-1}$  and  $\mathcal{V}^{\beta+1,2c}$  as

$$\begin{aligned} \max_{\mathbf{y}^{\beta,c}} \sum_{i \neq j} W_{i,j}^{\beta,c} \frac{1 - y_i^{\beta,c} y_j^{\beta,c}}{2}, \\ \text{s.t. } y_i^{\beta,c} \in \{-1, +1\}, \forall i = 1, \dots, |\mathcal{V}^{\beta,c}|, \end{aligned} \quad (5.19)$$

where  $\mathbf{y}^{\beta,c} \triangleq [y_1^{\beta,c}, \dots, y_{|\mathcal{V}^{\beta,c}|}^{\beta,c}]^T$ . Here,  $(1 - y_i^{\beta,c} y_j^{\beta,c})/2$  in (5.19) is equal to 1 if  $k_i^{\beta,c}$  and  $k_j^{\beta,c}$  are not in the same subset or 0 otherwise, which indicates whether or not edge  $(k_i^{\beta,c}, k_j^{\beta,c})$  is disconnected. This problem in (5.19) can be relaxed into a convex

#### 5.4. Proposed Actor-Critic Graph Representation Learning Algorithm

optimisation problem [25] as

$$\begin{aligned} \hat{\mathbf{X}}^{\beta,c} = \arg \max_{\mathbf{X}^{\beta,c}} \sum_{i \neq j} W_{i,j}^{\beta,c} \frac{1 - X_{i,j}^{\beta,c}}{2}, \\ \text{s.t. , } \mathbf{diag}\{\mathbf{X}^{\beta,c}\} = \mathbf{1} , \mathbf{X}^{\beta,c} \succeq 0 . \end{aligned} \quad (5.20)$$

Here,  $\mathbf{X}^{\beta,c}$  in (5.20) is a  $|\mathcal{V}^{\beta,c}| \times |\mathcal{V}^{\beta,c}|$  matrix whose elements,  $X_{i,j}^{\beta,c}$ , is a real number that approximates the multiplication,  $y_i^{\beta,c} y_j^{\beta,c}$ , in (5.19),  $\forall i, j$ . The above problem can be solved by existing convex optimisation solvers [49].

After solving the problem in (5.20), we can obtain the graph cut indicators,  $\mathbf{y}^{\beta,c}$ , by a rounding method [25]. Specifically, it performs singular value decomposition (SVD) on the optimal solution,  $\hat{\mathbf{X}}^{\beta,c}$ , that maximises the objective in (5.20). Since  $\hat{\mathbf{X}}^{\beta,c}$  is positive semidefinite, we can obtain a SVD of  $\hat{\mathbf{X}}^{\beta,c}$  with the following structure [48],

$$\mathbf{U}^{\beta,c} \mathbf{\Sigma}^{\beta,c} (\mathbf{U}^{\beta,c})^T = \text{svd}(\hat{\mathbf{X}}^{\beta,c}) , \quad (5.21)$$

where  $\mathbf{U}^{\beta,c}$  and  $\mathbf{\Sigma}^{\beta,c}$  are both  $|\mathcal{V}^{\beta,c}| \times |\mathcal{V}^{\beta,c}|$  matrices and further  $\mathbf{\Sigma}^{\beta,c}$  is a diagonal matrix with non-negative eigenvalues of  $\hat{\mathbf{X}}^{\beta,c}$  on its diagonal. Then, we randomly select a vector in  $\mathbb{R}^{|\mathcal{V}^{\beta,c}|}$  as  $\delta^{\beta,c}$  and set  $\mathbf{y}^{\beta,c}$  as<sup>1</sup>

$$\mathbf{y}^{\beta,c} = [y_1^{\beta,c}, \dots, y_{|\mathcal{V}^{\beta,c}|}^{\beta,c}]^T = \mathbf{sgn}(\mathbf{U}^{\beta,c} (\mathbf{\Sigma}^{\beta,c})^{\frac{1}{2}} \delta^{\beta,c}) . \quad (5.22)$$

We then configure  $\mathcal{V}^{\beta+1,2c-1}$  and  $\mathcal{V}^{\beta+1,2c}$  as

$$\begin{aligned} \mathcal{V}^{\beta+1,2c-1} &= \{k_i^{\beta,c} | y_1^{\beta,c} = -1, i = 1, \dots, |\mathcal{V}^{\beta,c}|\} , \\ \mathcal{V}^{\beta+1,2c} &= \{k_i^{\beta,c} | y_1^{\beta,c} = +1, i = 1, \dots, |\mathcal{V}^{\beta,c}|\} . \end{aligned} \quad (5.23)$$

---

<sup>1</sup>Each row of  $\mathbf{U}^{\beta,c} (\mathbf{\Sigma}^{\beta,c})^{\frac{1}{2}}$  is a unit vector, and the size of the angle between the  $i$ -th and  $j$ -th row indicates how likely user  $i$  and  $j$  should be assigned in the same subset,  $\forall i \neq j$  (smaller the angle, more likely they are in the same subset). Then, by multiplying  $\mathbf{U}^{\beta,c} (\mathbf{\Sigma}^{\beta,c})^{\frac{1}{2}}$  with a random vector, users' grouping indicators generated by (5.22) are more likely to have the same sign if the angle between their corresponding row vectors is smaller. It has been proved [25] that this method achieves at least 0.87854 of optimal achievable max cut objective in (5.19), which provides a near-optimal solution for max cut over given edge weights.

#### 5.4. Proposed Actor-Critic Graph Representation Learning Algorithm

Next, we repeat the graph cut process on  $\mathcal{V}^{\beta+1,2c-1}$  and  $\mathcal{V}^{\beta+1,2c}$ . Finally, when the process at the  $(\log_2(Z) - 1)$ -th level is done, we have  $Z$  disjoint groups,  $\mathcal{V}^{\beta,1} \dots, \mathcal{V}^{\beta,Z}$ , where  $\beta = \log_2(Z)$ , and the grouping decisions,  $\mathbf{z}$ , is obtained as

$$z_k = c, \forall k \in \mathcal{V}^{\beta,c}, \beta = \log_2(Z), \forall c = 1, \dots, Z. \quad (5.24)$$

Algorithm 3 summarises the recursive graph cut procedure, namely DoGraphCut( $\cdot$ ). The initial call of this procedure at the root of its tree diagram is DoGraphCut( $Z, \mathbf{W}, \mathcal{V}' = \mathcal{V}, \beta = 0, c = 1$ ), as shown in Fig. 5.5. Here,  $Z, \mathbf{W}$  and  $\mathcal{V}$  are the number of groups required, the adjacency matrix and the vertices of  $\mathcal{G}$ , respectively, as defined before. We denote the user grouping decisions computed based on this procedure as

$$\mathbf{z} = \text{DoGraphCut}(Z, \mathbf{W}, \mathcal{V}, 0, 1). \quad (5.25)$$

Note that the problem in (5.20) is a semidefinite programming problem that can be solved in polynomial computational complexity [47], [104]. Also, note that the graph cut procedure requires solving  $Z - 1$  (or  $\sum_{\beta=0}^{\log_2(Z)-1} 2^\beta$ ) times of the problem in (5.20). Since  $Z$  is constant in the grouping problem, the whole graph cut procedure, DoGraphCut( $Z, \mathbf{W}, \mathcal{V}, 0, 1$ ), has polynomial computational complexity.

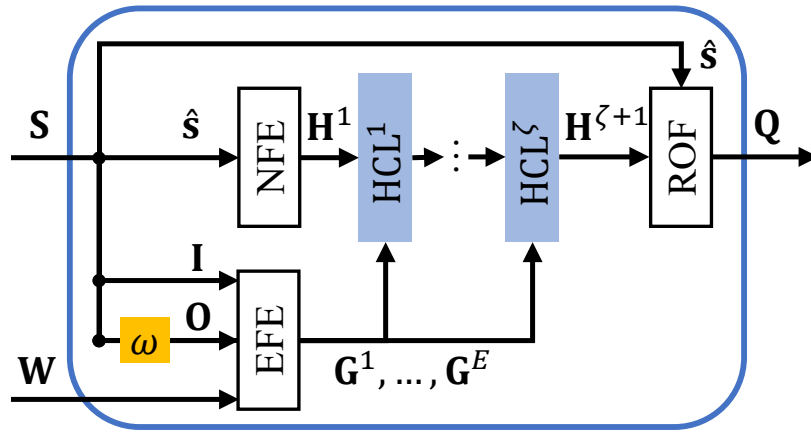
---

#### Algorithm 3 Recursive Graph Cut Procedure

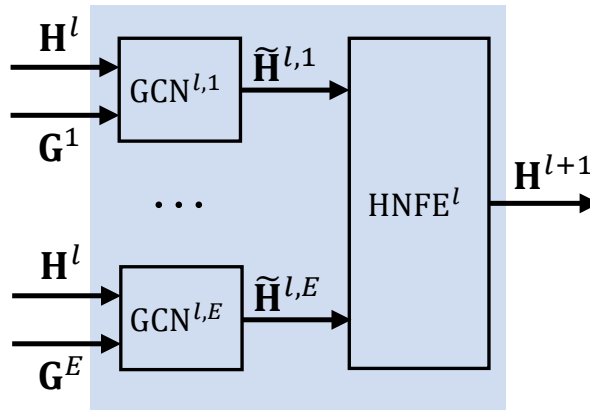
---

- 1: **procedure** DoGraphCut( $Z, \mathbf{W}, \mathcal{V}', \beta, c$ )
  - 2:   **if**  $\beta = \log_2(Z)$  **then**
  - 3:     Set  $z_k = c, \forall k \in \mathcal{V}'$ .
  - 4:   **else**
  - 5:     Set  $\mathbf{W}^{\beta,c}$  as (5.18), where  $\mathcal{V}^{\beta,c} = \mathcal{V}'$ .
  - 6:     Construct the problem in (5.20) and find  $\hat{\mathbf{X}}^{\beta,c}$ .
  - 7:     Generate  $\mathcal{V}^{\beta+1,2c-1}$  and  $\mathcal{V}^{\beta+1,2c}$  using (5.21)(5.22)(5.23).
  - 8:     Call DoGraphCut( $Z, \mathbf{W}, \mathcal{V}^{\beta+1,2c-1}, \beta + 1, 2c - 1$ ).
  - 9:     Call DoGraphCut( $Z, \mathbf{W}, \mathcal{V}^{\beta+1,2c}, \beta + 1, 2c$ ).
  - 10:   **end if**
  - 11:   **return**
  - 12: **end procedure**
-

#### 5.4. Proposed Actor-Critic Graph Representation Learning Algorithm



(a) The overall structure of the critic.



(b) The internal structure of HCL in each layer of the critic.

Figure 5.6: The structure of the critic.

#### 5.4.3 Design of the Graph-Evaluating Critic

We use a NN,  $\mathbf{Q}(\cdot)$ , to approximate the expected value of the users' throughput,  $\mathbf{r}$ , for the given network state,  $\mathbf{S}$ , and the given edge weights,  $\mathbf{W}$ , in the graph as

$$\begin{aligned}
 & \mathbb{E}[\mathbf{r}|\mathbf{S}, \mathbf{z}] \\
 &= \mathbb{E} \left[ [r_1, \dots, r_K]^T | \mathbf{S}, \text{DoGraphCut}(Z, \mathbf{W}, \mathcal{V}, 0, 1) \right] \\
 &\approx \mathbf{Q}(\mathbf{S}, \mathbf{W}|\theta^Q) = [Q_1(\mathbf{S}, \mathbf{W}|\theta^Q), \dots, Q_K(\mathbf{S}, \mathbf{W}|\theta^Q)]^T
 \end{aligned} \tag{5.26}$$

where  $\mathbf{z}$  is the grouping decisions generated based on (5.25) and  $Q_k(\mathbf{S}, \mathbf{W}|\theta^Q)$  is the  $k$ -th element in the output of  $\mathbf{Q}(\mathbf{S}, \mathbf{W}|\theta^Q)$  that approximates  $r_k$ ,  $\forall k$ . The overall structure of the critic is shown in Fig. 5.6a. To help the critic abstract information

#### 5.4. Proposed Actor-Critic Graph Representation Learning Algorithm

on the contention and interference from the network states, we apply the same pre-processing on the network states in the actor's design from Section 5.4.1. Note that we define the vector and matrices to collect all user's pre-processed states and simplify the notations as

$$\begin{aligned}\hat{\mathbf{S}} &\triangleq [s_{1,\hat{a}(1)}, \dots, s_{K,\hat{a}(K)}] , \\ \mathbf{I} &\triangleq [I_{i,j} | I_{i,j} = s_{i,\hat{a}(j)}, \forall i \neq j; I_{k,k} = 0, \forall k] , \\ \mathbf{O} &\triangleq [O_{i,j} | O_{i,j} = \omega(\mathbf{s}_i, \mathbf{s}_j | \theta^\omega), \forall i \neq j; O_{k,k} = 0, \forall k] .\end{aligned}\tag{5.27}$$

Here,  $\hat{\mathbf{S}}$  contains all path losses from users to their associated APs, determining the receiving signal power and the packet duration of users.  $\mathbf{I}$  contains path losses from each user to all other users' associated APs, indicating the interference power.  $\mathbf{O}$  indicates how likely each pair of users can sense each other. Then, the critic has a structure as follows

$$\mathbf{Q}(\mathbf{S}, \mathbf{W} | \theta^Q) = \dot{\mathbf{Q}}(\hat{\mathbf{S}}, \mathbf{I}, \mathbf{O}, \mathbf{W} | \theta^Q) |_{O_{i,j} = \omega(\mathbf{s}_i, \mathbf{s}_j | \theta^\omega), \forall i \neq j} ,\tag{5.28}$$

where we note that  $\hat{\mathbf{S}}$  and  $\mathbf{I}$  can be directly taken from  $\mathbf{S}$ , while  $\mathbf{O}$  is computed based on  $\mathbf{S}$  by using the inference NN,  $\omega(\cdot | \theta^\omega)$  defined in (5.15). Thus,  $\omega(\cdot | \theta^\omega)$  is a shared part of both the actor and the critic.

The structure of  $\dot{\mathbf{Q}}(\cdot)$  is designed based on GNNs that are flexible to the dimensions of its inputs,  $\hat{\mathbf{S}}$ ,  $\mathbf{I}$ ,  $\mathbf{O}$  and  $\mathbf{W}$ , as follows. First, we use a FNN, referred to as the edge feature embedder (EFE), to embed the user-pair-wise state information,  $\mathbf{I}$  and  $\mathbf{O}$  and the actor-generated edge weights,  $\mathbf{W}$ , into  $E$  embedded edge features, e.g.,  $\mathbf{G}^1, \dots, \mathbf{G}^E$ , whose the  $(i, j)$ -th elements are

$$[G_{i,j}^1, \dots, G_{i,j}^E]^T = \text{EFE}(I_{i,j}, O_{i,j}, W_{i,j}), \forall i \neq j ,\tag{5.29}$$

and we set  $G_{k,k}^e = 0, \forall e, k$ . Also, we use a FNN, namely the node feature embedder (NFE), to embed the per-user-wise state information,  $\hat{\mathbf{S}}$ , into a higher dimension

#### 5.4. Proposed Actor-Critic Graph Representation Learning Algorithm

as

$$\begin{aligned} \mathbf{H}^1 &\triangleq [\mathbf{h}_1^1, \dots, \mathbf{h}_K^1]^T \\ &= [\text{NFE}(s_{1,\hat{a}(1)}), \dots, \text{NFE}(s_{K,\hat{a}(K)})]^T, \end{aligned} \quad (5.30)$$

where  $\mathbf{h}_1^1, \dots, \mathbf{h}_K^1$  are  $M$ -dimensional vectors and  $\mathbf{H}^1$  is a  $K \times M$  matrix that is the input of the first hidden critic layer (HCL). There are  $\zeta$  HCLs in the critic, e.g.,  $\text{HCL}^l$  is the  $l$ -th HCL,  $l = 1, \dots, \zeta$ .  $\text{HCL}^l$  has two parts of inputs as 1) the embedded node features from the previous HCL's output (or from the NFE's output when  $l = 1$ ),  $\mathbf{H}^l$ , and 2) the embedded edge features from the EFE,  $\mathbf{G}^1, \dots, \mathbf{G}^E$ , as

$$\mathbf{H}^{l+1} \triangleq [\mathbf{h}_1^{l+1}, \dots, \mathbf{h}_K^{l+1}]^T = \text{HCL}^l(\mathbf{H}^l, \mathbf{G}^1, \dots, \mathbf{G}^E), \forall l, \quad (5.31)$$

where the internal structure of each HCL is shown in Fig. 5.6b. Specifically, we use  $E$  graph convolutional networks (GCNs) [118]<sup>2</sup>,  $\text{GCN}^{l,e}$ ,  $e = 1, \dots, E$ , in  $\text{HCL}^l$  to aggregate  $\mathbf{H}^l$  and  $\mathbf{G}^e$  into hidden node features as

$$\begin{aligned} \tilde{\mathbf{H}}^{l,e} &\triangleq [\tilde{\mathbf{h}}_1^{l,e}, \dots, \tilde{\mathbf{h}}_K^{l,e}]^T = \text{ReLU}(\text{GCN}^{l,e}(\mathbf{H}^l, \mathbf{G}^e)) \\ &= \text{ReLU}((\mathbf{D}^{l,e})^{-\frac{1}{2}}(\mathbf{G}^e + \mathbb{I}_K)(\mathbf{D}^{l,e})^{-\frac{1}{2}}\mathbf{H}^l\Theta^{l,e}), \forall l, e, \end{aligned} \quad (5.32)$$

where  $\text{ReLU}(\cdot)$  is the rectified linear activation function and  $\Theta^{l,e}$  are trainable parameters of  $\text{GCN}^{l,e}$ . Here,  $\mathbb{I}_K$  is a  $K \times K$  identity matrix and  $\mathbf{D}^{l,e}$  is the diagonal degree matrix of  $\mathbf{G}^e + \mathbb{I}_K$  (i.e.,  $D_{i,i}^{l,e} = \sum_{j=1}^K G_{i,j}^e + 1$ ) in (5.32). Next, a FNN, namely hidden node feature embedder (HNFE), aggregates each user's hidden node features from all GCNs as

$$\begin{aligned} \mathbf{H}^{l+1} &\triangleq [\mathbf{h}_1^{l+1}, \dots, \mathbf{h}_K^{l+1}]^T \\ &= \left[ \text{HNFE}^l \left( [(\tilde{\mathbf{h}}_1^{l,1})^T, \dots, (\tilde{\mathbf{h}}_1^{l,E})^T]^T \right), \dots, \text{HNFE}^l \left( [(\tilde{\mathbf{h}}_K^{l,1})^T, \dots, (\tilde{\mathbf{h}}_K^{l,E})^T]^T \right) \right]^T, \forall l, \end{aligned} \quad (5.33)$$

where  $\text{HNFE}^l$  is the HNFE in the  $l$ -th layer and  $\mathbf{H}^{l+1}$  is forwarded to the next HCL.

<sup>2</sup>we use GCNs to construct the critic because most inputs of the critic in (5.28) are user-pair-wise features (e.g.,  $\mathbf{I}$ ,  $\mathbf{O}$  and  $\mathbf{W}$ ) that can be viewed as edge features on a graph, as GCNs dedicated for. Alternatively, other GNNs can take edge features as inputs can be used, e.g., GNNs listed in [119]. However, what the optimal GNN structure is for the critic is not the focus of this work.

#### 5.4. Proposed Actor-Critic Graph Representation Learning Algorithm

Last, a FNN is used as a readout function (ROF) to map the node features from the output of the last HCL and the initial per-user-wise network states,  $\hat{\mathbf{s}}$ , into the approximated average throughput of users as

$$\dot{\mathbf{Q}}(\hat{\mathbf{s}}, \mathbf{I}, \mathbf{O}, \mathbf{W}|\theta^{\dot{Q}}) = [\text{ROF}(\mathbf{h}_1^{\zeta+1}, s_{1,\hat{a}(1)}), \dots, \text{ROF}(\mathbf{h}_K^{\zeta+1}, s_{K,\hat{a}(K)})]^T, \quad (5.34)$$

where  $\mathbf{h}_k^{\zeta+1}, \forall k$ , are computed as (5.29)-(5.33).

#### 5.4.4 The Flow of the AC-GRL Algorithm

Finally, we explain the flow of the AC-GRL algorithm that trains NNs in the above components, where the initial values of all NN parameters are randomly initialised.

##### Pre-training of the Inference NN

We first train the inference NN,  $\omega(\cdot|\theta^\omega)$ , that is the shared part of the actor and the critic. In each training step, we simulate a realisation of the wireless network where we measure the network states,  $\mathbf{S}$ . We also acquire whether or not each pair of users can sense each other as a  $K \times K$  matrix of binary indicators, defined as

$$\mathbf{O}^* \triangleq [O_{i,j}^* | O_{i,j}^* = \mathbf{1}_{\{\text{user } j \text{ can sense user } i\}}, \forall i \neq j; O_{k,k}^* = 0, \forall k]. \quad (5.35)$$

We note that  $\mathbf{O}^*$  does not need to be measured when deploying our methods in a real-world network. Then, the parameter of  $\omega(\cdot|\theta^\omega)$  is optimised by minimising a loss function as the cross entropy between the inferred expected value of  $O_{i,j}^*$ ,  $O_{i,j}$ , and its true value, e.g.,

$$L(\theta^\omega) = \mathbb{E} \left[ \sum_{i \neq j} -O_{i,j}^* \log(\omega(\mathbf{s}_i, \mathbf{s}_j|\theta^\omega)) - (1 - O_{i,j}^*) \log(1 - \omega(\mathbf{s}_i, \mathbf{s}_j|\theta^\omega)) \right], \quad (5.36)$$

#### 5.4. Proposed Actor-Critic Graph Representation Learning Algorithm

where we update  $\theta^\omega$  using the stochastic gradient descent (SGD) method [42] based on the gradient of (5.36) as

$$\nabla_{\theta^\omega} L(\theta^\omega) = \sum_{i \neq j} \left( \frac{-O_{i,j}^*}{O_{i,j}} - \frac{1 - O_{i,j}^*}{1 - O_{i,j}} \cdot (-1) \right) \nabla_{\theta^\omega} \omega(\mathbf{s}_i, \mathbf{s}_j | \theta^\omega). \quad (5.37)$$

The training step of the inference NN is repeated  $N$  times.

#### Main Training Process of the AC-GRL Algorithm

After the inference NN is trained to estimate the probability of hidden users, we train the remaining parameters of the actor and the critic, as shown in Fig. 5.3. We measure  $\mathbf{S}$  from a randomised realisation of the wireless network at the start of each training step. Then, the graph-constructing actor generates  $\mathbf{W}$  as (5.16). With a probability of  $\nu$  ( $\nu \in [0, 1]$ ), we set edge weights in the off-diagonal of  $\mathbf{W}$  with random numbers in  $[0, 1]$  for the purpose of exploration of good edge weights. After that, the graph cut procedure computes the grouping decisions,  $\mathbf{z}$ , based on  $\mathbf{W}$  by calling DoGraphCut( $\cdot$ ) in Algorithm 3 as (5.25). Then, we measure the users' throughput,  $\mathbf{r}$ , from the network for the given  $\mathbf{z}$ . The measured users' throughput is used to optimise the graph-evaluating critic that approximates the throughput for given network states and edge weights. Thus, the loss function of the critic is the difference between the estimated throughput and the actual measured users' throughput from the network as

$$L(\theta^Q) = \mathbb{E} \left[ \sum_{k=1}^K (r_k - Q_k(\mathbf{S}, \mathbf{W} | \theta^Q))^2 \right], \quad (5.38)$$

whose gradient of  $L(\theta^Q)$  with respect to  $\theta^{\dot{Q}}$  is

$$\begin{aligned} \nabla_{\theta^{\dot{Q}}} L(\theta^Q) = \sum_{k=1}^K \left\{ 2[r_k - \dot{Q}_k(\hat{\mathbf{s}}, \mathbf{I}, \mathbf{O}, \mathbf{W} | \theta^{\dot{Q}})] \right. \\ \left. \times \nabla_{\theta^{\dot{Q}}} \dot{Q}_k(\hat{\mathbf{s}}, \mathbf{I}, \mathbf{O}, \mathbf{W} | \theta^{\dot{Q}}) \right\} \Big|_{O_{i,j} = \omega(\mathbf{s}_i, \mathbf{s}_j | \theta^\omega), \forall i \neq j}. \end{aligned} \quad (5.39)$$

Also, the critic is used to optimise the actor. Note that the actor is optimised to generate the best edge weights that maximise the worst-case user throughput in (5.12),



### 5.5. Proposed Online Fine-Tuning Architecture

which can be expressed as a loss function based on the critic as

$$L(\theta^\mu) = \mathbb{E} \left[ - \min_k Q_k(\mathbf{S}, \mathbf{W} | \theta^Q) \Big|_{W_{i,j} = \mu(\mathbf{s}_i, \mathbf{s}_j | \theta^\mu)} \right], \quad (5.40)$$

where  $\min_k Q_k(\cdot)$  is the worst-case user throughput approximated by the critic and we can derive the gradients of  $L(\theta^\mu)$  with respect to  $\theta^\mu$  as

$$\begin{aligned} \nabla_{\theta^\mu} L(\theta^\mu) &= - \nabla_{\mathbf{W}} \min_k Q_k(\mathbf{S}, \mathbf{W} | \theta^Q) \times \nabla_{\theta^\mu} \mathbf{W} \\ &= - \sum_{i \neq j} \frac{\partial \min_k \dot{Q}_k(\hat{\mathbf{s}}, \mathbf{I}, \mathbf{O}, \mathbf{W} | \theta^Q)}{\partial W_{i,j}} \\ &\quad \times \nabla_{\theta^\mu} \dot{\mu}(s_{j, \hat{a}(j)}, s_{i, \hat{a}(j)}, s_{i, \hat{a}(i)}, O_{i,j} | \theta^\mu) \Big|_{O_{i,j} = \omega(\mathbf{s}_i, \mathbf{s}_j | \theta^\omega), \forall i \neq j}. \end{aligned} \quad (5.41)$$

Then, the SGD method updates the actor and the critic's parameters based on the above gradients in (5.39) and (5.41), respectively. Here, we repeat  $N$  times the training step for the actor and the critic.

Algorithm 4 summarises the NNs' training process, where lines 2-6 are the inference NN's training process and lines 7-16 are the actor and the critic's training process. Note that  $\eta$  is the learning rate of the NNs.

## 5.5 Proposed Online Fine-Tuning Architecture

In this section, we explain the online architecture that further fine-tunes the graph's edge weights for a given realisation of the wireless network, as shown in Fig. 5.7. Note that we assume that Algorithm 4 has returned the trained parameters of the inference NN, the actor, and the critic. We will not make any updates on them in the online architecture.

Specifically, to fine-tune the edge weights for a given network, we measure the network states  $\mathbf{S}$  as defined in Section 5.2.2 and infer  $O_{i,j}, \forall i \neq j$  as (5.15). Then, we compute  $\mathbf{W}$  using the actor as (5.14), and we then use computed  $\mathbf{W}$  to generate the user grouping decisions as (5.25). Next, we measure the average throughput of users,  $r_k$ ,

### 5.5. Proposed Online Fine-Tuning Architecture

---

#### Algorithm 4 Proposed Actor-Critic Graph Representation Learning Algorithm

---

- 1: Randomly initialise NNs' parameters,  $\theta^\omega$ ,  $\theta^{\hat{Q}}$  and  $\theta^{\hat{\mu}}$ .
  - 2: **for** step  $n = 1, \dots, N$  **do**
  - 3:   Generate a network randomly.
  - 4:   Measure  $\mathbf{S}$  and  $\mathbf{O}^*$  from the network.
  - 5:   Compute  $\nabla_{\theta^\omega} L(\theta^\omega)$  as (5.37) and perform the SGD on the inference NN as  

$$\theta^\omega \leftarrow \theta^\omega - \eta \nabla_{\theta^\omega} L(\theta^\omega).$$
  - 6: **end for**
  - 7: **for** step  $n = 1, \dots, N$  **do**
  - 8:   Generate a network randomly.
  - 9:   Measure  $\mathbf{S}$  from the network.
  - 10:   Generate  $\mathbf{W}$  as (5.16) using the graph-constructing actor.
  - 11:   With probability  $\nu$ , randomly set the values in  $\mathbf{W}$ .
  - 12:   Compute  $\mathbf{z}$  as (5.25) using the graph cut procedure.
  - 13:   Execute  $\mathbf{z}$  in the network and measure  $\mathbf{r}$ .
  - 14:   Compute  $\nabla_{\theta^{\hat{Q}}} L(\theta^{\hat{Q}})$  as (5.39) using the graph-evaluating critic and perform the SGD on the critic as  

$$\theta^{\hat{Q}} \leftarrow \theta^{\hat{Q}} - \eta \nabla_{\theta^{\hat{Q}}} L(\theta^{\hat{Q}}).$$
  - 15:   Compute  $\nabla_{\theta^{\hat{\mu}}} L(\theta^{\hat{\mu}})$  as (5.41) using the actor and the critic and perform the SGD on the actor as  

$$\theta^{\hat{\mu}} \leftarrow \theta^{\hat{\mu}} - \eta \nabla_{\theta^{\hat{\mu}}} L(\theta^{\hat{\mu}}).$$
  - 16: **end for**
  - 17: **return**  $\theta^\omega$ ,  $\theta^{\hat{Q}}$  and  $\theta^{\hat{\mu}}$  for online fine-tuning.
- 

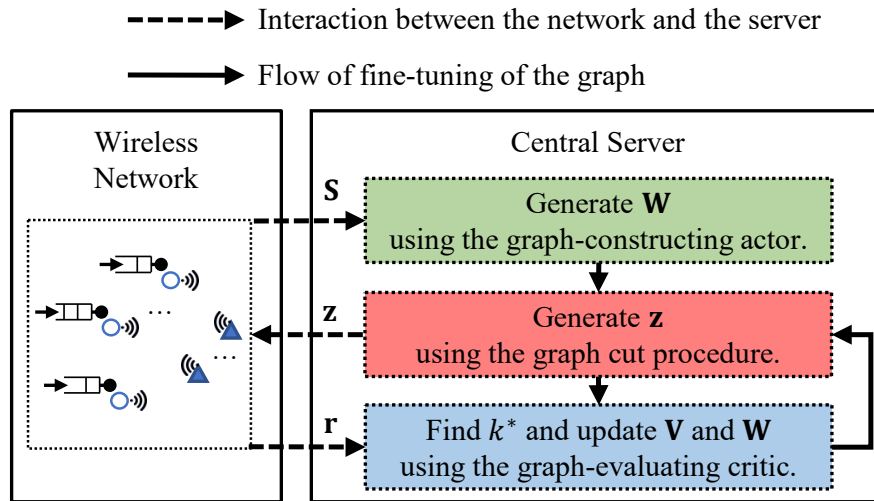


Figure 5.7: The proposed online fine-tuning architecture.

$k = 1, \dots, K$ , over  $T$  RAW slots and find the worst-case user's index as

$$k^* \leftarrow \arg \min_k r_k . \quad (5.42)$$

### 5.6. Evaluation of Proposed Methods

The elements in  $\mathbf{W}$  are then fine-tuned based on the worst-case user's index and the trained critic by minimising the following loss function,

$$L(\mathbf{W}) = -\dot{Q}_{k^*}(\hat{\mathbf{s}}, \mathbf{I}, \mathbf{O}, \mathbf{W} | \theta^{\dot{Q}}) \quad (5.43)$$

In order to prevent the updated weight values from going out of the bounds on weights, i.e.,  $[0, 1]$ , we update an intermediate  $K \times K$  matrix,  $\mathbf{V}$ , instead, where

$$\mathbf{V} \triangleq [V_{i,j} | V_{i,j} = \text{Sigmoid}^{-1}(W_{i,j}), \forall i \neq j; V_{k,k} = 0, \forall k] . \quad (5.44)$$

Here,  $\text{Sigmoid}^{-1}$  is the inverse of the Sigmoid function with the output range as  $[0, 1]$ . Then,  $\mathbf{V}$  is updated based on the gradient of the critic as

$$\begin{aligned} \nabla_{\mathbf{V}} L(\mathbf{W}) &= -\nabla_{\mathbf{V}} \dot{Q}_{k^*}(\hat{\mathbf{s}}, \mathbf{I}, \mathbf{O}, \mathbf{W} | \theta^{\dot{Q}}) \\ &= -\nabla_{\mathbf{W}} \dot{Q}_{k^*}(\hat{\mathbf{s}}, \mathbf{I}, \mathbf{O}, \mathbf{W} | \theta^{\dot{Q}}) \times \nabla_{\mathbf{V}} \mathbf{W} , \end{aligned} \quad (5.45)$$

where  $\nabla_{\mathbf{V}} \mathbf{W}$  can be computed based on the derivative of the Sigmoid function in (5.44). Next,  $\mathbf{V}$  and  $\mathbf{W}$  are updated as

$$\mathbf{V} \leftarrow \mathbf{V} - \eta \nabla_{\mathbf{V}} L(\mathbf{W}) , \mathbf{W} \leftarrow \text{Sigmoid}(\mathbf{V}) , \quad (5.46)$$

where  $\eta$  is the learning rate. After each time that  $\mathbf{W}$  is updated, we regenerate the grouping decisions as (5.25) and measure users' throughput again for the next  $T$  RAW slots. Then, the worst-case user is found as (5.42). Finally, the update of  $\mathbf{V}$  and  $\mathbf{W}$  in (5.45) and (5.46) is repeated until the network stops.

## 5.6 Evaluation of Proposed Methods

This section provides the simulation results that evaluate our proposed methods.

### 5.6.1 Simulation Configurations

We use NS-3 [31], [120] to simulate Wi-Fi HaLow networks, where all devices are located in a  $2\text{km} \times 2\text{km}$  squared area. We assume there are 4 APs ( $A = 4$ ), and they are located at the grid in the simulated area. Unless specifically stated, we set the number of users,  $K$ , as 20, and they are randomly distributed in the simulated area. The duration of a RAW slot,  $\Delta_0$ , is configured as 10 milliseconds. The channel bandwidth,  $B$  is set as 1 MHz at a 1 GHz carrier frequency. The transmission power of users is set as  $\mathbf{P}_0 = 0$  dBm, and the noise power spectral density is set as  $\mathbb{N}_0 = -90$  dBm/Hz. The path losses between any two devices (including APs and users) follow Friis model [121] as  $10 \log_{10}(\lambda^2/(4\pi d)^2)$  in dB, where  $\lambda$  is the wavelength at 1 GHz and  $d$  is the distance in meters between two devices. The threshold on path losses where a user can sense a transmission,  $\tilde{s}_{\max}$ , is set to 95 dB. The packet size,  $L$ , is 800 bits, the maximum queue size is 5, and each user has a Poisson packet arrival process with intervals of 20 milliseconds. The maximum decoding error probability,  $\epsilon_{\max}$ , for each user with no interference is  $10^{-5}$ . The decoding error probability of each transmitted packet is computed using the same equation in (5.2), where  $\phi$  is the signal-to-interference-plus-noise of each transmission instead.

All FNNs used in the actor and the critic have one input layer, one output layer, and two hidden layers. The activation functions of all hidden layers in FNNs are set as ReLU functions. Further, the size of each layer and the output activation functions (OAFs) of FNNs are listed in Table 5.1, where  $M = E = 5$ . The size of trainable parameters of GCNs in (5.32),  $\Theta^{l,e}$ , is  $M \times M$ ,  $\forall l, e$ . The exploration rate,  $\nu$ , is set to 10%, and the learning rate of NNs,  $\eta$ , is set to  $10^{-4}$ . The number of training steps,  $N$ , in Algorithm 4 is set to 1000.

## 5.6. Evaluation of Proposed Methods

Table 5.1: Configurations of FNNs in the Actor and the Critic

NNs	Dimensions of layers	OAFs
$\omega$	$2A, 20A, 20A, 1$	Sigmoid( $\cdot$ )
$\dot{\mu}$	$4, 40, 40, 1$	Sigmoid( $\cdot$ )
EFE	$3, 30, 30, E$	ReLU( $\cdot$ )
NFE	$1, 10, 10, M$	ReLU( $\cdot$ )
$\text{HNFE}^l, \forall l$	$EM, 10EM, 10EM, M$	ReLU( $\cdot$ )
ROF	$M + 1, 10(M + 1), 10(M + 1), 1$	None

### 5.6.2 Compared Methods in Simulations

#### Applying Heuristics

We compare our method to the heuristics method that simply randomly allocates users in each group with equal probability, referred to as the ‘‘RAND’’ scheme. We also compare a method that uniformly balances the number of users in each group, referred to as the ‘‘UNIF’’ scheme. Specifically, we sort the users based on their associated AP’s index as  $k'_1, \dots, k'_K$ , where  $\hat{a}(k'_1) \leq \hat{a}(k'_2) \leq \dots \leq \hat{a}(k'_K)$ . Then, we set  $z_{k'_i} = (i \bmod Z) + 1, \forall k$ , which balances the number of users in different groups for each AP.

#### Applying Markov-model-based Approach

We apply the Markov-model-based approach to the user grouping problem, where the Markov models in [109], [110] are used to estimate each user’s throughput following the approach taken in [32], [33]. Note that the model in [110] requires full knowledge of whether each pair of users are hidden users or not. Since this information is not measured in the network states, the model [110] cannot be applied directly to the user grouping problem in this work. Thus, we estimate the users’ throughput using the original model [109] that assumes all users can sense each other, i.e., no hidden users. Here, the grouping decision is generated by using the iterative algorithm developed in [32]. Specifically, In each iteration, the algorithm selects the best user among all un-grouped users and the best grouping decision of the selected user to maximise the worst-case user throughput in grouped users. Then, the selected user is marked

## 5.6. Evaluation of Proposed Methods

as grouped, and another un-grouped user will be selected in the next iteration until all users are grouped. We refer to the above method as the “MC-based” scheme in simulations.

### Applying Graph-based Approach

We also applied the graph-based approach in the grouping problem in RAW. Note that the works in [26] formulate the user grouping problem as a graph colouring problem in which an edge connects the users if the same AP can detect them. Then, connected users must have different colours/groups and be assigned different RAW slots. This approach requires many groups in dense networks since users will be densely connected in the above graph construction. This is unsuitable for the problem in (5.8) where the number of groups/slots in RAW,  $Z$ , is limited. Then, we consider using the max-cut-based schemes in [27], [28], which computes the user grouping decisions as (5.11) but using some manually designed rules to generate the graph’s edge weights. For example, since the user throughput is mostly affected by the contention and interference in the network, we can design the edge weights as  $W_{i,j} = O_{i,j}, \forall i \neq j$ , where  $O_{i,j}$  is computed using the trained inference NN as (5.15), indicating how likely user  $j$  can sense user  $i$ , i.e., how likely user  $i$  can trigger the CSMA/CA contention of user  $i$ . Alternatively, we can set the edge weights as  $W_{i,j} = 1 - O_{i,j}, \forall i \neq j$ , to indicate how likely users are hidden or how likely they will make concurrent transmissions causing interference. Also, we can set edge weights,  $W_{i,j}, \forall i \neq j$ , to indicate the interference caused by user  $i$  to the associated AP of user  $j$ , e.g.,  $W_{i,j} = \phi'_{i,j} = (\mathbf{P}_0/s_{j,\hat{a}(j)})/(\mathbb{N}_0B + \mathbf{P}_0/s_{i,\hat{a}(j)})$ . We referred to the above max-cut-based schemes, e.g., setting  $W_{i,j}$  as  $O_{i,j}$ ,  $1 - O_{i,j}$  and  $\phi'_{i,j}, \forall i \neq j$ , as the “MCON”, “MHID” and “MINT” schemes, respectively.

### Applying ML-based Approach

The ML-based approach is also applied to the problem, where a NN is trained to directly generate grouping decisions,  $\mathbf{z}$ , based on network states,  $\mathbf{S}$ . For example, we use the random-edge GNN (REGNN) design in [29], whose structure of each layer

## 5.6. Evaluation of Proposed Methods

is

$$\mathbf{g}^{l+1} = \sigma\left(\sum_{n=1}^{\xi} \alpha_n^l (\mathbf{I})^n \mathbf{g}^l\right), \quad l = 1, \dots, \chi - 1. \quad (5.47)$$

where  $\mathbf{I}$  is defined in (5.27) and  $\mathbf{g}^i$  are  $K$ -dimensional vectors ( $\mathbf{g}^1$  is set as  $\hat{\mathbf{s}}$  defined in (5.27)). Here,  $\chi$  is the number of layers in the REGNN,  $\alpha_n^l, \forall n, l$ , are trainable parameters in each layer, and  $\sigma$  is the activation function (e.g.,  $\text{ReLU}(\cdot)$ ) of each layer. Note that REGNNs only output one scalar feature for each user. Thus, We encode user grouping decisions  $z_k, k = 1, \dots, K$ , as a binary number, e.g.,  $z_k = 1, 2, 3, 4$  as  $z_k = 00, 01, 10, 11$ , and use two REGNNs with different parameters, where each one REGNN outputs one bit of the grouping decisions. we can also use message-passing GNN (MPGNN) that is studied in [30] to generate the user grouping decisions, e.g.,

$$\mathbf{g}_i^{l+1} = \sigma\left(\text{FNN2}\left(\mathbf{g}_i^l, \max_{j \neq i} \text{FNN1}\left(\mathbf{g}_i^l, s_{i, \hat{a}(j)}\right)\right)\right), \quad l = 1, \dots, \chi - 1, i = 1, \dots, K, \quad (5.48)$$

where FNN1 and FNN2 are two FNNs,  $\mathbf{g}_i^l$  is the user-wise feature of  $i$ -th user in the  $l$ -th layer and  $\mathbf{g}_i^1$  can be set as  $\mathbf{s}_i$  for the problem. We use a policy gradient algorithm [29], [122] to train the REGNN and the MPGNN.

### 5.6.3 Performance of Proposed AC-GRL Algorithms

We evaluate the performance of the AC-GRL algorithm in Algorithm 4 as follows.

#### Performance of the inference NN

We show the training information of the inference NN in Fig. 5.8, which is the first part of Algorithm 4 (lines 2-6). In Fig. 5.8a, the value of the loss function in (5.38) decreases over training steps and converges around 400 steps. Also, we measure the accuracy of the inference NN during training. For example, we sample the binary value  $O'_{i,j}$  from  $O_{i,j}$  computed as (5.15) and measure the probability that  $O'_{i,j}$  is equal to its true value,  $O_{i,j}^*$ , as shown in Fig. 5.8a. The results show the accuracy of the inference NN also converges around 400 steps. Further, we measure the accuracy of

## 5.6. Evaluation of Proposed Methods

the NN separately for two possible values of  $O_{i,j}^*$  in Fig. 5.8b. The results show that the accuracy converges in either case, which implies that the inference NN makes no biased guess on the value of  $O_{i,j}^*$ , e.g., simply guessing one of the possible values, and also implies that the inference NN is well-trained and can be used in the following part of the algorithm.

### Performance of the actor and the critic

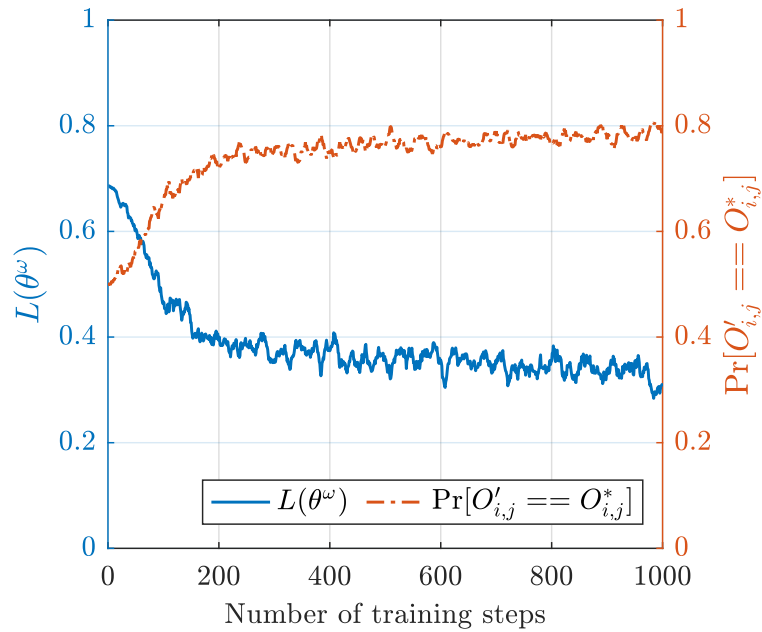
Next, we evaluate the training of the actor and the critic in Algorithm 4 (lines 7-16). We measure the worst-case user throughput,  $\min_k r_k$ , and the average throughput of users,  $\frac{1}{K} \sum_k r_k$ , during training in Fig. 5.9a and Fig. 5.9b, respectively. We compare our AC-GRL method (with legend “Proposed”) with the methods directly using the REGNN or the MPGNN as the user grouping policy. The results show that our method converges after 500 training steps, and it achieves over 80% more throughput than two other schemes without degradation in the average throughput of users. This is because the NNs, REGNN and MPGNN, aggregate all neighbour users’ information of a given user, e.g., due to the matrix-vector multiplication,  $(\mathbf{I})^n \mathbf{g}^l$ , or max function in (5.47) and (5.48), respectively, and thus, they struggle to express the user-pair-wise correlation between grouping decisions, i.e., how likely a pair of users should be assigned in the same slot. However, our method uses a max cut process to generate the user grouping decisions, where the correlation between pairwise of users’ decisions is retained and indicated by the edge weights between each user pair.

#### 5.6.4 Performance of Designed User Grouping Decisions

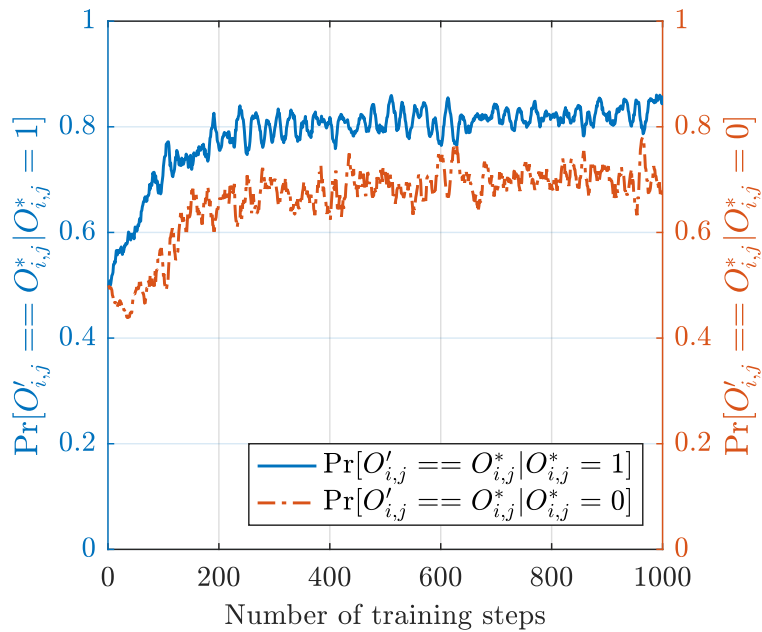
Next, we use the trained actor to generate the graph’s edge weights and compare the user grouping decisions made by the graph cut procedure with the decisions made by other methods. We measure all users’ throughput and the worst-case user’s throughput in 1000 random realisations of the network and plot the cumulative distribution functions (CDFs) of them in Fig. 5.10. Fig. 5.10a shows the CDFs of user throughput achieved by the trained NN in our methods (with legend “Proposed”) and achieved by



## 5.6. Evaluation of Proposed Methods



(a) The value of the loss function and the accuracy of the inference NN during training.

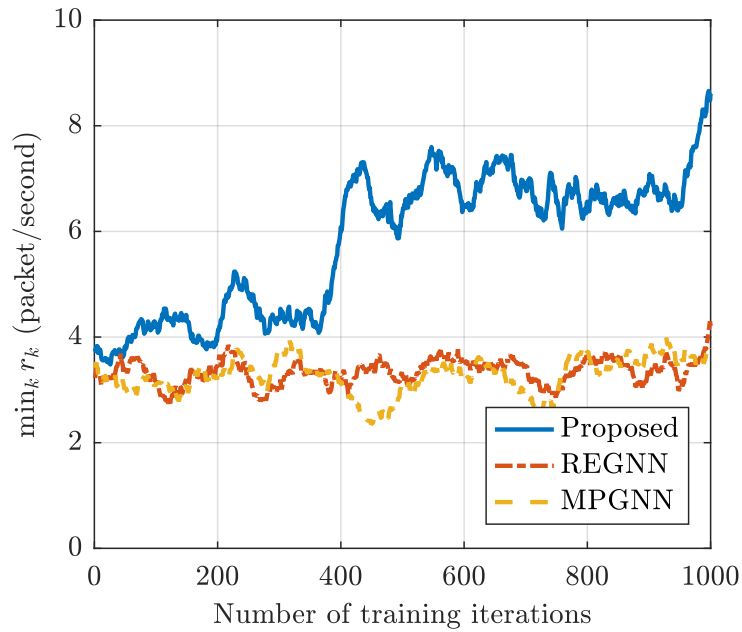


(b) The accuracy of the inference NN for two different cases of  $O_{i,j}^*$ ,  $\forall i, j$ , during training.

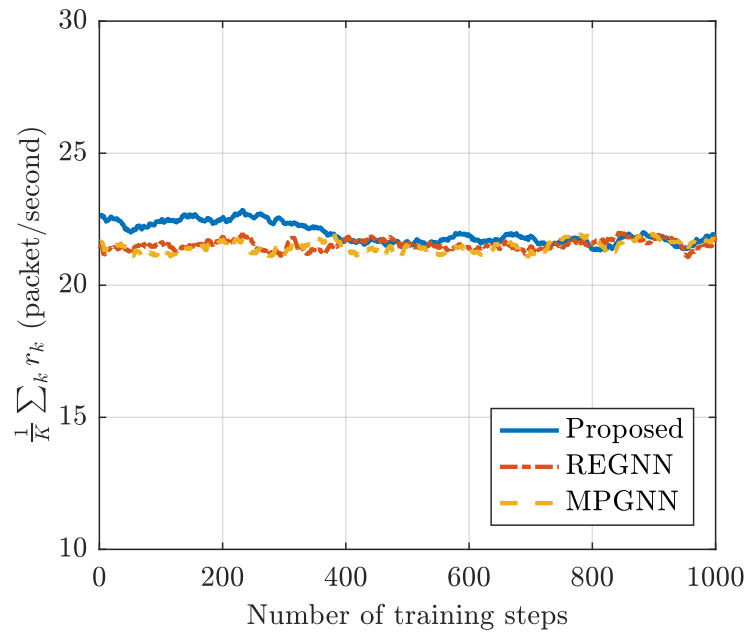
Figure 5.8: The training information of the inference NN.

heuristic schemes, RAND and UNIF, and the MC-based schemes. The results show that the RAND heuristic scheme performs the worst since it randomly allocates users in RAW slots, which can possibly allocate highly contended or interfered users into the

## 5.6. Evaluation of Proposed Methods



(a) The worst-cause user throughput during training.

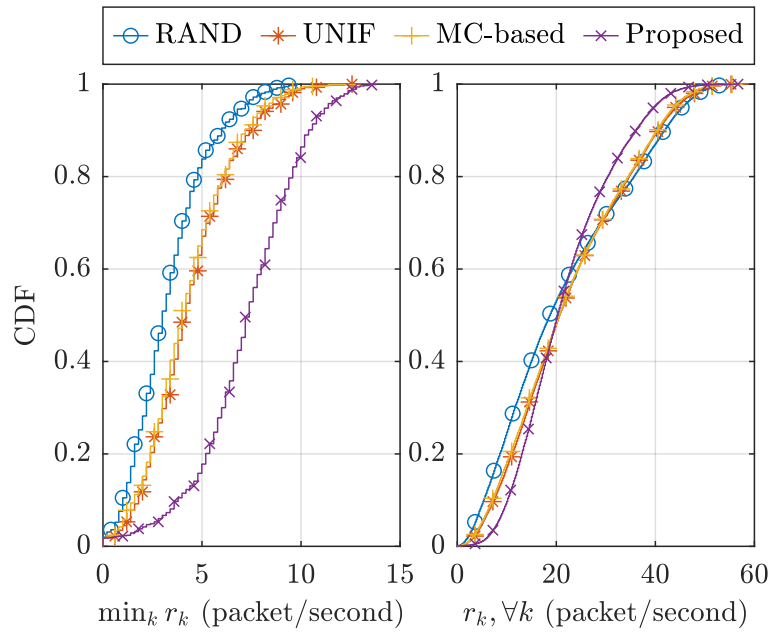


(b) The averaged user throughput during training.

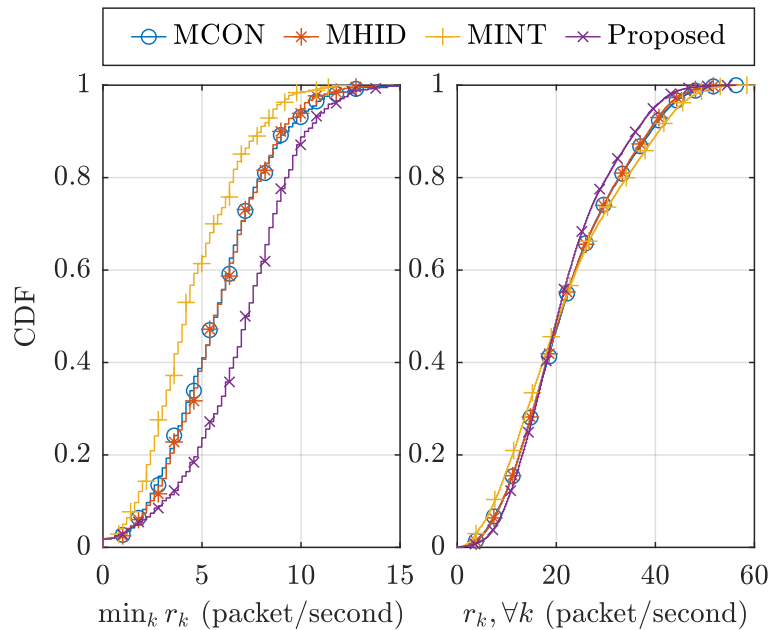
Figure 5.9: The training information of the actor and the critic.

same slot. Also, the UNIF and the MC-based schemes have close performance because both methods have a strategy that balances the number of transmissions in each slot. Further, the proposed method performs the best, e.g., 65% ~ 100% higher worst-cast

## 5.6. Evaluation of Proposed Methods



(a) Comparison with heuristic and Markov-model-based methods



(b) Comparison with max-cut-based methods

Figure 5.10: Comparison of the proposed method with other methods.

user throughput on average, which is due to well-exploitation on the contention and interference information in the optimised edge weights. Further, we compare the CDF of user throughput achieved by our method and achieved by max-cut-based methods

## 5.6. Evaluation of Proposed Methods

in Fig. 5.10b. The results show that the MCON and MHID schemes achieve higher worst-case user throughput than the other max-cut-based method, MINT. This is because MCON and MHID aim to cut more edges between either contending or hidden users (the two main causes of throughput starvation), which is not considered in MINT. Also, it is shown that our proposed methods achieve better worst-case user throughput than the MCON and MHID scheme (around 30% on average). This is because our method considers the path losses from users to APs as the input when deciding the edge weights, whose values thus have more information on the contention and interference in the network, e.g., the interference power/duration, etc. Furthermore, the CDFs of all users' throughput in Fig. 5.10a and Fig. 5.10b show that the improvement in the worst-case user's throughput has a minor reduction in users' throughput above the median.

Fig. 5.11 shows the locations of users with low throughput with respect to the location of APs in four schemes, namely RAND, MC-based, MHID, and our scheme. Specifically, we show the locations of users whose throughput is in the range of  $[0, 2.5)$  and  $[2.5, 5)$  in the 1000 random realisations of the network. The results show that our methods significantly reduce the number of users with low throughput and improve the users' throughput far from the APs, e.g., on the boundary of the simulated area.

### 5.6.5 Performance of Proposed Online Architecture

We then evaluate the performance of the proposed online architecture that fine-tunes the edge weights. We measure the users' throughput and update the weights as Section 5.5 every 2 seconds (i.e., every 200 RAW slots) for 200 seconds. We use the NNs trained with 20 users to generate the edge weights. Then, we use the architecture to fine-tune the edge weights in networks with 10, 20 and 40 users, as shown in Fig. 5.12a, 5.12b and 5.12c, respectively. Specifically, we perform the moving average of user throughput measured every 10 seconds. Then, we compute the ratio of average throughput during 200 seconds to their initial value in the first 10 seconds. We plot

5.6. Evaluation of Proposed Methods

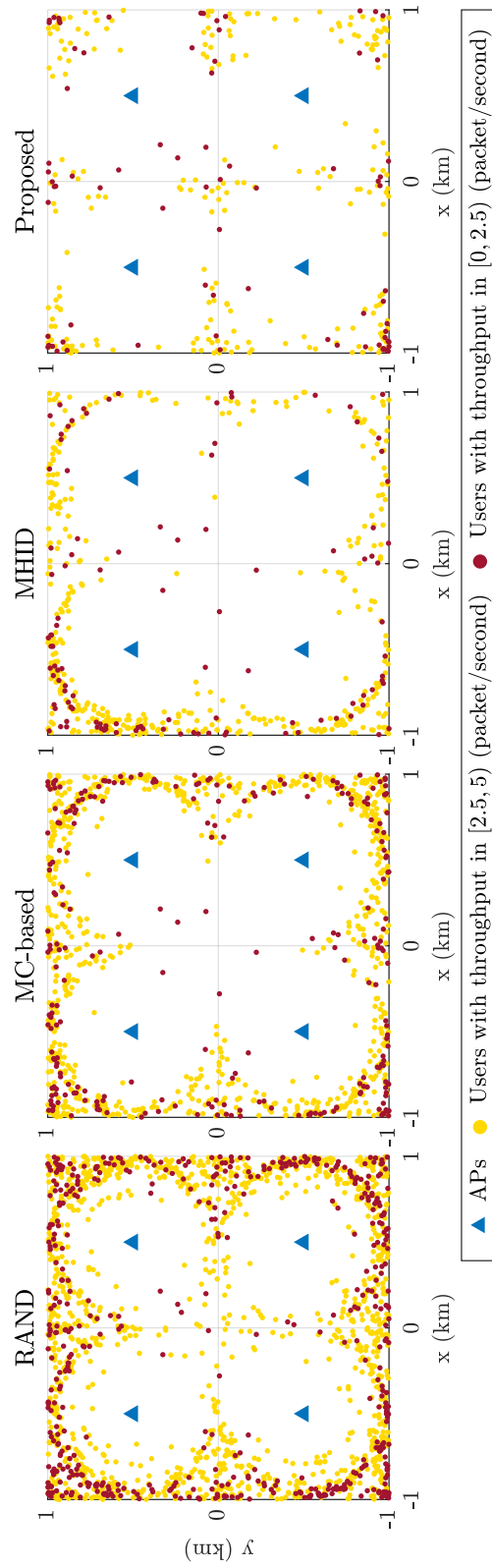


Figure 5.11: Locations of low-throughput users in the simulated area.

## 5.6. Evaluation of Proposed Methods

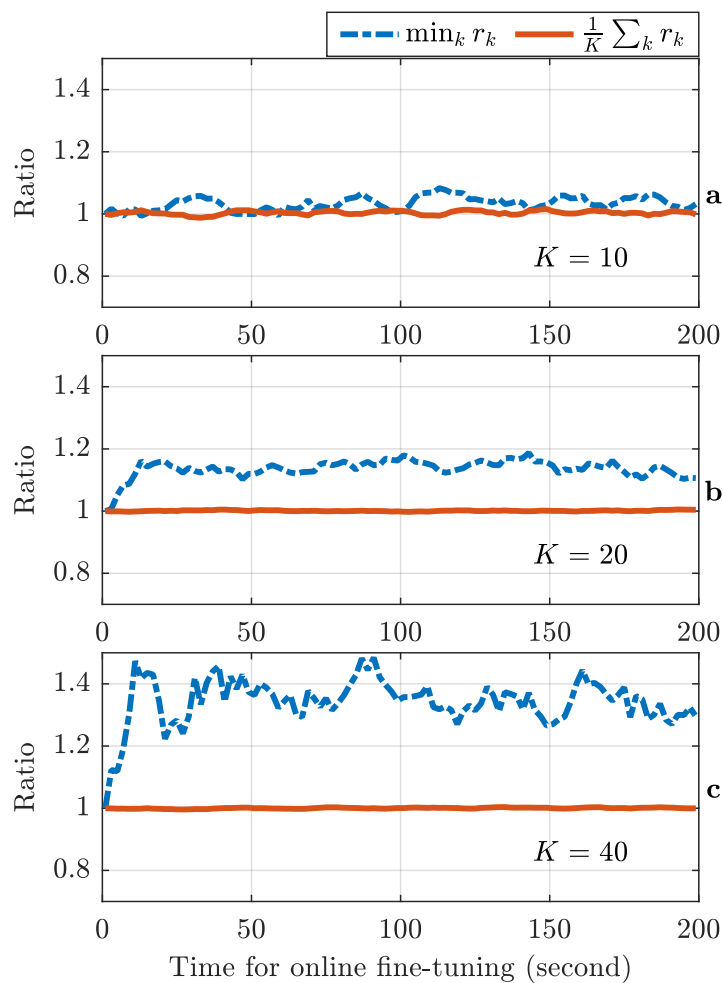


Figure 5.12: Ratios of worst-case and average user throughput to their initial values during fine-tuning for different numbers of users,  $K$ .

the above ratios for both the worst-case user throughput and the average throughput (with legends “ $\min_k r_k$ ” and “ $\frac{1}{K} \sum_k r_k$ ”, respectively), and we further average the above ratios in 100 realisations of networks for each case. The results show that the proposed architecture can further improve 15 ~ 20% of the worst-case user’s throughput compared to its initial value when 20 users are in the network. Also, the results show that when the number of users is small (e.g., 10 users), there is no significant improvement, while a large improvement, 30 ~ 40%, of the worst-case user’s throughput is achieved when more users are in the network (e.g., 40 users). This is because when there are fewer users, users are sparsely located in the area, causing less contention and interference with each other. Thus, the margin of system

### 5.7. Summary

performance is small. Meanwhile, when more users exist in the network, they are more densely located and make heavy contention and interference in the network, where fine-tuning the edge weights can significantly improve the network performance. Also, the results show that fine-tuning the edge weights does not decrease the average user throughput. Further, the results also indicate that the proposed method can optimise the network performance for the varying number of users, which validates the scalability of the method.

## 5.7 Summary

In this chapter, we have designed the GRL method that trains NN to generate edge weights representing the contention and interference between each user pair. Such a weighted graph is then cut to divide users into multiple groups, and each user group transmits their packets in a periodical RAW slot. We optimise the edge weights such that the grouping decisions can maximise the worst-case user throughput. Simulation results show that our method can significantly improve the worst-case user throughput compared to existing methods. We also developed the architecture that fine-tunes the edge weights according to online measurements, further improving the worst-case user throughput compared to the one achieved by edge weights that offline-trained NN generates.

# Chapter 6

## Conclusion

*In this thesis, we studied scheduler design methods in wireless networks. We proposed several schemes to enable automatic scheduler design with flexibility, to improve the convergence rate of the scheduler design algorithm and to coordinate scheduling decisions across the network. This chapter provides a summary of the content, contributions and results in the thesis, and sheds light on potential future directions.*

### 6.1 Summary of Content and Results

In Chapter 3, we developed the DRL algorithm, K-DDPG, in wireless scheduler design for time-sensitive traffic in 5G NR. We found that the straightforward implementation of DDPG converges slowly, has poor QoS performance, and can hardly be implemented in real-world 5G NR systems. We first proposed a T-DRL framework based on the theoretical models and results to address these issues. Then, different kinds of expert knowledge of the scheduler design problem were exploited to reduce the convergence time and improve each user's individual QoS. Furthermore, we developed an online DDPG architecture that enables offline initialisation and online fine-tuning. Our simulation and experimental results indicated that by using K-DDPG in the T-DRL framework, the convergence time and the individual QoS of each user can be improved significantly. In addition, with our online architecture, the scheduling policy can be



## 6.1. Summary of Content and Results

updated according to real-world feedback every few milliseconds and executed in each TTI in 5G NR.

Next, in Chapter 4, we proposed a method to design schedulers using limited prior knowledge of statistical CSI. Specifically, we considered the optimisation of weights in MWSs. We computed MWSs' average rates by solving the rate estimation problem based on the mean and variance of users' SNRs. We formulated the MVWO problem based on the estimated MWSs' rates and proposed an iterative solver, where the iterated weights are proved to converge to the optimal weights. Also, we designed an online architecture to apply our MVWO method in networks with varying SNRs' mean and variance. We conducted simulations to validate the accuracy of the rate estimation, the convergence of the proposed solver, and the optimality of the weights designed by our MVWO method. Simulations show that our MVWO method consumes 4 ~ 10 times fewer time slots in finding the optimal weights and achieves 5 ~ 15% better average data rates of users than SUWO methods.

Finally, In Chapter 5, we investigate the coordination of the scheduling decisions of RAW slots for multiple APs. We studied how to use the RAW mechanism in Wi-Fi HaLow to improve worst-case user throughput. We proposed the framework that formulates the user grouping problem in the RAW slot assignment as the graph construction problem. Here, the graph's edge weights are adjusted to represent the contention and interference in each user pair, and the graph's max cut can obtain the user grouping decisions. We developed the AC-GRL algorithm to train NNs that generate the optimal edge weights based on users' path losses measured at AP. Further, we designed the architecture to fine-tune the edge weights generated by trained NNs according to online feedback. Simulation results show that our approach achieves much better worst-case user throughput than the existing approaches. Also, our online architecture can further improve worst-case user throughput by fine-tuning the NN-generated edge weights.

We also summarised the connections between proposed schedulers. Specifically, the proposed DRL-based and MVWO-based schedulers are designed for per-BS scheduling

problems. They can be combined to flexibly approximate schedulers using NNs and accelerate the NN training process using the statistics of channels. Furthermore, since the proposed GRL-based scheduler coordinates the radio resource allocation across multiple BSs (or APs), we can use it to decide the available resources for the proposed DRL-based and MVWO-based schedulers in multiple BSs (or APs).

## 6.2 Future Directions

To conclude this thesis, we list several promising research directions from the works conducted herein.

### 6.2.1 Extension to DRL-Based Scheduler Design

The current DRL-based scheduler uses an FNN to approximate the scheduling policy that is not scalable in the user population, i.e., the FNNs need to be trained for each number of users. We can extend the work by using GNNs as the policy instead. As a result, a trained policy can be used for different numbers of users. We can also extend the DRL-based scheduler design to 5G networks with multiple network services. One possible approach is to define the system's reward as the weighted sum of utility functions of different service classes. We can define one reward function for each class of users (or flows) and combine reward functions by scalarisation (e.g., linear weighted average). The state and the action are the available measurements, e.g., delays and data rate, and the scheduling decisions, e.g., indicators of which user (or flow) transmits or when the user (the flow) transmits. In order to train NNs that can differentiate multiple classes (or flows), the NNs will be constructed using a shared part and dedicated parts for different classes (or flows), where each dedicated part approximates one class's utility function or scheduling policy. The current scheduler is designed for orthogonal user transmissions, while we can extend it to non-orthogonal multiple access (NOMA). In such a case, the rewards are the summation of both primary and secondary users in NOMA, and scheduling actions can be designed to

## 6.2. Future Directions

select which users are primary or secondary [123]. Also, we have shown how much training time can be saved by offline initialisation in the prototype, compared with direct online training with un-initialised NNs. We can apply other learning methods intended to speed up the adaption of NN to the changing environment. For example, few-shot learning can be used by adjusting NNs within a few transitions, reducing the online training time further [84]. Additionally, the transition samples can be collected from multiple BSs over the whole network. This can introduce significant communication, computation, and storage loads on edge servers. How to efficiently manage these loads requires further study.

### 6.2.2 Extension to MVWO-Based Scheduler Design

The MVWO-based scheduler design only considers the variance of each user's CSI. We can extend it to the case where the users' channels are correlated. In other words, the off-diagonal elements of the covariance matrix of users' CSI are non-zero. Efficient covariance estimation methods can be applied to reduce the memory requirements and the computational complexity [124]. For multiple service classes, we need to express each class's utility function in terms of the scheduling actions and the correlation between scheduling actions and CSIs. Then, the MVWO problem can be formulated and solved to find the optimal weights. Similarly, for NOMA, the utility function needs to be expressed in the data rate of primary and secondary users, and the correlation of the scheduling decisions on primary and secondary users needs to be expressed in the convex constraints of the MVWO problem. Also, the network state's correlation over time, including the queue state and the channel state, can be considered in formulating the feasible rate region, which may lead to the time-efficient scheduler design for more complicated utility functions. Finally, our works only give the boundness of the feasible rate region while it is not studied how tight the bound is. Note that the core of estimating the feasible rate region is the SDP problem. It might be possible to design a rounding method [25] to find the approximation ratio of MVWO.

### 6.2.3 **Extension to GRL-Based Scheduler Design**

We can extend the GRL-based scheduler design to consider more network performance objectives, e.g., latency and jitter, and study how to design the graph representation of the network in these cases. Moreover, we can extend the work to support multiple service classes based on the extension methods for the DRL-based scheduler mentioned before. Also, note that the proposed method requires measurement of the existence of the hidden users. We can investigate the method that can efficiently measure this information from the network with minimum overheads and keep the algorithm continuously optimising the network simultaneously. Further, we can design a two-level learning algorithm by combining the previously developed DRL-based and MVWO-based methods to exploit the network states' short-term variation, e.g., queue and channel states. In such a case, how to design the interaction between these two levels of the algorithm requires further study.

# Appendix A

## Proofs of Chapter 3

### Proof of the Markov Property of the System

To apply DRL, we prove the Markov property in this subsection. We first derive the transition probability of HoL delay.

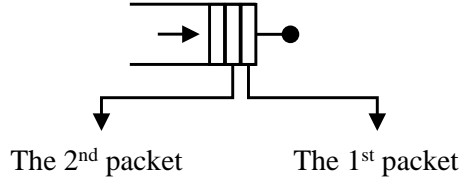


Figure A.1: Illustration of the queuing model.

If the  $k$ -th user is scheduled in the  $t$ -th slot, the transition probability is denoted by  $p_{i,j}^{k,+} = \Pr\{d_k(t+1) = j \mid d_k(t) = i, x_k(t) = 1\}$ , where  $i$  is the HoL delay in the  $t$ -th slot and  $j$  is the HoL delay in the  $t+1$ -th slot. Since users with empty buffers will not be scheduled, we have  $i > 0$ . To derive the transition probability, we consider the following three cases: 1)  $0 < i < j$ , 2)  $0 < j \leq i$  and 3)  $0 = j < i$ . As shown in Fig. A.1, the queuing delays of the first and the second packets in the  $t$ -th slot are  $i$  and  $j-1$ , respectively. The inter-arrival time between the first and the second packet is  $i - (j-1)$ . Since the inter-arrival time is strictly positive, we have  $i - (j-1) > 0$  and  $j \leq i$ . This means that for all  $0 < i < j$ ,  $p_{i,j}^{k,+} = 0$ . For the case  $0 < j \leq i$ ,  $p_{i,j}^{k,+}$

equals to the probability that the second packet arrived at the buffer  $i - (j - 1)$  slots later than the first packet. For the Bernoulli arrival process,  $p_{i,j}^{k,+} = p_k(1 - p_k)^{i-j}$ . For the case  $j = 0$ , the buffer becomes empty in the  $(t + 1)$ -th slot. It means that no packet arrived at the buffer during the past  $i$  slots. Thus,  $p_{i,j}^{k,+} = (1 - p_k)^i$ .

If the  $k$ -th user is not scheduled in the  $t$ -th slot, the transition probability is denoted by  $p_{i,j}^{k,-} = \Pr\{d_k(t + 1) = j \mid d_k(t) = i, x_k(t) = 0\}$ . To derive  $p_{i,j}^{k,-}$ , we consider three cases: 1) the buffer is empty,  $i = 0$ , 2)  $0 < i < D_{\max}$  and 3)  $i = D_{\max}$ . When the buffer is empty in the  $t$ -th slot,  $i = 0$ . With probability  $p_k$ , a packet arrives at the buffer in the  $t$ -th slot and  $j = 1$ . Otherwise,  $j = 0$ . When the HoL delay is smaller than the maximum delay bound,  $0 < i < D_{\max}$ , the HoL delay will increase by one slot. When the HoL delay equals the maximum delay bound,  $i = D_{\max}$ , the first packet will be discarded. The HoL delay in the next slot depends on the queuing delay of the second packet. If the second packet arrived within the previous  $D_{\max}$  slots,  $p_{i,j}^{k,-} = p_k(1 - p_k)^{D_{\max}-j}$ . Otherwise,  $p_{i,j}^{k,-} = (1 - p_k)^{D_{\max}}$ .

Since the above transition probabilities only depend on the states and actions in the  $t$ -th slot, the HoL delay is Markovian. By assuming that the wireless channel fading is Markovian, the problem is an optimal control problem of a Markov decision process. This completes the proof.

# Appendix B

## Proofs of Chapter 4

### Appendix: The Proof of Corollary 1

*Proof.* By applying the linearity of the inner product, we obtain

$$\begin{aligned} \langle \mathbf{w}, \mathbf{r}^{\sim\mu(\cdot|\mathbf{w})} \rangle &= \langle \mathbf{w}, \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^T \mathbf{v}(\mu(\mathbf{s}(t)|\mathbf{w}), \mathbf{s}(t)) \rangle \\ &= \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^T \langle \mathbf{w}, \mathbf{v}(\mu(\mathbf{s}(t)|\mathbf{w}), \mathbf{s}(t)) \rangle \stackrel{(a)}{\geq} \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^T \langle \mathbf{w}, \mathbf{v}(\omega(\mathbf{s}(t)), \mathbf{s}(t)) \rangle \quad (\text{B.1}) \\ &= \langle \mathbf{w}, \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^T \mathbf{v}(\omega(\mathbf{s}(t)), \mathbf{s}(t)) \rangle = \langle \mathbf{w}, \mathbf{r} \rangle, \forall \mathbf{r} \in \mathcal{F}, \end{aligned}$$

where (a) uses the fact that  $\mu(\cdot|\mathbf{w})$  maximises  $\langle \mathbf{w}, \mathbf{v}(\mathbf{x}(t), \mathbf{s}(t)) \rangle$  in each slot, as shown in (4.11). Thus,  $\mathbf{r}^{\sim\mu(\cdot|\mathbf{w})}$  maximises  $\langle \mathbf{w}, \mathbf{r} \rangle$  among all other elements in  $\mathcal{F}$ , which proves the statement.  $\square$

## Appendix: Proof of the Boundedness of the Bounding Set

*Proof.* For a given  $k \in \{1, \dots, K\}$ , let the elements of a vector,  $\mathbf{z}$ , be 0 except its  $k$ -th and  $(k + K)$ -th element,  $z_k$  and  $z_{k+K}$ . Then, the positive semi-definiteness of  $\mathbf{H}$  leads to

$$\mathbf{z}^T \mathbf{H} \mathbf{z} = (z_k)^2 H_{k,k}^{xx} + 2z_k z_{k+K} H_{k,k}^{x\phi} + (z_{k+K})^2 H_{k,k}^{\phi\phi} \geq 0, \quad \forall (z_k, z_{k+K}) \in \mathbb{R}^2, \forall k, \quad (\text{B.2})$$

which implies  $H_{k,k}^{xx} H_{k,k}^{\phi\phi} \geq (H_{k,k}^{x\phi})^2, \forall k$ . By substituting (C5)(C6)(C7) into the above, we obtain

$$p_k(1 - p_k) \cdot v_k^\phi \geq (y_k - p_k m_k^\phi)^2, \quad \Rightarrow \quad y_k \leq \sqrt{p_k(1 - p_k) \cdot v_k^\phi + p_k m_k^\phi}, \quad \forall k. \quad (\text{B.3})$$

Consider the inequality in (C2), where we assume  $\Delta_0 = 1$  and  $B = 1$  in order to simplify the notation without generality,

$$\begin{aligned} r_k &\leq p_k \log_2 \left( 1 + \frac{y_k}{p_k} \right) \stackrel{\text{(a)}}{\leq} p_k \log_2 \left( 1 + \frac{[\sqrt{p_k(1 - p_k) \cdot v_k^\phi + p_k m_k^\phi}]}{p_k} \right) \\ &\stackrel{\text{(b)}}{\leq} p_k \log_2 \left( 1 + \frac{\sqrt{v_k^\phi + m_k^\phi}}{p_k} \right), \end{aligned} \quad (\text{B.4})$$

where (a) uses the inequality in (B.3) and (b) uses the fact that  $p_k \leq 1$  and  $1 - p_k \leq 1$  in the numerator of the fraction. We write  $\alpha_k \triangleq \sqrt{v_k^\phi + m_k^\phi}$  to simplify the above notation as

$$r_k \leq p_k \log_2 \left( 1 + \frac{\alpha_k}{p_k} \right) = (p_k + \alpha_k) \log_2(p_k + \alpha_k) - \alpha_k \log_2(p_k + \alpha_k) - p_k \log_2(p_k), \quad \forall k. \quad (\text{B.5})$$



Because  $(p_k + \alpha_k) \log_2(p_k + \alpha_k) \leq (1 + \alpha_k) \log_2(1 + \alpha_k)$ ,  $\alpha_k \log_2(p_k + \alpha_k) \geq \alpha_k \log_2(\alpha_k)$  and  $p_k \log_2(p_k) \geq \frac{1}{e} \log_2 \frac{1}{e}$ , we obtain an upper bound on  $r_k$  as

$$r_k \leq (1 + \alpha_k) \log_2(1 + \alpha_k) - \alpha_k \log_2(\alpha_k) - \frac{1}{e} \log_2 \frac{1}{e}, \forall k. \quad (\text{B.6})$$

Also,  $r_k \geq 0$ ,  $\forall k$ , as shown in (C1), which implies  $r_k$  is bounded  $\forall k$ . Since all dimensions of  $\mathbf{r}$  are bounded,  $\mathbf{r}$  is bounded and so is  $\mathcal{G}$ .  $\square$

## Appendix: Proof of Corollary 2

We first check the sign of  $a^{(i)}$  and  $b^{(i)}$ . Note that  $\mathbf{r}^{(i)}$  is the optimal solution of (4.31), which implies  $\langle \mathbf{w}^{(i)}, \mathbf{r}^{(i)} \rangle > \langle \mathbf{w}^{(i)}, \mathbf{r}^* \rangle$ . Based on this fact, we can determine that  $a^{(i)} > 0$ . Suppose  $b^{(i)} \leq 0$ , then  $\min\{(\mathbf{r}^* - \mathbf{r}^{(i)}) \oslash \mathbf{w}^{(i)}\} \geq 0$ . This implies  $(\mathbf{r}^* - \mathbf{r}^{(i)}) \in \mathbb{R}_{\geq 0}^K$  and  $\langle \mathbf{w}^{(i)}, \mathbf{r}^* - \mathbf{r}^{(i)} \rangle \geq 0$ , which is contradictory to the optimality of  $\mathbf{r}^{(i)}$ . Therefore,  $b^{(i)} > 0$ .

Based on sign of  $a^{(i)}$  and  $b^{(i)}$ , we can prove that  $\mathbf{u}^{(i)} \in \mathbb{R}_{>0}^K$ . To achieve this, we first check the sign of the elements in  $b^{(i)}\mathbf{w}^{(i)} + (\mathbf{r}^* - \mathbf{r}^{(i)})$ , whose  $k$ -th element is

$$\begin{aligned} b^{(i)}w_k^{(i)} + r_k^* - r_k^{(i)} &= w_k^{(i)}(b^{(i)} + \frac{r_k^* - r_k^{(i)}}{w_k^{(i)}}) \\ &\geq w^{(i)}(b^{(i)} + \min\{(\mathbf{r}^* - \mathbf{r}^{(i)}) \oslash \mathbf{w}^{(i)}\}) = 0, \forall k. \end{aligned} \quad (\text{B.7})$$

This implies  $b^{(i)}\mathbf{w}^{(i)} + (\mathbf{r}^* - \mathbf{r}^{(i)}) \in \mathbb{R}_{\geq 0}^K$ . By adding  $a^{(i)}\mathbf{w}^{(i)}$  to the above, we obtain that  $\mathbf{u}^{(i)} \in \mathbb{R}_{>0}^K$ . Also,  $\mathbf{w}^{(i+1)}$  is normalised  $\mathbf{u}^{(i)}$ , which implies  $\mathbf{w}^{(i+1)} \in \mathbb{R}_{>0}^K$  and  $\|\mathbf{w}^{(i+1)}\|_2 = 1$ .

## Appendix: Proof of Lemma 1

*Proof.* To prove the first statement, we substitute (4.32) and (4.34) into  $\langle \mathbf{w}^{(i+1)}, \mathbf{w}^* \rangle$  as

$$\begin{aligned} \langle \mathbf{w}^{(i+1)}, \mathbf{w}^* \rangle &= \left\langle \frac{a^{(i)} + b^{(i)}}{\|\mathbf{u}^{(i)}\|_2} \mathbf{w}^{(i)} + \frac{1}{\|\mathbf{u}^{(i)}\|_2} (\mathbf{r}^* - \mathbf{r}^{(i)}), \mathbf{w}^* \right\rangle \\ &= \frac{a^{(i)} + b^{(i)}}{\|\mathbf{u}^{(i)}\|_2} \langle \mathbf{w}^{(i)}, \mathbf{w}^* \rangle + \frac{1}{\|\mathbf{u}^{(i)}\|_2} \langle \mathbf{r}^* - \mathbf{r}^{(i)}, \mathbf{w}^* \rangle \stackrel{(a)}{>} \frac{a^{(i)} + b^{(i)}}{\|\mathbf{u}^{(i)}\|_2} \langle \mathbf{w}^{(i)}, \mathbf{w}^* \rangle, \end{aligned} \quad (\text{B.8})$$

where (a) is because  $\langle \mathbf{w}^*, \mathbf{r}^* \rangle > \langle \mathbf{w}^*, \mathbf{r}^{(i)} \rangle$ . Also, we have

$$\begin{aligned} \langle \mathbf{u}^{(i)}, \mathbf{r}^* - \mathbf{r}^{(i)} \rangle &= \langle (a^{(i)} + b^{(i)}) \mathbf{w}^{(i)} + (\mathbf{r}^* - \mathbf{r}^{(i)}), \mathbf{r}^* - \mathbf{r}^{(i)} \rangle \\ &= \frac{\|\mathbf{r}^* - \mathbf{r}^{(i)}\|_2^2}{\langle \mathbf{w}^{(i)}, \mathbf{r}^{(i)} - \mathbf{r}^* \rangle} \langle \mathbf{w}^{(i)}, \mathbf{r}^* - \mathbf{r}^{(i)} \rangle + \langle b^{(i)} \mathbf{w}^{(i)}, \mathbf{r}^* - \mathbf{r}^{(i)} \rangle + \|\mathbf{r}^* - \mathbf{r}^{(i)}\|_2^2 \\ &= 0 + b^{(i)} \langle \mathbf{w}^{(i)}, \mathbf{r}^* - \mathbf{r}^{(i)} \rangle \stackrel{(a)}{<} 0, \end{aligned} \quad (\text{B.9})$$

where (a) uses the fact that  $\langle \mathbf{w}^{(i)}, \mathbf{r}^* \rangle < \langle \mathbf{w}^{(i)}, \mathbf{r}^{(i)} \rangle$ . The square of  $\ell_2$ -norm of  $\mathbf{u}^{(i)}$  is

$$\begin{aligned} \|\mathbf{u}^{(i)}\|_2^2 &= \langle \mathbf{u}^{(i)}, \mathbf{u}^{(i)} \rangle = \langle (a^{(i)} + b^{(i)}) \mathbf{w}^{(i)} + (\mathbf{r}^* - \mathbf{r}^{(i)}), \mathbf{u}^{(i)} \rangle \\ &\stackrel{(a)}{<} \langle (a^{(i)} + b^{(i)}) \mathbf{w}^{(i)}, \mathbf{u}^{(i)} \rangle = (a^{(i)} + b^{(i)})^2 + (a^{(i)} + b^{(i)}) \langle \mathbf{w}^{(i)}, \mathbf{r}^* - \mathbf{r}^{(i)} \rangle \end{aligned} \quad (\text{B.10})$$

where (a) uses the inequality in (B.9). By dividing  $(a^{(i)} + b^{(i)})^2$  at each term in the last inequality, we obtain

$$\begin{aligned} \left( \frac{\|\mathbf{u}^{(i)}\|_2}{a^{(i)} + b^{(i)}} \right)^2 &< 1 + \frac{\langle \mathbf{w}^{(i)}, \mathbf{r}^* - \mathbf{r}^{(i)} \rangle}{a^{(i)} + b^{(i)}} \\ \Rightarrow \left( \frac{\|\mathbf{u}^{(i)}\|_2}{a^{(i)} + b^{(i)}} \right)^2 &< 1 - \frac{(\langle \mathbf{w}^{(i)}, \mathbf{r}^{(i)} - \mathbf{r}^* \rangle)^2}{(a^{(i)} + b^{(i)}) \langle \mathbf{w}^{(i)}, \mathbf{r}^{(i)} - \mathbf{r}^* \rangle}, \end{aligned} \quad (\text{B.11})$$

and substituting  $a^{(i)}$  and  $b^{(i)}$  of (4.33) into the denominator in the RHS of the above

$$\begin{aligned}
& (a^{(i)} + b^{(i)}) \langle \mathbf{w}^{(i)}, \mathbf{r}^{(i)} - \mathbf{r}^* \rangle \\
&= \|\mathbf{r}^* - \mathbf{r}^{(i)}\|_2^2 + \langle b^{(i)} \mathbf{w}^{(i)}, \mathbf{r}^{(i)} - \mathbf{r}^* \rangle = \|\mathbf{r}^* - \mathbf{r}^{(i)}\|_2^2 + \sum_k b^{(i)} w_k^{(i)} (r_k^{(i)} - r_k^*) \\
&< \|\mathbf{r}^* - \mathbf{r}^{(i)}\|_2^2 + \sum_{k:r_k^{(i)} - r_k^* \leq 0} \frac{r_k^{(i)} - r_k^*}{w_k^{(i)}} w_k^{(i)} (r_k^{(i)} - r_k^*) + \sum_{k:r_k^{(i)} - r_k^* > 0} b^{(i)} w_k^{(i)} (r_k^{(i)} - r_k^*) \quad (\text{B.12}) \\
&\approx \|\mathbf{r}^* - \mathbf{r}^{(i)}\|_2^2 + \sum_k \frac{r_k^{(i)} - r_k^*}{w_k^{(i)}} w_k^{(i)} (r_k^{(i)} - r_k^*) = 2 \|\mathbf{r}^* - \mathbf{r}^{(i)}\|_2^2,
\end{aligned}$$

where we note that both  $\mathbf{r}^{(i)}$  and  $\mathbf{r}^*$  are vectors in  $\mathcal{G}$ , which implies  $\|\mathbf{r}^* - \mathbf{r}^{(i)}\|_2 \leq \hat{R}$  based on the definition of  $\hat{R}$ . By applying (B.12) into (B.11), we can obtain

$$\begin{aligned}
& \left( \frac{\|\mathbf{u}^{(i)}\|_2}{a^{(i)} + b^{(i)}} \right)^2 < 1 - \frac{(\langle \mathbf{w}^{(i)}, \mathbf{r}^{(i)} - \mathbf{r}^* \rangle)^2}{2\hat{R}^2} < 1, \\
& \Rightarrow \frac{\|\mathbf{u}^{(i)}\|_2}{a^{(i)} + b^{(i)}} < \left[ 1 - \frac{(\langle \mathbf{w}^{(i)}, \mathbf{r}^{(i)} - \mathbf{r}^* \rangle)^2}{2\hat{R}^2} \right]^{\frac{1}{2}} < 1.
\end{aligned} \quad (\text{B.13})$$

Note that  $\hat{R}$  is finite because  $\mathcal{G}$  is bounded. By substituting (B.13) in (B.8), we obtain (4.35).

For the second statement, the inner product between  $\mathbf{w}^{(i)}$  and  $\mathbf{w}^*$  is

$$\langle \mathbf{w}^{(i)}, \mathbf{w}^* \rangle \stackrel{(a)}{\geq} \langle \mathbf{w}^{(1)}, \mathbf{w}^* \rangle = \frac{1}{\sqrt{K}} \sum_{k=1}^K w_k^* \stackrel{(b)}{\geq} \frac{1}{\sqrt{K}} \sum_{k=1}^K (w_k^*)^2 = \frac{1}{\sqrt{K}}, \quad (\text{B.14})$$

where (a) uses the fact that  $\langle \mathbf{w}^{(i)}, \mathbf{w}^* \rangle$  is monotonic increasing based on the first statement in Lemma 1 and (b) is because  $w_k$  is less than or equal to 1,  $k = 1, \dots, K$ . Further, by applying Cauchy–Schwarz inequality, we obtain that  $\langle \mathbf{w}^{(i)}, \mathbf{w}^* \rangle = |\langle \mathbf{w}^{(i)}, \mathbf{w}^* \rangle| \leq \|\mathbf{w}^{(i)}\|_2 \|\mathbf{w}^*\|_2 = 1$ .

□

# Appendix C

## Proofs of Chapter 5

### Appendix: The Proof of Lemma 2

*Proof.* We prove the statement by contradiction. Suppose the statement is false, which means there exist grouping decisions,  $\mathbf{z}'$ , maximising the objective in (5.8) other than  $\mathbf{z}^*$ . Then, we construct edge weights as  $W'_{i,j}, \forall i \neq j$ , where  $W'_{i,j} = \mathbf{1}_{\{z'_i \neq z'_j\}}$ . Then,  $\mathbf{z}'$  is the optimal solution that maximises the LLP of (5.12) for the given edge weights as  $W'_{i,j}, \forall i \neq j$ , which means that  $\mathbf{z}'$  is the grouping decisions if the edge weights are  $W'_{i,j}, \forall i \neq j$ . Note that the network objectives in (5.8) and (5.12) are the same and only depend on the grouping decisions. Thus, the values of the objective achieved by  $\mathbf{z}'$  in (5.8) and  $W'_{i,j}, \forall i \neq j$ , in (5.12) are the same because their grouping decisions are the same. It also implies that the value of the objective achieved by  $\mathbf{z}^*$  in (5.8) is equal to the one achieved by  $W^*_{i,j}, \forall i \neq j$ , in (5.12) since  $\mathbf{z}^*$  is the grouping decisions by cutting  $W^*_{i,j}, \forall i \neq j$ . We note that  $W^*_{i,j}, \forall i \neq j$ , maximises (5.12), which means the objective in (5.12) achieved by  $W^*_{i,j}, \forall i \neq j$ , is greater than or equal to the one in (5.12) achieved by  $W'_{i,j}, \forall i \neq j$ . This implies that the objective achieved by  $\mathbf{z}^*$  in (5.8) is greater than or equal to the one achieved by  $\mathbf{z}'$  in (5.8), which is contradictory to the assumption on the optimality of  $\mathbf{z}'$  and the non-optimality of  $\mathbf{z}^*$  in (5.8) at the start of the proof. Therefore, the statement in Lemma 2 is true.  $\square$

# Bibliography

- [1] 3GPP, “Study on Scenarios and Requirements for Next Generation Access Technologies,” 3GPP, TS 38.913, 2022, v17.0.0.
- [2] T. Barnett, S. Jain, U. Andra, and T. Khurana, “Cisco visual networking index (VNI) complete forecast update, 2017–2022,” *Americas/EMEAR Cisco Knowledge Network (CKN) Presentation*, pp. 1–30, 2018.
- [3] A. Gupta and R. K. Jha, “A survey of 5G network: Architecture and emerging technologies,” *IEEE Access*, vol. 3, pp. 1206–1232, 2015.
- [4] Z. Zhang, Y. Xiao, Z. Ma, M. Xiao, Z. Ding, X. Lei, G. K. Karagiannidis, and P. Fan, “6G wireless networks: Vision, requirements, architecture, and key technologies,” *IEEE Vehicular Technology Magazine*, vol. 14, no. 3, pp. 28–41, 2019.
- [5] T. Adame, A. Bel, B. Bellalta, J. Barcelo, and M. Oliver, “IEEE 802.11 ah: The Wi-Fi approach for M2M communications,” *IEEE Wireless Communications*, vol. 21, no. 6, pp. 144–152, 2014.
- [6] C. Deng, X. Fang, X. Han, X. Wang, L. Yan, R. He, Y. Long, and Y. Guo, “IEEE 802.11 be Wi-Fi 7: New challenges and opportunities,” *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2136–2166, 2020.
- [7] P. Popovski, K. F. Trillingsgaard, O. Simeone, and G. Durisi, “5G wireless network slicing for eMBB, URLLC, and mMTC: A communication-theoretic view,” *IEEE Access*, vol. 6, pp. 55 765–55 779, 2018.
- [8] M. Wollschlaeger, T. Sauter, and J. Jasperneite, “The future of industrial communication: Automation networks in the era of the Internet of Things and

- Industry 4.0,” *IEEE Industrial Electronics Magazine*, vol. 11, no. 1, pp. 17–27, 2017.
- [9] 3GPP, “Service requirements for cyber-physical control applications in vertical domains,” 3GPP, TS 22.104, 2018, v16.0.0.
- [10] P. Popovski, J. J. Nielsen, C. Stefanovic, E. De Carvalho, E. Strom, K. F. Trillingsgaard, A.-S. Bana, D. M. Kim, R. Kotaba, J. Park, *et al.*, “Wireless access for ultra-reliable low-latency communication: Principles and building blocks,” *IEEE Network*, vol. 32, no. 2, pp. 16–23, 2018.
- [11] M. Giordani, M. Polese, M. Mezzavilla, S. Rangan, and M. Zorzi, “Toward 6G networks: Use cases and technologies,” *IEEE Communications Magazine*, vol. 58, no. 3, pp. 55–61, 2020.
- [12] X. Xu, Y. Pan, P. P. M. Y. Lwin, and X. Liang, “3D holographic display and its data transmission requirement,” in *2011 International Conference on Information Photonics and Optical Communications*, IEEE, 2011, pp. 1–4.
- [13] 3GPP. “The 3rd Generation Partnership Project.” (2019), [Online]. Available: <http://www.3gpp.org/about-3gpp>.
- [14] IEEE, “IEEE standard for information technology telecommunications and information exchange between systems local and metropolitan area networks specific requirements part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications,” *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pp. 1–3534, 2016. DOI: 10.1109/IEEESTD.2016.7786995.
- [15] IEEE, “IEEE standard for information technology–telecommunications and information exchange between systems - local and metropolitan area networks–specific requirements - part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications amendment 2: Sub 1 GHz license exempt operation,” *IEEE Std 802.11ah-2016 (Amendment to IEEE Std 802.11-2016, as amended by IEEE Std 802.11ai-2016)*, pp. 1–594, 2017. DOI: 10.1109/IEEESTD.2017.7920364.
- [16] D. Tse and P. Viswanath, *Fundamentals of wireless communication*. Cambridge university press, 2005.

- [17] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press, 2011.
- [18] I. Gomez-Migueluez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith, “srsLTE: An open-source platform for LTE evolution and experimentation,” in *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*, 2016, pp. 25–32.
- [19] D. Tse *et al.*, “Transmitter directed, multiple receiver system using path diversity to equitably maximize throughput,” *patent filed, May*, vol. 24, 1999.
- [20] R. Agrawal and V. Subramanian, “Optimality of certain channel aware scheduling policies,” in *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, Citeseer, vol. 40, 2002, pp. 1533–1542.
- [21] C. D. Kolstad and L. S. Lasdon, “Derivative evaluation and computational experience with large bilevel mathematical programs,” *Journal of Optimization Theory and Applications*, vol. 65, no. 3, pp. 485–499, 1990.
- [22] G. Savard and J. Gauvin, “The steepest descent direction for the nonlinear bilevel programming problem,” *Operations Research Letters*, vol. 15, no. 5, pp. 265–272, 1994.
- [23] B. Colson, P. Marcotte, and G. Savard, “An overview of bilevel optimization,” *Annals of Operations Research*, vol. 153, no. 1, pp. 235–256, 2007.
- [24] A. Sinha, P. Malo, and K. Deb, “A review on bilevel optimization: From classical to evolutionary approaches and applications,” *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 276–295, 2017.
- [25] M. X. Goemans and D. P. Williamson, “Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming,” *Journal of the ACM (JACM)*, vol. 42, no. 6, pp. 1115–1145, 1995.
- [26] Q. Chen, “An energy efficient channel access with target wake time scheduling for overlapping 802.11 ax basic service sets,” *IEEE Internet of Things Journal*, 2022.

- [27] R. Y. Chang, Z. Tao, J. Zhang, and C.-C. J. Kuo, “Multicell OFDMA down-link resource allocation using a graphic framework,” *IEEE Transactions on Vehicular Technology*, vol. 58, no. 7, pp. 3494–3507, 2009.
- [28] L. Liang, S. Xie, G. Y. Li, Z. Ding, and X. Yu, “Graph-based resource sharing in vehicular communication,” *IEEE Transactions on Wireless Communications*, vol. 17, no. 7, pp. 4579–4592, 2018.
- [29] M. Eisen and A. Ribeiro, “Optimal wireless resource allocation with random edge graph neural networks,” *IEEE Transactions on Signal Processing*, vol. 68, pp. 2977–2991, 2020.
- [30] Y. Shen, Y. Shi, J. Zhang, and K. B. Letaief, “Graph neural networks for scalable radio resource management: Architecture design and theoretical analysis,” *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 101–115, 2020.
- [31] NS-3. “Network Simulator 3.” (2022), [Online]. Available: <https://www.nsnam.org/about>.
- [32] T.-C. Chang, C.-H. Lin, K. C.-J. Lin, and W.-T. Chen, “Traffic-aware sensor grouping for IEEE 802.11 ah networks: Regression based analysis and design,” *IEEE Transactions on Mobile Computing*, vol. 18, no. 3, pp. 674–687, 2018.
- [33] C. Kai, J. Zhang, X. Zhang, and W. Huang, “Energy-efficient sensor grouping for IEEE 802.11 ah networks with max-min fairness guarantees,” *IEEE Access*, vol. 7, pp. 102 284–102 294, 2019.
- [34] C. E. Shannon, “A mathematical theory of communication,” *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.
- [35] Y. Polyanskiy, H. V. Poor, and S. Verdú, “Channel coding rate in the finite blocklength regime,” *IEEE Transactions on Information Theory*, vol. 56, no. 5, pp. 2307–2359, 2010.
- [36] T. Erseghe, “Coding in the finite-blocklength regime: Bounds based on Laplace integrals and their asymptotic approximations,” *IEEE Transactions on Information Theory*, vol. 62, no. 12, pp. 6854–6883, 2016.



## Bibliography

- [37] A. M. Tulino, G. Caire, S. Shamai, and S. Verdú, “Capacity of channels with frequency-selective and time-selective fading,” *IEEE Transactions on Information Theory*, vol. 56, no. 3, pp. 1187–1215, 2010.
- [38] W. Yang, G. Durisi, T. Koch, and Y. Polyanskiy, “Quasi-static multiple-antenna fading channels at finite blocklength,” *IEEE Transactions on Information Theory*, vol. 60, no. 7, pp. 4232–4265, 2014.
- [39] W. Yang, G. Durisi, T. Koch, and Y. Polyanskiy, “Block-fading channels at finite blocklength,” in *ISWCS 2013; The Tenth International Symposium on Wireless Communication Systems*, VDE, 2013, pp. 1–4.
- [40] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [41] M. Fey and J. E. Lenssen, “Fast graph representation learning with PyTorch Geometric,” in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.
- [42] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *International Conference on Learning Representations (ICLR)*, 2015.
- [43] R. Bellman, “Dynamic programming,” *Science*, vol. 153, no. 3731, pp. 34–37, 1966.
- [44] C. J. Watkins and P. Dayan, “Q-learning,” *Machine learning*, vol. 8, pp. 279–292, 1992.
- [45] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [46] 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,” *arXiv preprint arXiv:1509.02971*, 2015.

## Bibliography

- [47] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
- [48] G. Strang, *Linear algebra and its applications*. 2012.
- [49] S. Diamond and S. Boyd, “CVXPY: A Python-embedded modeling language for convex optimization,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2909–2913, 2016.
- [50] A. Aijaz and M. Sooriyabandara, “The tactile internet for industries: A review,” *Proceedings of the IEEE*, vol. 107, no. 2, pp. 414–435, 2018.
- [51] M. Bennis, M. Debbah, and H. V. Poor, “Ultrareliable and low-latency wireless communication: Tail, risk, and scale,” *Proceedings of the IEEE*, vol. 106, no. 10, pp. 1834–1853, 2018.
- [52] C. She, C. Yang, and T. Q. Quek, “Radio resource management for ultra-reliable and low-latency communications,” *IEEE Communications Magazine*, vol. 55, no. 6, pp. 72–78, 2017.
- [53] Y. Huang, S. Li, Y. T. Hou, and W. Lou, “GPF: A GPU-based design to achieve  $\sim 100 \mu\text{s}$  scheduling for 5G NR,” in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, 2018, pp. 207–222.
- [54] M. T. Kawser, H. Farid, A. R. Hasin, A. M. Sadik, and I. K. Razu, “Performance comparison between round robin and proportional fair scheduling methods for LTE,” *International Journal of Information and Electronics Engineering*, vol. 2, no. 5, pp. 678–681, 2012.
- [55] M. Andrews, “Probabilistic end-to-end delay bounds for earliest deadline first scheduling,” in *IEEE INFOCOM 2000-IEEE Conference on Computer Communications*, IEEE, vol. 2, 2000, pp. 603–612.
- [56] S. Schwarz, C. Mehlführer, and M. Rupp, “Low complexity approximate maximum throughput scheduling for LTE,” in *2010 Conference Record of the Forty Fourth Asilomar Conference on Signals, Systems and Computers*, IEEE, 2010, pp. 1563–1569.

- [57] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. ElBakoury, “Ultra-low latency (ULL) networks: The IEEE TSN and IETF DetNet standards and related 5G ULL research,” *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 88–145, 2018.
- [58] J. Specht and S. Samii, “Synthesis of queue and priority assignment for asynchronous traffic shaping in switched Ethernet,” in *2017 IEEE Real-Time Systems Symposium (RTSS)*, IEEE, 2017, pp. 178–187.
- [59] M. Khoshnevisan, V. Joseph, P. Gupta, F. Meshkati, R. Prakash, and P. Tinakornsrisuphap, “5G industrial networks with CoMP for URLLC and time sensitive network architecture,” *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 4, pp. 947–959, 2019.
- [60] D. Ginthör, J. von Hoyningen-Huene, R. Guillaume, and H. Schotten, “Analysis of multi-user scheduling in a TSN-enabled 5G system for industrial applications,” in *2019 IEEE International Conference on Industrial Internet (ICII)*, IEEE, 2019, pp. 190–199.
- [61] 3GPP, “Analysis on traffic model and characteristics for MTC and text proposal,” 3GPP, TR R1-120056, 2012, TSG-RAN Meeting WG1#68, Dresden, Germany.
- [62] H. A. Omar, W. Zhuang, A. Abdrabou, and L. Li, “Performance evaluation of VeMAC supporting safety applications in vehicular networks,” *IEEE Transactions on Emerging Topics in Computing*, vol. 1, no. 1, pp. 69–83, 2013.
- [63] 3GPP, “Study on scenarios and requirements for next generation access technologies,” 3GPP, TR 38.913, 2017, v14.2.0.
- [64] S.-C. Tseng, Z.-W. Liu, Y.-C. Chou, and C.-W. Huang, “Radio resource scheduling for 5G NR via deep deterministic policy gradient,” in *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, IEEE, 2019, pp. 1–6.
- [65] C. Qi, Y. Hua, R. Li, Z. Zhao, and H. Zhang, “Deep reinforcement learning with discrete normalized advantage functions for resource management in network slicing,” *IEEE Communications Letters*, vol. 23, no. 8, pp. 1337–1341, 2019.

- [66] J. Li and X. Zhang, “Deep reinforcement learning-based joint scheduling of eMBB and URLLC in 5G networks,” *IEEE Wireless Communications Letters*, vol. 9, no. 9, pp. 1543–1546, 2020.
- [67] J. A. Ayala-Romero, A. Garcia-Saavedra, M. Gramaglia, X. Costa-Perez, A. Banchs, and J. J. Alcaraz, “vrAIn: A deep learning approach tailoring computing and radio resources in virtualized RANs,” in *The 25th Annual International Conference on Mobile Computing and Networking*, 2019, pp. 1–16.
- [68] X. Foukas, M. K. Marina, and K. Kontovasilis, “Iris: Deep reinforcement learning driven shared spectrum access architecture for indoor neutral-host small cells,” *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 8, pp. 1820–1837, 2019.
- [69] A. Y. Ng, D. Harada, and S. Russell, “Policy invariance under reward transformations: Theory and application to reward shaping,” in *ICML*, Citeseer, vol. 99, 1999, pp. 278–287.
- [70] H. He, S. Jin, C.-K. Wen, F. Gao, G. Y. Li, and Z. Xu, “Model-driven deep learning for physical layer communications,” *IEEE Wireless Communications*, vol. 26, no. 5, pp. 77–83, 2019.
- [71] H. He, C.-K. Wen, S. Jin, and G. Y. Li, “Model-driven deep learning for MIMO detection,” *IEEE Transactions on Signal Processing*, vol. 68, pp. 1702–1715, 2020.
- [72] L. Gu, D. Zeng, W. Li, S. Guo, A. Y. Zomaya, and H. Jin, “Intelligent VNF orchestration and flow scheduling via model-assisted deep reinforcement learning,” *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 279–291, 2019.
- [73] A. Neumann, L. Wisniewski, R. S. Ganesan, P. Rost, and J. Jasperneite, “Towards integration of Industrial Ethernet with 5G mobile networks,” in *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, IEEE, 2018, pp. 1–4.
- [74] 3GPP, “Physical layer procedures for data,” 3GPP, TS 38.214, 2018, v15.2.0.

- [75] C. She, C. Yang, and T. Q. Quek, “Joint uplink and downlink resource configuration for ultra-reliable and low-latency communications,” *IEEE Transactions on Communications*, vol. 66, no. 5, pp. 2266–2280, 2018.
- [76] A. Goldsmith, *Wireless communications*. Cambridge university press, 2005.
- [77] H. Van Seijen, M. Fatemi, J. Romoff, R. Laroche, T. Barnes, and J. Tsang, “Hybrid reward architecture for reinforcement learning,” *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [78] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, “Prioritized experience replay,” in *International Conference on Learning Representations (ICLR)*, 2015.
- [79] M. Plappert, R. Houthoofd, P. Dhariwal, S. Sidor, R. Y. Chen, X. Chen, T. Asfour, P. Abbeel, and M. Andrychowicz, “Parameter space noise for exploration,” in *International Conference on Learning Representations (ICLR)*, 2018.
- [80] 3GPP, “Study on New Radio (NR) access technology; physical layer aspects (Release 14),” 3GPP, TR 38.802, 2017, v2.0.0.
- [81] N. Nikaiein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet, “OpenAirInterface: A flexible platform for 5G research,” *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 33–38, 2014.
- [82] P. Mogensen, W. Na, I. Z. Kovács, F. Frederiksen, A. Pokhariyal, K. I. Pedersen, T. Kolding, K. Hugl, and M. Kuusela, “LTE capacity compared to the Shannon bound,” in *2007 IEEE 65th Vehicular Technology Conference-VTC2007-Spring*, IEEE, 2007, pp. 1234–1238.
- [83] 3GPP, “Physical layer procedures,” 3GPP, TS 36.213, 2009, v8.8.0.
- [84] S. Jadon, “An overview of deep learning architectures in few-shot learning domain,” *arXiv preprint arXiv:2008.06365*, 2020.
- [85] P. Viswanath, D. N. C. Tse, and R. Laroia, “Opportunistic beamforming using dumb antennas,” *IEEE Transactions on Information Theory*, vol. 48, no. 6, pp. 1277–1294, 2002.

- [86] A. Asadi and V. Mancuso, “A survey on opportunistic scheduling in wireless communications,” *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1671–1688, 2013.
- [87] M. J. Neely, “Convergence and adaptation for utility optimal opportunistic scheduling,” *IEEE/ACM Transactions on Networking*, vol. 27, no. 3, pp. 904–917, 2019.
- [88] Z. Zhang, S. Moola, and E. K. Chong, “Approximate stochastic dynamic programming for opportunistic fair scheduling in wireless networks,” in *2008 47th IEEE Conference on Decision and Control*, IEEE, 2008, pp. 1404–1409.
- [89] W. Ouyang, S. Murugesan, A. Eryilmaz, and N. B. Shroff, “Exploiting channel memory for joint estimation and scheduling in downlink networks—a Whittle’s indexability analysis,” *IEEE Transactions on Information Theory*, vol. 61, no. 4, pp. 1702–1719, 2015.
- [90] V. S. Borkar, G. S. Kasbekar, S. Pattathil, and P. Y. Shetty, “Opportunistic scheduling as restless bandits,” *IEEE Transactions on Control of Network Systems*, vol. 5, no. 4, pp. 1952–1961, 2017.
- [91] I. Kadota, A. Sinha, E. Uysal-Biyikoglu, R. Singh, and E. Modiano, “Scheduling policies for minimizing age of information in broadcast wireless networks,” *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2637–2650, 2018.
- [92] J. Chen, Y. Wang, and T. Lan, “Bringing fairness to actor-critic reinforcement learning for network utility optimization,” in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, IEEE, 2021, pp. 1–10.
- [93] S. Wang, H. Liu, P. H. Gomes, and B. Krishnamachari, “Deep reinforcement learning for dynamic multichannel access in wireless networks,” *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 2, pp. 257–265, 2018.
- [94] S. Borst and P. Whiting, “Dynamic rate control algorithms for HDR throughput optimization,” in *IEEE INFOCOM 2001-IEEE Conference on Computer Communications*, IEEE, vol. 2, 2001, pp. 976–985.

## Bibliography

- [95] X. Liu, E. K. Chong, and N. B. Shroff, “A framework for opportunistic scheduling in wireless networks,” *Computer networks*, vol. 41, no. 4, pp. 451–474, 2003.
- [96] M. J. Neely, E. Modiano, and C.-P. Li, “Fairness and optimal stochastic control for heterogeneous networks,” *IEEE/ACM Transactions On Networking*, vol. 16, no. 2, pp. 396–409, 2008.
- [97] S. Mandelli, M. Andrews, S. Borst, and S. Klein, “Satisfying network slicing constraints via 5G MAC scheduling,” in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, IEEE, 2019, pp. 2332–2340.
- [98] V. Vukadinovic and G. Karlsson, “Video streaming performance under proportional fair scheduling,” *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 3, pp. 399–408, 2010.
- [99] E. Liu, Q. Zhang, and K. K. Leung, “Asymptotic analysis of proportionally fair scheduling in Rayleigh fading,” *IEEE Transactions on Wireless Communications*, vol. 10, no. 6, pp. 1764–1775, 2011.
- [100] J. M. Holtzman, “Asymptotic analysis of proportional fair algorithm,” in *12th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications. PIMRC 2001. Proceedings (Cat. No. 01TH8598)*, IEEE, vol. 2, 2001, F–F.
- [101] J. Choi, W.-H. Lee, Y.-H. Kim, J.-H. Lee, and S.-C. Kim, “Throughput estimation based distributed base station selection in heterogeneous networks,” *IEEE Transactions on Wireless Communications*, vol. 14, no. 11, pp. 6137–6149, 2015.
- [102] W. Feller, *An introduction to probability theory and its applications, vol 2*. John Wiley & Sons, 2008.
- [103] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, “Rate control for communication networks: Shadow prices, proportional fairness and stability,” *Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237–252, 1998.
- [104] S. Bubeck *et al.*, “Convex optimization: Algorithms and complexity,” *Foundations and Trends® in Machine Learning*, vol. 8, no. 3-4, pp. 231–357, 2015.

## Bibliography

- [105] E. Tanghe, W. Joseph, L. Verloock, L. Martens, H. Capoen, K. Van Herwegen, and W. Vantomme, “The industrial indoor channel: Large-scale and temporal fading at 900, 2400, and 5200 MHz,” *IEEE Transactions on Wireless Communications*, vol. 7, no. 7, pp. 2740–2751, 2008.
- [106] K. Miettinen, *Nonlinear multiobjective optimization*. Springer Science & Business Media, 2012, vol. 12.
- [107] L. B. Jiang and S. C. Liew, “Improving throughput and fairness by reducing exposed and hidden nodes in 802.11 networks,” *IEEE Transactions on Mobile Computing*, vol. 7, no. 1, pp. 34–49, 2007.
- [108] L. Tian, E. Khorov, S. Latré, and J. Famaey, “Real-time station grouping under dynamic traffic for IEEE 802.11 ah,” *Sensors*, vol. 17, no. 7, p. 1559, 2017.
- [109] G. Bianchi, “Performance analysis of the IEEE 802.11 distributed coordination function,” *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, pp. 535–547, 2000.
- [110] M. Garetto, T. Salonidis, and E. W. Knightly, “Modeling per-flow throughput and capturing starvation in CSMA multi-hop wireless networks,” *IEEE/ACM Transactions on Networking*, vol. 16, no. 4, pp. 864–877, 2008.
- [111] D. B. West *et al.*, *Introduction to graph theory*. Prentice Hall Upper Saddle River, 2001, vol. 2.
- [112] S. Szott, K. Kosek-Szott, P. Gawłowicz, J. T. Gómez, B. Bellalta, A. Zubow, and F. Dressler, “Wi-Fi meets ML: A survey on improving IEEE 802.11 performance with machine learning,” *IEEE Communications Surveys & Tutorials*, vol. 24, no. 3, pp. 1843–1893, 2022.
- [113] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, “A comprehensive survey on graph neural networks,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2020.
- [114] X. L. Huang and B. Bensaou, “On max-min fairness and scheduling in wireless ad-hoc networks: Analytical framework and implementation,” in *Proceedings*



- of the 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing, 2001, pp. 221–231.
- [115] C. Tang, L. Song, J. Balasubramani, S. Wu, S. Biaz, Q. Yang, and H. Wang, “Comparative investigation on CSMA/CA-based opportunistic random access for Internet of Things,” *IEEE Internet of Things Journal*, vol. 1, no. 2, pp. 171–179, 2014.
- [116] J. P. Pavon and S. Choi, “Link adaptation strategy for IEEE 802.11 WLAN via received signal strength measurement,” in *IEEE International Conference on Communications, 2003. ICC’03.*, IEEE, vol. 2, 2003, pp. 1108–1113.
- [117] M. Karasuyama and H. Mamitsuka, “Adaptive edge weighting for graph-based learning algorithms,” *Machine Learning*, vol. 106, pp. 307–335, 2017.
- [118] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *International Conference on Learning Representations (ICLR)*, 2017.
- [119] PyTorch Geometric. “PyTorch Geometric: Tables of GNN operators.” (2023), [Online]. Available: [https://pytorch-geometric.readthedocs.io/en/latest/cheatsheet/gnn\\_cheatsheet.html](https://pytorch-geometric.readthedocs.io/en/latest/cheatsheet/gnn_cheatsheet.html).
- [120] L. Tian, S. Deronne, S. Latré, and J. Famaey, “Implementation and validation of an IEEE 802.11 ah module for NS-3,” in *Proceedings of the 2016 Workshop on NS-3*, 2016, pp. 49–56.
- [121] H. T. Friis, “A note on a simple transmission formula,” *Proceedings of the IRE*, vol. 34, no. 5, pp. 254–256, 1946.
- [122] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” *Advances in Neural Information Processing Systems*, vol. 12, 1999.
- [123] Q. Wang, H. Chen, C. Zhao, Y. Li, P. Popovski, and B. Vucetic, “Optimizing information freshness via multiuser scheduling with adaptive NOMA/OMA,” *IEEE Transactions on Wireless Communications*, vol. 21, no. 3, pp. 1766–1778, 2021.

*Bibliography*

- [124] Y. Chen, Y. Chi, and A. J. Goldsmith, “Exact and stable covariance estimation from quadratic sampling via convex programming,” *IEEE Transactions on Information Theory*, vol. 61, no. 7, pp. 4034–4059, 2015.