

ML-based Optimization of Large-scale Systems: Case Study in Smart Microgrids and 5G RAN

by

Hao Zhou

A thesis submitted to the University of Ottawa
in partial fulfillment of the thesis requirement for the degree of

Doctor of Philosophy in Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

© Hao Zhou, Ottawa, Canada, 2023

Abstract

The recent advances in machine learning (ML) have brought revolutionary changes to every field. Many novel applications, such as face recognition and natural language processing, have demonstrated the great potential of ML techniques. Indeed, ML can significantly enhance the intelligence of many existing systems, including smart grid, wireless communications, mechanical engineering, and so on. For instance, microgrid (MG), a distribution-level power system, can exchange energy with the main grid or work in islanded mode, which enables higher flexibility for the smart grid. However, it suffers considerable management complexity by including multiple entities such as renewable energy resources, energy storage system (ESS), loads, etc. In addition, each entity may have unique observations and policies to make autonomous decisions.

Similarly, 5G networks are designed to provide lower latency, higher throughput and reliability for a large number of user devices, but the evolving network architecture also leads to great complexity for network management. The 5G network management should jointly consider various user types and network resources in a dynamic wireless environment. In addition, the integration of new techniques, such as reconfigurable intelligent surfaces (RISs), requires more efficient algorithms for network optimization. Consequently, intelligent management schemes are crucial to schedule network resources.

In this work, we aim to develop state-of-the-art ML techniques to improve the performance of large-scale systems. As case studies, we focus on MG energy management and 5G radio access network (RAN) management. Multi-agent reinforcement learning (MARL) is presumed to be an ideal solution for MG energy management by considering each entity as an independent agent. We further investigate how communication failures will affect MG energy trading by using Bayesian deep reinforcement learning (BA-DRL). On the 5G side, we use MARL, transfer reinforcement learning (TRL), and hierarchical reinforcement learning (HRL) to improve network performance. In particular, we study the performance of those algorithms under various scenarios, including radio resource allocation for network slicing, joint radio and computation resource for mobile edge computing (MEC), joint radio and cache resource allocation for edge caching. Additionally, we further investigate how HRL can improve the energy efficiency (EE) of RIS-aided heterogeneous networks.

The findings of this research highlight the capabilities of various ML techniques under different application domains. Firstly, different MG entities can be well coordinated by applying MARL, enabling intelligent decision-making for each agent. Secondly, Bayesian theory can be used to solve partially observable Markov decision process (POMDP) problems caused by communication failures in MARL. Thirdly, MARL is capable of balancing

the heterogeneous requirements of different slices in 5G networks, guaranteeing satisfactory overall network performance. Then, we find that TRL can significantly improve the convergence performance of conventional reinforcement learning or deep reinforcement learning by transferring the knowledge from experts to learners, which is demonstrated over a 5G network slicing case study. Finally, we find that long-term and short-term decisions are well coordinated by HRL, and the proposed cooperative hierarchical architecture achieves higher throughput and EE than conventional algorithms.

Acknowledgements

Firstly, I would like to express my deepest gratitude to my supervisor, professor Melike Erol-Kantarci, for all her guidance on my research plan, useful discussions in each meeting, corrections on my paper writing, support in finding collaborators and internships, and every effort she has made for our work and lab. It is one of my luckiest decisions to be your student.

Then, I would like to thank all my collaborators, Professor H Vincent Poor, Professor Ivona Brandic, Professor Damla Turgut, Professor Yuanwei Liu, Dr. Atakan Aral, and Dr. Mina Razghandi. I sincerely appreciate all your support in our collaboration, which considerably contributes to this thesis.

I would also like to thank NETCORE lab members. Special thanks to Dr. Medhat Elsayed, Dr. Shahram Mollahasani, and Dr. Long Kong for their generous help when I started my new research direction. I would like to thank Han Zhang, Yujie Yao, Md Arafat Habib, Pedro Enrique Iturria-Rivera, Ycaro Dantas, and Shavbo Salehi for all their support and collaboration.

I thank my parents in China for supporting and understanding me during pursuing my Ph.D. in a new country. Finally, I thank my fiancée Wenyan for all her love, patience and support. You are the first one that changes me so much, and I have a different understanding of the world because of you.

There is a long distance in terms of time and physical distance from a small rural village in China to University of Ottawa in Canada. At this moment, the feeling in my mind is beyond description.

Table of Contents

List of Tables	x i
List of Figures	x ii
List of Abbreviations	x v
List of Symbols	xviii
1 Introduction	1
1.1 Motivations	1
1.1.1 Machine Learning for Microgrid Energy Trading	1
1.1.2 AI-enabled Wireless Communications	3
1.2 Thesis Contributions and Organization	6
1.2.1 Smart MG-related Contributions	6
1.2.2 5G RAN-related Contributions	7
1.3 Publications	8
1.3.1 Journal Publications	8
1.3.2 Conference Publications	9
1.3.3 Under Review Submissions	10

2	Background and Literature Review	12
2.1	Background	12
2.1.1	Markov Decision Process and Q-learning	12
2.1.2	Deep Q-learning and Double Deep Q-learning	15
2.1.3	Multi-agent Reinforcement Learning	17
2.1.4	Transfer Reinforcement Learning	19
2.1.5	Hierarchical Reinforcement Learning	22
2.2	Literature Review	23
2.2.1	Microgrid Energy Trading	23
2.2.2	Machine Learning based 5G Resource Allocation for Network Slicing	26
2.2.3	Optimization Techniques for RIS-aided Wireless Networks	35
3	Multi-agent Correlated Q-learning for Decentralized Microgrid Energy Management	39
3.1	Community Microgrid System Model	39
3.1.1	Demand Side Management Model	40
3.1.2	PV Model	40
3.1.3	Energy Storage System Model	41
3.1.4	Energy Trading Model	41
3.1.5	Problem Formulation	43
3.2	Correlated Q-learning Algorithm	43
3.2.1	State and Actions	44
3.2.2	Reward	44
3.2.3	Correlated Equilibrium	45
3.3	Simulation Result	47
3.3.1	Parameter Settings	47
3.3.2	Energy Management with Correlated Q-learning	48
3.3.3	Comparison under Different PV Power	49
3.3.4	Comparison under Different Initial SOC of ESS	50

4	Multi-agent Bayesian Deep Reinforcement Learning for Microgrid Energy Management under Communication Failures	52
4.1	MG System Architecture and Energy Model	53
4.1.1	System Architecture	53
4.1.2	Microgrid Agents Model	54
4.1.3	Bidding based Microgrid Energy Trading	55
4.1.4	Problem Formulation	55
4.2	Multi-agent POMDP	57
4.2.1	MA-POMDP Definition	57
4.2.2	Bayesian Belief Update Method	60
4.3	Bayesian Deep Reinforcement Learning	63
4.3.1	BA-DRL Architecture and Network Training	63
4.3.2	Belief-based Correlated Equilibrium	64
4.3.3	Baseline Algorithms: Nash-DQN and ADMM	65
4.4	Performance Evaluation	68
4.4.1	Simulation Settings	68
4.4.2	Microgrid Energy Trading with Communication Failure	69
4.4.3	BA-DQL Performance under Supply Uncertainty	71
4.4.4	Comparison under Communication Failure	73
4.4.5	Comparison of Various Communication Failure Probabilities	74
4.4.6	Convergence and Complexity analyses	76
5	Multi-agent Correlated Q-Learning for 5G RAN Slicing	78
5.1	System Model and Problem Formulation	78
5.2	Correlated Q-learning Based Resource Allocation	80
5.2.1	COQRA Architecture	80
5.2.2	Markov Decision Process and Q-learning	80
5.2.3	Correlated Equilibrium	83

5.2.4	Nash Equilibrium	84
5.2.5	LRTQ and PPF Algorithms	84
5.3	Performance evaluation	85
5.3.1	Parameter Settings	85
5.3.2	Simulation Results	86
6	Transfer Reinforcement Learning-based Radio and Computation Resource Allocation in 5G RAN Slicing	89
6.1	System Model and Problem Formulation	90
6.1.1	Network Architecture	90
6.1.2	Communication Model	90
6.1.3	Computation Model	92
6.1.4	Problem Formulation	92
6.2	Knowledge Transfer based Radio and Computation Resource Allocation .	93
6.2.1	Knowledge Transfer Strategy and MDP definition.	94
6.2.2	Knowledge Transfer Reinforcement Learning	95
6.2.3	Baseline Algorithms: QLRA	97
6.3	Performance Evaluation	97
6.3.1	Parameter Settings	97
6.3.2	Simulation Results	98
7	Deep Transfer Reinforcement Learning based Joint Radio and Cache Resource Allocation in 5G RAN Slicing	101
7.1	System Model and Problem Formulation	102
7.1.1	Overall Architecture	102
7.1.2	Communication Model	103
7.1.3	Slicing-based Caching Model	103
7.1.4	Problem Formulation	105
7.2	Deep Transfer Reinforcement Learning based Resource Allocation	107

7.2.1	Overall Framework	107
7.2.2	Q-learning based Experts	108
7.2.3	Deep Transfer Reinforcement Learning based Learners	109
7.2.4	Baseline: Exploration bonus-DQN and PPF-TTL	116
7.2.5	Computational Complexity Analyses	117
7.3	Performance Evaluation	120
7.3.1	Parameter Settings	120
7.3.2	Performance Analyses of Various Learning Parameters	121
7.3.3	Network Performance Analyses	123
7.3.4	Content Caching Performance Analyses	126
8	Cooperative Hierarchical Deep Reinforcement Learning based Joint Sleep, Power, and RIS Control for Energy-Efficient HetNet	128
8.1	Network and System Model	129
8.1.1	RIS-Assisted Channel Model	130
8.1.2	Energy Consumption Model	131
8.1.3	Problem Formulation	131
8.2	Fractional Programming based RIS Phase-shift Control	132
8.3	Cooperative Hierarchical Deep Reinforcement Learning for Sleep Control and Power Control	136
8.3.1	Hierarchical SMDP, Meta-controller and Sub-controller Definitions .	136
8.3.2	Co-HDRL Framework and Network Training	140
8.3.3	High-level Cross-entropy enabled Meta-controller Policy	141
8.3.4	Low-level Sub-controllers Cooperation	143
8.3.5	Computational Complexity Analyses	145
8.4	Baseline Algorithms	146
8.4.1	Hierarchical Deep Reinforcement Learning	146
8.4.2	Surrogate Optimization based RIS phase control	147

8.5	Performance Evaluation	148
8.5.1	Simulation Settings	148
8.5.2	Performance Comparison under Sleep Control and RIS	148
8.5.3	RIS Control Strategy Analyses	150
8.5.4	Machine Learning Algorithm Comparison	153
8.5.5	Performance under various cell sizes and SBS numbers.	154
9	Conclusion	155
9.1	Multi-agent CEQ for Decentralized MG Energy Management	155
9.2	Multi-agent BA-DRL for Microgrid Energy Management under Communi- cation Failures	156
9.3	Multi-agent CEQ for 5G RAN Slicing	156
9.4	TRL-based Radio and Computation Resource Allocation in 5G RAN Slicing	156
9.5	DTRL-based Joint Radio and Cache Resource Allocation in 5G RAN Slicing	157
9.6	Co-HDRL based Joint Sleep, Power, and RIS Control for Energy-Efficient HetNet	157
9.7	Challenges and Open Issues	157
A	Mathematical Proofs	159
A.1	Proof of LSTM network complexity	159
A.2	Proof of cache hit ratio	160
A.3	Proof of Proposition 1	161
A.4	Proof of Proposition 2	161

List of Tables

2.1	Comparison of reinforcement learning algorithms	21
2.2	Summary of network resources in 5G networks.	28
2.3	Summary of AI techniques for 5G resource management-Part 1.	29
2.4	Summary of AI techniques for 5G resource management-Part 2.	30
2.5	Summary of control and optimization techniques for RIS-aided wireless networks	36
3.1	Strategy combination of players with cooperation and threat	42
3.2	Parameter settings of DSM loads with time limit and duration time	47
3.3	Parameters settings of MG with ESS capacity and electricity price	47
3.4	Comparison of CEQ and QLWC under different initial SOC (\$)	51
4.1	Parameter settings of MG and BA-DRL algorithm.	69
4.2	Properties of deferrable devices with power consumption and duration time	69
4.3	Observation function parameters with 8 possible actions and observations	70
5.1	Parameter settings of wireless networks and Q-learning	86
6.1	Parameter settings of 5G networks and edge server	98
7.1	Summary of strategies and MDP definitions	107
7.2	Parameters settings of 5G networks, cache capacity, and learning algorithm	120
8.1	Parameters settings of wireless networks, RISs, and learning algorithms . .	149

List of Figures

1.1	Microgrid system with PV, ESS, and consumers' load.	2
2.1	Markov decision process with state, action, and reward.	13
2.2	Tabular-based Q-learning scheme with environment interaction.	14
2.3	Deep Q-learning architecture with double neural networks.	16
2.4	Multi-agent Q-learning with information sharing	17
2.5	Multi-agent correlated Q-learning system with parameter sharing server . .	18
2.6	Comparison of transfer reinforcement learning and reinforcement learning.	20
2.7	Hierarchical reinforcement learning with meta-controller and sub-controller.	22
2.8	Microgrid energy trading and control framework	23
2.9	5G resource management architecture for network slicing	26
3.1	Microgrid energy trading model with ESS, DSM and PV agents.	42
3.2	Daily PV power and crucial load settings	47
3.3	Daily Time-Of-Use electricity price	47
3.4	Average DSM cost and ESS profit with iterations.	48
3.5	Optimal microgrid energy management policy	48
3.6	Deferred operation time of each device in 24 hours	49
3.7	Comparison of DSM cost under different PV power	50
3.8	Comparison of ESS profit under different PV power	50
3.9	Comparison of DSM cost under different initial SOC	50

3.10	Comparison of ESS profit under different initial SOC	50
4.1	Microgrid system architecture with PV, DSM and ESS agents.	53
4.2	Bidding-based microgrid energy trading.	56
4.3	Definitions of MDP, POMDP, and MA-POMDP	58
4.4	Observation function definition with partial observability.	60
4.5	Sequential Bayesian belief update method.	62
4.6	Multi-agent BA-DRL architecture for MG energy trading.	64
4.7	MG energy management and pricing under various scenarios.	71
4.8	MG system performance under uncertainty and different algorithms	72
4.9	MG system performance under uncertainty and different algorithms.	73
4.10	Comparison of different communication failure probabilities	75
4.11	Convergence performance of multi-agent BA-DRL	76
5.1	Network slicing-based two-step resource allocation.	79
5.2	Proposed COQRA architecture for resource management among slices. . .	81
5.3	URLLC latency distribution[ms] under 2Mbps load per cell.	87
5.4	URLLC latency [ms] with varying offered load [Mbps] per cell.	87
5.5	eMBB throughput with varying URLLC loads [Mbps].	87
5.6	Average packet drop rate under varying URLLC loads [Mbps].	87
5.7	Convergence performance of COQRA and NQL	88
6.1	Proposed system architecture for hierarchial resource allocation.	91
6.2	Learning strategy comparison between RL and HRL	94
6.3	Simulation results comparison between KTRA and QLRA.	99
7.1	Proposed radio and cache resources allocation scheme	102
7.2	Proposed DTRL architecture for resource management.	110
7.3	Proposed Q-value mapping function for deep transfer reinforcement learning.	113

7.4	Convergence and network performance comparison.	122
7.5	Performance comparison under various traffic load and backhaul capacity. .	124
7.6	Network performance comparison under various caching capacities.	127
8.1	RIS-aided heterogeneous network.	129
8.2	Comparisons between MDP and SMDP.	137
8.3	Overall architecture of the proposed Co-HDRL.	139
8.4	Co-HDRL system update between meta-controller and sub-controllers. . . .	145
8.5	Performance comparison of sleep control and RIS.	150
8.6	Performance comparison of difference RIS control strategies.	151
8.7	Performance analyses of Co-HDRL and HDRL.	152
8.8	Algorithm performance under various cell sizes and SBS numbers.	153

List of Abbreviations

Acronym	Meaning
3GPP	Third Generation Partnership Project
ADMM	Alternating Direction Method of Multipliers
ADTRL	action selection-based deep transfer reinforcement learning
AI	artificial intelligence
BA-DRL	Bayesian deep reinforcement learning
BS	base station
CEQ	correlated Q-learning
Co-HDRL	cooperative hierarchical deep reinforcement learning
COQRA	correlated Q-learning based inter-slice RB allocation
CPU	central processing unit
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DDPG	deep deterministic policy gradient
DDQN	double Deep Q-Network
DQN	deep Q-Network
DRL	deep reinforcement learning
DSM	demand side management
DTRL	deep transfer reinforcement learning
EB-DQN	exploration bonus deep Q-learning
ECDF	empirical cumulative distribution function
EE	energy efficiency
eMBB	Enhanced Mobile Broadband
ESS	energy storage system
ETSI	European Telecommunications Standards Institute
FP	fractional programming
GRM	global resource manager
HetNet	heterogeneous network

HRL	hierarchical reinforcement learning
IoT	Internet of Things
KTRA	knowledge transfer based resource allocation
LRTQ	Latency-Reliability-Throughput Q-learning
LSTM	long short-term memory
LTE	Long-Term Evolution
MA-POMDP	multi-agent partially observable Markov decision process
MANO	Management and Orchestration
MARL	Multi-agent reinforcement learning
MBS	macro base station
MDP	Markov decision process
MEC	mobile edge computing
MG	microgrid
ML	machine learning
mMTC	massive Machine Type Communications
NAs	normal agents
Nash-DQN	Nash deep Q-learning
NOMA	non-orthogonal multiple access
NQL	Nash Q-learning
OFDM	Orthogonal frequency-division multiplexing
PA	problematic agent
POMDP	partially observable Markov decision process
PPF	priority proportional fairness
PPF-TTL	priority proportional fairness and time-to-live
PV	photovoltaic power
QDTRL	Q-value-based deep transfer reinforcement learning
QLRA	Q-learning based resource allocation
QLWA	Q-learning with aggregator
QLWC	Q-learning without correlation
QoS	Quality of Service
RAN	radio access network
RB	resource block
RBG	resource block group
RIS	reconfigurable intelligent surface
RL	reinforcement learning
RNN	recurrent neural network
SBS	small base station
SCA	successive convex approximation

SDN	software defined networks
SOC	state of charge
SRM	slice resource manager
TL	transfer learning
TOU	Time-of-Use
TRL	transfer reinforcement learning
TTI	Transmission Time Interval
TTL	time-to-live
UAV	unmanned aerial vehicle
UE	user equipment
URLLC	Ultra Reliable and Low Latency Communications
VM	virtual machines
VNF	virtual network functions

List of Symbols

α	learning rate
ϵ	exploration probability
γ	discount factor
ι	waiting time indicator of deferrable devices
λ	agent number indicator
$\mathbb{E}$	expected value
$\mathcal{D}$	distance metric
$\mathcal{F}$	mapping function
$\mathcal{M}$	set of user devices
$\mathcal{N}$	set of slices
$\mathcal{P}$	proportion of buying/selling energy
$\mathcal{K}$	knowledge
$\mathcal{T}$	time-to-live of the cached content
$\mathcal{X}$	buying energy choice
$\mathcal{Y}$	selling energy choice
Ω	set of observations
ω	network performance metric weight

ϕ	packet request rate
π	policy
σ	transfer learning rate
ϖ	edge caching decision indicator
ϱ	on/off status indicator if deferrable devices
ς	ESS charging status indicator
ϑ	edge computing decision indicator
A	The set of actions
a	action
B	backhaul capacity
b	belief
b_{RB}	bandwidth of one resource block
C	edge caching/computing capacity
C^{ESS}	energy storage system (ESS) capacity
d	network delay
E	channel capacity
Er	error
h	cache hit ratio
J	set of base station (BS)s
L	packet size
N^{de}	total number of deferrable devices
N^{RB}	Set of resource blocks
N_0	noise power density

O	observation function
o	observation
P	power
Q	state-action value
R	The set of rewards
r	reward
$rand$	a random number between 0 and 1
S	The set of states
s	state
SOC	state of charge
T	transition probability
t	time slot
T^{op}	total simulation period
U	utility function
V	state value
w	network weight
x	binary indicator

Chapter 1

Introduction

1.1 Motivations

machine learning (ML) has been successfully applied in many fields and achieved great success, including smart grid, Internet of Things (IoT), wireless communications, robot control, natural language processing, facial recognition, etc [1]. Various ML techniques have been designed to understand and leverage the data and improve the system performance, including supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning (RL), and so on. The great success of ML techniques motivates us to investigate the possible applications in other fields. There are many types of complicated large-scale systems, but this work considers smart microgrid (MG) and 5G networks for the case study due to the critical importance of energy and information systems [2, 3]. Following, we will present the motivations in detail.

1.1.1 Machine Learning for Microgrid Energy Trading

MG refers to a distribution-level power system that is connected to the main grid or works in island mode. Integrating MG to the electrical distribution system will improve the efficiency, reliability and resilience of future smart grid [4]. In particular, they provide promising opportunities for enhancing the reliability of the electricity grid by forming energy-sharing communities [5, 6].

An MG usually contains an ESS, one or more generators and loads [7]. In the smart grid, these entities are interconnected and form an IoT environment where power loads,

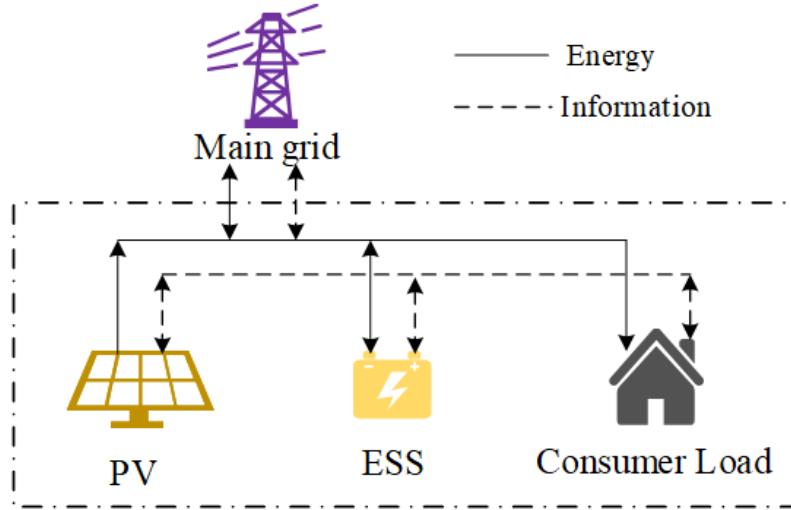


Figure 1.1: Microgrid system with PV, ESS, and consumers' load.

storage, and generation communicate and collaborate to improve energy management. Many centralized algorithms have been proposed for MG energy management, including the gradient search, interior point algorithm, and lambda iteration approach [8]. However, these centralized methods raise concerns about reliability and security issues, such as the failure of the central controller, which may result in a shutdown of the whole MG. In addition, resilience, privacy protection, high computation burden, and high bandwidth requirements are all further challenges for a centralized MG architecture.

The decentralized control is considered the future of MG energy management due to high flexibility and reliability, as well as low computation burden [4]. The decentralized MG architecture can be modelled as a multi-agent system, which includes a generator (e.g., photovoltaic power (PV) agent), a demand side management (DSM) agent, an ESS agent, and so on. In a multi-agent system, multiple intelligent agents work collaboratively, and each agent optimizes its objective in a parallel or sequential manner. Having physically separate units with different objectives and reasonable autonomy makes the MG an ideal application of multi-agent systems [9].

In recent years, multi-agent MG energy management has been widely studied with various methodologies, including Alternating Direction Method of Multipliers (ADMM), event-trigger, consensus theory, game theory, and so on. These model-based methods generally require dedicated optimization models. For example, to deploy the convex optimization, the convexity should be checked first, and non-convex problems must be converted into convex optimizations, which may lead to great complexity [10]. On the other hand, the

uncertainty of renewable energy resources, heterogeneous cost definitions of agents, and different energy trading strategies are all potential challenges for model-based solutions, since they will greatly increase the difficulty of building an optimization model [11, 12]. The limitations of conventional methods and the future challenges motivate us to find more ideal solutions for the MG energy management. Recently, data-driven approaches have emerged as a model-free alternative where ML algorithms are used for the management of MG [13]. In particular, Multi-agent reinforcement learning (MARL) has become a promising method for MG energy management and control [14]. Various RL algorithms have been used for MG energy management in the literature, e.g., Bayesian reinforcement learning, Nash Q-learning, fuzzy Q-learning, cooperative reinforcement learning, and deep Q-Network (DQN).

It is worth noting that communication between agents is the key to a multi-agent system in both model-based and model-free methods. By exchanging collaboration information, agents can choose optimal joint actions and maintain satisfactory overall performance. Unlike RL, missing collaboration information may lead to sub-optimal decisions, and in some cases, may even impact the stability of the whole MG [15]. However, most existing researches assume perfect communication environment without information loss [16]. As bandwidth, latency and reliability requirements of different smart grid applications vary widely, and packet loss may happen while agents exchange the collaboration information [15]. The communication system can be prone to errors and signal loss due to harsh smart grid environments, contributing to a high packet loss rate [17]. Consequently, it is critical to involve the packet loss in the MG energy management to guarantee the MG performance and reduce potential losses.

To this end, in Chapter 3, we introduce our proposed correlated Q-learning (CEQ)-based MG energy management method in [18], and correlated deep Q-learning scheme in [19]. To further overcome the communication failure issue, in Chapter 4, we present a Bayesian deep reinforcement learning algorithm [20], which is robust to communication failures in MG energy management.

1.1.2 AI-enabled Wireless Communications

The forthcoming 5G networks will support a vast number of services and applications, where heterogeneous requirements for latency, bandwidth, and reliability will coexist [21, 22]. Three major traffic types are supported in 5G, namely Enhanced Mobile Broadband (eMBB), Ultra Reliable and Low Latency Communications (URLLC), and massive Machine Type Communications (mMTC). The eMBB is regarded as an extension of Long-

Term Evolution (LTE)-Advanced services, which aims to provide a high data rate for applications such as video streaming [23]. The URLLC is proposed to provide a sub-millisecond latency and 99.999% reliability, which is critical for applications such as autonomous vehicles and remote surgery [24]. The mMTC is designed to connect a large number of IoT devices, where data transmissions occur sporadically.

The stringent and heterogeneous Quality of Service (QoS) requirements have become a challenging problem in 5G, especially when different traffic types are multiplexed on the same channel. Network slicing is one of the fundamental concepts in 5G that enables support for multiple services and applications with heterogeneous demands [25, 26]. With software-defined networks and network function virtualization techniques, various slices can be defined over the same physical network devices for flexibility and scalability [27]. However, network slicing also leads to significant complexities for resource management, since the requirements of heterogeneous slices should be satisfied concurrently with limited resources. On the other hand, new services and applications are constantly emerging, requiring more stringent service level agreements for network management. In addition, user types are not limited to eMBB, URLLC, and mMTC, and an enterprise slice with various needs can be dynamically created, as well. To this end, artificial intelligence (AI) and ML offer promising opportunities to reduce the management complexity and improve system efficiency [4, 28].

AI-based network management has attracted interest from both academia and industry. The Management and Orchestration (MANO) is the most notable framework defined by European Telecommunications Standards Institute (ETSI) [29]. Similarly, the Third Generation Partnership Project (3GPP) also defined the network data analytics function and management data analytics function to incorporate AI into resource management [30]. These ingeniously designed schemes, nevertheless, are still ongoing, and these frameworks mainly focus on system-level definitions of key elements.

Meanwhile, AI-enabled wireless networks have been extensively investigated by academia. For example, [31] summarizes the applications for employing AI in cellular networks, and identifies the challenges and roadmaps for AI-enabled 5G and 6G networks. A thorough survey is presented in [32] on recent advances and future challenges of applying AI to wireless networks, including algorithms, applications, standards, and so on.

On the other hand, numerous schemes have been proposed to make the most of AI for network management. For instance, AI is considered as a built-in architectural feature in [33] that enables resource elasticity, and it includes different use cases that utilize AI for elastic management. In these studies, huge amounts of samples are generally required to train their algorithms, and consequently the long training time may hamper the system

efficiency and applicability in practice. Moreover, in these schemes, the costly training needs to be repeated for each arriving new task.

Although learning-based methods, such as RL and deep reinforcement learning (DRL), have been generally applied for network resource allocation, most existing works do not consider the possibility of knowledge transfer. Specifically, an agent is designed for a specific task in these works, and it interacts with its environment from scratch, which usually leads to a lower exploration efficiency and longer convergence time. The agent must be retrained whenever a new task is assigned, even though similar tasks have been completed before. The poor generalization capability of straightforward RL methods motivates us to find a learning method with better generalization and knowledge transfer capability. On the other hand, humans can reuse the knowledge learned from previous tasks to solve new tasks more efficiently, and this capability can be built into ML as well [34]. Such knowledge transfer and reuse can significantly reduce the need for a large number of training samples, which is a common issue in many ML methods. By incorporating knowledge transfer capability into ML, it is expected to reduce the algorithm design and training efforts, and achieve better performance, such as faster convergence and higher average reward.

Moreover, although ML has been widely used for wireless network management, it is worth noting that most existing works require actions with the same timescales, which means all the actions must be decided simultaneously. But in practice, the actions may belong to different agents that apply various decision-making timescales. Hence, handling such time scale differences in wireless networks is important, which will benefit the overall coordination of short-term and long-term decisions. To this end, hierarchical reinforcement learning (HRL) is proposed, which includes a sub-controller to implement short-term actions under given policies, and a meta-controller to decide long-term policies based on the average performance of the sub-controller. HRL enables more flexible network management.

Motivated by these challenges, Chapter 5 presents a correlated Q-learning based inter-slice RB allocation (COQRA) algorithm for radio resource allocation of 5G radio access network (RAN) slicing [35]. Then, Chapter 6 introduces a transfer learning algorithm for joint radio and computation resource allocation of 5G mobile edge computing (MEC) [36]. Chapter 7 includes a deep transfer reinforcement learning (DTRL)-enabled joint radio and cache resource allocation scheme for 5G edge caching [37]. Finally, Chapter 8 introduces a cooperative hierarchical deep reinforcement learning (Co-HDRL) algorithm for joint sleep, power, and RIS control for energy-efficient heterogeneous network (HetNet).

1.2 Thesis Contributions and Organization

This thesis aims at applying state-of-the-art ML techniques to optimize large-scale systems such as smart MG and 5G RAN. In particular, the ML algorithms include CEQ, DQN, MARL, transfer reinforcement learning (TRL), DTRL, and HRL, and the application scenarios consider MG energy management and trading, RAN slicing, MEC, edge caching, HetNet, and so on. This thesis demonstrates the capability of ML techniques for optimizing large-scale systems. The detailed contributions and organizations are summarized as follows.

1.2.1 Smart MG-related Contributions

1) Multi-agent CEQ for Decentralized MG Energy Management

In Chapter 3, we present multi-agent CEQ algorithms for MG energy management [18,19]. They apply correlated equilibrium to coordinate the actions of MG agents such as DSM, ESS and PV. The correlated equilibrium allows for dependencies among agents' strategies, while the actions in conventional Nash equilibrium are independent. Compared with the iterative method, which is generally used for finding Nash equilibrium, correlated equilibrium could be easily found by linear programming. However, the proposed multi-agent scheme for MG energy management is vulnerable to communication failures, which may lead to performance loss, and we address this problem in Chapter 4.

2) Multi-agent BA-DRL for MG Energy Management under Communication Failures

In Chapter 4, to overcome communication failures, we investigate the ML-enabled energy trading under imperfect communication [20]. In particular, a multi-agent partially observable Markov decision process (MA-POMDP) based scheme is proposed to describe agents with communication failures. Based on MA-POMDP scheme, we propose a Bayesian belief update method, where each agent can update its beliefs on the actions of peer agents by Bayesian rules, and no direct communication is needed for the belief updating. Moreover, we propose a Bayesian deep reinforcement learning (BA-DRL) algorithm to handle the communication failures between agents in an MG. In BA-DRL, the double Deep Q-Network (DDQN) architecture is applied for Q-value approximation to prevent over-estimation. We also propose a novel action selection method called belief-based correlated equilibrium. It utilizes beliefs to select optimal joint actions under communication failures, which is implemented in a decentralized manner.

1.2.2 5G RAN-related Contributions

1) Multi-agent CEQ for 5G RAN Slicing

In 5G networks, QoS requirements of various services and limited spectrum resources make radio resource allocation a challenging problem. In Chapter 5, we develop a MARL-based RAN resource slicing scheme for 5G networks [35]. In the proposed multi-agent COQRA, each agent makes decisions autonomously and is coordinated by exchanging Q-values with each other. It applies a two-tier resource allocation scheme, including inter-slice and intra-slice resource allocations. The proposed method in [35] is designed for radio resource allocation only, but the evolving network architecture requires joint allocation of multiple resources, and we further investigate this problem in Chapters 6 and 7.

2)TRL-based Radio and Computation Resource Allocation in 5G RAN Slicing To implement network slicing in 5G, joint resource allocation is a key enabler, but it also leads to higher complexity for network management. In Chapter 6, we propose a knowledge transfer based resource allocation (KTRA) method to jointly allocate radio and computation resources for 5G RAN slicing [36]. Compared with existing works, the main difference is that the proposed KTRA method has a knowledge transfer capability. It is designed to use the prior knowledge of similar tasks to improve the performance of the target task, e.g., faster convergence speed or higher average reward. However, the proposed KTRA algorithm is still limited to tabular-based Q-learning schemes, and we investigate the knowledge transfer for DQN in Chapter 7.

3)DTRL-based Radio and Cache Resource Allocation in 5G RAN Slicing In Chapter 7, we propose a DTRL algorithm for joint radio and cache resource allocation in 5G RAN slicing. First, we define a hierarchical architecture for joint resource allocation of cellular networks. The global resource manager (GRM) will intelligently allocate resources between slices, then slice resource manager (SRM)s will distribute radio and cache resources within each slice based on specific rules. The proposed hierarchical architecture can reduce the control overhead and achieve higher flexibility. Secondly, We propose two DTRL-based solutions for inter-slice resource allocation, namely Q-value-based deep transfer reinforcement learning (QDTRL) and action selection-based deep transfer reinforcement learning (ADTRL). The QDTRL can use experts' Q-values as prior knowledge to improve the learning process, while the ADTRL focuses on action selection knowledge. Compared with RL or DRL, the proposed DTRL solutions show a better knowledge transfer capability.

4) Co-HDRL based Joint Sleep, Power, and RIS Control for Energy-Efficient HetNet

energy efficiency (EE) is a critical metric for 5G and envisioned 6G networks, and the RIS is considered a promising technique to improve the EE due to its low power consumption

and hardware cost. Chapter 8 investigates using Co-HDRL for joint sleep, power, and RIS control for energy-efficient HetNet. Specifically, we first propose a fractional programming (FP) method for RIS phase-shift control, which aims to maximize the sum-rate under given transmission power levels. Then, considering the timescale difference between sleep control and power control, we introduce a Co-HDRL algorithm, including a cross-entropy enabled meta-controller for sleep control, and correlated equilibrium-based sub-controllers for power control. Moreover, we proposed a surrogate optimization method as one baseline for RIS control, and conventional HDRL as another baseline for sleep and power control.

1.3 Publications

1.3.1 Journal Publications

- [J05] **H. Zhou**, M. Erol-Kantarci, and H. V. Poor, “Knowledge Transfer and Reuse: A Case Study of AI-enabled Resource Management in RAN Slicing,” *IEEE Wireless Communications* (Early Access), pp.1-10, Nov. 2022.
- [J04] **H. Zhou**, M. Erol-Kantarci, and H. V. Poor, “Learning from Peers: Transfer Reinforcement Learning for Joint Radio and Cache Resource Allocation in 5G Network Slicing,” *IEEE Transactions on Cognitive Communications and Networking*, vol.8, no.4, pp.1925-1941, Dec.2022.
- [J03] **H. Zhou**, A. Aral, I. Brandic, and M. Erol-Kantarci, “Multi-agent Bayesian Deep Reinforcement Learning for Microgrid Energy Management under Communication Failures,” *IEEE Internet of Things Journal*, vol.9, no.14, pp.11685-11698, Jul. 2022.
- [J02] M. Razghandi, **H. Zhou**, M. Erol-Kantarci, and D. Turgut, “Smart Home Energy Management: VAE-GAN Synthetic Dataset Generator and Q-learning,” *IEEE Transactions on Smart Grid*(Accepted), pp. 1-11, May 2023.
- [J01] PE. Iturria-Rivera, H. Zhang, **H. Zhou**, S. Mollahasani, and M. Erol-Kantarci, “Multi-Agent Team Learning in Virtualized Open Radio Access Networks (O-RAN),” *Sensors*, vol.22, no.14, pp.1-13, Jul. 2022.

1.3.2 Conference Publications

- [C15] M. A. Habib, **H. Zhou**, PE. Iturria-Rivera, M. Elsayed, M. Bavand, R. Gaigalas, Y. Ozcan, and M. Erol-Kantarci, “Hierarchical Reinforcement Learning Based Traffic Steering in Multi-RAT 5G Deployments,” (**Accepted to 2023 IEEE ICC, Best paper award**) .
- [C14] Y. Dantas, PE. Iturria-Rivera, **H. Zhou**, M. Elsayed, M. Bavand, R. Gaigalas, and M. Erol- Kantarci, “Beam Selection for Energy-Efficient mmWave Network Using Advantage Actor Critic Learning,” (**Accepted to 2023 IEEE ICC**).
- [C13] M. A. Habib, **H. Zhou**, PE. Iturria-Rivera, M. Elsayed, M. Bavand, R. Gaigalas, S. Furr, and M. Erol-Kantarci, “Traffic Steering for 5G Multi-RAT Deployments using Deep Reinforcement Learning,” in **Proc. of 2022 IEEE CCNC**, pp. 1-6, Oct. 2022.
- [C12] **H. Zhou**, L. Kong, M. Elsayed, M. Bavand, R. Gaigalas, S. Furr, and M. Erol-Kantarci, “Hierarchical Reinforcement Learning for RIS-Assisted Energy-Efficient RAN,” in **Proc. of 2022 IEEE GLOBECOM**, pp.1-6.
- [C11] H. Zhang, **H. Zhou**, and M. Erol-Kantarci, “Federated Deep Reinforcement Learning for Resource Allocation in O-RAN Slicing,” in **Proc. of 2021 IEEE GLOBECOM**, pp.1-6, Dec. 2022.
- [C10] Y. Yao, **H. Zhou**, M. Erol-Kantarci, “Joint Sensing and Communications using Deep Reinforcement Learning-based Beam Management in 6G Networks,” in **Proc. of 2021 IEEE GLOBECOM**, pp.1-6, Dec. 2022.
- [C09] Y. Yao, **H. Zhou**, M. Erol-Kantarci, “Deep Reinforcement Learning-Based Radio Resource Allocation and Beam Management Under Location Uncertainty in 5G mmWave Networks,” in **Proc. of 2022 IEEE Symposium on Computers and Communications**, pp.1-6, Jul.2022.
- [C08] H, Zhang, **H. Zhou**, M. Erol-Kantarci, “Team Learning-Based Resource Allocation for Open Radio Access Network (O-RAN),” in **Proc. of 2022 IEEE ICC**, pp.1-6, Jan. 2022.
- [C07] M. Razghandi,**H. Zhou**, M. Erol-Kantarci, and D. Turgut, “Variational Autoencoder Generative Adversarial Network for Synthetic Data Generation in Smart Home,” in **Proc. of 2022 IEEE ICC**, pp.1-6, Jan. 2022.

- [C06] **H. Zhou**, and M. Erol-Kantarci, “Knowledge Transfer based Radio and Computation Resource Allocation for 5G RAN Slicing,” in **Proc. of 2022 IEEE CCNC**, pp.1-6, Jan. 2022.
- [C05] M. Razghandi, **H. Zhou**, M. Erol-Kantarci, and D. Turgut, “Smart Home Energy Management: Sequence-to-Sequence Load Forecasting and Q-Learning,” in **Proc. of 2021 IEEE GLOBECOM**, pp.1-6, Dec. 2021.
- [C04] **H. Zhou**, M. Elsayed, and M. Erol-Kantarci, “RAN Resource Slicing in 5G Using Multi-Agent Correlated Q-Learning,” in **Proc. of 2021 IEEE PIMRC**, pp.1-6, Sep. 2021.
- [C03] M. Razghandi, **H. Zhou**, M. Erol-Kantarci, and D. Turgut, “Short-Term Load Forecasting for Smart Home Appliances With Sequence to Sequence Learning,” in **Proc. of 2021 IEEE ICC**, pp.1-6, Jun. 2021.
- [C02] **H. Zhou**, and M. Erol-Kantarci, “Decentralized Microgrid Energy Management: A Multi-agent Correlated Q-learning Approach,” in **Proc. of 2020 IEEE conference on Smart Grid Communication**, pp.1-7, Nov. 2020.
- [C01] **H. Zhou**, and M. Erol-Kantarci, “Correlated Deep Q-learning based Microgrid Energy Management,” in **Proc. of 2020 IEEE International Workshop CAMAD**, pp.1-6. Sep. 2020.

1.3.3 Under Review Submissions

- [U06] **H. Zhou**, M. Elsayed, M. Bavand, R. Gaigalas, S. Furr, and M. Erol-Kantarci, “Cooperative Hierarchical Deep Reinforcement Learning Based Joint Sleep, Power, and RIS Control for Energy-Efficient HetNet,” arXiv:2304.13226, pp. 1-14, Apr. 2023.
- [U05] **H. Zhou**, M. Erol-Kantarci, Y. Liu, and H. V. Poor, “A Survey on Model-based, Heuristic, and Machine Learning Optimization Approaches in RIS-aided Wireless Networks,” Under review of *IEEE Communications Surveys & Tutorials*, pp. 1-37, Mar. 2023.
- [U04] H. Zhang, **H. Zhou**, M. Elsayed, M. Bavand, R. Gaigalas, Y. Ozcan, and M. Erol-Kantarci, “On-Device Intelligence for 5G RAN: Knowledge Transfer and Federated Learning enabled UE-Centric Traffic Steering,” Under review of *IEEE Transactions on Cognitive Communications and Networking*, pp.1-15, Apr. 2023.

- [U03] H. Zhang, **H. Zhou**, M. Elsayed, M. Bavand, R. Gaigalas, Y. Ozcan, and M. Erol-Kantarci, “On the Security of Federated Learning-enabled Wireless Networks,” (**Submitted to 2023 IEEE Globecom**), May 2023.
- [U02] S. Salhi, **H. Zhou**, M. Elsayed, M. Bavand, R. Gaigalas, and M. Erol-Kantarci, “Policy Poisoning Attacks on Transfer Learning enabled Resource Allocation for Network Slicing,” (**Submitted to 2023 IEEE Globecom**), May 2023.
- [U01] Y. Dantas, PE. Iturria-Rivera, **H. Zhou**, M. Elsayed, M. Bavand, R. Gaigalas, and M. Erol-Kantarci, “Split Learning for Sensing-Aided Single and Multi-Level Beam Selection in Multi-Vendor RAN,” (**Submitted to 2023 IEEE Globecom**), May 2023.

Chapter 2

Background and Literature Review

This section first introduces the research background, especially RL techniques such as Q-learning, DRL, MARL, TRL, and HRL. Then, we investigate existing studies on MG energy trading, ML-enabled 5G resource allocation for network slicing, and optimization techniques for reconfigurable intelligent surface (RIS)-aided wireless networks.

2.1 Background

2.1.1 Markov Decision Process and Q-learning

In this section, we introduce Markov decision process (MDP) and Q-learning to illustrate the basic framework for RL techniques. In particular, the task problem is first described by the MDP, and then it is optimized by the RL [38].

MDP can be defined by a tuple $\langle S, A, T, R \rangle$, where S is the state set, A is the action set, T is the transition probability, and R is the reward function, respectively. As shown in Fig.2.1, the agents select an action a^t based on current state s^t , then it receives the reward r^t and moves to the next state $s^{t+\Delta t}$. The MDP can transform one optimization problem into maximizing the long-term expected reward r in MDP context, and r is closely related to the objective function in the given optimization problem.

With the MDP, optimization problems are easily transformed into a unified scheme that can be solved by using RL. In RL, we define an agent to interact with the environment

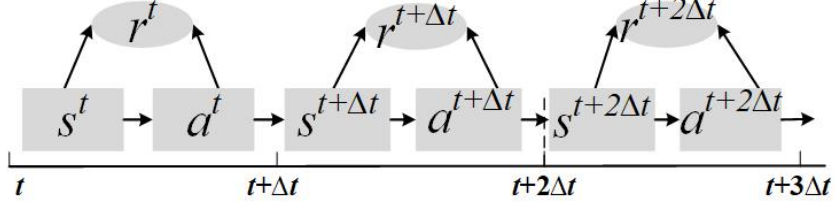


Figure 2.1: Markov decision process with state, action, and reward.

using MDP context, and it aims to maximize the long-term expected reward by:

$$V^\pi(s) = Ex_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s^t, a^t) | s = s^0 \right] \quad (2.1)$$

where t is the number of iterations, and π is the policy. $V^\pi(s)$ is the state value that indicates the potential expected reward at state s . $r(s^t, a^t)$ is the reward at iteration t by selecting action a^t under state s^t . s^0 is the initial state, and Ex indicates the expected value. γ is the discount factor ($0 < \gamma < 1$) to balance the future reward and the current reward. A higher γ value means that the agent focuses on the future reward; otherwise, a lower γ value indicates that the current reward is more important.

Then, for a specific state s and action a , we define the state-action value:

$$Q^\pi(s^t, a^t) = r(s^t, a^t) + \gamma \sum_{s^{t+1} \in S} Pr(s^{t+1} | s^t, a^t) V^\pi(s^{t+1}) \quad (2.2)$$

where $Q^\pi(s^t, a^t)$ is the state-action value that represents the long-term expected reward of selecting action a^t under state s^t . $Pr(s^{t+1} | s^t, a^t)$ is the probability of moving to state s^{t+1} from state s^t . $Q^\pi(s^t, a^t)$ consists of two terms: $r(s^t, a^t)$ is the current reward, and $\gamma \sum_{s^{t+1} \in S} Pr(s^{t+1} | s^t, a^t) V^\pi(s^{t+1})$ indicates the potential reward in following time slots. γ is the discount factor to balance two terms.

Consequently, we have the following relationship:

$$V^\pi(s^t) = \sum_{a^t \in A} \pi(a^t) Q(s^t, a^t) \quad (2.3)$$

where $\pi(a^t)$ is the action selection policy that represents the probability of selecting action a^t .

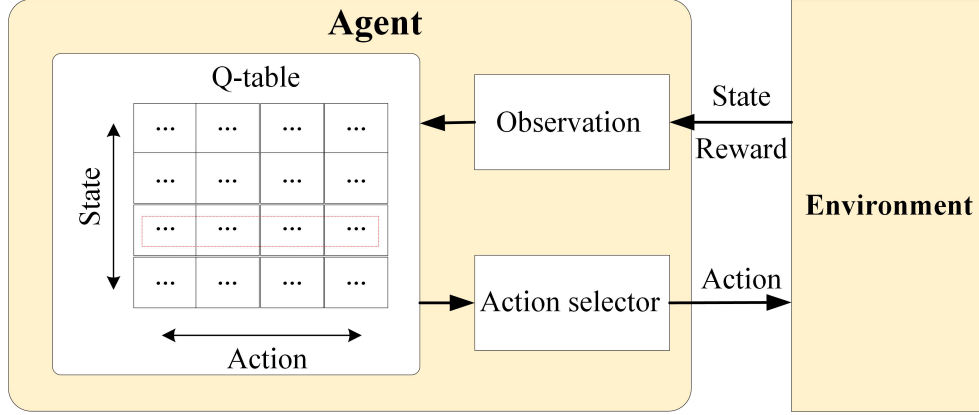


Figure 2.2: Tabular-based Q-learning scheme with environment interaction.

In Q-learning, the agent interacts with the environment as in Fig.2.2. The agent selects an action using the Q-table, and then receives a state and a reward from the environment, which will be further used to update the Q-values. The Q-values are updated by:

$$Q^{new}(s^t, a^t) = Q^{old}(s^t, a^t) + \alpha(r^t + \gamma \max_a Q(s^{t+1}, a) - Q^{old}(s^t, a^t)) \quad (2.4)$$

where $Q^{old}(s^t, a^t)$ and $Q^{new}(s^t, a^t)$ are old and new Q-values, respectively. Here α is the learning rate ($0 < \alpha < 1$), which indicates the Q-value update rate. A high learning rate such as 0.9 means the Q-values are quickly updated by using the current reward and state value of the next state; otherwise, a low learning rate such as 0.1 means the Q-values are updated slowly. It is worth noting that a high learning rate may lead to unstable training results, but a low learning rate requires an increased number of iterations to converge. In practice, the learning rate value is usually selected by trial and error method.

In Fig.2.2, the Q-values index consists of combinations of states and actions, and it relies on a Q-table to record all the Q-values. However, when the state-action space increases, the agent has to visit all the possible state-action combinations to search optimal solution, and it may require an increased number of iterations to converge all the Q-values. Moreover, the tabular-based Q-learning can not handle problems with continuous action spaces, because the Q-table can only record finite actions. To this end, DRL is proposed, which is introduced in the next section.

2.1.2 Deep Q-learning and Double Deep Q-learning

The main idea of DQN is to apply a deep neural network to predict Q-values [39]. It overcomes the scalability issue of Q-learning by using a neural network to record all the Q-values [40, 41]. In Q-learning, the Q-values are updated by equation (2.4). When Q-values converge, we have $Q^{old}(s^t, a^t) = Q^{new}(s^t, a^t)$, which means $Q^{old}(s^t, a^t) = r^t + \gamma \max_a Q(s^{t+1}, a)$. Then we can define a loss function for the network training of DQN:

$$Loss(w) = Er(r^t + \gamma \max_a Q(s^{t+1}, a, w') - Q(s^t, a^t, w)) \quad (2.5)$$

where w and w' are the weight of main and target networks, respectively, and Er is the error function. $r^t + \gamma \max_a Q(s^{t+1}, a, w')$ indicates target Q-values are generated by the target network, and $Q(s^t, a^t, w)$ means current Q-values are predicted by the main network.

In Q-learning, note that Q-values change dynamically in each iteration, and the target values also change, which may lead to an unstable result for network training. Meanwhile, the training data is expected to be discontinuous, but the state and action transition is consecutive in Q-learning. Two techniques are proposed to improve the DQN, namely experience replay and target network [39].

For the experience replay, the tuple $\langle s^t, a^t, r^t, s^{t+1} \rangle$ generated by Q-learning is saved in an experience pool. After some iterations, the agent will implement a random minibatch, and the sample data will be used for training. Meanwhile, old experiences will be replaced by new data gradually; thus, the network can learn new experiences step by step.

On the other hand, in equation (2.5), note that we use the main network to predict current Q-values, and target network to generate target Q-values. If using one single network to predict current and target Q-values simultaneously, both $Q(s^{t+1}, a, w')$ and $Q(s^t, a^t, w)$ will change after network training, leading to unstable training results. By contrast, the double network scheme can provide a more stable environment for main network training. After training several times, the weight of the main network will be copied to the target network, and this late updating provides a stable target for the main network training.

The DQN can be summarized as Fig.2.3. Given the current state s^t , the main network selects the action a^t first, and then the action a^t will be sent to the environment. The agent receives a new state s^{t+1} and reward r^t from the environment. The tuple $\langle s^t, a^t, r^t, s^{t+1} \rangle$ is saved in the experience pool, which will be used for the network training as shown by equation (2.5). Finally, the target network will copy the weight of the main network after training several times.

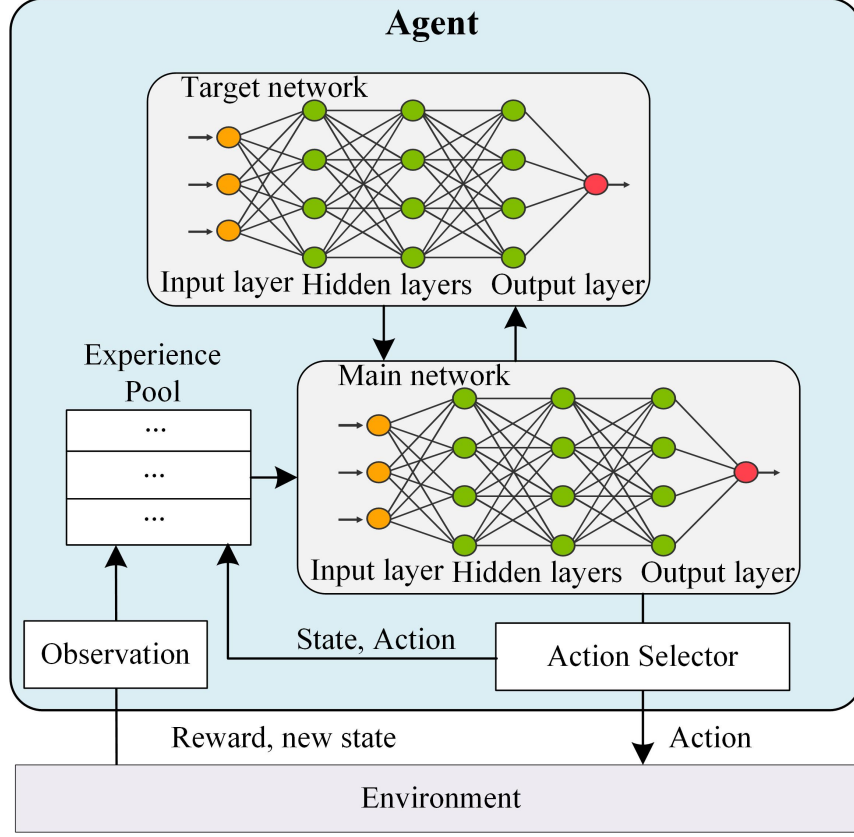


Figure 2.3: Deep Q-learning architecture with double neural networks.

In DQN, note that the action selection and evaluation are both implemented by the target network, which is indicated by $\max_a Q(s^{t+1}, a, w')$ in equation (2.5). Meanwhile, Q-values are calculated by the reward added to the maximum Q-value of the next state. If every time the Q-value is calculated at a higher value under a certain state, then the Q-value predicted by the neural network will be obviously higher every time. As a result, some Q-values are overestimated by a higher value, and the selected action can be sub-optimal, since the real state-action value is lower than the predicted value.

To this end, the DDQN has been proposed in which action selection and evaluation are decoupled. The loss function of DDQN is defined as:

$$Loss(w) = Er(r^t + \gamma Q(s^{t+1}, \arg \max_a Q(s^{t+1}, a, w), w') - Q(s^t, a^t, w)) \quad (2.6)$$

where main network will select actions by $\arg \max_a Q(s^{t+1}, a, w)$, and target network eval-

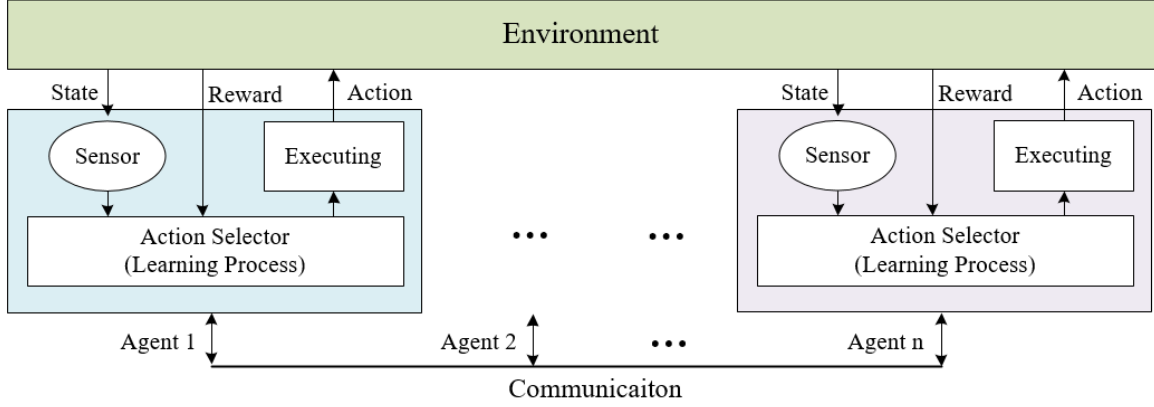


Figure 2.4: Multi-agent Q-learning with information sharing

uate the action by $Q(s^{t+1}, \arg \max_a Q(s^{t+1}, a, w), w')$. By decoupling the action selection and evaluation, the DDQN can provide more accurate Q-values than DQN.

2.1.3 Multi-agent Reinforcement Learning

In this section, we introduce the MARL technique, which is an important variant of RL. Compared with RL, the main difference of MARL is that multiple agents are involved simultaneously [42]. As shown by Fig. 2.4, each agent can define its state and reward and select actions independently. Since the action of one agent may affect the environment and rewards of other agents, the coordination method becomes a critical issue for MARL [43].

The main idea behind MARL is to guarantee that all agents can achieve satisfactory performance while maintaining good results for the whole system [44]. There have been many methods for multi-agent coordination, and here we introduce three strategies: the fully cooperative method, nash equilibrium, and correlated equilibrium.

Firstly, the fully cooperative method indicates that all agents work cooperatively to maximize overall performance. A typical cooperation method assumes all agents share the same global reward, and consequently they will select actions that bring higher rewards for the whole system, instead of maximizing individual potential rewards. This strategy means there is no competition between agents, and they share the same optimization objective. However, this can be impractical in practice because agents may belong to different entities and aim to maximize their individual reward regardless of other agents.

Secondly, the Nash equilibrium has been very widely applied in solving multi-agent

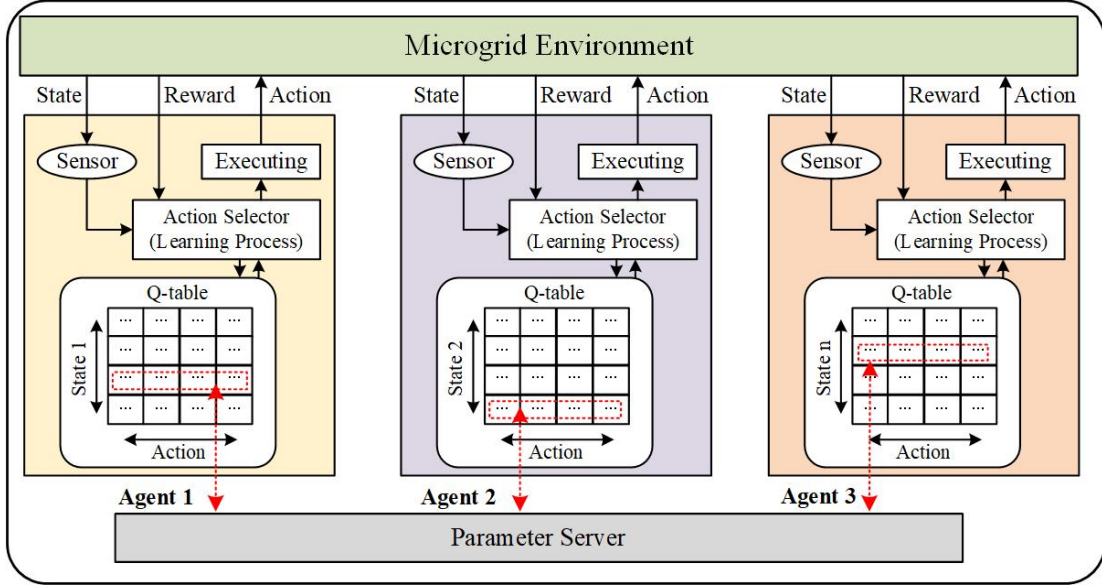


Figure 2.5: Multi-agent correlated Q-learning system with parameter sharing server

problems, and it is given by:

$$U_i(\vec{a}_{-i}, a_i) \geq U_i(\vec{a}), \vec{a} \in A, \vec{a}_{-i} \in A_{-i}, \quad (2.7)$$

where $\vec{a}$ is the action combination, $\vec{a}_{-i}$ is the action combination except agent i , A is the set of $\vec{a}$, and A_{-i} is the set of $\vec{a}_{-i}$. U_i is the utility function for agent i , which refers to Q-values or rewards in RL. The idea behind equation (2.8) is that no agent can improve its utility by only changing its action selection. It provides a solution for non-cooperative multi-agent coordination.

Thirdly, we present the correlated equilibrium. Although Nash equilibrium is a widely applied method, it can not guarantee convergence or reach a global optimal. There may exist multiple Nash equilibria in one given problem, or the Nash equilibrium does not even exist. By contrast, correlated equilibrium can be easily found by linear optimization, and the convergence and existence are further guaranteed. In correlated equilibrium, agents exchange utilities to share information, and the joint action is chosen according to the

following equations:

$$\max \sum_{\vec{a} \in A} Pr(\vec{s}, \vec{a}) Q(\vec{s}, \vec{a}) \quad (2.8a)$$

$$sub.to \sum_{\vec{a} \in A} Pr(\vec{s}, \vec{a}) = 1 \quad (2.8b)$$

$$\sum_{\vec{a}_{-i} \in A_{-i}} Pr(\vec{s}, \vec{a}_i) (Q(s, \vec{a}_i) - Q(\vec{s}, \vec{a}_{-i}, a_i)) \geq 0 \quad (2.8c)$$

$$0 \leq Pr(\vec{s}, \vec{a}) \leq 1 \quad (2.8d)$$

where $\vec{s}$ is the system state agents, $\vec{a}$ is the joint action, $Pr(\vec{s}, \vec{a})$ is the probability of choosing action combination $\vec{a}$ under state $\vec{s}$, a_i is the action of agent i , $\vec{a}_{-i}$ is the action combination except agent i , and A_{-i} is the set of $\vec{a}_{-i}$. Equation (2.8) uses Q-values to represent the expected reward. It aims to maximize the total potential reward of all agents by finding an optimal action selection probability distribution $Pr(\vec{s}, \vec{a})$. Constraints (2.8b) and (2.8d) are probability distribution constraints. The constraint (2.8c) indicates that $\vec{a}_i$ produces higher expected reward with probability $pr(\vec{a}_{-i} | s_{sub})$, which is indicated by $Pr(\vec{s}, \vec{a}_i) (Q(s, \vec{a}_i) - Q(\vec{s}, \vec{a}_{-i}, a_i)) \geq 0$.

A correlated Q-learning framework is shown in Fig.2.5, in which each agent applies Q-learning to make decisions independently. Meanwhile, they will exchange the Q-values with a parameter server for action selection coordination.

2.1.4 Transfer Reinforcement Learning

In this section, we present the TRL. Although learning-based methods, such as RL and DRL, have been generally applied for network resource allocation, most existing works do not consider the possibility of knowledge transfer. Specifically, an agent is designed for a specific task in these works, and it interacts with its environment from scratch, which usually leads to a lower exploration efficiency and longer convergence time. Furthermore, the agent must be retrained whenever a new task is assigned, even though similar tasks have been completed before. The poor generalization capability of straightforward RL methods motivates us to find a learning method with better generalization and knowledge transfer capability. On the other hand, humans can reuse the knowledge learned from previous tasks to solve new tasks more efficiently, and this capability can be built into ML as well. Such knowledge transfer and reuse can significantly reduce the need for a large number of training samples, which is a common issue in many ML methods. By

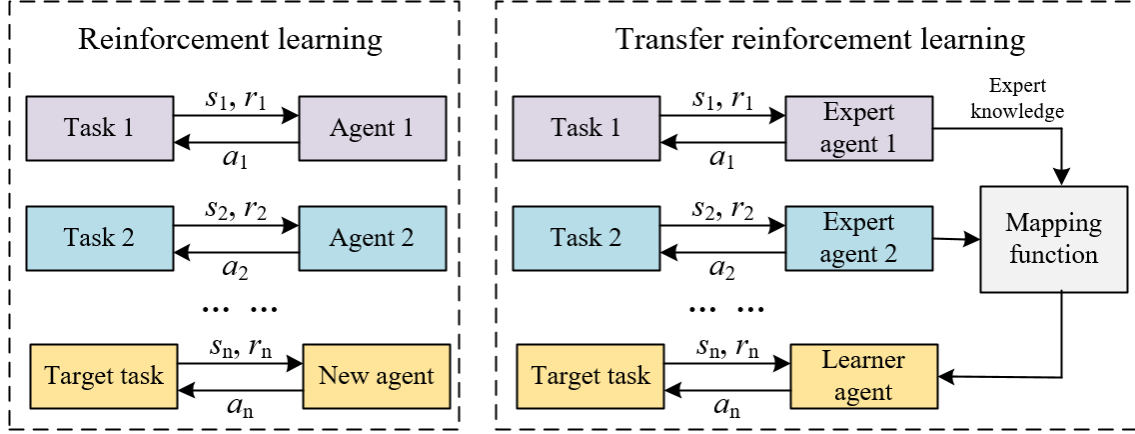


Figure 2.6: Comparison of transfer reinforcement learning and reinforcement learning.

incorporating knowledge transfer capability into ML, it is expected to reduce the algorithm design and training efforts and achieve better performance such as faster convergence and higher average reward.

In this section, we introduce the background of TRL to distinguish it from RL. As shown in Fig.2.6, given a task pool, the interactions between tasks and agents can be described by the MDP $\langle S, A, R, T \rangle$, where S is the state space, A is the action space, R is the reward function, and T is the transition probability. In RL, each agent works independently to try different actions, arrives at new states and receives rewards. The learning phase of RL can be defined as:

$$\mathcal{L}_{RL} : s \times \mathcal{K} \rightarrow a, r (a \in A), \quad (2.9)$$

where the $\mathcal{K}$ is the knowledge of this agent, s is the current state, a is the selected action, and r is the reward. Equation (2.9) indicates that RL agent utilizes the collected knowledge to select action a and receive reward r under state s .

On the contrary, TRL includes two phases: the knowledge transfer phase and the learning phase. In the knowledge transfer phase, as shown in Fig.2.6, considering task differences, a mapping function is defined to make the knowledge of experts digestible for the learner. Then the learner explores target tasks on its own and forms its knowledge. The whole process of TRL can be defined as:

$$\mathcal{L}_{TRL} : s \times \mathcal{F}(\mathcal{K}_{expert}) \times \mathcal{K}_{learner} \rightarrow a, r (a \in A), \quad (2.10)$$

Table 2.1: Comparison of reinforcement learning algorithms

Algorithms	Features	Difficulties	Applications
RL	Agent has no prior knowledge about tasks. It explores new tasks from scratch.	Long convergence time, low exploration efficiency.	Tasks with limited state-action space and no prior knowledge.
transfer learning (TL)	Improving generalization across different distributions between source and target tasks.	Negative transfer, automation of task mapping.	It is mainly designed for the supervised learning domain, e.g., classification, regression.
DRL	Combining artificial neural networks with RL architecture. Applying neural networks to estimate state-action values.	Time-consuming network training and tuning; Low sample and exploration efficiency; Training stability.	Large state-action space or continuous-action problem.
TRL	Utilizing prior knowledge from experts to improve performance of learners such as higher average reward or faster convergence.	Transfer function needs to be defined to digest prior knowledge.	Optimization of tasks with existing prior knowledge.

where the $\mathcal{F}$ is the mapping function, $\mathcal{K}_{expert}$ is the knowledge from experts, $\mathcal{K}_{learner}$ is the knowledge of the learner. In equation (2.10), knowledge from experts is utilized in the action selection of the learner, and it is expected to accelerate the learning process $\mathcal{L}_{TRL}$. In TRL, new tasks can be better handled based on the knowledge of experts.

The RL, TL, DRL and TRL techniques are compared in Table 2.1. Compared with RL, TRL has higher exploration efficiency and better generalization capability. On the other hand, although DRL is a breakthrough approach by combining neural networks with RL, the time-consuming network training is a well-known issue, and the training stability and generalization capability of DRL can cause problems. Finally, although TL has been extensively studied in the ML literature, it mainly focuses on the supervised learning domain such as classification and regression. Compared with TL, TRL can be more complicated because the knowledge needs to be transferred in the context of the MDP. Moreover, due to the dedicated components of MDP, the knowledge may exist in

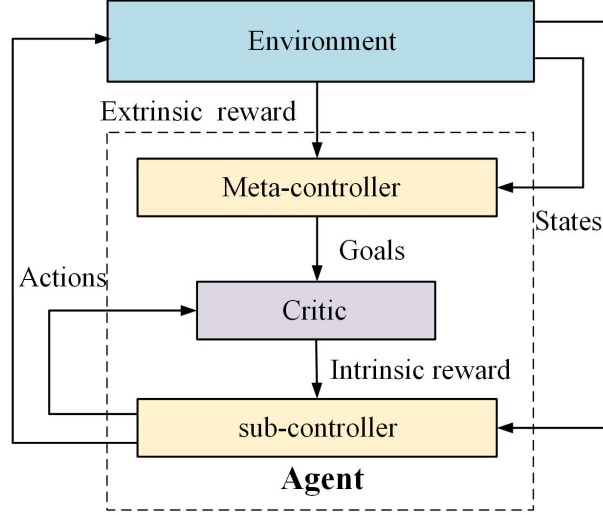


Figure 2.7: Hierarchical reinforcement learning with meta-controller and sub-controller.

different forms, which must be transferred in different ways.

2.1.5 Hierarchical Reinforcement Learning

In conventional RL, the action of agents can only apply one fixed timescale, which means all the action elements have to be decided simultaneously. However, conventional RL may have difficulty defining more complicated and realistic problems such as tasks with sparse rewards and different timescale actions. To this end, the HRL is proposed.

In HRL shown by Fig.2.7, the agent consists of two controllers, namely meta-controller and sub-controller. Accordingly, the MDP is rewritten by $\langle S, A, T, R, \mathcal{G} \rangle$, where $\mathcal{G}$ indicates the set of goals. Based on the current state $s \in S$, the meta-controller will generate high-level goals $g \in \mathcal{G}$ for sub-controllers. Then, these goals are transformed into high-level policies by the critic. Consequently, the sub-controller chooses low-level actions $a \in A$ according to high-level policies, and receives an intrinsic reward r_{in} . Finally, the meta-controller will receive an extrinsic reward r_{ex} from the environment and select new goals g' for the sub-controller. The idea behind the HRL is to introduce hierarchy architecture in RL. In particular, the meta-controller will produce high-level policies to guide the low-level action selection of the sub-controller. Compared with traditional RL, HRL is considered a more efficient learning method due to the hierarchical architecture, allowing better task management by dividing sub-goals.

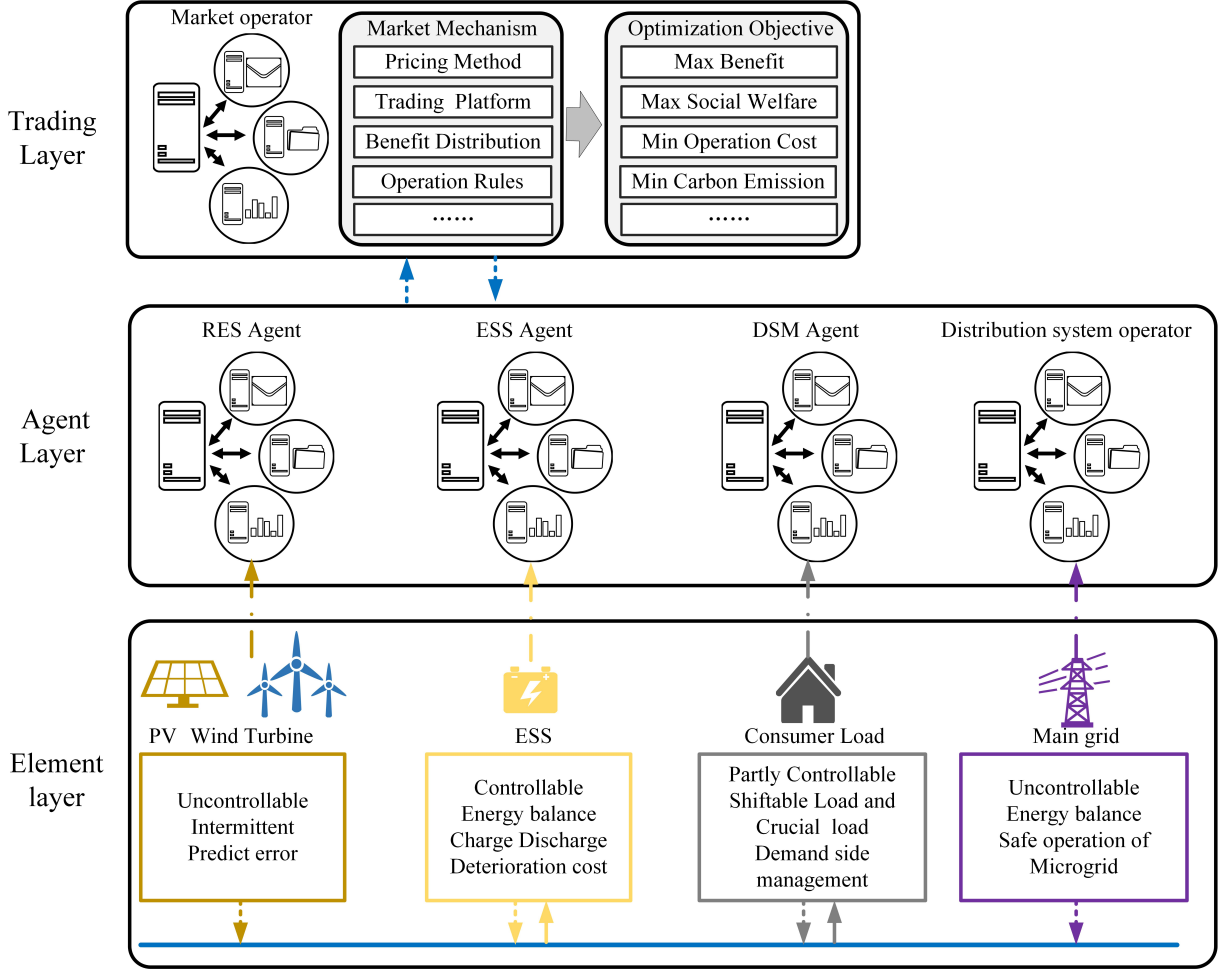


Figure 2.8: Microgrid energy trading and control framework

2.2 Literature Review

This section investigates related works on MG energy trading, ML-enabled 5G resource allocation, and optimization techniques for RIS-aided wireless networks.

2.2.1 Microgrid Energy Trading

The MG energy trading and control scheme is shown in Fig.2.8. It contains three layers: elements, agents, and the operator. The element layer contains basic MG elements, includ-

ing the electricity load, ESS, renewable energy resources such as PV and wind turbines, and so on. Then, the agent layer can aggregate and control these elements to maximize the profit or minimize the operation cost. For instance, the PV agent may control all the solar panels in the MG community, which will decide the power supply capacity of the whole MG. Meanwhile, in a MG, the electricity load devices can be divided into crucial and deferrable loads. The crucial load mainly includes light and cooking devices, which cannot change their operation time. However, the operation time of deferrable load could be changed without affecting the end user’s satisfaction much, e.g., washing machine. We assume the DSM agent is authorized by consumers to control the deferrable load of the MG. Finally, the top level is the energy trading and operating layer, including pricing method, trading platform, profit distribution, operation rules, etc. The trading layer can apply various optimization goals such as maximizing profit or social welfare and minimizing operation costs and carbon emissions.

Many centralized algorithms have been proposed for MG energy management, including the gradient search, interior point algorithm, and lambda iteration approach [45]. Moreover, considering the great potential of ML algorithms, the power loss during MG energy trading is considered in [46], and a Bayesian-based reinforcement learning technique is applied for the optimal energy trading strategy among MGs. Furthermore, the environment in a MG is related to the state of each agent, which means the MG system state could be rather complicated. DRL [39] is applied in the MG energy management field in several prior works. The DRL is used for Peer-to-Peer energy trading in [47], and an improvement is shown when compared with the rule-based strategy. Furthermore, the energy management problem is formulated to be a MDP in [48], and actor-critic based DRL is proposed to minimize the energy cost. In addition, considering the overestimation issue of standard DQN, the DDQN is used in [49] for community ESS, and the result proves the feasibility of DDQN.

Most of the works in the literature [46–49] mainly focus on the one-agent case. However, these centralized methods raise concerns about reliability and security issues, such as the failure of the central controller, which may result in a shutdown of the whole MG. In addition, resilience, privacy protection, high computation burden, and high bandwidth requirements are all further challenges for a centralized MG architecture. The MG can be modelled as multi-agent systems where storage, generation, and load are each represented by an agent. For example, a renewable energy resource unit can sell energy to the grid or the ESS. The decision on whom to sell and the price will determine the revenue for the renewable energy resource agent. Cooperation and competition may exist between agents to maximize individual utility or the overall profit [50]. In this case, the agents need to trade energy to maximize their profit, and the competition may increase the difficulty of

balancing the profit.

Recently there have been many studies on MARL based MG energy management. A MARL-based method is proposed in [51], where each agent has the learning ability to maximize its profit. In addition, cooperative reinforcement learning is proposed in [52] for the MG economic dispatch, where the diffusion strategy guarantees cooperation. In MARL, each agent is expected to learn the cooperation strategy. Naturally, MARL can also be generalized to multi-agent DRL. Nash-based methods are generally used to distribute profit among cooperative agents [53]. However, it must exchange information iteratively, which may lead to a heavy computation burden.

In [54], power loss in energy trading is minimized, where the coalition information is transferred between MGs. A decentralized MG energy management method is proposed in [55], where Nash equilibrium is used for coordination by sharing utility values. Furthermore, fuzzy Q-learning is applied in [56] for MG energy management. A diffusion strategy is used in [57] for cooperative reinforcement learning where the agent only needs to communicate with its neighbours. Considering the huge state-action space of MG, DQN is applied for energy trading in [58], and the utility of MG is balanced by Nash equilibrium. In most of the prior works, communication has been a key element as the agents collaborate by sharing critical information, e.g., coalition information [54], utility values [55, 58], state variables [56, 57] to make decisions.

Communications between agents are the key to a multi-agent system. However, the underlying communication system is not always reliable in practice, and IoT devices in the MG may experience system failures, packet loss, and delayed packets. Although MG energy management has been investigated by various models and algorithms, the communication failure problem has not been considered before. Note that, in [59], the effect of communication delay on MG economic dispatch is investigated, and it presents that communication delay will lead to a system performance fluctuation. [60] proposed a novel consensus-based economic dispatch algorithm to handle communication delay in MG, where the communication delay results in a higher cost for MG. The work of [59] and [60] show that it is necessary to consider the effect of communication systems in the MG energy management. However, to the best of our knowledge, the impacts of communication failure on MG energy management have not been investigated before. We have proposed a MARL scheme for MG energy management in [18], and a multi-agent DQN method in [19]. But communication-related problems were still not studied in those works. Nevertheless, handling such uncertainty brings substantial performance improvements as demonstrated in systems other than MG such as electric vehicles.

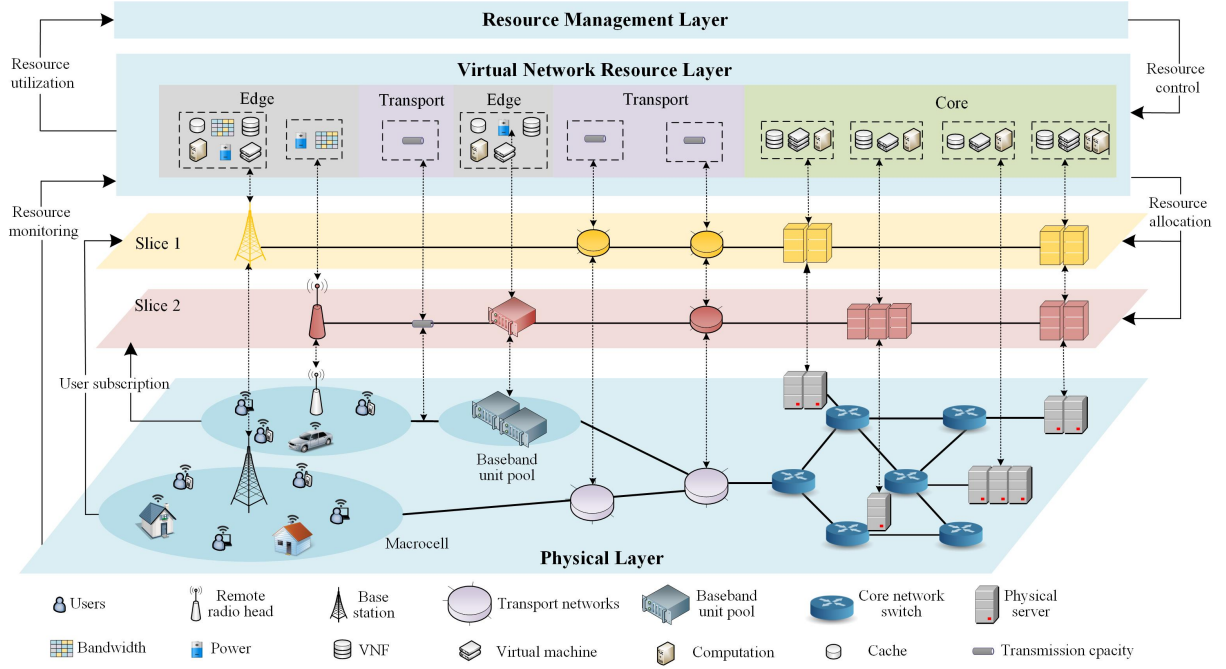


Figure 2.9: 5G resource management architecture for network slicing

2.2.2 Machine Learning based 5G Resource Allocation for Network Slicing

Network slicing presents demanding requirements for resource management, especially at the edge and radio domains, since tight resources are allocated concurrently between multiple slices in a very short timescale to satisfy different service level agreements. Although managing one single resource has been widely studied in the literature, the evolving network architecture requires the holistic management of multiple resources [61]. For example, in the MEC-enabled 5G RAN, the computation offloading and bandwidth allocation need to be jointly considered to guarantee the network performance. Furthermore, end-to-end network slicing calls for cross-domain resource allocation that involves RAN, edge, transmission and core network [62]. Thus a unified architecture is needed to include all the cross-domain resources in a systematic view.

Fig.2.9 shows our defined resource management architecture, including the physical layer, virtual network resource layer, and management layer. First, the physical layer consists of physical network devices that can be split into different network slices. For instance, slices 1 and 2 in Fig.2.9 may require multiple network resources simultaneously,

and both slices can share one single physical device. Then, by monitoring the status of the physical layer, we define a virtual network resource layer to aggregate the resources of different network elements. For example, the baseband unit pool in Fig.2.9 includes power, cache, computation resources, and so on, while the remote radio head mainly involves bandwidth and power as manageable resources. Finally, the management layer works as a control plane for the virtual resources, and these resources will be allocated to slices to fulfill their requirements.

Meanwhile, we summarize diverse network resources in Table 2.2, including the features, applications, and management objectives of each resource. Note that the computation resource allocation is involved in both edge and core networks. The network edge devices usually have limited computation capacity, which makes the computation task offloading a critical concept. On the contrary, the core network has an abundant computation capacity, and more computationally intense tasks without real-time requirements can be processed. virtual machines (VM)s can also be deployed in both edge and core networks to implement virtual network functions (VNF)s, but the edge side only has limited hardware resources to support the VMs or containers. Table 2.2 implies that resource management of RAN and the edge is more complicated than transport and core networks due to limited available resources, more dynamic conditions and the need for actions on a shorter timescale.

Then we briefly review the latest AI/ML techniques for resource management applications, including supervised learning, unsupervised learning, RL, federated learning, graph learning, and transfer learning. Note that the main goal of this work is not to review all the AI/ML techniques on wireless networks but serves as a compressed taxonomy for new techniques [31]. Table 2.3 and 2.4 summarizes the algorithms, main features, difficulties and typical applications of each learning method.

1) Supervised and Unsupervised Learning

Supervised learning algorithms map input data to labelled output data for classification or prediction purposes. Well-known algorithms and neural network architectures include artificial neural networks, recurrent neural network (RNN), long short-term memory (LSTM) network, support vector machines, decision trees, and so on [63–65]. Supervised learning can be applied for traffic prediction and classification for resource management. However, supervised learning with neural networks suffers because network training is too much dependent on hyperparameter tuning and the difficulty in obtaining labelled data.

By contrast, unsupervised learning intends to analyze and cluster unlabeled data [66]. Well-known approaches include the k-means and Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm. The deep belief network can also be used for unsupervised learning to extract the features of unlabeled data. These techniques discover

Table 2.2: Summary of network resources in 5G networks.

Domains	Resources	Features	Applications	Typical management objectives
RAN and edge	Radio	Spectrum requires careful allocation, such that transmission requirements of various slices are satisfied using this very scarce resource.	Spectrum allocation	Maximizing throughput or minimizing the delay; improving spectrum efficiency.
	Power	Power is a key resource in RAN. Power control can mitigate interference and improve energy efficiency for network devices.	Power allocation for interference management	Minimizing interference and energy cost; maximizing network throughput.
	Cache	Edge caching can significantly reduce the network delay by storing popular contents in the edge, and caching capacity is a limited resource.	Content replacement and delivery in network edge.	Maximizing hit ratio in a cache for better network performance (lower delay) or economic metrics.
	Computation	Computation capacity is a key resource in the edge, where computationally intensive tasks from user devices (UEs) or IoT devices can be processed in the network edge instead of the cloud.	MEC offloading between UEs, MEC server and cloud.	Minimizing processing delay or computation cost.
	VNFs, VMs, containers	VMs or containers are required when VNFs are deployed in the mobile edge, and thus they become important resources to run VNFs.	VMs and VNF placement.	Minimizing response time, energy cost and hardware usage.
Transport	Fronthaul	Connections between remote radio heads and baseband unit pool in the edge cloud RAN.	The connection capacity is considered as critical transmission constraints. Pricing models can be applied to maximize the profit by managing the transmission capacity allocation.	
	Backhaul	Connections from RAN to the core network, or from edge (base stations or baseband unit pools) to the core network.		
Core	VMs and VNFs	Compared with mobile edge, core network is capable of instantiating a higher number of VMs to implement VNFs. The number of deployed VMs determines the service capability of a single network node.	VMs and VNF placement.	Minimizing cost or energy consumption; guaranteeing network performance and reliability.
	Computation	In the core network, available computation resources of physical hosts in network nodes should be allocated to VNFs to process tasks.	CPU allocation; work load assignment.	Minimizing computation delay or cost.

the hidden similarities and differences of unlabeled data. However, the long training time and high computation complexity are potential issues for unsupervised learning. Mean-

Table 2.3: Summary of AI techniques for 5G resource management-Part 1.

Learning methods	Typical algorithms	Main features	Difficulties	Typical applications
Supervised learning	Artificial neural network, RNN, LSTM, support vector machine, etc.	Training the algorithm to best map inputs to outputs in a dataset, and thus labeled data is required for training. Note that each algorithm has its features, e.g., LSTM can better handle the long-term dependencies of data.	Algorithm training relies heavily on fine-grained datasets, which may be inaccessible in practice. Network training can be time-consuming due to hyperparameter tunings such as learning rate and the number of hidden layers or units.	Traffic load prediction and classification for slices.
Unsuper-vised learning	K-means, DBSCAN	Discovering hidden patterns or similarities in unlabeled datasets.	High computation complexity; noisy results.	User clustering to create new slices.
Reinfor- cement learning	Q-learning	Agent interacts with an environment to maximize the expected reward. A Q-table is used to record state-action values in Q-learning.	Requiring long convergence time, especially for problems with large state-action space.	RL is the most generally applied AI technique for the resource management and optimization wireless networks, e.g., minimizing network delay or energy consumption, maximizing network throughput or revenue. For example, multi-agent reinforcement learning may be used for resource allocation of slices.
	Actor-critic learning	The actor is considered as a policy structure to select actions, and the critic will estimate the value function for the actions of the actor.	Long convergence time; instability caused by the interplay between actor and critic.	
	Deep rein-forcement learning	Combining neural networks with RL framework to predict state-action values instead of using Q-tables. DRL can be applied to address the large state-action space problem.	Time-consuming network training; tedious hyperparameter tuning; network training stability; low sample efficiency.	
	Double deep Q-learning	Decoupling the action selection and evaluation of deep Q-learning to prevent overestimation and provide a better Q-value estimation.		
	Multi-agent reinforcement learning	Each agent implements the RL or DRL independently to achieve its goal or optimize an overall objective.	The coordination mechanism of multiple agents requires dedicated design.	

while, the output results are hard to validate due to the unlabeled dataset. Unsupervised learning can be deployed for the user or traffic clustering, especially mobile users, for resource management of 5G networks [67].

2) Reinforcement Learning (RL)

RL has been widely used to optimize the resource management problems such as physical

Table 2.4: Summary of AI techniques for 5G resource management-Part 2.

Learning methods	Typical algorithms	Main features	Difficulties	Typical applications
Federated learning	Federated deep learning, federated DRL	Training models by distributed datasets of multiple learners, then feeding back global model parameters to local learners for their use. Private and sensitive data can be well protected with federated learning.	Communication overhead for updating local models; systems heterogeneity caused by various local devices such as different storage and computation capacity.	Each slice can train a local model, and the central controller produces a global model for coordination.
Graph learning	Graph neural networks, graph convolutional networks	Graph learning maps the feature of a graph to the vectors without projecting the graph into a low dimensional space, which can be used for classification, link prediction, and matching.	Scalability issue; generative graph learning; dynamic graph.	GNN-based digital twin for network slicing.
Transfer learning	Supervised transfer learning	Improving generalization capability by reusing the pre-trained model such as neural networks of source tasks to related target tasks.	Task mapping function definition; negative knowledge transfer.	Classification and prediction for resource management and RAN slicing.
	Transfer reinforcement learning	Utilizing knowledge of experts to improve learner's performance within the context of MDP, which aims at faster convergence and higher average reward.	The knowledge transfer function requires dedicated design.	Optimization of network resource management and RAN slicing with fewer training samples and faster convergence.

resource block allocation, power allocation, and so on [68]. RL algorithms include Q-learning, actor-critic learning, DRL, etc ¹. In RL, the agent aims to maximize the long-term expected reward within the MDP. Q-learning is the most widely used RL algorithm due to its simplicity, but it suffers from a long convergence time, especially for problems with large state-action space. Meanwhile, actor-critic learning defines an actor for action selection and a critic for action evaluation, which uses the value function as a baseline for policy gradients. DRL has been used to overcome the slow convergence problem of Q-learning and the challenge of storing a large Q-table. Instead of deploying huge Q-tables, the state-action values in DRL are predicted by neural networks, and thus it can better handle large state-action space problems. Furthermore, double deep Q-learning is designed to prevent the over-estimation issue of DQN, in which the main network selects actions, and the target network evaluates actions.

By transforming diverse optimization problems into unified MDPs, RL methods can avoid the possible complexity of defining a dedicated optimization model. RL methods

¹There are many versions of RL techniques in the literature. Summarizing all is out of the scope of this work. We provide examples of the most widely used algorithms.

have been used in many network resource management problems. Nevertheless, it demands a huge amount of samples for training, and the trained algorithm can only handle specific problems with poor generalization capability.

3) Federated Learning

Federated learning is one of the state-of-the-art ML methods that is applied in wireless networks. Federated learning algorithms are trained across multiple decentralized local datasets without sharing training data, where only the local model parameters are sent back to a central server for an improved global model. Then, the well-trained global model parameters are fed back to local learners to update local models. As an example in wireless communications, user equipment (UE) can train its federated learning model based on its data, then the BS can form a global model and send it back to UEs as in [69]. Federated learning addresses data privacy and security issues, since each learner can keep its training data on board. However, the learners still need to exchange the parameters of the trained models. As such, the communication overhead becomes a critical issue for federated learning, especially for wireless networks with limited bandwidth resources. Meanwhile, considering various storage, computational, and communication capabilities of local devices, the system heterogeneity can be another challenge for applying federated learning on wireless networks.

4) Graph Learning

Graphs are widely used to represent the network architecture, and graph learning refers to ML algorithms on graphs. In particular, graph learning is designed to extract the main features of graphs, such as node and edge connections, to vectors with the same dimensions in the embedding space. For instance, the graph neural network aims to learn a state embedding that contains the neighborhood information of each node, and the graph data is directly mapped to the neural network's output without projecting the graph into a low-dimensional space. Graph learning has been successfully applied in node classification, graph classification, link predictions, and so on, which shows superior performance than other existing approaches. Considering the graph nature of wireless networks, graph learning shows significant potential for resource management solutions such as power and spectrum allocation [70]. Meanwhile, graph learning still faces challenges, e.g., dynamic graphs, scalability issues, and generative graph learning. For example, users with high mobility may lead to dynamic graph structures, but graph learning architectures cannot change adaptively if edges and nodes appear or disappear frequently.

5) Transfer learning

Although various ML methods have been introduced before, it is worth noting that: i) most algorithms require a large number of samples to explore the target task; ii) these algorithms are designed for specific tasks with very limited generalization capability. As

a result, even though similar tasks have been completed, the algorithms still need to be retrained for new problems. To this end, TL aims to reuse the knowledge and existing model of completed tasks. Specifically, a model developed for one task is reused as the starting point for other related tasks. Indeed, humans can apply their previous knowledge to solve new works more rapidly. This reduces the need for a huge amount of training samples, which is a common issue in ML. For example, one gNodeB can learn the user-association experience of other gNodeBs for interference mitigation in mmWave networks as in [67], and it achieves a faster convergence speed than conventional Q-learning. Despite the potential advantages, the knowledge may exist in different forms in ML algorithms, and knowledge transfer methods vary between tasks. For instance, Q-values, learning rate, and action selections in Q-learning can all be considered as knowledge, and one needs to define various mapping functions to make these prior knowledge digestible for target tasks.

The stringent and heterogeneous QoS requirements of services have become a challenging problem in 5G [71, 72]. Considering the limited radio resources and increasing bandwidth demand, different methodologies are proposed for 5G radio resource allocation. A joint link adaptation and resource allocation policy is proposed in [73], which dynamically adjusts the block error probability of URLLC transmissions based on cell load. A risk-sensitive method is used in [74] to allocate resources for the incoming URLLC traffic, while minimizing the risk of the eMBB transmissions and ensuring URLLC reliability. Puncturing technique is applied in [75] to guarantee minimum latency of URLLC, where eMBB traffic is scheduled at the beginning of slots, while URLLC traffic can puncture at any time with a higher priority.

A common feature of aforementioned works is that the URLLC traffic is scheduled on top of the eMBB traffic such as puncturing technique [74, 75], and a potential priority is applied to guarantee the latency and reliability of URLLC traffic [74, 75]. As a result, the eMBB traffic will be affected by degraded throughput. Allocation of limited bandwidth resources among slices has been the main challenge in RAN slicing [76]. The allocation should meet various QoS requirements under a limited bandwidth budget constraint. Meanwhile, another important problem is the increasing complexity of wireless networks, e.g., the evolving network architecture, dynamic traffic patterns and increasing device numbers, which makes it harder to build a dedicated optimization model for resource allocation.

To this end, the emerging ML techniques have become a promising solution. For example, in RL, the agent interacts with environments to maximize the long-term reward based on MDP, and the complexity of defining a dedicated optimization model is avoided. In [77], a Latency-Reliability-Throughput Q-learning (LRTQ) algorithm is proposed for jointly optimizing the performance of both URLLC and eMBB users. [78] develops an RL method to select different scheduling rules according to the scheduler states, which aims

to minimize the traffic delay and packet drop rate. The random forest algorithm is applied in [79] to accomplish the Transmission Time Interval (TTI) selection for each service, and the result shows a lower delay and lower PDR for URLLC traffic while guaranteeing the eMBB throughput requirements. [80] presents an RL method for resource optimization in 5G network slicing, and the results show improved network utility and scalability. DQN is deployed in [81] for end-to-end network slicing resource allocation in multi-slice and multi-service scenarios. Furthermore, [62, 82] use DRL for resource allocation in 5G, in which neural networks are used to learn allocation rules.

The previous works mainly concentrate on RAN slicing. While the spectrum is indisputably the most critical resource of RAN, other resources are equally important to guarantee the network performance, such as computation and cache resources [83, 84]. To support the emerging computation-intensive applications, deploying MEC servers becomes an ideal solution [85, 86]. The computation tasks can be processed in the MEC server instead of the central cloud; thus, a lower delay is achieved. The work of [87] shows that joint resource allocation slicing has a better performance than slicing one single resource, and [88] introduced a mathematical model to slice mobile network and edge computation resources jointly.

Although incorporating computation capability into RAN will bring significant benefits, it also increases network management complexity, especially when multiple slices are involved. [89] introduced a RAN slicing method to allocate radio and computation resources dynamically, and a constrained learning scheme is defined. A decentralized DRL method is presented in [90] for network slicing, which ensures service level agreements under networking and computation resources constraints. The joint RAN slicing and computation offloading problem is investigated in [91], and multi-agent DQN is applied to maximize the communication and computation resources utilization. Furthermore, [92] proposed an actor-critic network-based DRL method for joint radio and computation resource allocation of virtualized RAN.

Similarly, incorporating caching into the RAN has attracted interest from academia and industry. Some novel network architectures have been proposed to harvest the potential advantages of edge caching [93]. Indeed, edge caching allows storing data closer to the users by utilizing the storage capacity at the network devices, reducing traffic to the core network. Moreover, caching can save the backhaul capacity without affecting the network delay. Cache placement and caching strategies have been extensively studied in numerous works, but the problem is usually addressed disjointly, without considering the RAN. For example, a location-customized caching method is presented in [94] to maximize the cache hit ratio, and content caching locations are optimized in [95] by considering both cloud-centric and edge-centric caching.

Augmenting edge devices in RAN with caching will significantly improve 5G networks. However, this leads to higher complexity. In addition, the complex network dynamics, especially the stochastic arrival requests of slices, make the underlying network optimization challenging. Fortunately, ML techniques offer promising solutions. By applying a RL scheme, the potential complexity of defining a dedicated optimization model is avoided. For instance, DDQN is applied for computation offloading within sliced RANs in [96]. DRL is deployed in [97] for spectrum allocation in integrated access and backhaul networks with dynamic environments. MARL is used in [98] for distributed dynamic spectrum access sharing of communication networks. Moreover, a DRL-based method is proposed in [99] for mobility-aware proactive resource allocation, which pre-allocates resources to mobile UEs in time and frequency domains. Finally, [100] proposed a DQN based intelligent resource management method to improve the quality of service for 5G cloud RAN. The aforementioned works show that various ML methods have been applied for the resource management of wireless networks, including RL, DRL, DDQN, multi-agent reinforcement learning, and so on.

In these works, the main motivations for deploying ML algorithms are the increasing complexity of wireless networks and the difficulties of building dedicated optimization models. Indeed, the evolving network architecture, emerging new network performance requirements and increasing device numbers make conventional methods such as convex optimization more complicated. ML, especially RL, offers a good opportunity to reduce the optimization complexity and make data-driven decisions. However, a large number of samples are needed for the learning process, which means a long training time and low system efficiency. Furthermore, after the long learning process, the agent can only handle one specific task without any generalization. This learning process has to be repeated when a new task is assigned. DRL can be considered a solution to reduce the number of samples and improve convergence. However, it is still limited to a localized domain, and the neural network training can be time-consuming because of the hyperparameter tuning. To this end, we propose DTRL-based solutions for the joint radio and cache resource allocation of 5G RAN. The TRL has been applied in [67] for intra-beam interference management of 5G mm-Wave, but here, the resource allocation problem is more complicated due to a much bigger state-action space. The proposed scheme has a satisfactory knowledge transfer capability by utilizing the knowledge of expert agents, and it outperforms conventional DRL by faster convergence and better network performance.

2.2.3 Optimization Techniques for RIS-aided Wireless Networks

While 5G has entered the commercialization phase, the research community has started the exploration of future 6G networks. Compared with previous generations, 6G networks are expected to present more stringent performance requirements, i.e., terabits per second data rates for virtual reality, and more than $10^7/km^2$ connection densities with significantly lower latencies than 5G networks [101]. One of the main obstacles to wireless network evolution is the uncontrollable radio environment with reflections, diffractions, and scattering. Recently, RISs have emerged as promising techniques to enhance wireless signal propagation [102]. In particular, the core feature of RISs is to manipulate the signal propagation path by intelligently configuring numerous small elements. Each RIS element can independently tune the phase of the incident signal, creating a smart radio environment [103]. RISs not only are technically attractive but also require low energy consumption and incur less hardware cost, making it promising to enhance the spectral efficiency for real-world deployments. Given these advantages, RISs can be combined with other emerging techniques, including multiple-input multiple-output, millimeter-wave communications, unmanned aerial vehicle (UAV) networks, non-orthogonal multiple access (NOMA), vehicle-to-everything networks, and so on. Many existing studies and implementations have demonstrated RIS's capability of improving network capacity, coverage, energy efficiency, and security.

Despite their potential, integrating RISs into wireless networks will significantly increase the complexity of network management [104]. For example, each RIS element requires independent phase-shift configurations, leading to large solutions spaces for optimization algorithms. The RIS configuration is more complicated when other control variables are jointly involved, such as beamforming, spectrum allocation, NOMA decoding order, or UAV trajectory design. Therefore, advanced optimization techniques are of paramount importance to handle such complexity and take full advantage of RISs.

Existing methods for optimizing RIS-aided wireless networks can be classified as three approaches: model-based, heuristic, and ML approaches. We compare these approaches in Table 2.5, including main features, advantages, drawbacks, difficulties, and applications for RIS.

Table 2.5: Summary of control and optimization techniques for RIS-aided wireless networks

Optimization approaches	Main features	Advantage	Drawbacks	Difficulties	Application scenarios for RISs
Model-based algorithms	Model-based algorithms usually require full knowledge of the problem to find near-optimal solutions by using transformation and relaxation.	Model-based algorithms can provide detailed proofs and explanations for the optimality. Target problems are efficiently solved with guaranteed optimality once the closed-form solution is achieved.	Model-based solutions are usually problem-specific with certain requirements such as convexity and continuity. It has difficulty adapting to dynamically changing environments.	It has to apply transformations, division, and relaxation to convert the problem to specific forms. These transformations need a dedicated design for each problem.	Numerous algorithms have been developed to solve RIS-aided optimization problems, such as alternating optimization to decouple the active and passive beamforming.
Heuristic algorithms	It applies heuristic rules to find a trade-off between optimality and computational complexity, providing local optima and low-complexity solutions.	Heuristic algorithms have much lower computational complexity. It has few requirements for the properties of target problems.	It only presents local optima in the current stage, indicating a bad performance in some cases.	Heuristic rules should be carefully selected and designed, directly affecting the algorithm performance.	Heuristic algorithms can provide low-complexity alternatives, i.e., sequential phase shift and on/off control using greedy rule, phase-shift optimization using GA.
ML techniques	ML techniques provide unified control and optimization algorithms for certain types of problems. Most algorithms are easily applied without requiring dedicated design.	Data-driven approaches avoid the complexity of building dedicated optimization models. It can better adapt to the dynamic wireless environment given the learning capability.	It may require long iteration numbers for the algorithm training. ML optimization techniques do not guarantee optimality.	Algorithm training is the main difficulty of applying ML, which is data and computation-demanding.	Various ML techniques have been applied for RIS-related optimizations, e.g., neural networks for CSI prediction, and DDPG for continuous RIS phase-shift optimization.

1) **Model-based method:** Table 2.5 shows that model-based methods usually require full knowledge of the defined problem for transformation and approximation. Model-

based approaches can provide efficient and stable solutions once the problem is properly reformulated. RIS passive beamforming has been widely investigated in various scenarios, i.e., multiple-input multiple-output networks [105–108], multiple-input single-output networks [109–112], single-input single-output networks [113], NOMA [114], millimeter-wave [115] and vehicle communications [115]. These works mainly consider BS beamforming vectors and RIS phase shifts as control variables, which are known as joint active and passive beamforming, and the total transmit power and RIS phases are included as constraints. However, the joint optimization problem is very challenging due to network dynamics and the large number of RIS elements. The fractional terms of signal-to-interference-plus-noise ratio (SINR), logarithm introduced by Shannon theory, and non-convex constraint of RIS phase shifts lead to significant complexity for the joint optimization. Consequently, most existing studies decouple the control variables using alternating optimization [105–108, 111]. For example, a widely applied scheme is to first optimize the BS beamforming vectors, then solve the RIS phase shifts sub-problem iteratively. Finally, as one of the core control variables, the RIS phase-shift control is supposed to be continuous in many studies. The first reason is that discrete control variables lead to integer constraints that are NP-hard; another reason is that the achieved results can be converted into the nearest discrete values using the rounding method [116]. However, model-based algorithms are usually complicated to design, indicating a series of transformations, e.g., decoupling the denominator and numerator in SINR terms and relaxing integer constraints.

2) **Heuristic algorithms:** The primary benefit of the heuristic method is the low implementation complexity, decoupling the problem into multiple stages and solving sequentially, i.e., optimizing RIS control in an element-by-element manner. Heuristic algorithms achieve a trade-off between optimality and low computational complexity. For example, genetic algorithm (GA) and particle swarm optimization (PSO) are used for RIS phase shifts in [117] and [118] to maximize the data rate. Meanwhile, statistical CSI is investigated in [119] to obtain a closed-form expression of the uplink ergodic data rate, then GA is deployed for phase control to maximize the data rate. In addition, Tabu search is applied to irregular RIS to decide the element design in [105]. Note that heuristic algorithms may be combined with model-based algorithms. For instance, the energy-efficiency maximization problem in [120] is decoupled into the beamforming optimization, phase control, and RIS on/off optimization, in which beamforming and phase control is solved by successive convex approximation (SCA), and RIS on/off is optimized by the greedy algorithm. Other combinations can be found in [121] by combining semi-definite relaxation (SDR) with greedy heuristic, and in [122] by combining SCA and SDR with matching methods.

3) **ML algorithms:** Data-driven ML algorithms present unified optimization schemes, applying to diverse problems without dedicated design. Most optimization problems can

be converted into unified MDPs that include state, action, transition probability and rewards, and then RL is utilized to maximize the reward for a higher sum-rate or energy efficiency. For example, DRL has been used for RIS phase-shift optimization in [123–125]. In these studies, continuous phase shifts are quantized to form discrete action spaces for DQN or DDQN. On the contrary, deep deterministic policy gradient (DDPG) can handle continuous action spaces directly without quantization, which has been used for continuous RIS phase-shift control in [126–128]. Then, supervised learning applies neural networks for prediction-based control, utilizing CSI to predict achievable data rate, but it relies on fine-grained labelled datasets [129, 130]. Meanwhile, neural networks may be used in an unsupervised manner by involving the objective function in the loss function [131, 132]. Such an unsupervised learning approach can reduce the dependence on labelled datasets.

Chapter 3

Multi-agent Correlated Q-learning for Decentralized Microgrid Energy Management

The energy management of MGs is complicated when renewable energy resources, ESS and DSM are jointly optimized. Meanwhile, these systems may belong to different entities with competition. Nash equilibrium is the most commonly used approach for coordinating such entities; however, Nash equilibrium cannot guarantee convergence and existence. To this end, this Chapter applies correlated equilibrium to coordinate agents with guaranteed convergence. We build an energy trading model using the mid-market rate and propose a CEQ algorithm to maximize the revenue of each agent. Our results show that the proposed CEQ can balance the agents' revenue without harming total profit. In addition, compared with Q-learning without correlation, CEQ could save 19.3% cost for the DSM agent and obtain 44.2% more profit for the ESS agent.

3.1 Community Microgrid System Model

This section introduces the demand side management model, the PV model and the energy storage system model. Then, an energy trading model is built using the mid-market rate.

3.1.1 Demand Side Management Model

The user demand is divided into crucial loads and deferrable loads. The DSM agent does not interfere with the crucial loads, e.g., lighting and cooking devices [133]. However, deferrable loads, such as dishwashers or water heaters, can be deferred to another time by the DSM agent [134].

For n sets of deferrable devices, the state is described as:

$$\vec{\varrho}_t = [\varrho_{t,1}, \varrho_{t,2}, \dots, \varrho_{t,i}, \dots, \varrho_{t,N^{de}}] \quad (3.1)$$

where $\varrho_{t,i} = 1$ represents turning on the devices in set i at time t , and $\varrho_{t,i} = 0$ represents turning off. N^{de} indicates the total number of deferrable devices.

The power demand of DSM agent at time t is:

$$P_t^{DSM} = \sum_{i=1}^{N^{de}} P_i \varrho_{t,i} \quad (3.2)$$

where P_i is the power of deferrable devices set i . We assume an average power consumption for the duration $[t, t+1]$.

The deferrable devices must be served before a certain time limit. The waiting time for devices is described as:

$$\vec{\iota}_t = [\iota_{t,1}, \iota_{t,2}, \dots, \iota_{t,i}, \dots, \iota_{t,n}], \quad (3.3)$$

where $\iota_{t,i} = 1$ indicates that device set i is still under service at time t ; $\iota_{t,i} = 0$ indicates that this device set has been serviced or its turn has not come. If one device reaches its time limit and has not been serviced, we assume it will be turned on mandatorily.

3.1.2 PV Model

In this work, we assume the PV power can be predicted with an acceptable error [135].

$$P_t^{PV} = \hat{P}_t^{PV} \quad (3.4)$$

where $\hat{P}_t^{PV}$ is the predicted PV power.

3.1.3 Energy Storage System Model

Here we consider a centralized ESS in the community MG. We use ς_t to denote the state of ESS at time t . To enhance the flexibility, two discharging levels are denoted as ς_t equals 0.5 and 1, meaning that the ESS can be either fully discharged or half discharged. The power of ESS is:

$$P_t^{ESS} = P^{char} \varsigma_t \quad (3.5)$$

$$\varsigma_t = \begin{cases} -1 & \text{charge} \\ 0 & \text{unchanged} \\ 0.5 & \text{discharge} \\ 1 & \text{discharge} \end{cases} \quad (3.6)$$

where P^{char} is a constant charging power.

The state of charge (SOC) of ESS is updated according to:

$$SOC_{t+1} = SOC_t - \frac{P^{char}}{C^{ESS}} \varsigma_t \quad (3.7)$$

where C^{ESS} is the capacity of ESS.

3.1.4 Energy Trading Model

The defined energy trading model is presented in Fig.3.1. The DSM agent could choose ESS or the main grid as its energy source. The ESS agent could use PV or grid power to charge, and it can sell its energy to the DSM agent or main grid. We make the following assumptions to build the energy trading model:

Assumption 1: Compared with the main grid, PV power has a lower price: $p^{PV} < p_t^{bgrid}$. Considering the rationality of the main grid, $p_t^{sgrid} < p_t^{bgrid}$.

Assumption 2: Surplus PV power is only available when PV power generation is more than the crucial load. Note that energy trading of the crucial load is out of the scope of this work.

Assumption 3: If DSM is unable to consume all the ESS energy, ESS could sell the rest energy to the main grid.

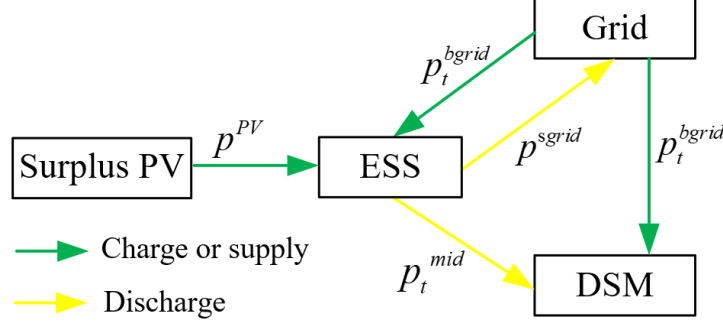


Figure 3.1: Microgrid energy trading model with ESS, DSM and PV agents.

Table 3.1: Strategy combination of players with cooperation and threat

Value of middle price		ESS agent	
		Cooperation	Threat
DSM Agent	Cooperation	$0.5p^{sgrid} + 0.5p^{bgrid}$	$0.25p^{sgrid} + 0.75p^{bgrid}$
	Threat	$0.75p^{sgrid} + 0.25p^{bgrid}$	Cooperation breaks

The mid-market rate has been widely used in peer-to-peer energy trading research as in [39]. We use p_t^{mid} to represent the price of ESS selling electricity to DSM.

$$p_t^{mid} = \frac{p_t^{sgrid} + p_t^{bgrid}}{2} \quad (3.8)$$

where p_t^{sgrid} and p_t^{bgrid} represent the prices of selling power to the main grid and buying power from the main grid separately, respectively.

However, in a competitive environment, it is reasonable to assume that one agent may take risks for higher profit, denoted as a threat strategy. The strategy combinations are presented in Table 3.1. If one agent chooses a threat strategy and another agent still cooperates, the former agent will earn more profit. On the other hand, if both choose threat strategies, cooperation breaks up, and both agents will exchange power with the main grid. In this work, we will use correlated equilibrium to coordinate agents' actions, avoid conflict and balance the revenue.

3.1.5 Problem Formulation

Both ESS and DSM agents make decisions independently to optimize their individual operations. The optimization objective of the ESS agent is maximizing its revenue:

$$\max(P^{char} \sum_{t=1}^{T^{op}} \varsigma_t p_t^{clear}) \quad (3.9)$$

where T^{op} is the simulation period; p_t^{clear} is the price of ESS agent buying/ selling electricity, which is known as the market clearing price [31]. Clearing price denotes an ideal situation, where the demand equals the supply and no shortage or surplus exist in the market. P^{char} and ς_t have been defined in equation (3.5).

On the other hand, the optimization objective of DSM agent is minimizing cost:

$$\min(\sum_{t=1}^{T^{op}} P_t^{DSM} p_t^{buy}) \quad (3.10)$$

where p_t^{buy} is the price of buying power at time t .

The optimization has to obey the following constraints:

$$P_t^{PV} + P_t^{grid} + P_t^{ESS} = P_t^{DSM} \quad (3.11)$$

$$SOC_{min} \leq SOC_t \leq SOC_{max} \quad (3.12)$$

$$\varrho_{t,i} \leq \iota_{t,i} \quad (3.13)$$

Equation (3.11) denotes the energy balance constraint, where P_t^{grid} is the power of the main grid. Equation (3.12) is the SOC constraint, where SOC_{min} and SOC_{max} are lower and upper bound of ESS, respectively. Equation (3.13) is the DSM constraint, which means only devices that have not been serviced could be turned on.

3.2 Correlated Q-learning Algorithm

This section proposes a CEQ based algorithm to coordinate the operation of agents. CEQ is a MARL algorithm. The coordination between agents is achieved by exchanging the state-action value matrix, which means it is implemented in a decentralized manner. We assume that all agents are selfless and share true information with each other. This section will introduce the system state, actions, rewards, and the correlated equilibrium.

3.2.1 State and Actions

The system state is defined as:

$$s = \{t, SOC, \vec{t}\} \quad (3.14)$$

The actions of DSM and ESS agents are:

$$a^{DSM} = \{\mathcal{Y}, \vec{a}\} \quad (3.15)$$

$$a^{ESS} = \{\mathcal{X}, \varsigma_t\}, \quad (3.16)$$

where $\mathcal{Y}$ belongs to a set of two choices: buying electricity from ESS or main grid; $\mathcal{X}$ belongs to a set of two choices: selling power to the main grid or DSM.

The Q-value spaces of DSM and ESS agents are:

$$Q_{DSM} = \{a^{ESS}, a^{DSM}, t, SOC, \vec{t}\} \quad (3.17)$$

$$Q_{ESS} = \{a^{ESS}, a^{DSM}, t, SOC, \vec{t}\} \quad (3.18)$$

3.2.2 Reward

The cost of ESS agent includes buying power from the PV or main grid, and the profit is comprised of selling power to the main grid or DSM. Meanwhile, DSM agent aims to minimize the cost by adjusting the operation time of deferrable devices and choosing a lower-priced energy source. The rewards of the following two cases are formulated corresponding to ESS charging or discharging.

1) At time t , if ESS agent chooses to charge, the instant reward is:

$$r_t^{ESS} = P_t^{ESS}(\mathcal{P}^{PV} p_t^{PV} + (1 - \mathcal{P}^{PV}) p_t^{bgrid}) \quad (3.19)$$

where $\mathcal{P}^{PV}$ denotes the proportion of buying power from PV ($0 \leq \mathcal{P}^{PV} \leq 1$).

ESS agent prefers PV power because of a lower price, but surplus PV power may not be enough for ESS charging. As a liner programming problem, we give the following equation of $\mathcal{P}^{PV}$:

$$\mathcal{P}^{PV} = \frac{P_t^{PV} - P_t^{crl}}{|P_t^{ESS}|} \quad (3.20)$$

where P_t^{crl} is the crucial load.

Meanwhile, the DSM can only get power from the main grid. The reward of DSM agent at time t is:

$$r_t^{DSM} = -P_t^{DSM} p_t^{bgrid} \quad (3.21)$$

2) If ESS agent chooses to discharge, the reward of ESS agent is:

$$r_t^{ESS} = P_t^{ESS} (\mathcal{P}^{ESS} p_t^{mid} + (1 - \mathcal{P}^{ESS}) p_t^{sgrid}) \quad (3.22)$$

where $\mathcal{P}^{ESS}$ denotes the proportion of selling power to DSM ($0 \leq \mathcal{P}^{ESS} \leq 1$). According to equation (3.8), the ESS will try to sell power to DSM agent for a higher price. Note that DSM may be unable to buy all ESS power, and therefore we give the following equation:

$$\mathcal{P}^{ESS} = \frac{P_t^{DSM}}{P_t^{ESS}} \quad (3.23)$$

The reward of DSM agent is:

$$r_t^{DSM} = -P_t^{DSM} (\mathcal{P}^{DSM} p_t^{mid} + (1 - \mathcal{P}^{DSM}) p_t^{bgrid}) \quad (3.24)$$

Similarly, denoting the proportion of buying power from ESS by:

$$\mathcal{P}^{DSM} = \min\left(\frac{P_t^{ESS}}{P_t^{DSM}}, 1\right) \quad (3.25)$$

3.2.3 Correlated Equilibrium

The aims of ESS and DSM agents are to maximize their rewards in a simulation epoch. For the agent λ , with the initial state s_0 , the expected accumulated discounted reward under a policy π is:

$$V_\lambda^\pi(s) = \mathbb{E}_\pi \left[\sum_{n=0}^{\infty} \gamma^n r_\lambda(s_n, a_n) \mid s = s_0 \right] \quad (3.26)$$

where γ is the reward discount factor.

For a state s and action a , we define the state-action value:

$$Q_\lambda^\pi(s, a) = r_\lambda(s, a) + \gamma \sum_{s' \in S} P(s' \mid s, a) V_\lambda^\pi(s) \quad (3.27)$$

Then we have the following relationship:

$$V_\lambda^\pi(s) = \sum_{a \in A} \pi_x(a) Q_\lambda(s, a) \quad (3.28)$$

Algorithm 3.1 Correlated Q-learning

```
1: Initialize: microgrid and Q-learning parameters
2: for  $j = 1$  to episode do
3:   Reset state  $s$ 
4:   for  $t = 1$  to  $T$  do
5:     Calculate eligible  $a_t^{DSM}, a_t^{ESS}$  according to state  $s$ 
6:     if  $rand < \epsilon$  then
7:       Randomly choose  $a_t^{DSM}, a_t^{ESS}$ 
8:     else
9:       Agents exchange current state-action matrix
10:      Find equilibrium by equation (3.29) and get optimal joint action  $a_t^{DSM}, a_t^{ESS}$ 
11:    end if
12:     $[r_t^{DSM}, r_t^{ESS}] = \text{Reward}(s, a_t^{DSM}, a_t^{ESS})$ 
13:    Update  $Q, t, SOC$  and  $\vec{w}$ 
14:  end for
15: end for
16: Output: Optimal action sequence from  $t = 1$  to  $T$ 
```

In CEQ algorithm, agents choose an optimal joint action by exchanging the Q-value matrix, and we assume both agents know possible actions of each other. The joint actions of agents are chosen according to correlated equilibrium:

$$\sum_{a_{-\lambda} \in A_{-\lambda}} \pi_x(a_{-\lambda}, a_\lambda) Q_\lambda(s, a_{-\lambda}, a_\lambda) \geq \sum_{a_{-\lambda} \in A_{-\lambda}} \pi_x(a_{-\lambda}, a_\lambda) Q_\lambda(s, a_{-\lambda}, a'_\lambda) \quad (3.29)$$

where the a_λ and a'_λ denote the action of agent i in correlated equilibrium and non-equilibrium, $a_{-\lambda}$ denotes the actions of agents except agent λ , $A_{-\lambda}$ denotes the action set of agents except λ . Meanwhile, equation (3.29) denotes that the joint-optimal action is chosen by exchanging Q-values, which means the private information of each agent could be well protected.

The policy is improved according to ϵ -greedy policy:

$$\pi_\lambda(s) = \begin{cases} random & rand \leq \epsilon \\ equation(3.29) & rand > \epsilon \end{cases} \quad (3.30)$$

Where ϵ is a small value between 0 and 1; $rand$ is a random number between 0 and 1.

It is worth noting that ESS will only choose eligible actions, e.g., the charge action is eliminated if $SOC=1$. The CEQ algorithm is summarized in Algorithm 3.1.

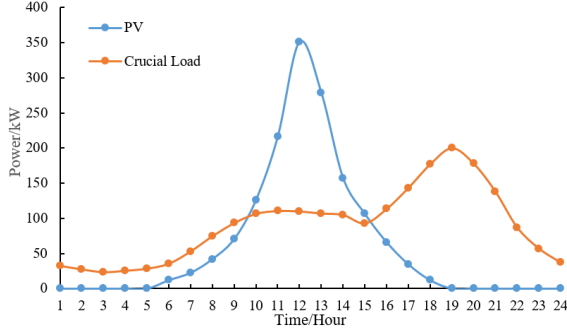


Figure 3.2: Daily PV power and crucial load settings

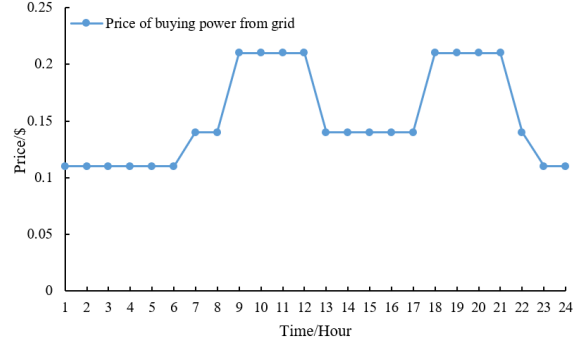


Figure 3.3: Daily Time-Of-Use electricity price

Table 3.2: Parameter settings of DSM loads with time limit and duration time

Device Number	Operation time limit (Hours)	Average duration time (Hours)
1	[1, 8]	1
2	[7, 13]	1
3	[10, 17]	2
4	[15, 21]	1
5	[20, 5 ^{+24h}]	3

Table 3.3: Parameters settings of MG with ESS capacity and electricity price

Parameters	value
Capacity of ESS: C^{ESS}	120 kWh
ESS charge power: P_{char}	30 kW
Price of PV selling electricity to ESS: p^{PV}	0.03 $\$/kWh$
Price of selling electricity to grid: p^{sgrid}	0.08 $\$/kWh$

3.3 Simulation Result

3.3.1 Parameter Settings

We use MATLAB for our simulations. The PV power and crucial loads are shown in Fig.3.2, which are extracted from [38]. Fig.3.3 shows the Time-of-Use (TOU) electricity prices

according to the winter TOU price of Ontario, Canada. Table 3.2 shows the operation time limits and the operation duration of five sets of deferrable devices [136]. For each device, the power (kW) is an integer number generated randomly from the set [8, 14], representing the average power consumed throughout the operation. We repeat the simulations for 30 runs and present the averaged values with 95% confidence intervals in plots. Table 3.3 shows other parameter settings.

3.3.2 Energy Management with Correlated Q-learning

First, we set the initial SOC of ESS to 0.25, and correlated Q-learning is used to optimize the cost of ESS agent and the profit of DSM agent. The results of average DSM cost and ESS profit are shown in Fig.3.4, where the cost decreases and profit increases through the iterations of the algorithm. It is observed that both agents converge.

Fig.3.5 presents the daily schedule of the optimal MG energy management. Positive bars represent load, charging or selling electricity to the main grid, while negative bars represent discharging or surplus PV power. The negative and positive bars have the same total height in each hour, meaning energy balance is well maintained. For example, from 1:00 to 2:00, ESS agent discharges with a power of 15kW, and DSM agent uses 10kW for operation, while the rest of 5kW is sold to the main grid. Due to a lower charging price, ESS agent uses surplus PV power to charge from 11:00 to 15:00. Meanwhile, DSM agent uses ESS power for operation, and both agents benefit from cooperation.

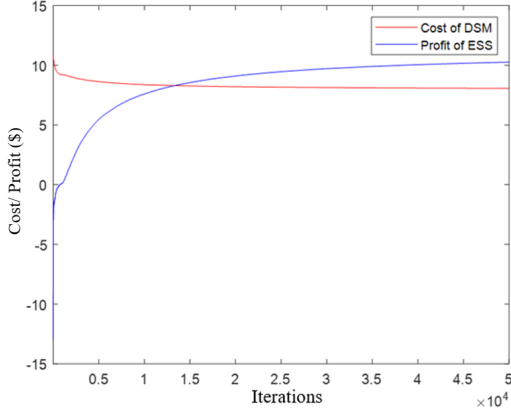


Figure 3.4: Average DSM cost and ESS profit with iterations.

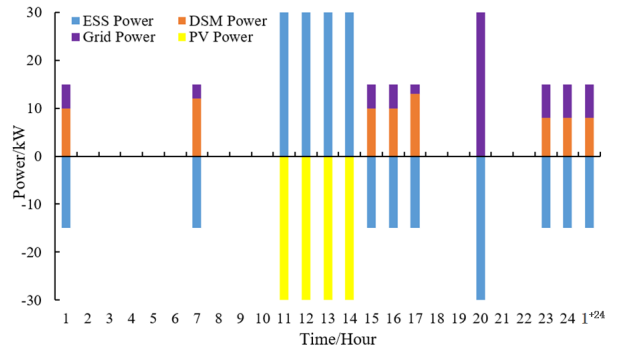


Figure 3.5: Optimal microgrid energy management policy

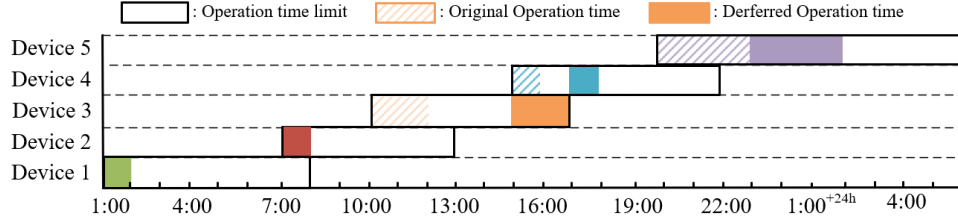


Figure 3.6: Deferred operation time of each device in 24 hours

Fig.3.6 shows how devices are deferred. Devices 1 and 2 start immediately because they have already worked under the minimum TOU price in their time limit. Devices 3 and 4 are deferred 5 and 2 hours later for a lower TOU price, respectively. Although the TOU price is lower from 13:00 to 18:00, note that device 3 starts operation at 15:00, because the ESS agent uses surplus PV power for charging from 11:00 to 15:00, and the ESS agent is unable to discharge in this period. Device 5 is deferred to 1:00 of the next day.

3.3.3 Comparison under Different PV Power

In the following two sections, we compare three cases:

Case I: CEQ is used to coordinate two agents.

Case II: Q-learning without correlation (QLWC) is implemented, where each agent maximizes their rewards and ignores others.

Case III: Q-learning with aggregator (QLWA) is conducted in this case, where all agents belong to one aggregator and maximize the total revenue. Considering that centralized optimization can usually get the global optimal result, we compare CEQ with QLWA to show that the distributed nature of CEQ does not harm total revenue.

The initial SOC of ESS is set to 0.25, and two cases are compared under different peak PV power. As shown in Fig.3.7 and Fig.3.8, the CEQ outperforms QLWC with a lower DSM cost and higher ESS profit, respectively. The 100% Peak PV power is defined as Fig.3.2. At 100% PV power, CEQ saves 16.5% cost for the DSM agent and earns 25.5% more profit for the ESS agent. At 80% PV power, the proportion for the DSM and ESS agents become 16.3% and 30.4%, respectively. CEQ earns at most 44.2% more profit than QLWC when peak PV power is only 20%. The main reason is the CEQ can coordinate the operation of the DSM and ESS agents. ESS uses PV power to charge and sell the energy to DSM. On the contrary, in QLWC, both agents try to maximize their rewards, and cooperation breaks. Instead of buying energy from the ESS agent, the DSM agent

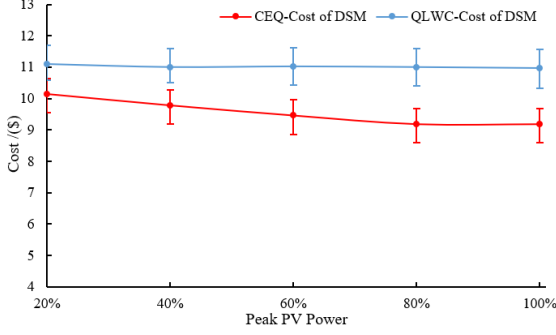


Figure 3.7: Comparison of DSM cost under different PV power

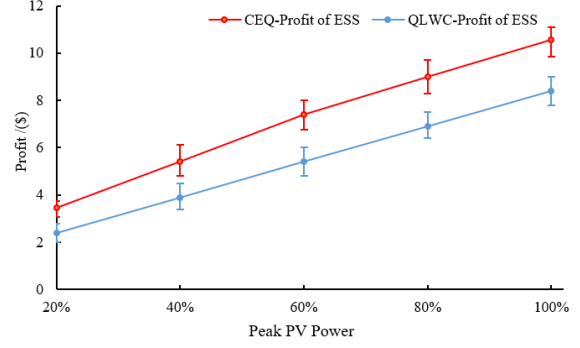


Figure 3.8: Comparison of ESS profit under different PV power

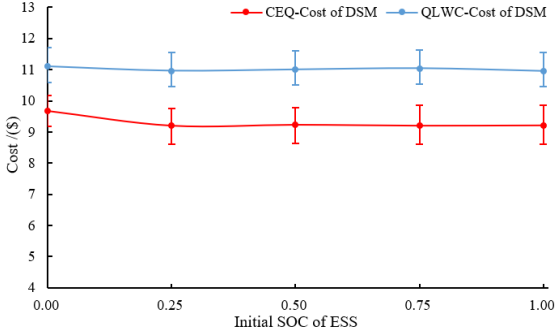


Figure 3.9: Comparison of DSM cost under different initial SOC

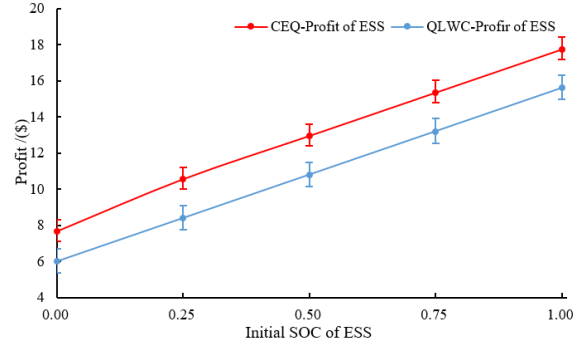


Figure 3.10: Comparison of ESS profit under different initial SOC

buys electricity from the main grid. Meanwhile, the ESS agent uses PV power to charge and sells the electricity to the main grid.

3.3.4 Comparison under Different Initial SOC of ESS

In this section, we set the PV power to 100% and compare three cases under different initial SOC of ESS. Note that the initial SOC cost is not considered.

As shown in Fig. 3.9 and Fig. 3.10, the CEQ has a lower DSM cost and higher ESS profit than QLWC. The DSM agent uses ESS energy for operation, and both agents benefit. At 0.75 initial SOC, the DSM agent in CEQ could save 19.3% cost, and ESS agent in CEQ could earn 16.3% more profit than QLWC. After the DSM agent receives enough energy

Table 3.4: Comparison of CEQ and QLWC under different initial SOC (\$)

Initial SOC		0.25	0.5	0.75	1.00
CEQ	Profit	10.55	12.95	15.35	17.75
	Cost	9.20	9.23	9.20	9.21
QLWA-Revenue		1.35	3.72	6.15	8.54

from ESS, the ESS will sell surplus energy to the main grid for profit. As a result, the DSM cost remains unchanged, but ESS profit still increases.

Furthermore, at 100% PV power, we compare the performance of CEQ with QLWA in Table 3.4. If we use the profit of CEQ minus the cost, we will get the same total revenue with QLWA. The result proves CEQ did not harm the total profit of the whole MG.

Chapter 4

Multi-agent Bayesian Deep Reinforcement Learning for Microgrid Energy Management under Communication Failures

This Chapter proposed a multi-agent BA-DRL algorithm for the MG energy management. The proposed algorithm is designed to be robust under both power supply uncertainty and communication failures uncertainty between IoT-enabled loads, storage and generation in MGs. We assume each agent employs BA-DRL independently and shares Q-values for collaboration, but the shared Q-values may be lost due to failures in the underlying communication system. These failures can be system failures, packet loss, or delayed/obsolete packets. To address this problem, we include a novel Bayesian belief update method, which estimates the actions of other agents by updating beliefs. Considering the large state-action space, a double deep Q-learning architecture is applied for Q-values estimation, and a novel belief-based correlated equilibrium is proposed for the collaborative action selection of agents. The BA-DRL is compared with both model-free Nash deep Q-learning (Nash-DQN) and model-based ADMM methods, and the results show that BA-DRL maintains better performance than baseline algorithms.

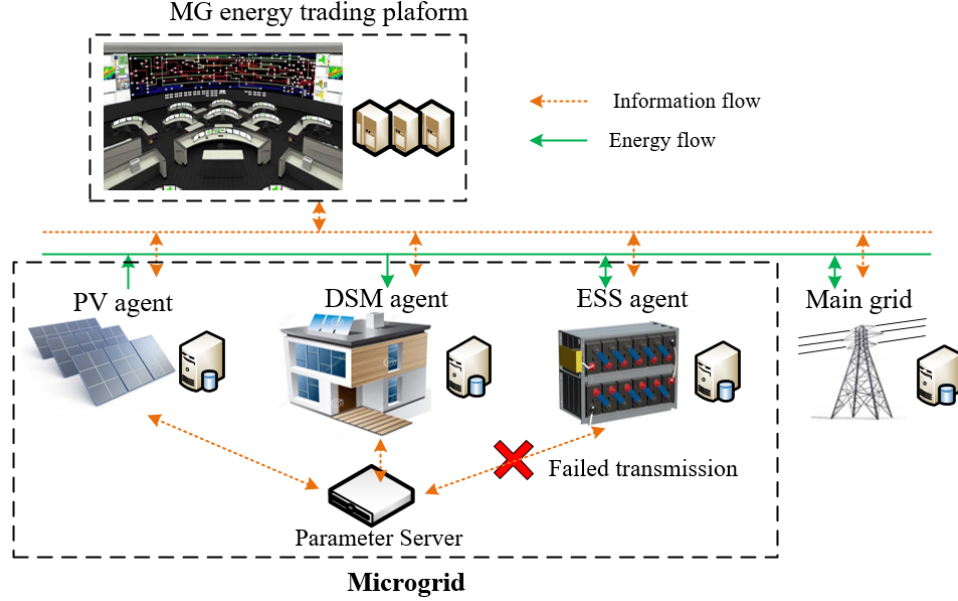


Figure 4.1: Microgrid system architecture with PV, DSM and ESS agents.

4.1 MG System Architecture and Energy Model

4.1.1 System Architecture

The MG system architecture is shown in Fig.4.1. We consider three types of agents, namely PV, DSM, and ESS agents. A PV agent may control multiple solar panels, ESS agent can control multiple storage devices, and DSM agent controls all controllable loads [137]. Agents exchange information with the energy trading platform to sell or buy energy, maximizing the profit. Meanwhile, to maintain the overall profit of the whole MG, we assume agents can exchange collaboration information with each other by a parameter server. In each time slot, the parameter server forwards the message of one agent to all other agents by wireless networks because of its low cost and high flexibility. This way, all agents share the collaboration information, which will be used for the decision-making of each agent [138]. However, communication failures may occur during information exchange, especially over the wireless networks, and some agents may be temporarily isolated [139, 140]. Due to the missing collaboration information, the isolated agent is very likely to take arbitrary actions, and the profit of other agents may be harmed. As a result, it is critical to consider the communication failures between agents in MG energy management.

4.1.2 Microgrid Agents Model

The DSM agent controls the demand of multiple deferrable devices such as water heaters or dishwashers. The devices connect to the DSM controller via IoT technologies as in [141, 142]. These devices can change their operation time to reduce costs, and the comfort level of end users will not be affected. We assume an average power consumption, and the total power demand of the DSM agent with D sets of devices is:

$$P_t^{dsm} = \sum_{g=1}^{N^{de}} P_g \varrho_{t,g} \quad (4.1)$$

where N^{de} is the total number of DSM devices sets, P_g is the average power consumption of devices in set g , and $\varrho_{t,g}$ is a binary value to represent the on/off status. Considering the end users' comfort level, these devices must be serviced before a certain time limit. The waiting time for devices is described as:

$$\vec{\iota}_t = [\iota_{t,1}, \iota_{t,2}, \dots, \iota_{t,g}, \dots, \iota_{t,N^{de}}], \quad (4.2)$$

where $\iota_{t,g}$ is the waiting time of device g at time t . Here we assume the device g should be serviced between the operation time limit $[t_{start}, t_{end}]$, which means $\iota_{t,g} \leq t_{end} - t_{start}$, and $\iota_{t,g}$ can be easily calculated by $\iota_{t,g} = t - t_{start}$, in which t is current time slot. We set $\iota_{t,g}$ to 0 when $t < t_{start}$ or $t > t_{end}$. Meanwhile, crucial devices, e.g., lighting and cooking devices, which cannot change their operation time, are not involved in this work since the agent has no control over them.

For the PV agent, we assume the PV power can be predicted with an error term, which describes the uncertainty of the PV power and the possible prediction inaccuracy [135].

The power of ESS agent is:

$$P_t^{ess} = P^{char} \varsigma_t \quad (4.3)$$

where P^{char} is the fixed charging power, and ς_t equals 1, -1 and 0 when discharging, charging, and unchanged, respectively. Note that we assume a centralized MG-level ESS, instead of one ESS for each house in MG [49].

The SOC of ESS is updated according to:

$$SOC_{t+1} = SOC_t - \frac{P^{char}}{C^{ess}} \varsigma_t \quad (4.4)$$

where C^{ess} is the ESS capacity.

4.1.3 Bidding based Microgrid Energy Trading

This section introduces the MG energy trading model, where power suppliers and consumers are involved [143]. In the proposed MG, PV agent is a power supplier, and DSM agent is a power consumer, while ESS agent can be a supplier when discharging or consumer when charging. The main grid can also participate in energy trading as a supplier or consumer for the energy balance of MG. Firstly, every supplier submits its power supply capacity P^{sup} and bidding price p^{sup} . All consumers submit their power demand P^{dem} , and the total power demand is calculated accordingly. Next, as shown in Fig.4.2, the suppliers will be ranked from lowest to highest bidding price. Then, the bidding price at the intersection of total power demand and total supply is determined as the clearing price for this market, which indicates that the total power supply is equal to power demand with this price.

Moreover, if $\sum P^{sup} < \sum P^{dem}$, the main grid participates in the market as a supplier, and the bidding price p^{grid} , which is offered by the main grid, will be the clearing price. On the contrary, if $\sum P^{sup} > \sum P^{dem}$, some suppliers need to sell the surplus electricity to the main grid, which is normally at a lower price. The bidding price higher than the p^{grid} will be unacceptable, because consumers always desire a lower energy price. Similarly, the suppliers are unlikely to offer a lower price than selling energy directly to the main grid. In this market, suppliers with a lower bidding price are more likely to be accepted, which will benefit the MG consumers [144]. It is worth noting that the bidding information is private for each agent, and no agent can manipulate the market.

4.1.4 Problem Formulation

Considering the DSM agent is always a consumer, it needs to minimize its cost:

$$\min(\sum_{t=1}^{T^{op}} \sum_{g=1}^D P_g q_{t,g} p_t^c) \quad (4.5)$$

where T^{op} is the optimization horizon, p_t^c is the clearing price.

The PV agent, in turn, aims to maximize the utility:

$$\max(\sum_{t=1}^{T^{op}} (P_t^{pv} p_t^c - (\beta(P_t^{pv})^2 + \zeta P_t^{pv} + \Phi))) \quad (4.6)$$

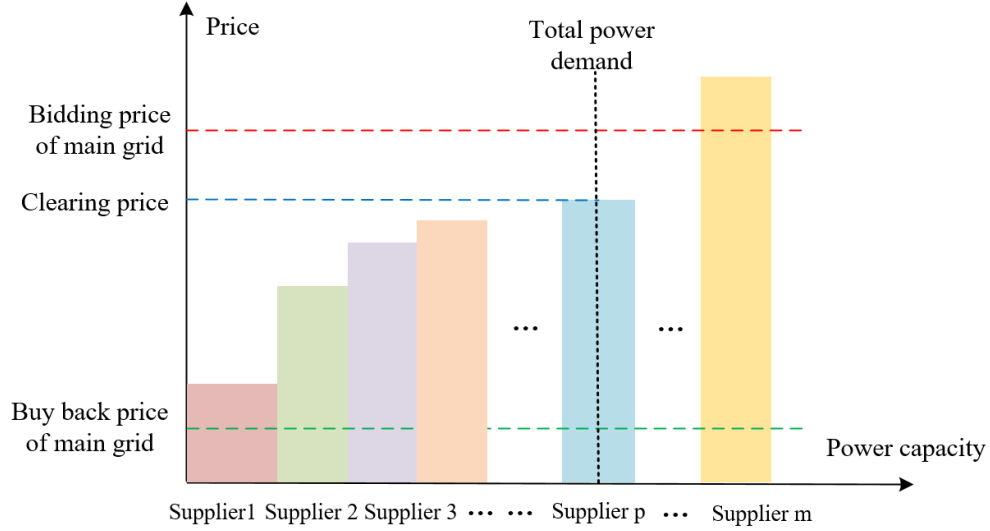


Figure 4.2: Bidding-based microgrid energy trading.

where P_t^{pv} is the PV power. The widely used quadratic function is adopted to present the generation cost of PV, and β, ζ, Φ are cost coefficients.

Finally, the ESS agent maximizes its utility using:

$$\max \left(\sum_{t=1}^{Top} P_t^{ess} p_t^c \right) \quad (4.7)$$

Equation (4.5), (4.6) and (4.7) define the objectives of DSM, PV and ESS agents, respectively. The DSM agent controls the on/off status of its devices to minimize its energy cost as a consumer. PV agent intends to maximize its profit by submitting an appropriate bidding price as an energy supplier. ESS agent buys energy when the energy price is lower and sells energy when the price is higher to maximize the total profit. Equations (4.5) to (4.7) show that each agent can make decisions autonomously to maximize its profit or reduce the cost, then all three objectives are simultaneously improved. Meanwhile, the problem is optimized under the following constraints:

$$P_t^{pv} + P_t^{grid} + P_t^{ess} = P_t^{dsm} \quad (4.8)$$

$$\varrho_{t,g} \leq \iota_{t,g} \quad (4.9)$$

$$\iota_{t,g} \leq \iota_g^{max} \quad (4.10)$$

$$SOC_{min} \leq SOC_t \leq SOC_{max} \quad (4.11)$$

Equation (4.8) is the energy balance constraint; (4.9) is the DSM constraint, which means only devices that have not been serviced can be turned on; (4.10) means the waiting time cannot exceed the maximum limit; (4.11) is the SOC lower and upper bound constraint.

4.2 Multi-agent POMDP

This section defines the MA-POMDP and introduces a Bayesian belief update method. In MA-POMDP, each agent maintains beliefs on other agents. When some agents are isolated by communication failures, other agents can make reasonable estimations about the actions of isolated agents.

4.2.1 MA-POMDP Definition

As shown in Fig.4.3(a), in the conventional MDP, an agent takes actions a^t based on the current state s^t , then receives a reward r^t and arrives to a new state s^{t+1} . However, in some cases, the agent cannot directly observe the underlying state s^t , which is common in practice. partially observable Markov decision process (POMDP) is proposed to describe cases in which the state is partially observable [145]. The POMDP is represented with $\langle S, A, T, R, \Omega, O, b \rangle$, in which S , A , T and R represent the set of states, set of actions, transition probability, and reward function, respectively. Ω is the set of observations o , O is the observation function, and $b(s^t)$ is the belief of state s^t . In POMDP, an agent is not sure about its state s^t , but the observation o^t is always available when it comes to a new state. The observation function $O(s^t|o^t)$ maps the observation result o^t to state s^t . Fig.4.3(b) shows the architecture of POMDP. It is worth noting that agents can only choose actions by observations o^t in POMDP, instead of s^t in MDP.

Although POMDP is defined for solving the state uncertainty, an agent in a multi-agent system is more likely to be uncertain about the actions of other agents instead of its state. These unknown actions will directly affect the system environment. The proposed MA-POMDP is shown in Fig.4.3(c), where each agent has its state, action, reward, observation and belief. Based on the current states and beliefs, the agent selects its actions and gets the corresponding reward; then, it moves to the next state. Meanwhile, each agent will make observations about the actions of its peers, which is represented by the dashed lines from the action of other agents to observations of one specific agent.

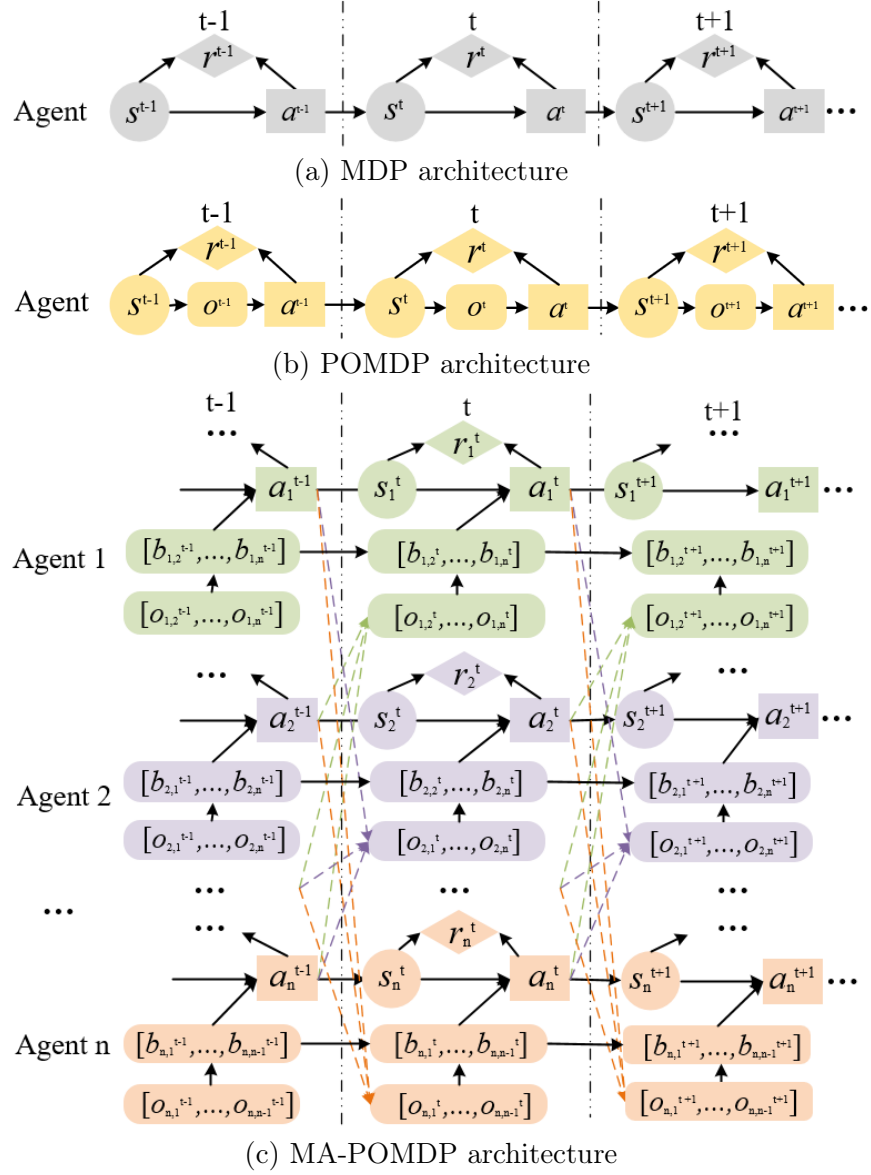


Figure 4.3: Definitions of MDP, POMDP, and MA-POMDP

For agent λ , the observation set is denoted as $\vec{o}_{-\lambda}^t = (o_{\lambda,1}^t, o_{\lambda,2}^t, \dots, o_{\lambda,\lambda-1}^t, o_{\lambda,\lambda+1}^t, \dots, o_{\lambda,n}^t)$. Here, $o_{\lambda,\varphi}^t$ means the observation made by the agent λ for the action of agent φ at time t ($\varphi \neq \lambda$). Indeed, one agent is unlikely to fully share its action selection with other agents, especially under communication failure. However, the observations do not rely on

actual communications, and they are generated according to the accumulated experience of agents, which means no direct communication is needed. Given this characteristic of the observation function, we can utilize the MA-POMDP as a solution for communication failure between agents. Specifically, we assume that each intelligent agent can maintain a belief set $\vec{b}_{-\lambda}^t = (b_{\lambda,1}^t, b_{\lambda,2}^t, \dots, b_{\lambda,\lambda-1}^t, b_{\lambda,\lambda+1}^t, \dots, b_{\lambda,n}^t)$ about actions of other $n - 1$ agents. $b_{\lambda,\varphi}^t$ means the belief of agent λ for the action of agent φ at time t ($\varphi \neq \lambda$). Based on observations at time t , the belief can be updated from t to $t + 1$, as described in the following Section 4.2.2. Noting that the proposed MA-POMDP is different from the existing decentralized POMDP. In decentralized POMDP, all agents work as a team to maximize the global reward, but each agent has its reward in our proposed MA-POMDP.

Based on the proposed MA-POMDP scheme, we define the state, action, reward, and observation function of MG agents:

- For the DSM agent, the state is defined by current time and device waiting time $\{t, \vec{W}\}$, and the action is the on/off status of DSM devices. The reward function is given by the objective of DSM agent to minimize energy cost, which is indicated by equation (4.5).
- For PV agent, the state is defined by current time and power $\{t, P_t^{PV}\}$, and the action is bidding price. The reward function is defined by equation (4.6) in the problem formulation to maximize its profit.
- For ESS agent, the state is defined by time and SOC $\{t, SOC\}$, and the action includes ESS power and bidding price. The reward function is defined as equation (4.7) to maximize its utility.

The proposed observation function is presented as Fig.4.4, which can be divided into two parts: observation generation and action estimation. Firstly, in the observation generation phase, when agent λ takes action a_1^t , an intermediate observation $\mathfrak{l}_{1,x}^t$ will be generated with a probability $p_{1,x}^t$ ($1 \leq x \leq |A_\lambda|$, $\sum_{x=1}^{|A_\lambda|} p_{1,x}^t = 1$), in which the $\mathfrak{l}_{1,x}^t$ means the x^{th} intermediate observation of action a_1^t (shown by the red dashed arrow in the left of Fig.4.4). Then the $\mathfrak{l}_{1,x}^t$ will be mapped to $o_{\varphi,\lambda,x}^t$, which is the final observation results received by agent φ . Secondly, in the action estimation phase, the observation results received by agent φ will be mapped to the estimated actions of agent λ . As shown by Fig.4.4, the relationship between observation and estimation results is defined by $o_{\varphi,\lambda,x}^t \rightarrow a_x^t$, which means receiving $o_{\varphi,\lambda,x}^t$ will lead to estimated action a_x^t . However, $o_{1,\lambda,x}^t$ can also be mapped by other intermediate observations like $p_{2,x}^t$ or $p_{|A_\lambda|,x}^t$. It denotes that the real action of agent λ is never known by agent φ , and observation error is inevitable. For example,

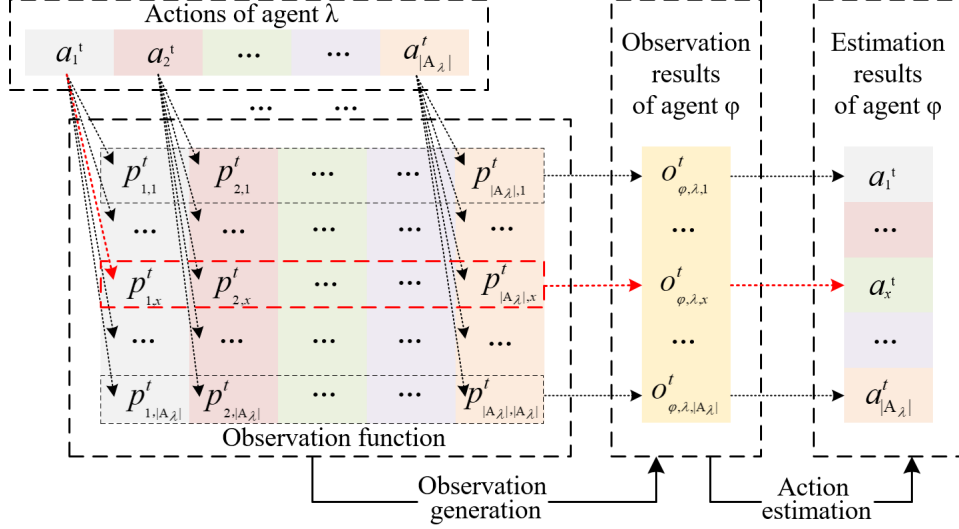


Figure 4.4: Observation function definition with partial observability.

when DSM agent takes an action a_{dsm}^t , PV and ESS agents will receive observations for the on/off status of DSM agent, which is an estimated action of DSM agent. Although the real action a_{dsm}^t is never known by PV and ESS agents, the long-term observation can still reflect the action selection routine of DSM agent, which can be utilized to update the beliefs. Note that the observation functions can be defined in various ways, and here we define our function according to the MG environment.

4.2.2 Bayesian Belief Update Method

This section introduces how to use observations to update beliefs by the Bayesian rule. Let the agent with communication failures be called as problematic agent (PA), and other agents are normal agents (NAs). Note that the proposed method can be applied for the multiple PA case without loss of generality. Here PA is isolated due to failed transmissions (which NAs can infer by a missing periodic update), and it will ignore other agents when choosing its actions. Meanwhile, considering all agents are independent, NAs cannot know the state and actions of the PA. However, NAs could make some observations about the actions of PA, and map the observation results to actions, which is a basic assumption in POMDP problems [146].

The historical experience from former $j-1$ episodes is used to form the beliefs of actions in j^{th} episode. Based on Bayesian rules, when o_i is observed, the posterior distribution of

action a_i is:

$$b_j(a_i|o_i) = \frac{O(o_i|a_i)b_j(a_i)}{\sum_{a_i \in A} O(o_i|a_i)b_j(a_i)} \quad (4.12)$$

where $b_j(a_i)$ is the prior belief of PA action a_i at episode j , and $\sum_{a_i \in A} O(o_i|a_i)b_j(a_i)$ is the normalizing constant. $b_j(a_i|o_i)$ represents the expected probability that PA takes action a_i when observing o_i . Note that $b_j(a_i|o_i)$ is the conditional probability based on observation o_i , which can not be used directly as a new prior belief for the next update. Then we need to use the $b_j(a_i|o_i)$ to update the $b_j(a_i)$ to $b_{j+1}(a_i)$, which can be used in the next episode.

Let A_P be the action set of PA. $\vec{C}_{P,j-1} = (c_{1,j-1}, c_{2,j-1}, \dots, c_{|A_P|,j-1})$ denotes that the action a_i is expected to be selected $c_{i,j-1}$ times in the past $j-1$ episode, and $\vec{b}_j(a) = (b_j(a_1), b_j(a_2), \dots, b_j(a_{|A_P|}))$ is the prior belief of PA actions. Considering $\sum_{a_i \in A_P} b_j(a_i) = 1$, the distribution of $b_j(a_i)$ obeys Dirichlet distribution $Dirt(\vec{b}_j(a)|\vec{C}_{P,j})$ [146].

$$Dirt(\vec{b}_j(a)|\vec{C}_{P,j}) = \frac{\tau(\sum_{i=1}^{|A_P|} c_{i,j})}{\prod_{i=1}^{|A_P|} \tau(c_{i,j})} \prod_{i=1}^{|A_P|} b_j(a_i)^{c_{i,j}-1} \quad (4.13)$$

where τ is the gamma function.

At episode j , a new observation o_j is received, which updates $\vec{C}_{P,j-1}$ to $\vec{C}_{P,j}$. Then the expected prior distribution of taking action a_i is calculated by:

$$\begin{aligned} \mathbb{E}(b_j(a_i)) &= \frac{\sum_{k=1}^j b_k(a_i|o_k)}{\sum_{k=1}^j \sum_{i=1}^{|A_P|} b_k(a_i|o_k)} \\ &= \frac{\sum_{k=1}^{j-1} b_k(a_i|o_k) + b_j(a_i|o_j)}{\sum_{k=1}^{j-1} \sum_{i=1}^{|A_P|} b_k(a_i|o_k) + \sum_{i=1}^{|A_P|} b_t(a_i|o_j)} \\ &= \frac{c_{i,j-1} + b_j(a_i|o_j)}{\sum_{i=1}^{|A_P|} (c_{i,j-1} + b_j(a_i|o_j))} \end{aligned} \quad (4.14)$$

Meanwhile, $Dirt(\vec{b}_{j-1}(a)|\vec{C}_{P,j-1})$ is the Dirichlet distribution, and the expected value

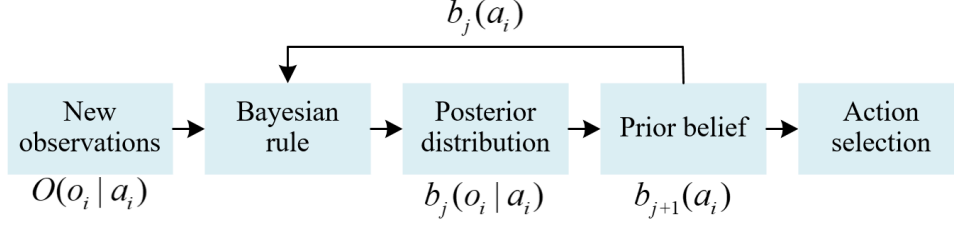


Figure 4.5: Sequential Bayesian belief update method.

of $b_{j-1}(a_i)$ is:

$$\begin{aligned}
\mathbb{E}(b_{j-1}(a_i)) &= \int b_{j-1}(a_i) \text{Dir}(\vec{b}_{j-1}(a) | \vec{C}_{P,j}) d(\vec{b}_{j-1}(a)) \\
&= \frac{\tau(\sum_{i=1}^{|A_P|} c_{i,j-1})}{\prod_{i=1}^{|A_P|} \tau(c_{i,j-1})} \int b_{j-1}(a_i)^{c_{i,j-1}+1-1} \prod_{k=1, k \neq i}^{|A_P|} b_{j-1}(a_k)^{c_{k,t-1}-1} d(\vec{b}_{j-1}(a)) \\
&= \frac{\tau(c_{i,j-1} + 1) \prod_{k=1, k \neq i}^{|A_P|} \tau(c_{i,j-1}) \tau(\sum_{i=1}^{|A_P|} c_{i,j-1})}{\tau(1 + \sum_{i=1}^{|A_P|} c_{i,j-1}) \prod_{i=1}^{|A_P|} \tau(c_{i,j-1})} \\
&= \frac{c_{i,j-1}}{\sum_{i=1}^{|A_P|} c_{i,j-1}}
\end{aligned} \tag{4.15}$$

In Dirichlet distribution, with the new experiment results Δb , the new expectation becomes:

$$\mathbb{E}(b_j(a_i)) = \frac{c_{i,j-1} + \Delta b}{\sum_{i=1}^{|A_P|} (c_{i,j-1} + \Delta b)} \tag{4.16}$$

It is obvious that $\Delta b = b_j(a_i|o_j)$ if we compare equation (4.16) with (4.14), which means we could use $b_j(a_i|o_j)$ to update the expectation of taking action a_i , and $Ex(b_j(a_i))$ is used as a new prior distribution of a_i in (4.16).

The proposed Bayesian belief update method can be summarized by Fig.4.5. Given observations $O(o_i|a_i)$ and prior belief $b_j(a_i)$, Bayesian rule is applied to calculate the posterior distribution $b_j(a_i|o_i)$. Then, we update the $b_{j+1}(a_i)$ by equation (13) to (16), which will be used in the next episode to calculate new posterior distribution $b_{j+1}(a_i|o_i)$. By repeating this cycle, the prior belief on the PA actions can be used in the action selection of BA-DRL.

4.3 Bayesian Deep Reinforcement Learning

Based on the defined MA-POMDP, we present the BA-DRL algorithm. In this section, we first introduce the overall architecture and network training method of BA-DRL, and then we propose a belief-based correlated equilibrium for the action selection under communication failure. Moreover, Nash-DQN and ADMM methods are introduced as baseline algorithms.

4.3.1 BA-DRL Architecture and Network Training

This work applies a DDQN framework for state-action value prediction. DDQN can better estimate the Q-values by decoupling the action selection and evaluation, and the loss function of DDQN is defined as:

$$Loss(w) = Er(r^t + \gamma Q(s^{t+1}, \arg \max_a Q(s^{t+1}, a, w), w') - Q(s^t, a^t, w)) \quad (4.17)$$

where α is the learning rate, γ is the discount factor, s is the state, a is the action. w and w' are the weight of main and target networks, respectively, and Er is the error function. The target network will evaluate the action, and the main network will select actions by $\arg \max_a Q(s^{t+1}, a, w)$. By decoupling the action selection and evaluation, the DDQN can provide more accurate Q-values than DQN.

With the DDQN architecture, the proposed multi-agent BA-DRL is shown in Fig.4.6, where each agent runs BA-DRL independently. For example, in ESS agent shown in yellow, after a_{ess}^t is sent to MG trading platform, a tuple $(s_{ess}^t, a_{ess}^t, r_{ess}^t, s_{ess}^{t+1})$ is received from MG trading environment, which will be stored in the experience pool. Then the network training part is shown at the bottom. ESS agent implements a random minibatch from the experience pool. For every data tuple $(s_{ess}^t, a_{ess}^t, r_{ess}^t, s_{ess}^{t+1})$, main network predicts $Q(s_{ess}^t, a_{ess}^t, w)$ and finds actions by $\arg \max_{a_{ess}} Q(s_{ess}^{t+1}, a_{ess}, w)$. Target network evaluates the action by $Q(s_{ess}^{t+1}, \arg \max_{a_{ess}} Q(s_{ess}^{t+1}, a_{ess}, w), w')$. Predicted values are utilized in loss function by gradient descent to update main network weight. After some iterations, the main network weight will be copied to the target network. This late update provides a stable reference for the main network, which helps BA-DRL to be robust to power supply uncertainty such as PV power.

Meanwhile, here we use the LSTM structure in target and main networks, which is a special RNNs. The hidden node of traditional RNNs only include a single activation function, but the hidden node in LSTM is a memory cell with forget, input, and output

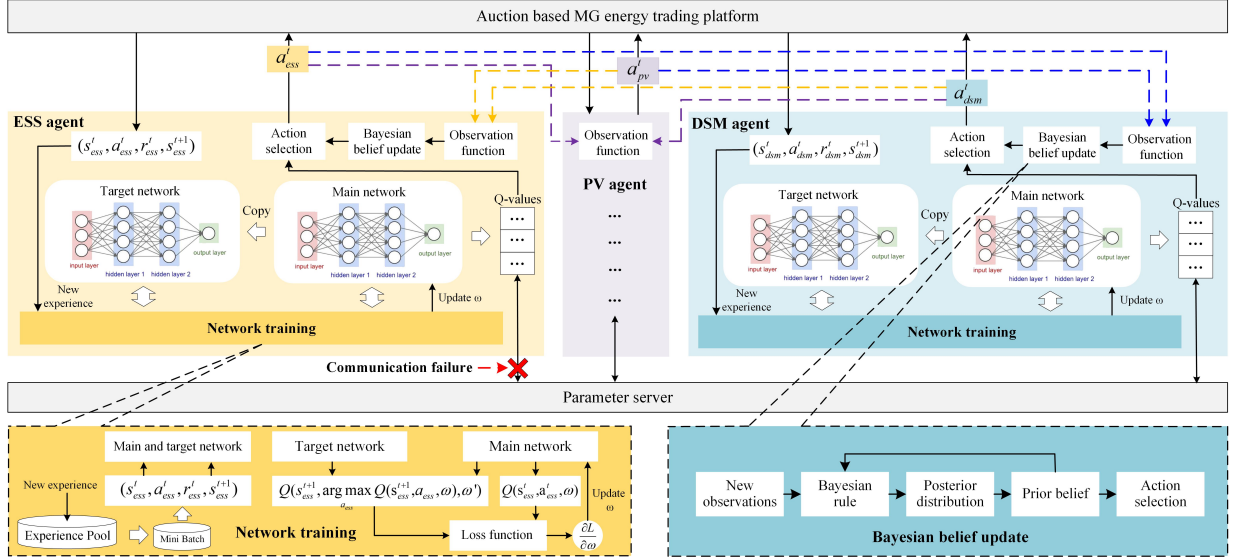


Figure 4.6: Multi-agent BA-DRL architecture for MG energy trading.

gates, and contents can be memorized, erased or exposed accordingly [147]. As such, LSTM can better capture the long-term data dependencies. Compared with existing DDQN frameworks that applied traditional deep neural networks, our BA-DRL uses LSTM to better learn the complicated long-term MG energy trading patterns.

Finally, in Fig.4.6, note that both beliefs and Q-values affect the action selection part. On the one hand, ESS agent will collect observations of a_{PV}^t and a_{DSM}^t , which is shown by the yellow dashed lines on the top. Given the observations as input, the Bayesian belief update is illustrated by the block at the bottom of Fig.4.6, and the updated prior beliefs will be finally used for action selection. On the other hand, the main network will predict Q-values of the current state, and agents exchange Q-values by the parameter server for collaboration. But the communication failure may occur (shown by the red cross at the bottom), and ESS agent (or another agent) can be temporarily isolated. However, with the Bayesian belief update method defined in Section 4.2.2, agents can use beliefs to select actions under communication failure, which will be introduced in the next section.

4.3.2 Belief-based Correlated Equilibrium

Action selection is a critical part of reinforcement learning algorithms. For one agent case, the ϵ -greedy policy is generally applied for action selection. However, multi-agent action

selection is much more complicated because the environment is affected by the actions of multiple agents simultaneously. With communication failures, some agents will lose collaboration information with the community, and it is hard to make an optimal joint-action for all agents. Although the collaboration information of isolated agents is missing, normal agents can still utilize their beliefs to estimate the possible actions of isolated agents and make the best response accordingly.

In this section, we propose a belief-based correlated equilibrium, which enables normal agents to use beliefs to make the best response to communication failure in a decentralized manner. Here we assume that all agents are selfless and they will share true information with each other. When the agent λ is isolated due to a communication failure, other agents choose the joint-action by:

$$\begin{aligned}
& \max_{a_\lambda \in A_\lambda} b(a_\lambda) \sum_{\vec{a}_{-\lambda} \in A_{-\lambda}} Pr(s, \vec{a}_{-\lambda}) Q(s, \vec{a}_{-\lambda}, a_\lambda) \\
& \text{sub.to } \sum_{\vec{a}_{-\lambda} \in A_{-\lambda}} Pr(s, \vec{a}_{-\lambda}) = 1 \\
& \sum_{a_{-\lambda, -k} \in A_{-\lambda, -k}} Pr(s, \vec{a}_{-\lambda}) (Q(s, \vec{a}_{-\lambda}) - Q(s, \vec{a}_{-\lambda, -k}, a_\lambda, a_k)) \geq 0 \\
& 0 \leq Pr(s, \vec{a}_{-\lambda}) \leq 1
\end{aligned} \tag{4.18}$$

where $b(a_\lambda)$ is the belief that agent λ will choose action a_λ . $\vec{a}_{-\lambda}$ means the action combination of other $n - 1$ agents except agent λ , $Pr(s, \vec{a})$ is the probability of taking joint-action $\vec{a} = (a_1, a_2, \dots, a_n)$ under state s , $A_{-\lambda}$ is the set of $\vec{a}_{-\lambda}$.

The objective of (4.18) is to maximize the total expected reward of NAs, and find an optimal probability distribution to enable each agent to choose an optimal action. We multiply the belief $b(a_\lambda)$ with the total expected reward of other $n - 1$ agents $\sum_{\vec{a}_{-\lambda} \in A_{-\lambda}} Pr(s, \vec{a}_{-\lambda}) Q(s, \vec{a}_{-\lambda}, a_\lambda)$ to represent the expected reward under this belief. Meanwhile, the constraints guarantee that action combination $\vec{a}_{-\lambda}$ brings more expected reward with probability distribution $Pr(s, \vec{a}_{-\lambda})$, which is indicated by $Pr(s, \vec{a}_{-\lambda}) (Q(s, \vec{a}_{-\lambda}) - Q(s, \vec{a}_{-\lambda, -k}, a_\lambda, a_k)) \geq 0$.

The BA-DRL is summarized in Algorithm 1. We apply the action selection constraints here to realize the optimization constraints in the learning process. For example, ESS agent cannot choose to discharge if $SOC = 0$.

4.3.3 Baseline Algorithms: Nash-DQN and ADMM

Algorithm 4.1 BA-DRL

```
1: Initialize: MG and BA-DRL parameters
2: for  $j = 1$  to  $episode$  do
3:   for  $t = 1$  to  $T^{op}$  do
4:     for all Agents do
5:       With probability  $\epsilon$ , choose actions randomly; otherwise, predict  $Q(s^t, a^t, w)$ ,
       and:
6:       if Communication Failure then
7:         PAs select actions by the greedy policy.
8:         NAs exchange Q-values and calculate belief-based correlated equilibrium by
         equation (4.18).
9:       else
10:        All agents exchange Q-values and find optimal joint-action by correlated equi-
        librium.
11:      end if
12:      Make observations of other agents' actions. Calculate the posterior distribution
      and update  $\vec{C}_{P,j}$  by equation (4.14).
13:    end for
14:    Agents take actions and update their state, and save  $(s^t, a^t, r^t, s^{t+1})$  to their expe-
    rience pool.
15:  end for
16:  Every  $C$  episodes, random sample a minibatch from the experience pool. Generate
  target Q-values  $Q^T(s^t, a^t) =$ 

$$\begin{cases} r^t & \text{if done} \\ r^t + \gamma Q(s^{t+1}, \arg \max_a Q(s^{t+1}, a, w), w') & \text{else} \end{cases}$$

17:  Update  $w$  using gradient descent by minimizing the loss  $Loss(w) = Er(Q^T(s^t, a^t) -$ 
 $Q(s^t, a^t, w))$ .
18:  Copy  $w$  to  $w'$  after several training.
19: end for
20: Output: Optimal action sequence from  $t = 1$  to  $T^{op}$ 
```

This work uses the Nash-DQN and ADMM as baseline algorithms. Nash equilibrium is generally applied for multi-agent coordination problems, and Nash-DQN is compared as a learning-based algorithm. Meanwhile, ADMM algorithm is a well-known model-based distributed optimization method, which can be applied for MG economic dispatch problems.

Algorithm 4.2 Nash-DQN

```
1: Initialize: MG and DQN parameters
2: for  $j = 1$  to episode do
3:   for  $t = 1$  to  $T^{op}$  do
4:     for all Agents do
5:       With probability  $\epsilon$ , choose actions randomly; otherwise, predict  $Q(s, a, w)$  and
       find its own Nash equilibrium by equation (4.19).
6:     end for
7:     Agents exchange equilibrium information and match the system equilibrium iter-
       atively. Neglecting the isolated agent if a communication failure occurs.
8:     Agents take actions and update its own state  $s$ , and save  $(s^t, a^t, r^t, s^{t+1})$  to their
       own experience pool.
9:   end for
10:  Every  $C$  episodes, random sample a minibatch from experience pool, the target
  Q-values  $Q^T(s^t, a^t)$  is:
11:

$$\begin{cases} r^t & \text{if done} \\ r^t + \gamma \max_a Q(s^{t+1}, a, w') & \text{else} \end{cases}$$

12:  Update  $w$  using gradient descent by minimizing the loss  $Loss(w) = Er(Q^T(s^t, a^t) -$ 
 $Q(s^t, a^t, w))$ .
13:  Copy  $w$  to  $w'$  after several trainings.
14: end for
15: Output: Optimal action sequence from  $t = 1$  to  $T^{op}$ 
```

Nash-DQN is a multi-agent DRL method, in which each agent runs a DQN independently, and Nash equilibrium is applied for the joint-action selection. In Nash equilibrium, no agent will change its action because it will result in a lower utility. The Nash equilibrium is described as:

$$U_\lambda(\vec{a}_{-\lambda}, a_\lambda) \geq U_\lambda(\vec{a}), \vec{a} \in A, \vec{a}_{-\lambda} \in A_{-\lambda}, \quad (4.19)$$

where U_λ is the utility function for agent λ , $\vec{a}$ is the joint-action of all agents, $\vec{a}_{-\lambda}$ is the joint-action of other agents except agent λ , A is the action set of $\vec{a}$, and $A_{-\lambda}$ is the action set of $\vec{a}_{-\lambda}$. Equation (4.19) indicates that agent λ will not change its action a_λ because it will lead to a lower utility U_λ for itself. Furthermore, the multi-agent system will reach Nash equilibrium when all agents get their equilibrium. It is worth noting that all agents need to exchange information iteratively to find a system-level Nash equilibrium, which increases the risk of communication failure. Here we assume other agents will neglect the

isolated agents under communication failure. Nash-DQN is given in Algorithm 2.

On the other hand, here we compare the ADMM with BA-DRL to show the superiority of the learning-based method. In the ADMM framework, the MG system is described as:

$$\begin{aligned} \min \quad & cost^{dsm}(P_t^{dsm}) - profit^{pv}(p_t^{pv}) - profit^{ess}(P_t^{ess}, p_t^{ess}) \\ \text{sub.to} \quad & (4.8) - (4.11) \end{aligned} \tag{4.20}$$

where P_t^{dsm} and P_t^{ess} are power of DSM and ESS agents, respectively. p_t^{pv} and p_t^{ess} are the bidding prices of PV and ESS agent at time t . The convexity of equation (4.20) can be easily proven as a combination of quadratic and linear problems. In ADMM, the problem will be first transformed to an augmented Lagrangian form. Then each sub-objective is updated sequentially, and the iteration will stop when the value converges. Noting that agents must frequently exchange dual variables in ADMM. We assume ADMM is applied in each time slot until convergence, then the next time slot comes.

4.4 Performance Evaluation

4.4.1 Simulation Settings

We assume there are PV, ESS and DSM agents in a MG. The predicted PV power is generated according to, and the prediction error obeys a Gaussian distribution. The bidding price is discretized as $[0.05, 0.09, 0.13, 0.17, 0.21, 0.25]$ \$/kW·h. MG and BA-DRL parameters are shown in Table 4.1. There are 5 sets of deferrable devices, where the operation time limit and average duration time are shown in Table 4.2. Here, 4^{+24h} means 4 am of the next day.

We apply the well-known experience replay technique for network training. In particular, the former experience $\langle \text{state}, \text{action}, \text{reward}, \text{next state} \rangle$ will be saved in the experience pool, considered the generated data for network training. After every 40 episodes, a random minibatch is implemented in the experience pool, and the batch data will be used for main network training. The neural network contains 4 layers, including one input layer, one output layer, and two LSTM hidden layers with 35 nodes. Hyperparameter tuning is a well-known issue for network training, and here we set our values using the grid search method by trying different parameter combinations. The simulation is repeated for 10 runs in MATLAB platform to get averaged values with 95% confidence interval.

Table 4.1: Parameter settings of MG and BA-DRL algorithm.

MG Parameters		BA-DRL Parameters	
PV capacity	30kW	Network layers	4
PV prediction error	$N(0, 0.1P_t^{pv})$	Training frequency	Every 40 episodes
ESS charging power	20kW	Hidden layers	2 LSTM (35 nodes)
ESS capacity	120kW·h	Batch size	120
ESS initial SOC	0.4	Experience Pool	1200
SOC limit	[0,1]	Learning rate	0.005
Grid feed-in price	40%	Discount factor	0.6

Table 4.2: Properties of deferrable devices with power consumption and duration time

Device Number	Average Power(kW)	Operation time limit (Hours)	Average duration time (Hours)
1	6	[1, 8]	2
2	15	[7, 13]	1
3	19	[10, 17]	1
4	10	[15, 22]	3
5	7	[20, 4 ^{+24h}]	1

4.4.2 Microgrid Energy Trading with Communication Failure

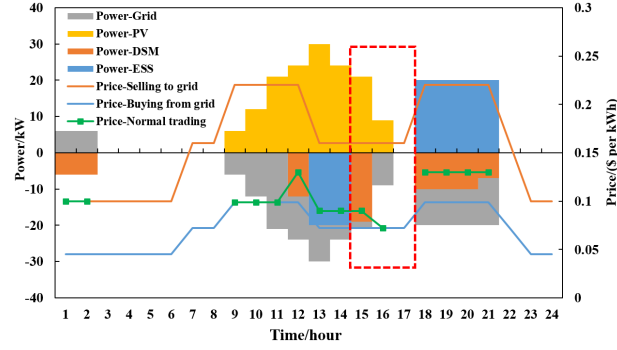
This section explains how MG energy trading and pricing work under BA-DRL. Fig.4.7(a) shows the performance under perfect communication. The ESS agent uses PV power to charge in daylight and sells the electricity to DSM devices at night, which will benefit the whole MG. In Fig.4.7(b), we assume the ESS agent suffers a communication failure at time 15. Then ESS agent will neglect other agents and make its own decision. Compared with Fig.4.7(a), the main difference is that ESS charges at time 15. Considering that the PV power is not enough for DSM devices and ESS charging at the same time, the main grid participates in the market. Then the uniform clearing price is dominated by the main grid, where the new price 0.16 \$/kW·h is much higher than the original clearing price 0.072 \$/kW·h. As a result, both ESS and DSM agents have to buy electricity at a higher price.

Table 4.3: Observation function parameters with 8 possible actions and observations

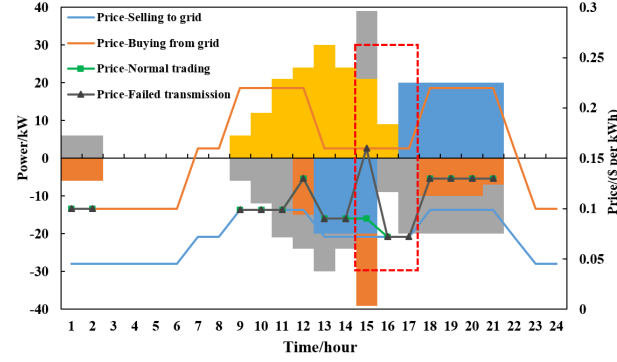
	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8
a_1	0.6	0.0571	0.0571	0.0571	0.0571	0.0571	0.0571	0.0571
a_2	0.0571	0.6	0.0571	0.0571	0.0571	0.0571	0.0571	0.0571
a_3	0.0571	0.0571	0.6	0.0571	0.0571	0.0571	0.0571	0.0571
a_4	0.0571	0.0571	0.0571	0.6	0.0571	0.0571	0.0571	0.0571
a_5	0.0571	0.0571	0.0571	0.0571	0.6	0.0571	0.0571	0.0571
a_6	0.0571	0.0571	0.0571	0.0571	0.0571	0.6	0.0571	0.0571
a_7	0.0571	0.0571	0.0571	0.0571	0.0571	0.0571	0.6	0.0571
a_8	0.0571	0.0571	0.0571	0.0571	0.0571	0.0571	0.0571	0.6

It means that other agents are affected by the communication failure of ESS agent, and the overall profit is harmed. The failed transmission does not only change the current action, but also affects the following actions. For example, compared with Fig.4.7(a), ESS also makes a different decision at time 17 in Fig.4.7(b).

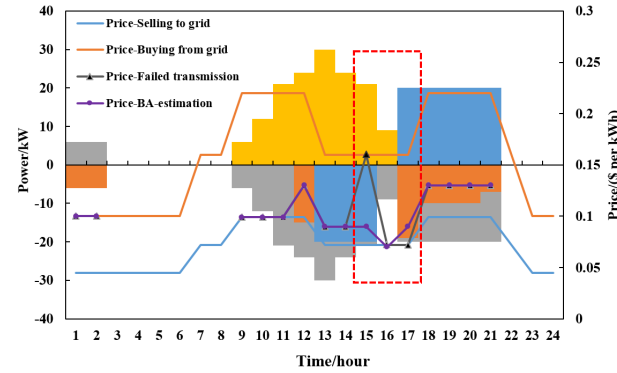
Furthermore, we apply the BA-DRL, and PV and DSM agents will maintain beliefs on ESS's actions. Based on the former definition shown as Fig.4.2 in Section 4.2.1, we set the observation function as Table 4.3. When action a_i of ESS agent is actually taken, only one observation result is received. With a probability 0.6, the observation result is o_i . However, there is a chance that other observation results will be received, and we assume an equal probability here. PV and DSM agents only know the observation results, but the real action is never known to them. We first explain the results under this particular observation function, and then we present the results under various observation functions in the following sections. The results are shown in Fig.4.7(c). At time 15, when ESS agent makes a wrong decision, DSM agent turns off the devices, and PV agent sells electricity to ESS agent with a lower clearing price. The main reason is that DSM and PV agent make a belief that ESS agent will charge at time 15 based on their retrospective experiences, and they apply the belief-based correlated equilibrium to select joint-action. Compared with selling electricity to the grid, the ESS agent sells electricity to DSM devices at time 17, and both agents are benefited. Compared with Fig.4.7(b), the MG system buys less electricity from the main grid, benefiting the MG consumers. This example helps us demonstrate how the proposed scheme works under communication failures.



(a) Energy trading with normal communication



(b) Energy trading with communication failure



(c) Energy trading with Bayesian estimation

Figure 4.7: MG energy management and pricing under various scenarios.

4.4.3 BA-DQL Performance under Supply Uncertainty

Power generation uncertainty is a critical concern for MG that rely on renewables. Hence the designed algorithm should maintain a satisfactory performance under uncertainty. In

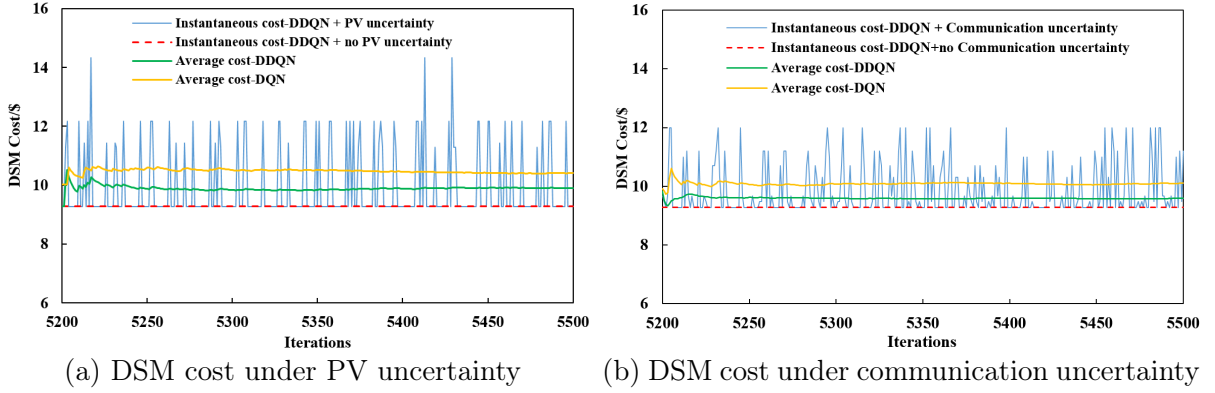


Figure 4.8: MG system performance under uncertainty and different algorithms

this section, we investigate the BA-DRL performance under 2 types of uncertainty: PV generation uncertainty and communication uncertainty resulting from the failed transmissions. Based on the BA-DRL scheme, we compare the DDQN architecture with normal DQN to present the superiority of DDQN.

Fig.4.8(a) shows the DSM daily cost from 5200 to 5500 iterations, where each iteration contains 24 time slots as shown in Fig.4.7. The result shows that the DDQN architecture maintains a good performance under PV uncertainty. Compared with DQN architecture, the DDQN method obtains 6.7% lower cost, and the main reason is that it overcomes the overestimation problem and provides a better estimation. Noting that the lower limit of daily DSM cost is because DSM agent has already obtained the lowest energy price.

Fig.4.8(b) presents the performance under communication uncertainty, where there is a 1% probability of communication failures between the parameter server and agents. It shows that daily DSM cost is affected by the communication failure. The missing collaboration information misleads agents, and they may make suboptimal decisions. Our proposed scheme still maintains good performance. Compared with DQN scheme, the DDQN architecture has a 5% lower DSM cost.

The good performance of DDQN architecture in Fig.4.8 (a) and (b) can be explained by its ability to mitigate the overestimation issue. In traditional DQN, both action selection and evaluation are conducted by the target network, and it constantly selects the maximum Q-values of the next state. If every time the Q-value is calculated a higher value, then the Q-value predicted by the neural network will be obviously higher every time, which will lead to an over-optimistic Q-value estimation. However, in DDQN, decoupling the action evaluation and estimation can provide more accurate Q-value estimations, leading

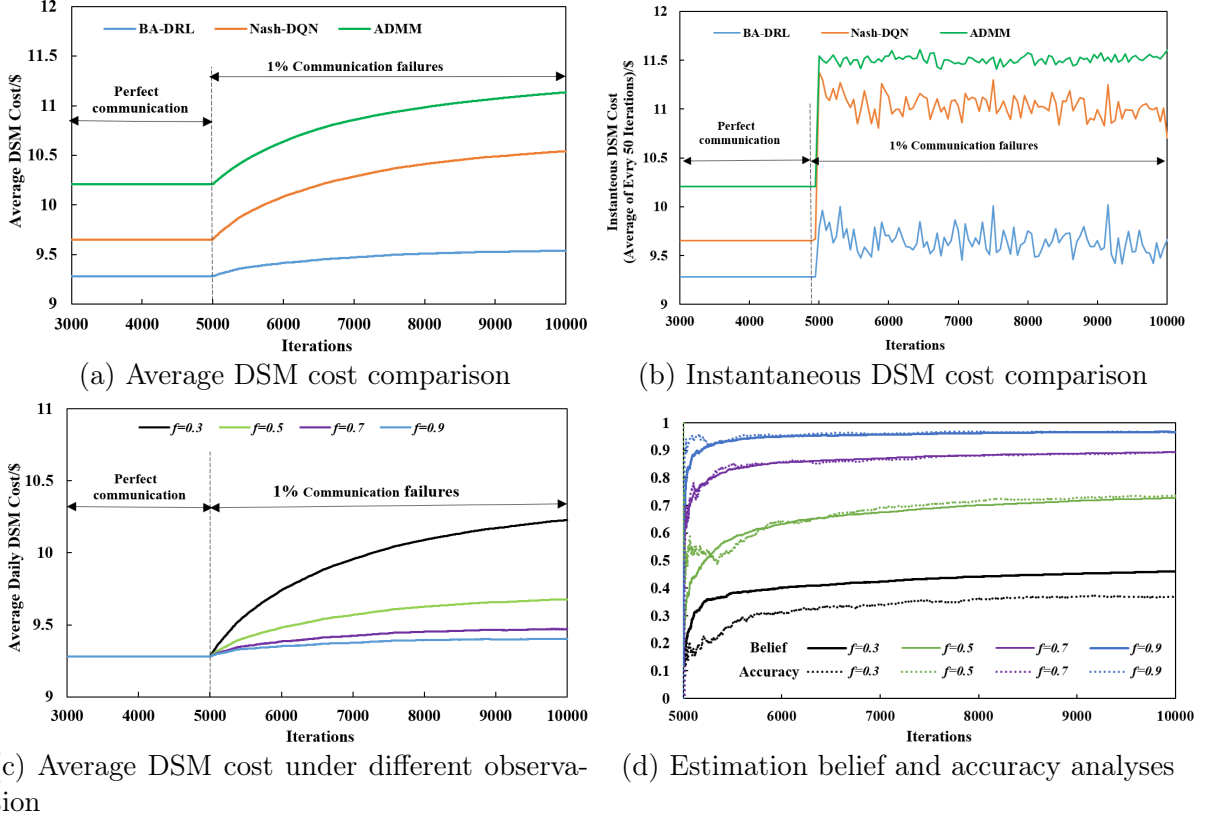


Figure 4.9: MG system performance under uncertainty and different algorithms.

to better action selection and overall performance.

4.4.4 Comparison under Communication Failure

In this section, we compare the BA-DRL with other baseline algorithms, including the Nash-DQN and ADMM methods. To better show the effect of communication failures, we assume normal communication between 3000-5000 episodes, and communication failure occurs with 1% probability from 5000 to 10000 episodes.

Fig.4.9(a) and (b) show the average and instantaneous daily DSM cost. In the first 3000-5000 iterations, which is assumed to be a perfect communication period, BA-DRL has a 4.3% lower cost than Nash-DQN, and 9.1% lower than ADMM method. The reasons are that Nash-DQN is affected by overestimation, and ADMM method cannot detect the long-

term reward. In 5000-10000 iterations, communication failures occur with 1% probability. Then all three algorithms have a higher cost, which means failed transmissions affect the performance of DSM agent. However, the BA-DRL achieves a lower cost than Nash-DQN and ADMM. In BA-DRL, normal agents can estimate isolated agents' action pattern, and make decisions by the belief-based correlated equilibrium. On the contrary, Nash-DQN and ADMM methods cannot estimate the behavior of other agents when communication fails, and the missing information leads to a higher cost.

It is obvious that the BA-DRL relies on the observation function, and Fig.4.9(c) analyzes the performance under different observation functions, where $f = 0.3$ means that we use 0.3 to replace the value 0.6 in Table 4.3. Fig.4.9(d) presents the belief and accuracy, in which the accuracy means the proportion of correct estimation. Fig.4.9(c) and (d) show that a higher observation accuracy leads to lower cost and higher estimation accuracy. With more observations, agents can learn the action patterns of isolated agents. Based on the observation results, DSM agent updates beliefs and makes a good estimation for ESS actions under communication failures.

4.4.5 Comparison of Various Communication Failure Probabilities

Finally, we present the performance of all agents and the whole MG under different communication failure probabilities. Fig.4.10(a) shows that the BA-DRL achieves higher profit for PV and ESS agents, and lower cost for DSM agent. Fig.4.10(b) shows that BA-DRL buys less energy from the grid, and sells more energy to the grid, which means a higher profit for the whole MG. Compared with Nash-DQN and ADMM, BA-DRL sells 5.9% and 12.7% more energy, and buying 8.2% and 13.9% less energy under 1% communication failure probability. Fig.4.10(c) also demonstrates that BA-DRL maintains a high reward for overall MG with varying communication failure probabilities. With 1% communication failure probability, BA-DRL has a 4.1% and 10.3% higher reward compared with Nash-DQN and ADMM.

In Fig.4.10(d), we investigate the number of failed transmissions of different algorithms. ADMM has many more failures, mainly because agents need to communicate iteratively until the result converges, while Nash-DQN has a similar problem. However, BA-DRL only exchanges Q-values once for each time slot, which causes fewer losses under the same failure probability. In Fig.4.10(e), we assume the Bayesian estimation is applied in part of agents, where All-BA, T-BA, O-BA and No-BA indicate all agents, two random agents, only one agent and no agent, respectively. It demonstrates that partial estimation will lead to a lower system reward, and O-BA has comparable performance to No-BA. The

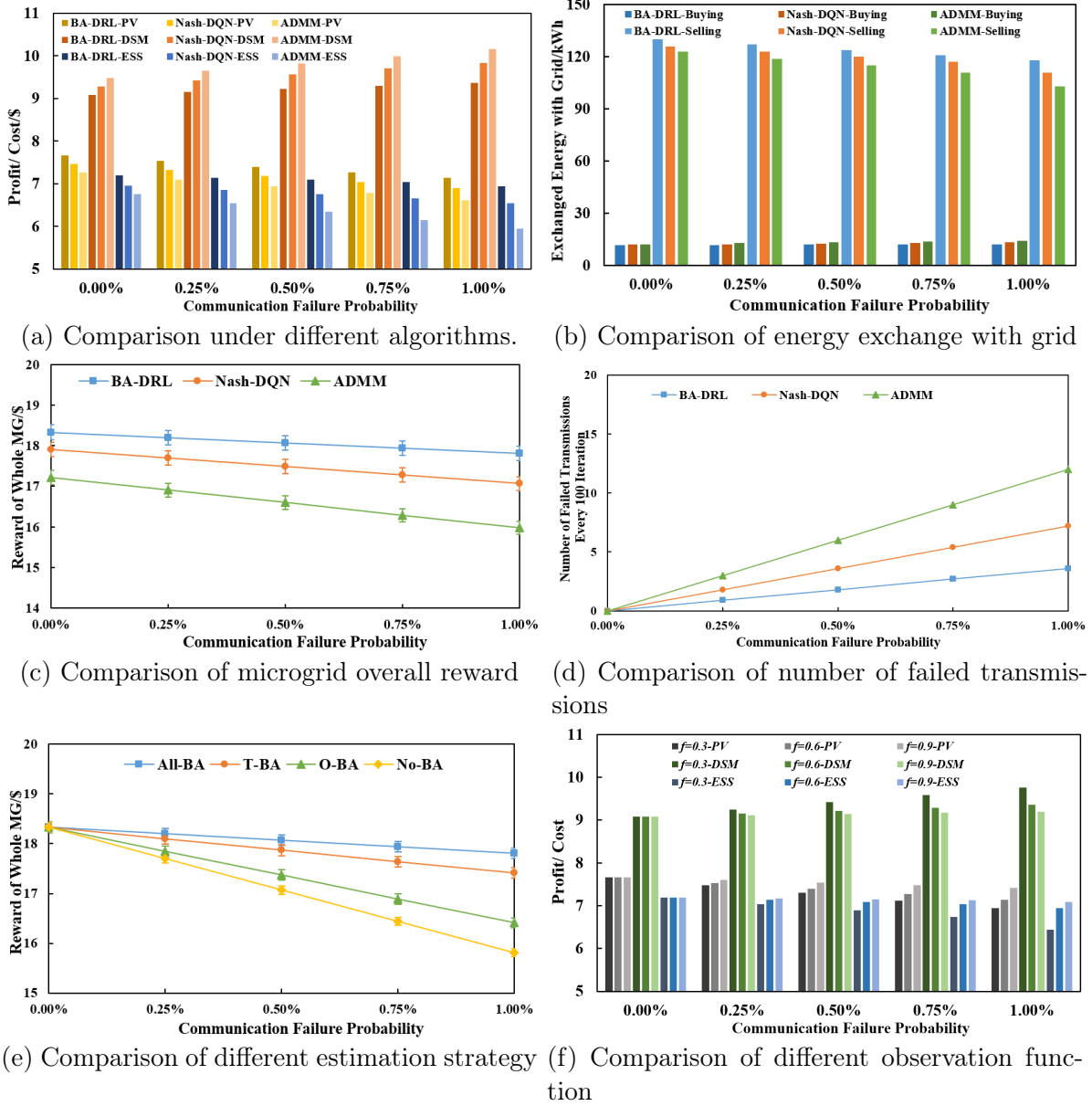


Figure 4.10: Comparison of different communication failure probabilities

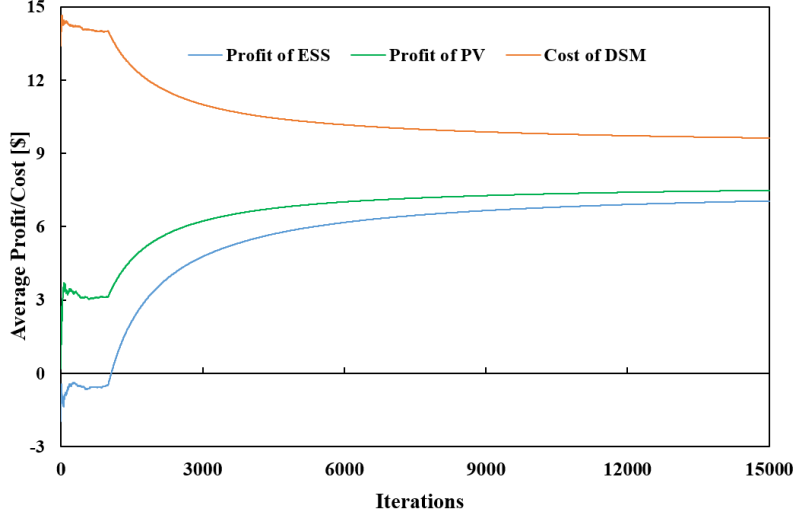


Figure 4.11: Convergence performance of multi-agent BA-DRL

cost and profit of different agents under various observation function values, investigated in Fig.4.10(f), demonstrates that an accurate observation function can lead to a higher profit and lower cost for agents.

4.4.6 Convergence and Complexity analyses

Finally, we present the convergence performance and complexity analyses. The convergence performance of the proposed multi-agent BA-DRL is shown in Fig.4.11. It is observed that all agents benefit from the BA-DRL algorithm, including the increasing ESS and PV profits as well as the decreasing DSM cost. The results show that the proposed multi-agent BA-DRL can coordinate the actions of all agents and converge to a satisfactory overall performance for the whole MG. The convergence performance of BA-DRL can be explained by the good convergence property of correlated equilibrium. In addition, we decay the learning rate every several iterations, contributing to a stable learning process and a good convergence performance.

In addition, we analyze the memory and runtime complexity of the proposed algorithms. In this work, the hardware environment is Intel core i7-7770 central processing unit (CPU) with 16 G memory size. For the memory complexity, both BA-DRL and Nash-DQN use the neural network for Q-values approximation. It means no huge Q-table is needed to record all the Q-values; thus, the memory space is greatly reduced. Meanwhile, the ADMM

algorithm also has no stringent requirements for memory space, because there is no need to store variables between iterations.

For the run time complexity, the BA-DRL is designed in a decentralized manner, and each agent runs BA-DRL parallelly, reducing the complexity and improving scalability. The computational complexity of LSTM network that we used in BA-DRL is given by $O((MC)^2)$ (see the appendix A.1), where M is the number of memory blocks, and C is the number of memory cells in each block. Meanwhile, the correlated equilibrium also contributes to the computational complexity. It can be easily formulated as a linear problem and solved in polynomial time $O(n^k)$, where n is the number of agents, and k is the power of polynomial. The average runtimes per iteration are 0.331s for BA-DRL, 0.325s for Nash-DQN and 0.129s for ADMM. The BA-DRL has a comparable runtime with Nash-DQN, but ADMM has a much lower runtime. This result can be explained by the time-consuming network training process, which is a well-known issue for DRL algorithms. However, the network training is only implemented after a certain number of iterations. The runtime can be alleviated by reducing network training frequency or simplifying network architecture.

Chapter 5

Multi-agent Correlated Q-Learning for 5G RAN Slicing

In 5G networks, heterogeneous QoS requirements of different services and limited spectrum resources make radio resource allocation a challenging problem [148, 149]. This Chapter proposed a MARL method for radio resource slicing in 5G networks. Similar to the multi-agent MG environment, we model each slice as an intelligent agent that competes for limited radio resources, and the correlated Q-learning is applied for inter-slice resource block (RB) allocation. However, it is worth noting that these two environments can be very different, requiring dedicated algorithm designs. The proposed COQRA scheme is compared with Nash Q-learning (NQL), LRTQ methods, and priority proportional fairness (PPF) algorithm. The simulations show that the proposed COQRA achieves 32.4% lower latency and 6.3% higher throughput when compared with LRTQ, and 5.8% lower latency and 5.9% higher throughput than NQL. In addition, significantly higher throughput and lower packet drop rate are observed compared with PPF algorithm.

5.1 System Model and Problem Formulation

As shown in Fig.5.1, we consider URLLC and eMBB slices where each slice serves several users. First, the slice manager collects QoS requirements of the users, such as bandwidth and latency, then the collected information is sent to the software defined networks (SDN) controller for the required resources. Based on the received requirements, SDN controller implements inter-slice RB allocation to distribute RBs between slices. Then, the users are

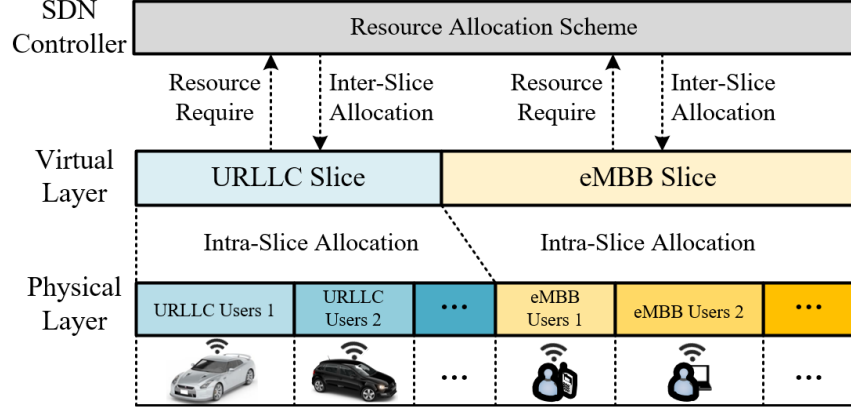


Figure 5.1: Network slicing-based two-step resource allocation.

scheduled within the allocated RBs for that particular slice. We consider numerology 0, where one RB contains 12 sub-carriers in the frequency domain.

Here we assume each slice manager is an intelligent agent making decisions autonomously. For the eMBB agent, it needs to maximize the throughput, as denoted by:

$$\max \sum_{j=1}^J \sum_{e=1}^{E_j} b_{j,e,t}, \quad (5.1)$$

where $b_{j,e,t}$ is the throughput of e^{th} eMBB user in j^{th} BS at time t , and E_j is the number of eMBB users of j^{th} BS.

URLLC agent needs to minimize the delay as follows:

$$\min \sum_{j=1}^J \sum_{u=1}^{U_j} d_{j,u,t}, \quad (5.2)$$

where $d_{j,u,t}$ is the delay of u^{th} URLLC user in j^{th} BS at time t , and U_j is the number of URLLC users of j^{th} BS.

The packet delay d mainly consists of three components:

$$d = d^{tx} + d^{que} + d^{rtx}, \quad (5.3)$$

where d^{tx} is the transmission delay, d^{que} is the queuing delay, and d^{rtx} is the re-transmission delay.

The transmission delay of a packet depends on the link capacity between the UE and the BS:

$$d^{tx} = \frac{L_u}{E_{u,j}}, \quad (5.4)$$

where L_u is the packet size of u^{th} UE, and $C_{u,j}$ is the link capacity between u^{th} UE and the BS it belongs to.

The link capacity is calculated as follows:

$$E_{u,j} = \sum_{r \in N_u^{RB}} b_{RB} \log\left(1 + \frac{p_{j,r} x_{j,r,u} g_{j,r,u}}{b_{RB} N_0 + \sum_{j' \in J_{-j}} p_{j',r'} x_{j',r',u'} g_{j',r',u'}}\right), \quad (5.5)$$

where N_u^{RB} is the set of RBs that the u^{th} UE uses, b_{RB} is the bandwidth of one RB, $p_{j,r}$ is the transmission power of r^{th} RB in j^{th} BS, $x_{j,r,u}$ is a binary variable to indicate whether this RB is distributed to u^{th} UE, $g_{j,r,u}$ is the channel gain between BS and UE, N_0 is the unit noise power density, and $j' \in J_{-j}$ is the BS set except j^{th} BS.

5.2 Correlated Q-learning Based Resource Allocation

5.2.1 COQRA Architecture

The architecture of the proposed multi-agent COQRA is illustrated in Fig.5.2. Each slice is an independent agent and observes its state from the environment. The agent exchanges Q-values with its peers to make decisions, and the action selection is determined by correlated equilibrium. Then, the selected actions are sent to the wireless environment, and eMBB and URLLC users are scheduled RBs within the slice resources according to the proportional fairness algorithm [73]. Users transmit packets based on allocated bandwidth, and the experienced throughput and delay are sent back to agents. Finally, the reward is received, and slice managers make the next decisions based on the new state and updated Q-values.

5.2.2 Markov Decision Process and Q-learning

In this section, based on the system model, we will define the MDP to describe agents and introduce the learning scheme. Here we define each slice manager as an intelligent agent, which will interact with the environment and make decisions autonomously.

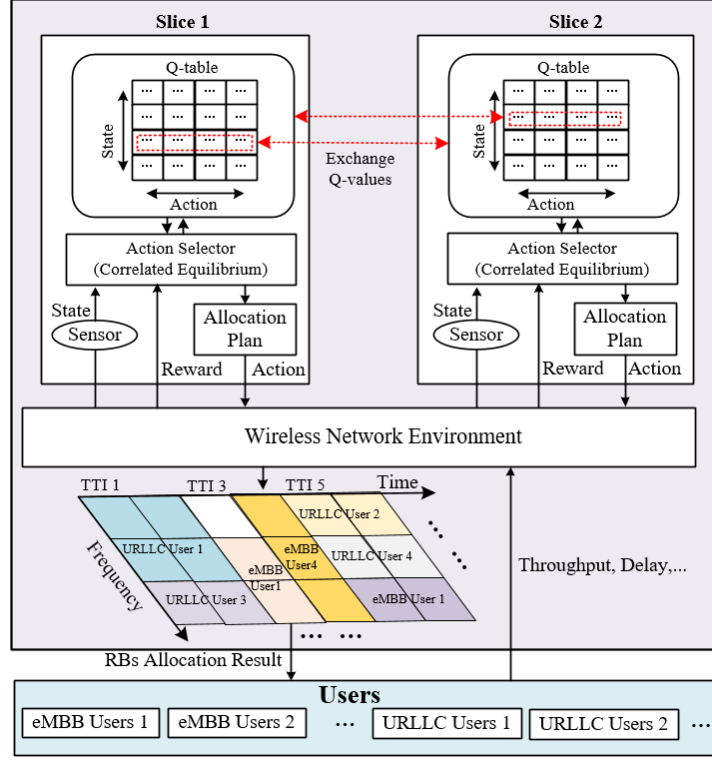


Figure 5.2: Proposed COQRA architecture for resource management among slices.

We assume each agent has its state, action and reward signal. The state s^u for URLLC slice manager agent (from hereon, referred to as URLLC agent) is the number of packets in its queue, and the action a^u is the number of RBs it allocates. The state and action for the eMBB slice manager agent (from hereon, referred to as eMBB agent) are defined similarly, namely s^e and a^e . Thus, the Q-space for both agents are $Q^u = \{s^u, a^u\}$ and $Q^e = \{s^e, a^e\}$.

The reward function for eMBB agent is given in (5.6), where obtaining higher throughput leads to a higher reward [77]

$$r_{j,t}^{eMBB} = \frac{2}{\pi} \arctan \left(\sum_{u=1}^{|\mathcal{M}_j^{eMBB}|} b_{j,u,t} \right), \quad (5.6)$$

where $\mathcal{M}_j^{eMBB}$ is the set of eMBB users in BS j .

The reward function for URLLC agent at time t is [77]

$$r_{j,t}^{URLLC} = \begin{cases} 1 - \max_{u \in H_t^u} (d_u^{que})^2, & |H_t^u| \neq 0, \\ 0, & |H_t^u| = 0, \end{cases} \quad (5.7)$$

where d_u^{que} is the queuing delay for u^{th} URLLC user, $|H_t^u|$ denotes the length of the queue for URLLC users at time slot t . In (5.7), a lower queuing delay means a higher reward, which depends on the number of RBs that the agent gets. URLLC agent competes for more RBs to reduce the queuing delay. Meanwhile, to guarantee the packet drop rate, we apply a constant negative reward as a penalty if any packet is dropped.

In Q-learning, one agent always aims to maximize the long-term accumulated reward. For one agent λ , the state value is:

$$V_\lambda^\pi(s_\lambda) = \mathbb{E}_\pi \left(\sum_{n=0}^{\infty} \gamma^n r_\lambda(s_{\lambda,n}, a_{\lambda,n}) | s_\lambda = s_{\lambda,0} \right), \quad (5.8)$$

where π is the policy, $s_{\lambda,0}$ is the initial state, $r_\lambda(s_{\lambda,n}, a_{\lambda,n})$ is the reward of taking action $a_{\lambda,n}$ at state $s_{\lambda,n}$, γ is the reward discount factor. $V_\lambda^\pi(s_\lambda)$ represents long-term expected accumulated reward at state s_λ .

Then we define the state-action value $Q_\lambda^\pi(s_\lambda, a_\lambda)$ to describe the expected reward of taking action a_λ under state s_λ :

$$Q_\lambda^\pi(s_\lambda, a_\lambda) = (1 - \alpha) Q_\lambda^\pi(s_\lambda, a_\lambda) + \alpha (r(s_\lambda, a_\lambda) + \gamma \max_{a'_\lambda} Q_\lambda^\pi(s'_\lambda, a'_\lambda)) \quad (5.9)$$

where s_λ and s'_λ are current and next state, and a_λ and a'_λ are current and next action, and α is the learning rate. By updating the Q-values, the agent will learn the best action sequence, and achieve a long term best reward.

When only one agent exists, the ϵ -greedy policy is generally applied to balance exploration and exploitation.

$$a_\lambda = \begin{cases} \text{random action}, & \text{rand} \leq \epsilon, \\ \arg \max (Q_\lambda^\pi(s_\lambda, a_\lambda)), & \text{otherwise,} \end{cases} \quad (5.10)$$

where ϵ is the exploration probability, and $0 < \epsilon < 1$, and *rand* indicates a random number between 0 and 1.

On the other hand, in a multi-agent environment, the action of one agent will affect both the environment and the reward of the other agents. Therefore, we propose a correlated Q-learning-based resource allocation approach to address the multi-agent 5G environment.

5.2.3 Correlated Equilibrium

Given the limited bandwidth resources, in our multi-agent environment, each slice manager agent will compete for more RBs to optimize their own goal, which may lead to a conflict. Therefore, we use the correlated equilibrium to balance the reward of each agent and maintain good overall performance for the whole multi-agent system. In correlated equilibrium, agents exchange Q-values for sharing information, and the joint action is chosen according to the following equations:

$$\begin{aligned}
& \max \sum_{\vec{a} \in A} Pr(\vec{s}, \vec{a}) Q(\vec{s}, \vec{a}) \\
& \text{sub.to} \sum_{\vec{a} \in A} Pr(\vec{s}, \vec{a}) = 1 \\
& \sum_{\vec{a}_{-\lambda} \in A_{-\lambda}} Pr(\vec{s}, \vec{a}_{-\lambda}) (Q(s, \vec{a}_{-\lambda}) - Q(\vec{s}, \vec{a}_{-\lambda}, a_{\lambda})) \geq 0 \\
& 0 \leq Pr(\vec{s}, \vec{a}) \leq 1
\end{aligned} \tag{5.11}$$

where $\vec{s}$ is the system state of eMBB and URLLC agents $\vec{s} = (s^e, s^u)$, $\vec{a} = (a^e, a^u)$ is the joint action, $Pr(\vec{s}, \vec{a})$ is the probability of choosing action combination $\vec{a}$ under state $\vec{s}$, a_{λ} is the action of agent i , $\vec{a}_{-\lambda}$ is the action combination except agent i , and $A_{-\lambda}$ is the set of $\vec{a}_{-\lambda}$. The correlated equilibrium is described as a linear program in (5.11). The objective is to maximize the total expected reward of all agents, and the constraints guarantee a probability distribution of action combination in which each agent chooses an optimal action.

Based on correlated equilibrium, an improved ϵ -greedy policy is applied for action selection:

$$\pi_{\lambda}(s) = \begin{cases} \text{random action,} & rand \leq \epsilon, \\ \text{equation(5.11),} & rand > \epsilon. \end{cases} \tag{5.12}$$

Exploration is performed whenever $rand \leq \epsilon$, i.e., random action is selected. Otherwise, the exploitation is implemented by correlated equilibrium. The COQRA scheme is summarized in Algorithm 1. In COQRA, the two-step resource allocation method avoids the complexity of processing all UE requirements by a central controller, which will reduce the computational complexity of the learning algorithm by a smaller action space.

Algorithm 5.1 COQRA based Resource Allocation

- 1: **Initialize:** Q-learning and wireless network parameters
 - 2: **for** $TTI = 1$ to T **do**
 - 3: **if** $rand < \epsilon$ **then**
 - 4: Choose a_t^u and a_t^e randomly.
 - 5: **else**
 - 6: Agents exchange Q-values under their current state. Find correlated equilibrium using eq. (5.11) and choose action a_t^u, a_t^e
 - 7: **end if**
 - 8: Complete inter-slice resource allocation.
 - 9: Implement intra-slice allocation. Schedule users on respective slice resources using the proportional fair algorithm.
 - 10: Agents calculate reward based on received QoS metrics.
 - 11: Update agent state s^u, s^e and Q-table Q^u and Q^e .
 - 12: **end for**
-

5.2.4 Nash Equilibrium

This section introduces the NQL algorithm, which is generally used in multi-agent problems. Compared with correlated equilibrium, the Nash equilibrium is an iterative-based coordination method. In some cases, there could be more than one Nash equilibrium or no equilibrium. We use NQL as a baseline algorithm. In NQL, agents select actions by:

$$U_\lambda(\vec{a}_{-\lambda}, a_\lambda) \geq U_\lambda(\vec{a}), \vec{a} \in A, \vec{a}_{-\lambda} \in A_{-\lambda}, \quad (5.13)$$

where $\vec{a}$ is the action combination, $\vec{a}_{-\lambda}$ is the action combination except agent i , A is the set of $\vec{a}$, and $A_{-\lambda}$ is the set of $\vec{a}_{-\lambda}$. U_λ is the utility function for agent i , which refers to Q-values in this work. At Nash equilibrium, agents are less likely to change their actions, leading to lower observed utility. Similarly, we apply an improved ϵ -greedy policy to select actions as (5.13). The NQL scheme is summarized in Algorithm 2. We assume the equilibrium is randomly selected if more than one equilibrium is found.

5.2.5 LRTQ and PPF Algorithms

To further investigate the performance of the proposed method, two more baseline algorithms are used in this work. LRTQ was proposed in [77]. LRTQ is also a learning-based resource allocation method, but it only defines one reward function for all users. PPF

Algorithm 5.2 NQL based Resource Allocation

```
1: Initialize: Q-learning and wireless network parameters
2: for  $TTI = 1$  to  $T$  do
3:   if  $rand < \epsilon$  then
4:     Choose  $a_t^u$  and  $a_t^e$  randomly.
5:   else
6:     Each agent, searching its own Nash equilibrium using eq. (5.13).
7:     Match each agent's equilibrium to get the system Nash equilibrium.
8:     Agents choose actions according to the system Nash equilibrium.
9:   end if
10:  Complete inter-slice resource allocation.
11:  Implement intra-slice allocation, using the proportional fair algorithm.
12:  Agents calculate reward based on received QoS metrics.
13:  Update agent state  $s^u$ ,  $s^e$  and Q-table  $Q^u$  and  $Q^e$ .
14: end for
```

algorithm is applied in [73]. In PPF, the RBs are first allocated to URLLC users to guarantee low latency, then the remaining RBs are distributed among eMBB users. Note that network slicing is not implemented in both of these baseline algorithms. These algorithms perform RB allocation decisions only.

5.3 Performance evaluation

5.3.1 Parameter Settings

We use MATLAB to implement our proposed algorithm. We consider five gNBs with 500 meters inter-site distance, each serving one eMBB and one URLLC slice. Each eMBB slice serves 5 users, and URLLC slice has 10 users, randomly distributed in the cell. The bandwidth for a cell is 20 MHz, and there are 100 RBs. Each RB contains 12 sub-carriers, and each sub-carrier has 15kHz. 100 RBs are divided into 13 RB groups, where the first 12 groups contain 8 RBs each, and the last group contains 4 RBs. The simulation period is 5000 TTIs, and each TTI contains 2 Orthogonal frequency-division multiplexing (OFDM) symbols (5G mini-slot length 0.143ms) allocations. The ϵ -greedy policy is implemented in the first 3000 TTI, and the rest 2000 TTI is pure exploitation. Other parameters are shown in Table 5.1, and the Q-learning parameters are selected by grid search method. Each scenario is repeated 10 runs to get an average value with 95% confidence interval.

Table 5.1: Parameter settings of wireless networks and Q-learning

Parameters	Value
Traffic Model	URLLC Traffic: 80% Poisson distribution and 20% constant bit rate.
	eMBB Traffic: Poisson distribution.
	Packet Size: 32 Bytes.
Propagation Model	$128.1 + 37.6 \log(\text{distance}(\text{km}))$.
	Log-Normal shadowing with 8 dB.
Transmission settings	Transmission power: 40 dBm (Uniform distributed)
	Tx/Rx antenna gain: 15 dB.
	3GPP urban network.
Q-learning	Learning rate: 0.9
	Discount factor: 0.5
	Epsilon value: 0.05

5.3.2 Simulation Results

First, we set eMBB load to 0.5 Mbps per cell and consider that URLLC load changes from 1 Mbps to 3 Mbps per cell. The latency distribution of four algorithms is shown in Fig.5.3. Meanwhile, Fig.5.4 presents the average URLLC latency against varying URLLC offered loads. The results show that COQRA, NQL and PPF have a comparable latency distribution, while the LRTQ has a relatively higher latency. The PPF has the lowest overall latency for URLLC traffic, and the main reason is that URLLC traffic has a priority in this method. Whenever URLLC packet arrives, it will be directly scheduled over eMBB traffic in the PPF algorithm. Meanwhile, the COQRA achieves 4.4% and 5.8% lower latency than NQL when URLLC load is 2 Mbps and 3 Mbps, respectively. Compared with LRTQ, the COQRA has a 27.1% lower latency under 2 Mbps load, and 32.4% lower latency under 3 Mbps load.

Furthermore, the eMBB throughput under different URLLC loads is shown in Fig.5.5. The result shows that COQRA, NQL and LRTQ have a close performance of throughput, while the PPF has a much lower value. When URLLC load is 3 Mbps, the COQRA method has a 5.9% higher throughput over NQL method, and 6.3% higher than LRTQ.

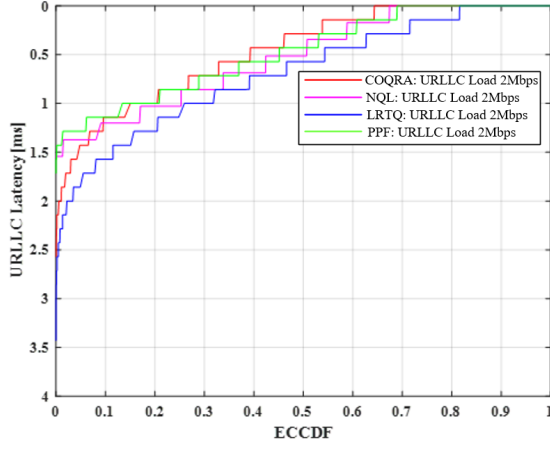


Figure 5.3: URLLC latency distribution[ms] under 2Mbps load per cell.

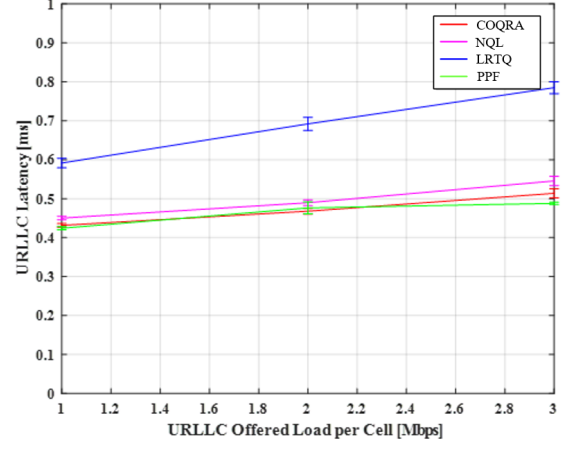


Figure 5.4: URLLC latency [ms] with varying offered load [Mbps] per cell.

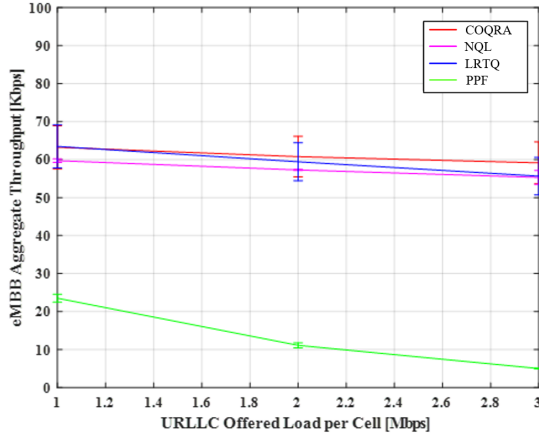


Figure 5.5: eMBB throughput with varying URLLC loads [Mbps].

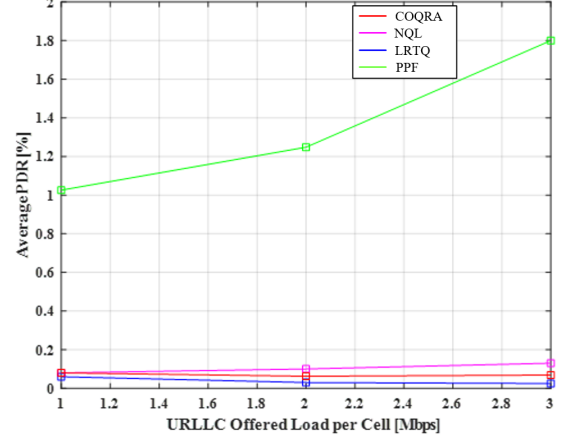


Figure 5.6: Average packet drop rate under varying URLLC loads [Mbps].

Although the PPF has a good latency performance for URLLC, the eMBB throughput is almost 90% lower than the other three algorithms. This result can still be explained by the priority settings in PPF, which means the eMBB throughput will decrease with increasing prioritized URLLC load. On the other hand, the COQRA, NQL and LRTQ benefit from the jointly optimizing scheme, and they maintain a good throughput performance. In Fig.5.6, we compare the packet drop rate of four algorithms. We show that COQRA,

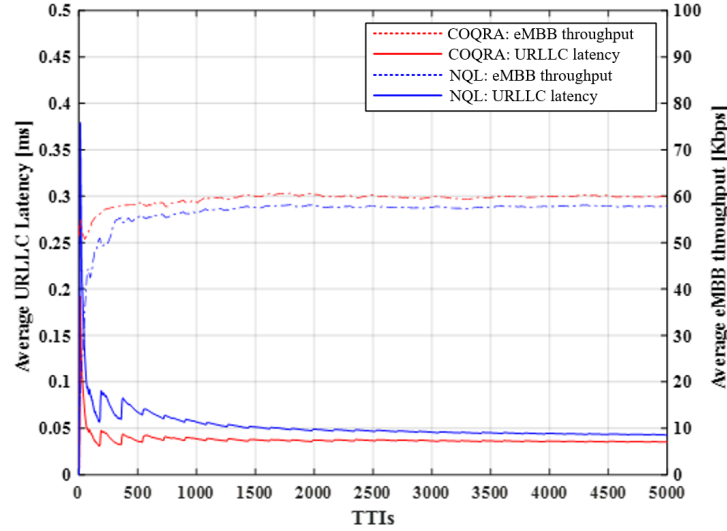


Figure 5.7: Convergence performance of COQRA and NQL

NQL and LRTQ maintain a much lower packet drop rate than PPF method. In learning algorithms, agents will get a negative reward when dropping packets. However, PPF fails to maintain a low packet drop rate under all traffic loads, where a worst-case packet drop rate of 1.8% is observed. Finally, we compare the convergence performance of COQRA and NQL, and a faster convergence is observed for COQRA in Fig.5.7. The reason is that COQRA has a more efficient way of finding the equilibrium. Overall, COQRA outperforms all baseline methods in terms of latency, throughput, packet loss and convergence time.

Chapter 6

Transfer Reinforcement Learning-based Radio and Computation Resource Allocation in 5G RAN Slicing

To implement network slicing in 5G, resource allocation is a key function to allocate limited network resources such as radio and computation resources to multiple slices [150,151]. However, the joint resource allocation also leads to higher complexity for network management. In addition, compared with MG energy management, 5G networks have more stringent requirements for fast and real-time responses, which indicates faster algorithm training and convergence. Therefore, this Chapter proposed a KTRA method to jointly allocate radio and computation resources for 5G RAN slicing. Compared with existing works, the main difference is that KTRA method has a knowledge transfer capability. It is designed to use the prior knowledge of similar tasks to improve the performance of the target task, e.g., faster convergence speed or higher average rewards. Here we assume the expert agent has been well-trained for solving source tasks, and no extra training is required for the existing expert agent. The proposed KTRA is compared with Q-learning based resource allocation (QLRA) algorithm, and KTRA method presents an 18.4% lower URLLC delay and a 30.1% higher eMBB throughput as well as a faster convergence speed.

6.1 System Model and Problem Formulation

6.1.1 Network Architecture

The proposed system model is shown in Fig.6.1. We assume the base station (BS) is equipped with a MEC server to process computation tasks of eMBB and URLLC slices [152]. We apply a two-step resource allocation scheme. In the inter-slice phase, the BS intelligently allocates radio and computation resources between different slices. Then these resources are utilized within each slice in the intra-slice phase. For example, the URLLC slice operator will distribute radio resources between attached UEs, and use available MEC server capacity to process the computation tasks of its UEs. In this work, we mainly focus on inter-slice resource allocation. Given this architecture, the delay experienced by a computation task is:

$$d = d^{tx} + d^{rtx} + d^{que} + \vartheta d^{edge} + (1 - \vartheta) d^{cloud}, \quad (6.1)$$

where d^{tx} and d^{rtx} are the task transmission and retransmission delays, respectively. d^{que} is the queuing delay in BS, which is considered the scheduling delay. d^{edge} is the processing delay in MEC server, and d^{cloud} is the processing delay in the central cloud. ϑ is a binary variable. $\vartheta = 1$ if the task is computed in the BS, and $\vartheta = 0$ means the task is forwarded to the central cloud. We presume the intelligent BS will decide whether to process the task in MEC server or offload it to the cloud. Equation(6.1) shows that the delay is affected by both radio and computation resources. The radio resource allocation will affect the transmission delay d^{tx} , and computation resource allocation can change the processing delay d^{edge} . Meanwhile, the scheduling efficiency will affect the queuing delay d^{que} . Following, we will explain the communication and computation model.

6.1.2 Communication Model

We consider RBs as the smallest time-frequency resource distributed to users. The transmission delay d^{tx} between BS and UE is:

$$d^{tx} = \frac{L_u}{E_{j,u}}, \quad (6.2)$$

where L_u is the transmitted packet size of UE u , $E_{j,u}$ is the link capacity between the BS j and the UE u . The link capacity depends on the number of RBs that are allocated to

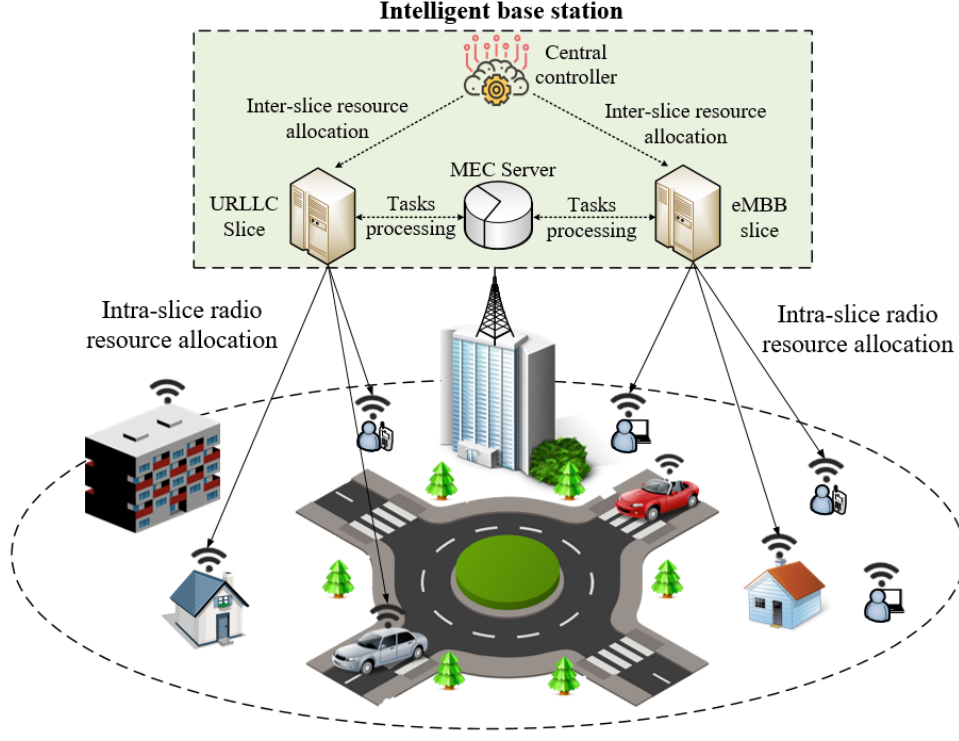


Figure 6.1: Proposed system architecture for hierarchial resource allocation.

this transmission:

$$E_{j,u} = \sum_{r \in N_u^{RB}} b_{RB} \log \left(1 + \frac{p_{j,r} x_{j,u,r} g_{j,u,r}}{b_{RB} N_0 + \sum_{j' \in \mathcal{J}_{-j}} \sum_{u' \in \mathcal{M}_{j'}} \sum_{r' \in N_{j'}^{RB}} p_{j',r'} x_{j',u',r'} g_{j',u',r'}} \right), \quad (6.3)$$

where N_u^{RB} denotes the set of RBs that is allocated to UE u , b_{RB} denotes the bandwidth of one RB, N_0 denotes the noise power density, $p_{j,r}$ denotes the transmission power of RB r in BS j . $x_{j,u,r}$ is a binary variable. $x_{j,u,r} = 1$ means the RB r is allocated to UE u ; otherwise $x_{j,u,r} = 0$. $g_{j,u,r}$ denotes the channel gain between BS j and UE u . $\mathcal{J}_{-j}$ denotes the set of BSs except j^{th} BS, $\mathcal{M}_{j'}$ denotes the set of UEs in BS j' , and $N_{j'}^{RB}$ denotes the set of total RBs in BS j' .

6.1.3 Computation Model

For a computation task from UEs, we assume it requires certain computation resources to complete the task (denoted by the number of CPU cycles). Then the processing delay in MEC server d^{edge} is:

$$d^{edge} = \frac{c_{u,q}}{\mathcal{B}C_j}, \quad (6.4)$$

where $c_{u,q}$ denotes required computation resources of task q from UE u , $\mathcal{B}$ denotes the proportion of computation resources allocated to this task ($0 \leq \mathcal{B} \leq 1$), and C_j denotes the total computation capacity of MEC server in BS j .

On the other hand, BS may offload the task to central cloud computation servers [153]. The d^{cloud} in equation(6.1) is described as:

$$d^{cloud} = d^{up} + d^{down} + d^{c,que} + d^{c,computation}, \quad (6.5)$$

where d^{up} and d^{down} are upload and download transmission delays of the computation task, respectively. $d^{c,que}$ is the cloud queuing delay, and $d^{c,computation}$ is cloud computation delay. The d^{down} and $d^{c,computation}$ can be omitted because: i) the downloaded packet size after computation is usually much smaller than the input packet; ii) the central cloud usually has a very high computation capacity [154, 155]. Then equation(6.5) can be rewritten as:

$$d^{cloud} = \frac{1}{\frac{B}{L_s} - \phi} + d^{c,que}, \quad (6.6)$$

where B is the backhaul capacity, B/L_u denotes the service rate, and ϕ is the packet arrival rate. We apply the M/M/1 queue model to describe the upload delay $d^{up} = \frac{1}{\frac{B}{L_s} - \phi}$. Meanwhile, this work mainly focuses on the RAN resource allocation, and it is reasonable to assume a fixed cloud queuing delay $d^{c,que}$. Finally, we assume: i) the task is preferred to be processed in the MEC server of the BS due to the potential benefit of MEC; ii) BS will offload the task to the central cloud if the queuing time expires the preset target delay [156].

6.1.4 Problem Formulation

Here we consider two typical slices: eMBB and URLLC slices. The eMBB slice intends to maximize the throughput, while the URLLC slice requires lower latency. The intelligent BS

needs to balance the requirements of both slices; then we define the problem formulation as follows:

$$\max \quad w^{emb} b_j^{emb,avg} + w^{urllc} (d^{tar} - d_j^{urllc,avg}) \quad (6.7)$$

$$\text{s.t.} \quad b_j^{emb,avg} = \frac{\sum_{u \in \mathcal{M}_j^{emb}} b_{j,u}^{emb}}{|\mathcal{M}_j^{emb}|} \quad (6.7a)$$

$$b_j^{urllc,avg} = \frac{\sum_{v \in \mathcal{M}_j^{urllc}} d_{j,v}^{urllc}}{|\mathcal{M}_j^{urllc}|} \quad (6.7b)$$

$$(6.1) (6.2) (6.3) (6.4) (6.5) \quad (6.7c)$$

$$\sum_{u \in \mathcal{M}_j^{emb}} x_{j,u,r'} + \sum_{v \in \mathcal{M}_j^{urllc}} x_{j,v,r'} = 1 \quad (6.7d)$$

$$\sum_{r' \in N_j^{RB}} \left(\sum_{u \in \mathcal{M}_j^{emb}} x_{j,u,r'} + \sum_{v \in \mathcal{M}_j^{urllc}} x_{j,v,r'} \right) \leq |N_j^{RB}| \quad (6.7e)$$

$$C_j^{emb} + C_j^{urllc} \leq C_j \quad (6.7f)$$

where $b_j^{emb,avg}$ and $d_j^{urllc,avg}$ denote the average throughput and latency of eMBB and URLLC slices, respectively, which are calculated by equation(6.7a) and (6.7b). w^{emb} and w^{urllc} are weighting factors to balance two metrics and form an overall objective. d^{tar} is the preset target delay of URLLC slice, which will bring a positive reward if $d^{tar} > d_j^{urllc,avg}$. $b_{j,u}^{emb}$ is the throughput of UE u in the eMBB slice of BS j , and $d_{j,v}^{urllc}$ is the latency of UE v in the URLLC slice. $\mathcal{M}_j^{emb}$ and $\mathcal{M}_j^{urllc}$ denote the UE set of eMBB and URLLC slices in BS j , respectively. In equation(6.7), a higher eMBB throughput and a lower URLLC latency are expected to maximize the objective. The equation(6.7d) guarantees one RB can only be allocated to one UE. Equation (6.7e) and (6.7f) mean that total allocated RBs and computation capacity should not exceed the available resources in the BS j .

6.2 Knowledge Transfer based Radio and Computation Resource Allocation

In this section, we first introduce the proposed KTRA method, including the knowledge transfer strategy, MDP definition, and the knowledge transfer reinforcement learning. Then we introduce the Q-learning-based resource allocation scheme as a baseline algorithm.

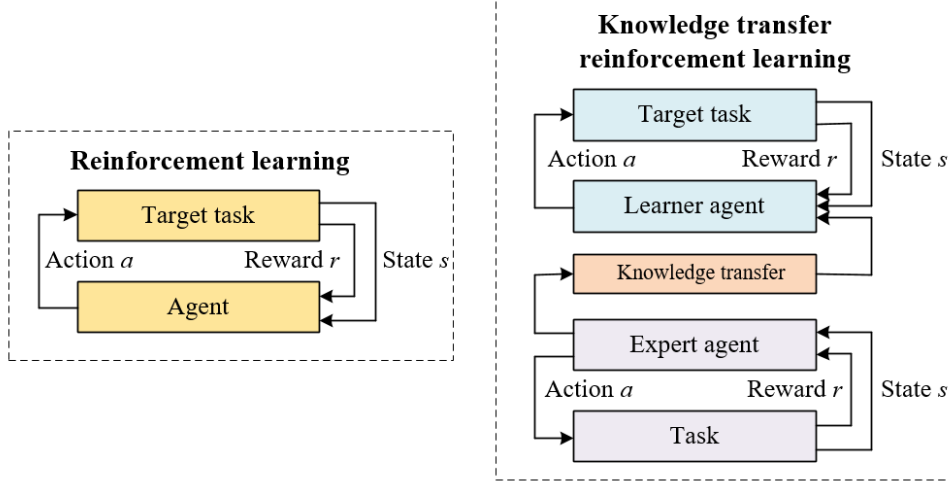


Figure 6.2: Learning strategy comparison between RL and HRL

6.2.1 Knowledge Transfer Strategy and MDP definition.

First, we compare knowledge transfer reinforcement learning with reinforcement learning to better explain the knowledge transfer strategy. The interaction between agent and task can be described by MDP $\langle S, A, T, R \rangle$, where S , A , T , R are the set of states, set of actions, transition probability, and reward function, respectively. As shown in Fig.6.2, in reinforcement learning, the agent selects an action, receives a reward, and arrives in a new state. The agent starts to explore the task from scratch, since no prior knowledge is available in the learning phase.

By contrast, in knowledge transfer reinforcement learning, two agents are involved, namely learner and expert agents. Compared with the single learning phase in RL, knowledge transfer reinforcement learning has two phases: the knowledge transfer phase and the learning phase. In the knowledge transfer phase, considering different MDP definitions of two agents, a map function is needed to transform the expert agent's knowledge to the learner agent. In the learning phase, the learner agent utilizes the expert agent's knowledge to improve its performance on the target task. The learner agent is expected to achieve a higher exploration efficiency, since it already has some prior knowledge at the beginning of exploration.

In this work, we define a learner agent for joint radio and computation resources allocation and an expert agent for radio resource allocation. The expert agent has no knowledge of computation resource allocation, but its knowledge of radio resource allocation can be

used by the learner agent. Following, we will define the state, action and reward of two agents.

- **State:** The states of both expert and learner agents are (q^{embb}, q^{urllc}) . q^{embb} denotes the number of tasks in the queue of eMBB slice, and q^{urllc} is defined similarly. (q^{embb}, q^{urllc}) represents the demands of two slices, and the agent can select actions accordingly.
- **Action:** For the expert agent, it only implements the radio resource allocation, and then the action is defined as (r^{embb}, r^{urllc}) , which represents the number of radio resources allocated to eMBB and URLLC slices. For the learner agent, it considers both radio and computation resource allocation, and then the action is defined as $(r^{embb}, r^{urllc}, c^{embb}, c^{urllc})$, where c^{embb} and c^{urllc} denote the computation capacity allocated to eMBB and URLLC slices, respectively.
- **Reward:** The reward is defined by the objective of slices, which is denoted by (6.7). We also apply a constant negative reward as a penalty to guarantee the packet drop rate.

6.2.2 Knowledge Transfer Reinforcement Learning

Here we will introduce the knowledge transfer reinforcement learning and define the map function for knowledge transfer. In reinforcement learning, to maximize the expected reward, the agent needs to improve its policy $\pi(s_t)$ to select the best action at state s_t , and arrives a new state s_{t+1} .

We define the state-action value to describe the expected reward of taking action a_t under state s_t . In Q-learning, the Q-values are updated by:

$$Q^{new}(s_t, a_t) = Q^{old}(s_t, a_t) + \alpha(r + \gamma \max_a Q(s_{t+1}, a) - Q^{old}(s_t, a_t)), \quad (6.8)$$

where Q^{old} and Q^{new} denote old and new Q-values, respectively, a_t is the action at time t , r is the reward, γ is the discount factor ($0 < \gamma < 1$), and α is the learning rate ($0 < \alpha < 1$). In Q-learning, the agent must explore the state-action space to find the optimal action sequence. This exploration usually takes a large number of iterations, because the agent needs to try huge amounts of action combinations without prior knowledge. On the contrary, in knowledge transfer reinforcement learning, we use the prior knowledge of

Algorithm 6.1 KTRA algorithm

```
1: Initialize: Wireless network parameters and Q-table of the expert.  
2: for  $TTI = 1$  to  $T$  do  
3:   for Every BS do  
4:     With probability  $\epsilon$  agent chooses actions randomly; otherwise, the agent selects  
        $a_{l,t}$  by the greedy policy.  
5:     BS allocates radio and computation resources between slices. Slices process com-  
       putation tasks by allocated MEC server capacity and allocate RBs by proportional  
       fairness algorithm.  
6:     Calculating reward based on received metrics.  
7:     Updating state  $s_{l,t}$ , find  $Q^T(\mathcal{F}(s_{l,t}), \mathcal{F}'(a_{l,t}))$ .  
8:     Updating Q-values by equation (6.9).  
9:   end for  
10: end for
```

experts to improve exploration efficiency. The Q-values are updated by:

$$Q^{new}(s_{l,t}, a_{l,t}) = Q^T(\mathcal{F}(s_{l,t}), \mathcal{F}'(a_{l,t})) + Q^{old}(s_{l,t}, a_{l,t}) + \alpha(r + \gamma \max_a Q(s_{l,t+1}, a) - Q^{old}(s_{l,t}, a_{l,t})), \quad (6.9)$$

where $s_{l,t}$ and $a_{l,t}$ are learner's state and action at time t , respectively. $Q^T(\mathcal{F}(s_{l,t}), \mathcal{F}'(a_{l,t}))$ is the mapped Q-values as an extra reward of selecting $a_{l,t}$ under state $s_{l,t}$, which aims to guide the exploration of learner agent. $\mathcal{F}$ and $\mathcal{F}'$ are state and action map functions, respectively. $Q^T(\mathcal{F}(s_{l,t}), \mathcal{F}'(a_{l,t}))$ can be generated by:

$$Q^T(\mathcal{F}(s_{l,t}), \mathcal{F}'(a_{l,t})) = Q^e(s_e, a_e), \quad (6.10)$$

where Q^e is the Q-values of expert agent, s_e and a_e are state and action of expert agent. The goal of equation(6.10) is to find a specific Q-value of expert agent to represent the potential reward of taking $a_{l,t}$ under $s_{l,t}$ in learner. Thus we need to find similar states and actions in the expert agent's Q-table. Note that the expert and learner agents have the same state definition, then we can always find a s_e that satisfies $s_{l,t} = s_e$, then $\mathcal{F}$ can be easily defined. Meanwhile, based on the definition of actions, for any $a_l = (r^{eM}, r^{UR}, c^{eM}, c^{UR})$, we consider $a_e = (r^{eM}, r^{UR})$ as a similar action for mapping, and $\mathcal{F}'$ can be defined accordingly.

The KTRA algorithm is summarized as Algorithm 1. Noting that we apply a classic proportional fairness algorithm for intra-slice radio resource allocation, because this work mainly focuses on inter-slice level scheduling, and there is no intra-slice computation resource allocation.

Algorithm 6.2 QLRA algorithm

```
1: Initialize: Wireless network parameters.
2: for  $TTI = 1$  to  $T$  do
3:   for Every BS do
4:     With probability  $\epsilon$ , the agent chooses actions randomly; otherwise, the agent selects  $a^{l,t}$  by the greedy policy.
5:     BS allocates radio and computation resources between slices. Slices process computation tasks by allocated MEC server capacity and allocate RBs by proportional fairness algorithm.
6:     Calculating reward based on received metrics.
7:     Updating state  $s_t$ .
8:     Updating Q-values by equation (6.8).
9:   end for
10: end for
```

6.2.3 Baseline Algorithms: QLRA

We apply QLRA as a baseline algorithm. Q-learning is the most generally used RL algorithm. The MDP definition of QLRA is the same as KTRA, but there is no prior knowledge. QLRA is given in Algorithm 2.

6.3 Performance Evaluation

6.3.1 Parameter Settings

We consider three different cases, including:

- **Case I:** Q-learning-based radio resource allocation. It works as an expert for Case II.
- **Case II:** KTRA based joint radio and computation resource allocation. As a learner, BSs in Case II can utilize the Q-tables of the expert agent as prior knowledge.
- **Case III:** QLRA based joint radio and computation resource allocation. It is considered a baseline algorithm without any prior knowledge of the task.

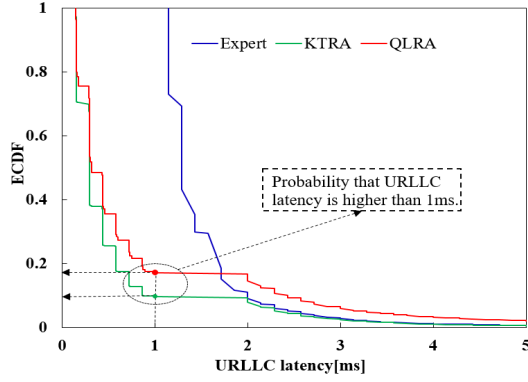
Table 6.1: Parameter settings of 5G networks and edge server

5G Networking	Computation Settings
3GPP Urban Macro network 2 OFDM symbols for each TTI Tx/Rx antenna gain: 15 dB. Number of subcarriers in each RB: 12 Subcarrier bandwidth: 15kHz Transmission power: 40 dBm Backhaul capacity: 10 Mbps	Computation capacity: 3 GHz CPU cycles required per bit: 200
	Traffic Model
	eMBB\URLLC traffic: Poisson distribution URLLC packet size: 50 Bytes eMBB packet size: 100 Bytes
	Problem Formulation
Propagation Model	eMBB\URLLC weight factor: 1
$128.1+37.6\log(\text{distance(km)})$ Log-Normal shadowing: 8 dB.	URLLC target delay: 2 ms Fixed cloud queuing delay: 1 ms
Retransmission Settings	Learning Settings
Max number of retransmissions: 1 Round trip delay: 4 TTIs	Learning rate: 0.9 Discount factor: 0.5

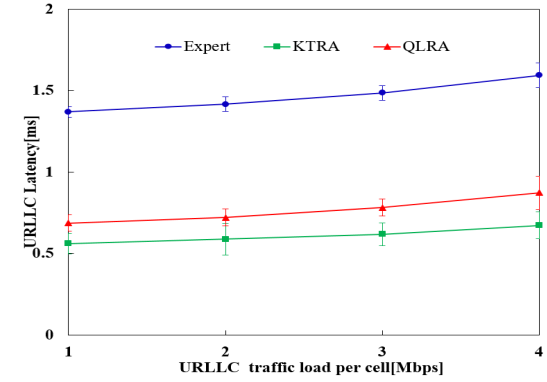
Each case contains 5 BSs with 500 m inter-site distance, and each BS is considered an independent agent to implement the proposed strategy. For example, all BSs in Case II will implement the KTRA independently to achieve their own goal. We assume each BS has one eMBB slice with 5 UEs and one URLLC slice with 10 UEs. The available bandwidth of one BS is 20 MHz, which contains 100 RBs. We assume there are 13 resource block group (RBG)s to reduce the allocation complexity. The first 12 RBGs contain 8 RBs each, while the last resource block group has 4 RBs. 200 CPU cycles are required to process 1 bit data [89]. Other parameters are shown in Table 6.1.

6.3.2 Simulation Results

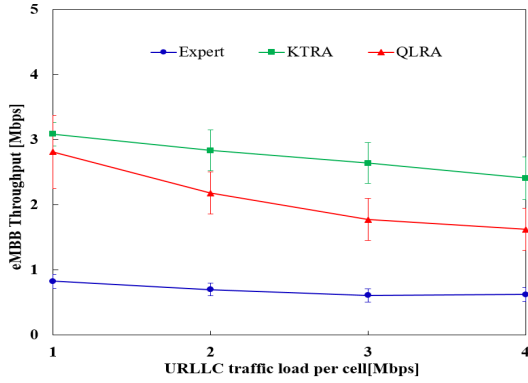
Firstly, Fig.6.3 (a) shows the empirical cumulative distribution function (ECDF) of URLLC latency of three cases under 2 Mbps eMBB and URLLC traffic, and 3 GHz MEC server per cell. The result shows that the expert case has the highest delay distribution, and the reason is that we assume MEC servers are not deployed in the expert. Thus all computation tasks



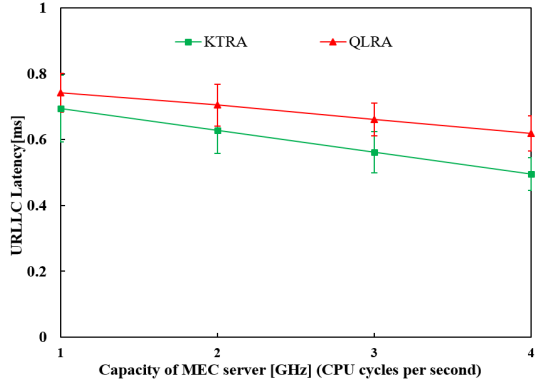
(a) URLLC latency distribution [ms].



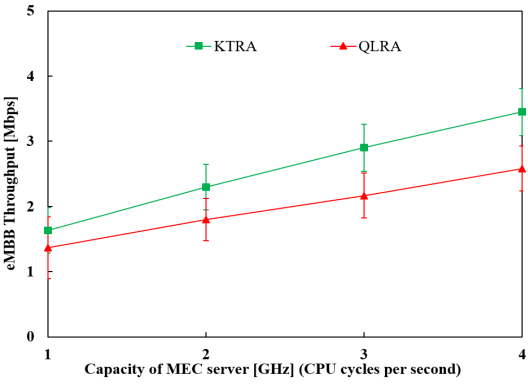
(b) Average URLLC latency [ms].



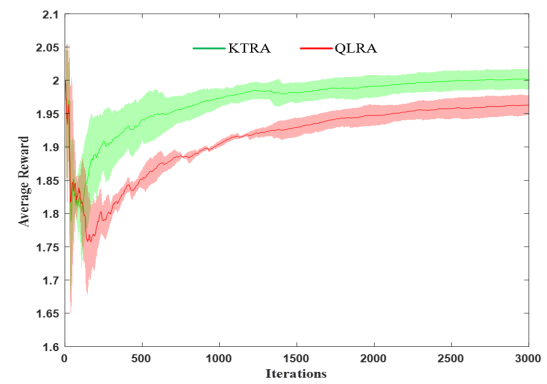
(c) eMBB throughput [Mbps] per cell .



(d) Average URLLC latency [ms] under various MEC server capacities.



(e) eMBB throughput [Mbps] per cell under various MEC server capacities.



(f) Convergence performance of KTRA and QLRA.

Figure 6.3: Simulation results comparison between KTRA and QLRA.

need to be processed in the central cloud, which leads to a higher delay. Meanwhile, the proposed KTRA method outperforms QLRA by a better delay distribution, as indicated by a lower ECDF curve in Fig.6.3 (a), and it can be explained by the knowledge transfer capability of KTRA.

Fig.6.3 (b) presents the average delay experienced by URLLC UEs, and Fig.6.3 (c) shows the average eMBB throughput per cell. Here the eMBB traffic load is fixed to 2 Mbps per cell, and URLLC traffic varies from 1 Mbps to 4 Mbps. Without MEC servers, the expert case still has the highest delay and the lowest throughput. Meanwhile, KTRA method achieves a lower URLLC delay and higher eMBB throughput than QLRA. KTRA has a 18.4% lower URLLC delay and 30.1% higher eMBB throughput under 2 Mbps URLLC traffic.

Shown by Fig.6.3 (d) and (e), we investigate the network performance under various MEC server capacities, which is indicated by CPU cycles per second. Considering the expert case has no MEC capability, we focus on the performance of KTRA and QLRA to further present the advantage of the proposed method. As expected, both algorithms have a lower URLLC delay and a higher eMBB throughput by increasing the MEC server capacity, because higher computation capacity means lower task processing delay. KTRA still outperforms QLRA by a 15.1% lower URLLC delay and a 33.8% higher eMBB throughput under 3 GHz MEC server capacity.

Furthermore, we compare the convergence performance in Fig.6.3 (f). Based on prior knowledge of the expert, KTRA has a significantly higher exploration efficiency, which is indicated by a shorter exploration period and a higher average reward. On the contrary, QLRA suffers a longer exploration phase and a lower average reward, because it needs to explore the task from scratch. To summarize, KTRA achieves a better performance in both network metrics (lower URLLC delay and higher eMBB throughput) and machine learning metrics (better convergence speed and higher average reward).

Finally, the packet drop rate of KTRA and QLRA are 0.042% and 0.048%, respectively, under 2 Mbps eMBB traffic and 4 Mbps URLLC traffic. The satisfactory packet drop rate is because we apply a penalty in both algorithms to prevent dropping packets.

Chapter 7

Deep Transfer Reinforcement Learning based Joint Radio and Cache Resource Allocation in 5G RAN Slicing

In 5G networks, network slicing is an important pillar that covers RAN, edge, transport and core networks. The evolving network architecture requires orchestrating multiple network resources such as radio and cache resources. In recent years, ML techniques have been widely applied for network management. However, most existing works do not take advantage of the knowledge transfer capability in ML. In this work, we propose a DTRL scheme for joint radio and cache resource allocation to serve 5G RAN slicing. We first define a hierarchical architecture for joint resource allocation. Then we propose two DTRL algorithms: QDTRL and ADTRL. In the proposed schemes, learner agents utilize expert agents' knowledge to improve their performance on target tasks. We assume the expert agent has been well-trained for solving source tasks, which is an existing expert with prior knowledge. The proposed algorithms are compared with the model-free exploration bonus deep Q-learning (EB-DQN) and the model-based priority priority proportional fairness and time-to-live (PPF-TTL) algorithms. Compared with EB-DQN, our proposed DTRL based method presents 21.4% lower delay for URLLC slice and 22.4% higher throughput for eMBB slice, while achieving significantly faster convergence than EB-DQN. Moreover, 40.8% lower URLLC delay and 59.8% higher eMBB throughput are observed with respect to PPF-TTL.

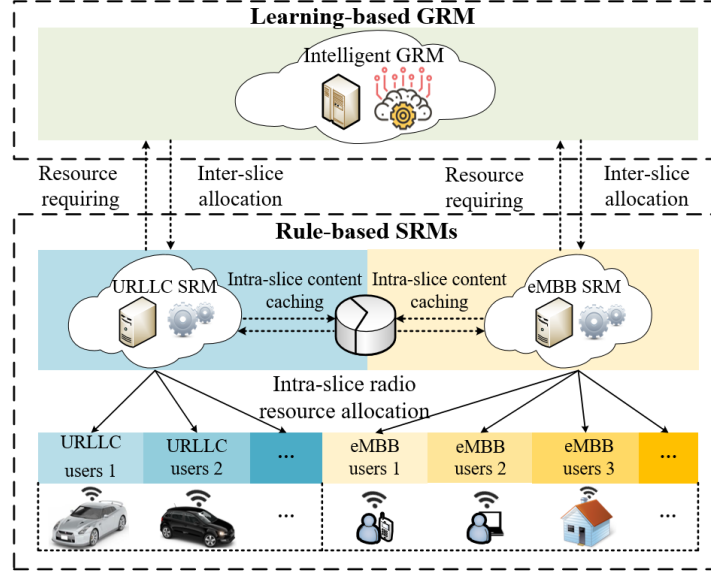


Figure 7.1: Proposed radio and cache resources allocation scheme

7.1 System Model and Problem Formulation

7.1.1 Overall Architecture

Fig.7.1 presents the proposed joint radio and cache resource allocation. A hierarchical architecture is defined for the resource allocation of cache-enabled macro cellular networks. We assume the BS has the caching capability to store content items. Our proposed scheme can be used for any number of slices without loss of generality, but here we mainly consider two typical slices, namely an eMBB slice and URLLC slice, to better illustrate the framework. Firstly, the SRMs collect the QoS requirements of associated UEs, and the collected information is sent to the GRM. Then the GRM will intelligently implement inter-slice resource allocation to divide the radio and cache resource between eMBB and URLLC SRMs. Finally, SRMs distribute RBs to associated UEs, and update cached content items within the allocated caching capacity. We apply learning-based methods to realize an intelligent GRM, while rule-based methods are deployed for SRMs. This hierarchical architecture can alleviate the burden of GRM, since the GRM only accounts for slice-level allocation instead of handling all the UEs directly.

7.1.2 Communication Model

In this section, we introduce the communication model. Firstly, the total delay consists of:

$$d_{j,m,g} = d_{j,m,g}^{tx} + d_{j,m,g}^{que} + (1 - \varpi_{j,g})d_{j,m,g}^{back}, \quad (7.1)$$

where $d_{j,m,g}^{tx}$ is the transmission delay of content item g from BS j to UE m , $d_{j,m,g}^{que}$ is the queuing delay. $d_{j,m,g}^{back}$ is the backhaul delay of fetching content items from the core network. $\varpi_{j,g}$ is a binary variable. $\varpi_{j,g} = 1$ when the content item g is cached at BS j , and $\varpi_{j,g} = 0$ otherwise. Equation (7.1) shows that the total delay is affected by both communication and cache resources. The transmission delay depends on radio resource allocation, and edge caching can prevent the backhaul delay. The scheduling efficiency will affect the queuing delay.

The transmission delay depends on the link capacity between the BS and UE:

$$d^{tx} = \frac{L_m}{E_{j,m}}, \quad (7.2)$$

where L_m is the size of content items required by UE m . $E_{j,m}$ is the link capacity between BS j and the UE m , which is calculated as follows:

$$E_{j,m} = \sum_{q \in N_j^{RB}} b_{RB} \log\left(1 + \frac{p_{j,q} x_{j,q,m} g_{j,q,m}}{b_q N_0 + \sum_{j' \in J_{-j}} p_{j',q'} x_{j',q',m'} g_{j',q',m'}}\right), \quad (7.3)$$

where N_m^R is the set of RBs that are allocated to UE m , b_{RB} is the bandwidth of one RB, N_0 is the noise power density, $p_{j,q}$ is the transmission power of the RB q of the BS j , $x_{j,q,m}$ is a binary indicator to denote whether RB q is allocated to the UE m , $g_{j,q,m}$ is the channel gain between the BS and UE, and $j' \in J_{-j}$ is the BS set except BS j .

7.1.3 Slicing-based Caching Model

This section will introduce the slicing-based caching model, in which the time-to-live (TTL) method is used as the content replacement strategy. The TTL indicates the time that a content item is stored in a caching system before it is deleted or replaced. The TTL value will be reset if this content item is required again, and thus popular content items will live longer. Although there have been many content replacement strategies, TTL is selected because: i) this work mainly focuses on inter-slice level resource allocation, and it is reasonable to apply a well-known caching method for intra-slice caching; ii) TTL

requires no prior knowledge of content popularity, which is more realistic. Nevertheless, our proposed architecture is compatible with any other caching methods without loss of generality. The complexity of the slicing-based caching model lies in effectively dividing the limited caching capacity between slices, which is far more complicated than the original TTL model. We use $\mathcal{N}_j$ to represent the slice set in the BS j , and the slice n contains $|\mathcal{M}_{j,n}|$ UEs, and each UE is denoted by m . Each slice has its own content catalog $\mathcal{G}_{j,n}$, and the variables g represents the content items ($g \in \mathcal{G}_{j,n}$). We assume all content items have the same packet size [157].

$\phi_{j,n,m,g}$ represents the request rate of UE m for the content item g , which denotes the frequency that content item g is demanded by the UE m . Then we have

$$\phi_{j,n,m,g} = p_{j,n,m,g} \phi_{j,n,m}, \quad (7.4)$$

where $\phi_{j,n,m}$ is the total request rate of UE m , and $p_{j,n,m,g}$ is the request rate distribution of UE m for content items ($\sum_{g \in \mathcal{G}} p_{j,n,m,g} = 1$).

The cache hit ratio indicates the probability of finding a content item in the cache, which can be calculated by [158]:

$$h_{j,n,g} = 1 - e^{-\sum_{m \in \mathcal{M}_{j,n}} \phi_{j,n,m,g} \mathcal{T}_{j,n,m,g}}, \quad (7.5)$$

where $h_{j,n,g}$ is the cache hit ratio of content item g in the slice n , and the TTL of the content item g will be reset to $\mathcal{T}_{j,n,m,g}$ when this content item is required. Note that the number of contents is large, and the request rate of each content item is relatively low. Based on $\lim_{x \rightarrow 0} e^x = x + 1$, we approximately have:

$$h_{j,n,g} = \sum_{m \in \mathcal{M}_{j,n}} \phi_{j,n,m,g} \mathcal{T}_{j,n,m,g}, \quad (7.6)$$

Meanwhile, the total cache hit ratio is related to the allocated storing capacity of slice n :

$$\sum_{g \in \mathcal{G}_n} \sum_{m \in \mathcal{M}_n} h_{j,n,m,g} = \frac{C_{j,n}}{C_{j,T}}, \quad (7.7)$$

where $C_{j,n}$ is the allocated storage capacity for slice n , $C_{j,T}$ is the total storage capacity of BS j , $h_{j,n,m,g}$ is the cache hit ratio of UE m for content item g . We assume the content item g has the same popularity for different UEs and thus $\mathcal{T}_{j,n,m,g} = \mathcal{T}_{j,n,g}$. Given equation (7.6) and (7.7), we have (the proof is given in the appendix A.2):

$$h_{j,n,g} = \frac{C_{j,n}}{C_{j,T}} \frac{\sum_{m \in \mathcal{M}_{j,n}} \phi_{j,n,m,g}}{\sum_{m \in \mathcal{M}_{j,n}} \phi_{j,n,m}}, \quad (7.8)$$

Equation (7.8) indicates that a higher caching capacity C_n leads to a higher cache hit ratio $h_{n,g}$ [157]. As such, popular content items, which is indicated by a higher request rate $\phi_{n,m,g}$, have a higher cache hit ratio.

Then we define a cache hit rate $\phi_{j,n,m}^{hit}$ to describe the frequency of requesting cached content items in the total request rate, which can be calculated by:

$$\phi_{j,n,m}^{hit} = \sum_{g \in \mathcal{G}_{j,n}} \phi_{j,n,m,g} h_{j,n,m,g}, \quad (7.9)$$

and the cache miss rate is:

$$\phi_{j,n,m}^{miss} = \phi_{j,n,m} - \phi_{j,n,m}^{hit}, \quad (7.10)$$

which indicates the frequency of requesting non-cached content items in the total request rate.

Finally, backhaul delay is only applied when a content item is not cached at the BS. The backhaul service is presumed to obey the M/M/1 queue, and the average delay is [159]:

$$d^{back} = \frac{1}{\frac{B}{L_m} - \sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{M}_{j,n}} (\phi_{j,n,m}^{miss})}, \quad (7.11)$$

where B is the backhaul capacity, L_m is the average size of content items, B/L_m denotes the service rate, and $\sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{M}_{j,n}} (\phi_{j,n,m}^{miss})$ is the total cache miss rate. Note that d^{back} will be infinite if the cache miss rate is higher than the service rate.

7.1.4 Problem Formulation

The objective of the eMBB slice is to maximize the total throughput, while the URLLC slice aims to minimize the average delay. It is assumed that the content catalogs of the two slices are not overlapped. To balance the requirements of the two slices, the GRM needs to jointly consider the objectives of the two slices and allocate the radio and cache

resources accordingly. For each BS, the GRM allocates radio and cache resources by:

$$\max \quad wb^{emb,avg} + (1-w)(d^{tar} - d^{urllc,avg}) \quad (7.12)$$

$$\text{s.t.} \quad (7.1) \ (7.2) \ (7.8) \ (7.9) \text{ and } (7.11) \quad (7.12a)$$

$$\sum_{n \in \mathcal{N}_j} N_{j,n}^{RB} \leq N_{j,T}^{RB} \quad (7.12b)$$

$$\sum_{n \in \mathcal{N}_j} C_{j,n} \leq C_{j,T} \quad (7.12c)$$

$$\sum_{n \in \mathcal{N}_j} \sum_{m \in \mathcal{M}_{j,n}} \phi_m^{miss} \leq \frac{B}{L_m} \quad (7.12d)$$

$$\sum_{n \in \mathcal{N}_j} \sum_{m \in \mathcal{M}_{j,n}} x_{j,q,m} \leq 1 \quad (7.12e)$$

$$\sum_{m \in \mathcal{M}_{j,n}} \sum_{q \in N_j^{RB}} x_{j,q,m} \leq N_{j,n}^{RB} \quad (7.12f)$$

$$\sum_{g \in \mathcal{G}_{j,n}} \varpi_{j,g} \leq C_{j,n} \quad (7.12g)$$

where $b^{emb,avg}$ is the total throughput of the eMBB slice, $d^{urllc,avg}$ is the average latency of the URLLC slice, and d^{tar} is the target delay. Here we use w as a weight factor in (7.12) to balance the objectives of two slices and maximize the overall objective. $\mathcal{N}_j$ is the slice set of BS j , which consists of eMBB and URLLC slices. $N_{j,n}^{RB}$ is the number of RBs that the GRM allocated to the slice n , and $N_{j,T}^{RB}$ is the total number of RBs of BS j , and the equation (7.12b) is the radio resource constraint. $C_{j,n}$ is the caching capacity of slice n , and $C_{j,T}$ is the total caching capacity in BS j , and (7.12c) ensures that the allocated caching capacity cannot exceed the upper limit. $\mathcal{M}_{j,n}$ is the set of UEs in slice n , and ϕ_m^{miss} is the miss rate of UE m , and (7.12d) denotes that the total miss rate should not exceed the backhaul service rate. $x_{j,q,m}$ has been defined in equation (7.3) as the RB allocation indicator. Equations (7.12e) and (7.12f) denote that one RB can only be allocated to one UE, and the total number of available RBs are $N_{j,n}^{RB}$ in slice n . Finally, $\varpi_{j,g}$ is a binary variable that has been defined in equation (7.1) to represent whether a content item is cached.

In the defined problem formulations, $N_{j,n}^{RB}$ and $C_{j,n}$ are the control variables of the GRM, which means the GRM only accounts for inter-slice resource allocation. $x_{j,q,m}$ and $\varpi_{j,g}$ are the control variables of SRMs. In SRMs, we apply the classic proportional fairness algorithm for intra-slice RB allocation to determine $x_{j,q,m}$, since all UEs in the same slice

Table 7.1: Summary of strategies and MDP definitions

Indices	Strategies	States	Actions	Instant Reward
Expert 1	Q-learning based radio resource allocation; No caching capability.	(s^{embb}, s^{urllc}) , where s^{embb} denotes the number of eMBB packets in the queue, and s^{urllc} is defined similarly.	(N^{embb}, N^{urllc})	$r = wr^{embb} + (1 - w)r^{urllc}$, where
Expert 2	Fixed radio resource allocation; Q-learning based caching capacity allocation.		(C^{embb}, C^{urllc})	$r^{embb} = \frac{2}{\pi} \tan^{-1}(b^{embb,avg})$ $r^{urllc} = d^{tar} - d^{urllc,avg}$.
Learner 1	DTRL-based joint radio and cache resource allocation with Q-values mapping function.		$(N^{embb}, N^{urllc}, C^{embb}, C^{urllc})$, where N^{embb} and C^{embb} denote the number of RBs and caching capacity allocated to the eMBB slice, respectively. N^{urllc} and C^{urllc} are defined similarly for the URLLC slice.	w is the weighting factor, r^{embb} and r^{urllc} are the rewards of eMBB and URLLC slices, respectively. $b^{embb,avg}$ and $d^{urllc,avg}$ are average throughput of eMBB slice and average delay of URLLC slice. d^{tar} is the target URLLC delay.
Learner 2	DTRL-based joint radio and cache resource allocation with action selection mapping function.			
Baseline ¹	EB-DQN for joint radio and cache resource allocation without prior knowledge.			

[1] We include PPF-TTL as the second baseline. However, since PPF-TTL is not a ML-based method, it is excluded from this table.

are presumed to be equally important. Meanwhile, cached content items are updated by TTL rule, which will determine $\varpi_{j,g}$.

7.2 Deep Transfer Reinforcement Learning based Resource Allocation

7.2.1 Overall Framework

In this section, we introduce the DTRL based inter-slice resource allocation, where each BS is considered an independent agent to make decisions autonomously. As shown in Table 7.1, five learning-based strategies are deployed. We assume experts 1 and 2 apply Q-learning for radio and cache resource allocation, respectively. Experts are only good at one of radio or cache resource allocation, but they have no multi-task knowledge. Then learners 1 and 2 can utilize various knowledge from experts to improve their performance on joint radio and cache resource allocation. Based on different mapping functions, we propose two DTRL based methods, namely QDTRL and ADTRL. Finally, we apply EB-DQN as a learning-based benchmark, and PPF-TTL method as a model-based baseline. In the following, we will introduce the experts, learners and baselines.

7.2.2 Q-learning based Experts

In this section, we assume the expert agents have learning experience on one specific task, but they have no knowledge of other tasks. For expert 1, it uses Q-learning for the radio resource allocation, and there is no caching capability. For expert 2, RBs are allocated by the numbers of UEs in each slice, and Q-learning is used for the caching capacity allocation.

To transform the problem formulation in equation (7.12) to the RL context, we first define the MDP $\langle S, A, T, R \rangle$ for experts, where S is the state set, A is the action set, T is the transition probability, and R is the reward function. The MDP definitions of experts are given below:

- **State:** In this work, we intend to coordinate the performance of various slices by inter-slice resource allocation. As such, the state definition should reflect the transmission demand of each slice. The states of expert 1 and 2 are both defined by (s^{embb}, s^{urllc}) , which indicates the number of packets waiting in the queues of eMBB and URLLC slices, respectively.
- **Action of expert 1:** The expert 1 only implements radio resource allocation, and consequently the action (N^{embb}, N^{urllc}) denotes the number of RBs allocated to eMBB and URLLC slices.
- **Action of expert 2:** In expert 2, the learning strategy is only applied for caching capacity allocation, and the action (C^{embb}, C^{urllc}) indicates the caching capacity allocated to eMBB and URLLC slices.
It is worth noting that the actions $N^{embb}, N^{urllc}, C^{embb}$ and C^{urllc} have been defined as control variables in the problem formulation (7.12), and they are transformed into actions here to serve the Q-learning scheme.
- **Reward function:** The reward functions of experts 1 and 2 are defined by the objectives of slices:

$$r = wr^{embb} + (1 - w)r^{urllc}, \quad (7.13)$$

where r^{embb} and r^{urllc} are rewards of eMBB and URLLC slices, respectively, and w is the weight factor.

For the eMBB slice, obtaining higher throughput leads to a higher reward, and we have.

$$r^{embb} = \frac{2}{\pi} \tan^{-1}(b^{embb, avg}) \quad (7.14)$$

where $b^{embb,avg}$ is the average throughput of the eMBB slice, and we apply the $\tan^{-1}$ function to normalize the reward ($0 < r^{embb} < 1$).

For the URLLC slice, a lower delay means a higher reward:

$$r^{urllc} = d^{tar} - d^{urllc,avg}, \quad (7.15)$$

where d^{tar} and $d^{urllc,avg}$ are target and achieved average delays for the URLLC slice, respectively.

Moreover, to guarantee the constraints of the problem formulation (7.12), we apply penalties to the reward when these constraints are violated.

Expert 1 and 2 apply Q-learning for optimization, which has been introduced in former Chapters. Then the state-action values of experts 1 and 2 are updated by:

$$Q^{new}(s_e, a_e) = Q^{old}(s_e, a_e) + \alpha(r_e + \gamma \max_a Q(s'_e, a) - Q^{old}(s_e, a_e)) \quad (7.16)$$

where $Q^{old}(s_e, a_e)$ and $Q^{new}(s_e, a_e)$ are old and new Q-values, s_e and s'_e are current and next states of experts, respectively, a_e is the action, r_e is the reward, and α is the learning rate ($0 < \alpha < 1$). By updating the Q-values iteratively, experts will learn the optimal action selections to achieve the best accumulated reward.

7.2.3 Deep Transfer Reinforcement Learning based Learners

Given the prior knowledge of experts, learners are expected to solve more complicated problems with higher state-action space. To apply the transfer reinforcement learning, we need to define the MDP by $(S, A, R, T, \mathcal{F})$, where $\mathcal{F}$ is the mapping function. In this section, we first define the states, actions and rewards of learners, then we introduce the DDQN based learning policy. Finally, we propose two DTRL based algorithms, namely QDTRL and ADTRL, and the main difference lies in the mapping function definitions.

The states, actions and reward functions of the two learners are given below:

- **State and reward function:** As shown in Table 7.1, we assume learners have the same state and reward definitions as experts. The main reason is that transfer learning is designed for tasks that share similarities with existing source tasks. As such, similar state and reward definitions can reduce the complexity of defining mapping functions, which can better transfer knowledge from experts to learners.

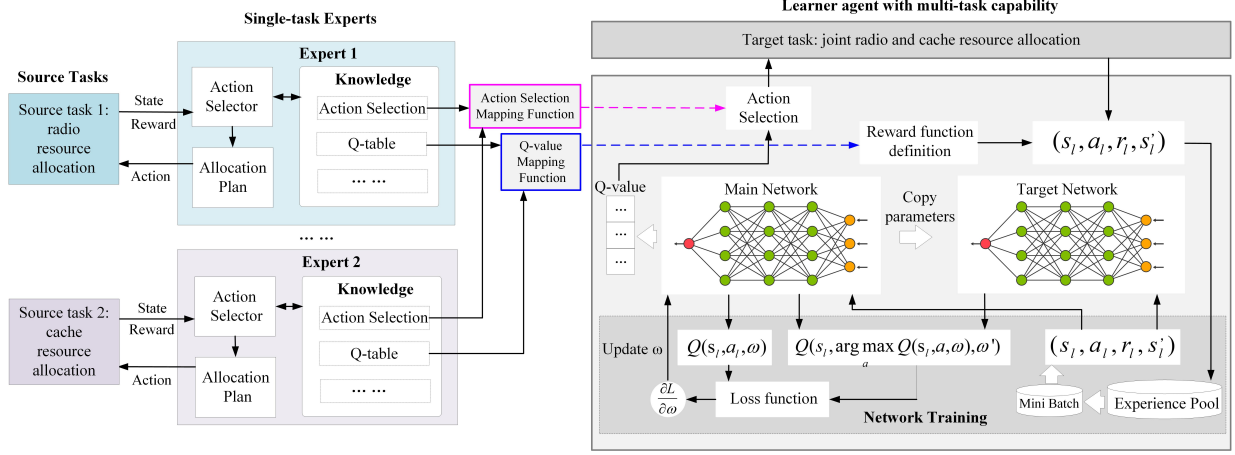


Figure 7.2: Proposed DTRL architecture for resource management.

- **Action of the learner 1 and 2:** Compared with experts, the learners have to jointly consider the radio and cache resource allocation, and then the actions are defined by $(N^{emb}, N^{urllc}, C^{emb}, C^{urllc})$. Compared with allocating one single resource, the joint resource allocation problem is much more complicated, especially when multiple slices are involved.

Based on MDP definitions, we introduce our DTRL algorithm. Conventional DQN may suffer an over-estimation issue, which has been introduced in former Chapters. To this end, DDQN is used to decouple the action selection and evaluation. The loss function of DDQN is defined as:

$$L(w) = Er(r_l + \gamma Q(s'_l, \arg \max_a Q(s'_l, a, w), w') - Q(s_l, a_l, w)) \quad (7.17)$$

where s_l and a_l are the state and action of the learner, respectively, s'_l is the next state, r_l is the reward, and Er is the error function. w is the weight of the main network, which will predict current Q-values $Q(s_l, a_l, w)$. w' is the weight of the target network. Here we use the notation l to indicate the learner. The main network chooses actions by $a_l = \arg \max_a Q(s'_l, a, w)$, and the target network evaluates the action by $Q(s'_l, a_l, w')$. By decoupling the action selection and evaluation, DDQN can prevent overestimation and better predict Q-values than DQN.

In this work, we deploy the DDQN architecture in the proposed DTRL. For the learner agent shown by color grey in Fig.7.2, an action a_l is first selected and sent to the target task environment. Then a tuple (s_l, a_l, r_l, s'_l) will be received from the environment, which

will be saved in the experience pool. The learner agent samples a random minibatch from the experience pool. For every tuple (s_l, a_l, r_l, s'_l) , the main network predicts $Q(s_l, a_l, w)$ and selects actions by $a = \arg \max_a Q(s'_l, a, w)$. The target network evaluates the action by $Q(s'_l, a, w')$. Then we utilize the loss function shown by equation (7.17) for gradient descent to update the weight w of the main network. After several training sessions, the target network will copy the weight parameters of the main network. Such a late update of the target network serves as a stable reference for the main network training. Here we deploy the LSTM network as hidden layers for main and target networks, which has been introduced in former Chapters.

Finally, it is worth noting that we include two different mapping functions in the proposed DTRL scheme. The Q-value mapping function will affect the reward calculation of the learner agent (indicated by the pink line in Fig.7.2), while the action selection mapping function influences the action selection (shown by the blue line in Fig.7.2). Accordingly, we propose two DTRL based methods, namely QDTRL and ADTRL, and in the following we will introduce these two mapping functions and corresponding algorithms.

1) Learner 1: Q-value based Deep Transfer Reinforcement Learning

In QDTRL, the Q-values of experts are presumed to be the prior knowledge of learner 1, which can guide the exploration efficiency of this learner [160]. The loss function in QDTRL is defined by:

$$L(w) = Er(\sigma_1 Q^E(\mathcal{F}(s_l), \mathcal{F}'(a_l)) + r_l + \gamma Q(s'_l, \arg \max_a Q(s'_l, a, w), w') - Q(s_l, a_l, w)) \quad (7.18)$$

where $\mathcal{F}$ and $\mathcal{F}'$ are state and action mapping functions, respectively. Compared with equation (7.17), the main difference is that $\sigma_1 Q^E(\mathcal{F}(s_l), \mathcal{F}'(a_l))$ is involved as an extra reward of selecting action a_l under state s_l . σ_1 is the transfer learning rate, which describes the importance of prior knowledge ($0 \leq \sigma_1 \leq 1$). A higher transfer learning rate means the prior knowledge utilization is more important than its learning process, while a lower value indicates the reverse.

In equation (7.18), we apply $\sigma_1 Q^E(\mathcal{F}(s_l), \mathcal{F}'(a_l))$ to guide the action selection of learner. The main idea is that the learner is encouraged to select actions with higher Q-values of the experts, which is indicated by a higher extra reward in the equation (7.18). Considering the similarities of tasks, these actions are expected to bring similar high rewards for the learner. However, due to different state-action spaces, the learner cannot directly utilize the Q-values of experts; thus, a function is needed to map experts' Q-values to the learner's Q-table. Q^E term in equation (7.18) is generated by:

$$Q^E(\mathcal{F}(s_l), \mathcal{F}'(a_l)) = Q_{e,1}(s_{e,1}, a_{e,1}) + Q_{e,2}(s_{e,2}, a_{e,2}), \quad (7.19)$$

where $Q_{e,1}$, $s_{e,1}$ and $a_{e,1}$ are the Q-value, state and action of expert 1, respectively. $Q_{e,2}$, $s_{e,2}$ and $a_{e,2}$ are defined similarly for expert 2. The idea behind the mapping functions $\mathcal{F}$ and $\mathcal{F}'$ is to find the states and actions of the experts that are similar to s_l and a_l . $\mathcal{F}$ and $\mathcal{F}'$ are defined by:

- **State mapping $\mathcal{F}$:** For a given state s_l , considering experts and learner 1 have the same state definition, we can always find $s_l = s_{e,1} = s_{e,2}$. Thus $\mathcal{F}$ can be easily defined.
- **Action mapping $\mathcal{F}'$:** The goal of $\mathcal{F}'$ is to find $(a_{e,1}, a_{e,2}) = \mathcal{F}'(a_l)$. For any action a_l , which is defined as $a_l = (N^{emb}, N^{urllc}, C^{emb}, C^{urllc})$, it can be decomposed into the combination of $a_{e,1} = (N^{emb}, N^{urllc})$ and $a_{e,2} = (C^{emb}, C^{urllc})$. Then $\mathcal{F}'$ can be defined accordingly.

Based on the state and action mapping relationships, if given s_l and a_l , we can always find specific $Q_{e,1}(s_{e,1}, a_{e,1})$ and $Q_{e,2}(s_{e,2}, a_{e,2})$. Then $Q^E(\mathcal{F}(s_l), \mathcal{F}'(a_l))$ can be directly used by learner 1. The defined Q-value mapping function is summarized by Fig.7.3 from steps 1 to 4. First, for any given (s_l, a_l) , we find state action pairs $(s_{e,1}, a_{e,1})$ and $(s_{e,2}, a_{e,2})$ by state mapping function $\mathcal{F}$ and action mapping function $\mathcal{F}'$. Then, we extract $Q_{e,1}(s_{e,1}, a_{e,1})$ and $Q_{e,2}(s_{e,2}, a_{e,2})$ from expert agents' Q-tables. After that, we generate $Q^E(\mathcal{F}(s_l), \mathcal{F}'(a_l))$ by equation (7.19), which will be considered as extra rewards when selecting a_l under s_l . This extra reward is added to r_l , and the new tuple $(s_l, a_l, r_l + \sigma Q^E(\mathcal{F}(s_l), \mathcal{F}'(a_l)), s'_l)$ will be saved in the experience pool. Finally, we implement the gradient descent by equation (7.18) for the network training.

2) Learner 2: Action Selection based Deep Transfer Reinforcement Learning

Learner 2 has the same MDP definition as learner 1, but the mapping function is different. In this work, the learners have a bigger action space than experts for the joint radio and cache resource allocation. Indeed, a huge action space lowers the exploration efficiency and results in a longer convergence time. To this end, we propose an ATDRL method to improve the exploration efficiency in learner 2.

It is worth noting that TL is mainly applied to target tasks related to source tasks. Specifically, it requires similarities between source and target MDPs. Then, we first introduce the Kantorovich distance $\mathcal{K}(\mathcal{D})(Y, Z)$ to describe the similarities between two

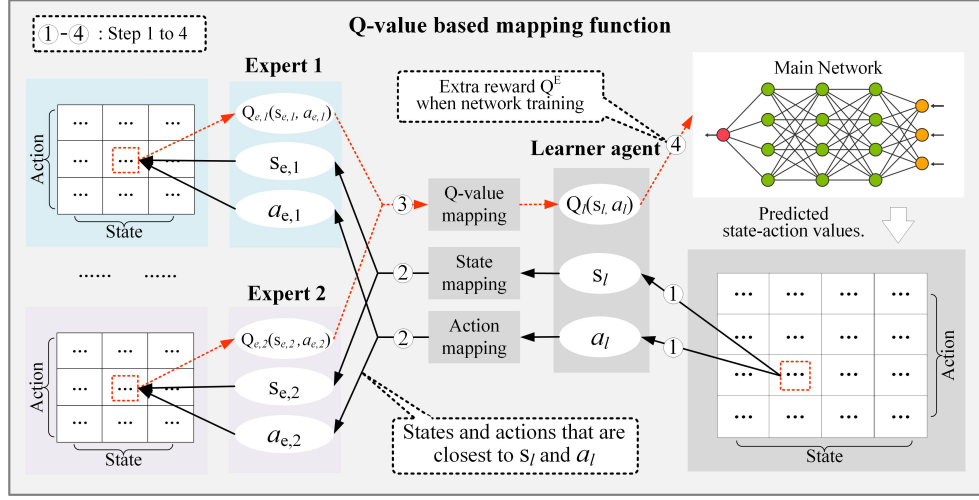


Figure 7.3: Proposed Q-value mapping function for deep transfer reinforcement learning.

distributions [161]:

$$\begin{aligned}
 \max_{u_f, f=1,2,\dots,|S|} \quad & \sum_{f=1}^{|S|} (Y(s_f) - Z(s_f))u_f \\
 \text{s.t.} \quad & u_f - u_k \leq \mathcal{D}(s_f, s_k) \quad f, k = 1, 2, \dots, |S|. \\
 & 0 \leq u_f \leq 1
 \end{aligned} \tag{7.20}$$

where Y and Z are two probability distributions of $s_f \in S$, u_f and u_k are internal optimization variables, and $\mathcal{D}(s_f, s_k)$ denotes a metric $\mathcal{D}$ to assess the distance between s_f and s_k . $\mathcal{K}(\mathcal{D})(Y, Z)$ shows the distance of two probability distributions Y and Z with the metric $\mathcal{D}$ in state set S . However, in TL, we focus more on the distance metric of different state sets, and $\mathcal{K}(\mathcal{D})(Y, Z)$ is rewritten by [162]:

$$\begin{aligned}
 \max_{u_f, f=1,2,\dots,|S_1|; v_k, k=1,2,\dots,|S_2|} \quad & \sum_{e=1}^{|S_1|} Y(s_f)u_f - \sum_{f=1}^{|S_2|} Z(s_k)v_k \\
 \text{s.t.} \quad & u_f - v_k \leq \mathcal{D}(s_f, s_k) \\
 & -1 \leq u_f \leq 1
 \end{aligned} \tag{7.21}$$

where S_1 and S_2 are two state sets with $s_f \in S_1$ and $s_k \in S_2$, respectively, and $\mathcal{K}(\mathcal{D})(Y, Z)$ evaluates the distance between state set S_1 and S_2 .

Then we include two MDPs $\langle S_e, A_e, \mathcal{T}_e, R_e \rangle$ and $\langle S_l, A_l, \mathcal{T}_l, R_l \rangle$ to identify the difference of their state-action pairs. The lax bisimulation metric is introduced to evaluate the distance between two state-action pairs [163]:

$$\mathcal{D}_{\sim}((s_e, a_e), (s_l, a_l)) := \theta_1 |r_e(s_e, a_e) - r_l(s_l, a_l)| + \theta_2 \mathcal{K}(\mathcal{D}'(Y(s_e, a_e), Z(s_l, a_l))) \quad (7.22)$$

where θ_1 and θ_2 are weight factors. The first term $|r_e(s_e, a_e) - r_l(s_l, a_l)|$ represents the reward distance, and $\mathcal{K}(\mathcal{D}'(Y(s_e, a_e), Z(s_l, a_l)))$ is the Kantorovich metric for state-action pairs under semi-metric $\mathcal{D}'$. $\mathcal{D}'$ is defined by the Hausdorff metric:

$$\mathcal{D}'(s_e, s_l) = \max(\max_{a_e \in A_e} \min_{a_l \in A_l} \mathcal{D}((s_e, a_e), (s_l, a_l)), \min_{a_e \in A_e} \max_{a_l \in A_l} \mathcal{D}(s_e, a_e), (s_l, a_l)) \quad (7.23)$$

Hausdorff distance is the maximum distance from one set to the nearest point of the other set [164], and here we define $\mathcal{D}'(s_e, s_l)$ to measure the distance between action sets A_e and A_l under state s_e and s_l .

To evaluate the potential optimality of selecting a_l under s_l , we include the Bellman optimality to find the state value difference:

$$\begin{aligned} & |Q_l^*(s_l, a_l) - V_e^*(s_e)| \\ &= |Q_l^*(s_l, a_l) - Q_e^*(s_e, \pi^*(s_e))| \\ &= |Q_l^*(s_l, a_l) - Q_e^*(s_e, a_e^*)| \\ &= |(r_l(s_l, a_l) + \gamma \sum_{s'_l \in S_l} Y(s'_l | s_l, a_l) V_l(s'_l)) - (r_e(s_e, a_e^*) + \gamma \sum_{s'_e \in S_e} Z(s'_e | s_e, a_e^*) V_e(s'_e))| \\ &\leq |r_l(s_l, a_l) - r_e(s_e, a_e^*)| + \gamma |\sum_{s'_l \in S_l} Y(s'_l | s_l, a_l) V_l(s'_l) - \sum_{s'_e \in S_e} Z(s'_e | s_e, a_e^*) V_e(s'_e)| \\ &\leq |r_l(s_l, a_l) - r_e(s_e, a_e^*)| + \max_{a_l \in A_l} \min_{a_e \in A_e} (\gamma |\sum_{s'_l \in S_l} Y(s'_l | s_l, a_l) V_l(s'_l) - \sum_{s'_e \in S_e} Z(s'_e | s_e, a_e^*) V_e(s'_e)|) \\ &\quad \text{(Using equation (7.21), (7.23))} \\ &= |r_e(s_e, a_e^*) - r_l(s_l, a_l)| + \gamma \mathcal{K}(\mathcal{D}'((s_e, a_e^*), (s_l, a_l))) \\ &\quad \text{(Setting } \theta_1 = 1 \text{ and } \theta_2 = \gamma \text{ in equation (7.22))} \\ &= \mathcal{D}_{\sim}((s_l, a_l), (s_e, a_e^*)) \end{aligned} \quad (7.24)$$

where $Q_l^*(s_l, a_l)$ is the optimal state-action value of (s_l, a_l) , $V_e^*(s_e)$ is the optimal state value of s_e , s'_l is the next state of s_l , and $Y(s'_l | s_l, a_l)$ is the probability of arriving s'_l by implementing a_l under s_l . Here we use $a_e^* = \pi^*(s_e)$ to represent the action selection of the expert agent. Equation (7.24) shows that there is an upper bound between the state-action

pairs (s_l, a_l) and (s_e, a_e^*) . Following, we will introduce how to utilize equation (7.24) to improve the action selection of the learner.

When selecting an action, we usually consider $V_l^*(s_l)$ as a target value for $Q_l^*(s_l, a_l)$, and we have $V_l^*(s_l) = \arg \max_{a_l} Q^*(s_l, a_l)$. Then $V_l^*(s_l) - Q_l^*(s_l, a_l)$ can be used to evaluate the potential optimality of a_l by:

$$\begin{aligned}
& V_l^*(s_l) - Q_l^*(s_l, a_l) \\
&= Q_l^*(s_l, \pi^*(s_l)) - Q_l^*(s_l, a_l) \\
&= Q_l^*(s_l, a_l^*) - Q_l^*(s_l, a_l) \\
&= |Q_l^*(s_l, a_l^*) - Q_l^*(s_l, a_l)| \\
&= |Q_l^*(s_l, a_l^*) - V_e^*(s_e) + V_e^*(s_e) - Q_l^*(s_l, a_l)| \\
&\leq |Q_l^*(s_l, a_l^*) - V_e^*(s_e)| + |V_e^*(s_e) - Q_l^*(s_l, a_l)| \\
&\leq \mathcal{D}_\sim((s_l, a_l^*), (s_e, a_e^*)) + \mathcal{D}_\sim((s_l, a_l), (s_e, a_e^*)) \\
&\quad (\text{By using equation (7.24)})
\end{aligned} \tag{7.25}$$

Equation (7.25) gives an upper bound for the potential optimality of selecting a_l under s_l , and then a lower bound $O_l(s_l, a_l)$ can be easily found by:

$$\begin{aligned}
& Q_l^*(s_l, a_l) - V_l^*(s_l) \\
&= Q_l^*(s_l, a_l) - Q_l^*(s_l, \pi^*(s_l)) \\
&\geq -\mathcal{D}_\sim((s_l, a_l^*), (s_e, a_e^*)) - \mathcal{D}_\sim((s_l, a_l), (s_e, a_e^*)) \\
&= O_l(s_l, a_l),
\end{aligned} \tag{7.26}$$

In $O_l(s_l, a_l)$, note that $\mathcal{D}_\sim((s_l, a_l^*), (s_e, a_e^*))$ will not affect a_l selection, since it is a constant value for given s_l and s_e . Considering we have two experts in this work, we rewrite $O_l(s_l, a_l)$ by:

$$\begin{aligned}
O_l(s_l, a_l) &= -\sigma_2(\mathcal{D}_\sim((s_l, a_l), (s_{e,1}, a_{e,1}^*))) \\
&\quad + \mathcal{D}_\sim((s_l, a_l), (s_{e,2}, a_{e,2}^*))) - (1 - \sigma_2)(\mathcal{D}_\sim((s_l, a_l^*), (s_{e,1}, a_{e,1}^*))) + \mathcal{D}_\sim((s_l, a_l^*), (s_{e,2}, a_{e,2}^*))),
\end{aligned} \tag{7.27}$$

where σ_2 is the transfer learning rate of ADTRL ($0 \leq \sigma_2 \leq 1$). If $\sigma_2 = 0$, then $O_l(s_l, a_l)$ becomes a constant value for all $a_l \in A_l$, which means transferred knowledge will not affect the action selection of the learner. By contrast, if $\sigma_2 = 1$, $O_l(s_l, a_l)$ totally depends on the lax bisimulation metric between (s_l, a_l) and (s_e, a_e^*) , which indicates the learner will imitate the action selection of experts.

In summary, given $\langle S_e, A_e, T_e, R_e \rangle$ as a source MDP and $\langle S_l, A_l, T_l, R_l \rangle$ as a target MDP, $O_l(s_l, a_l)$ defines the lower bound of potential optimality of a_l in terms of the distance between (s_l, a_l) and (s_e, a_e^*) . Here s_e is considered as the source state that is closest to s_l . In this work, we assume experts and learners have the same state definitions, and s_e can be easily found accordingly for any s_l . Finally, the probability of choosing a_l is given by:

$$Pr(a_l|s_l) = \frac{Sig(O_l(s_l, a_l))}{\sum_{a \in A_l} sig(O_l(s_l, a))} \quad (7.28)$$

where Sig denotes the Sigmoid function for normalization. Equation (7.28) means that actions with higher potential optimality have a higher chance of being selected, and consequently it will improve the exploration efficiency. QDTRL and ADTRL are summarized in Algorithm 1 and 2, respectively.

7.2.4 Baseline: Exploration bonus-DQN and PPF-TTL

In this section, we include two baseline algorithms. Firstly, EB-DQN serves as a benchmark to compare our DTRL method with other ML-based algorithms. The MDP definition of EB-DQN is the same as DTRL, which is shown by Table 7.1. EB-DQN agent explores the joint resource allocation task from scratch, and no prior knowledge is included. In EB-DQN, the loss function is defined by:

$$L(w) = Er(r + \frac{\Psi}{\sqrt{\psi(s, a)}} + \gamma \max_a Q(s', a, w') - Q(s, a, w)) \quad (7.29)$$

where Ψ is an extra reward, and $\psi(s, a)$ is the number of times that (s, a) is selected. $\frac{\Psi}{\sqrt{\psi(s, a)}}$ is regarded as a bonus for selecting actions that are less visited, and it encourages the agent to better explore the environment. EB-DQN based joint resource allocation is summarized in Algorithm 3.

On the other hand, to compare the ML methods with model-based algorithms, we apply a model-based PPF-TTL algorithm. The well-known PPF algorithm is applied for radio resource allocation, in which URLLC packets have a higher priority than eMBB packets [73]. The RBs will first serve URLLC transmission, then eMBB traffic will be processed. We deploy the TTL method for caching, but no slicing and learning are included. The PPF-TTL method is shown in Algorithm 4.

7.2.5 Computational Complexity Analyses

Moreover, we analyze the computational complexity of the proposed DTRL-based methods. Firstly, the complexity of the DTRL method is dominated by the training and updating of the LSTM network. The complexity of the LSTM network updating consists of the running time of recurrent connections and bias, and the updating time of input and output nodes. The computational complexity for updating the LSTM network in DTRL is $\mathcal{O}(l_{hd}m_{lstm}^2c_{lstm}^2)$, where l_{hd} is the number of hidden layers, c_{lstm} is the number of memory cells in each block and m_{lstm} is the number of memory blocks. It is worth noting that only the main network needs to be trained, and the target network can copy the weight from the main network.

On the other hand, the knowledge transfer process also contributes to the complexity. In QDTRL, the knowledge transfer consists of the state and action mapping functions $\mathcal{F}$ and $\mathcal{F}'$. Accordingly, the time complexity is $\mathcal{O}(\sum_{q=1}^{N^e}|S_{e,q}||A_{e,q}|)$, where $|N^e|$ is the number of experts, $S_{e,q}$ and $A_{e,q}$ are state and action sets of experts, respectively.

In ADTRL, the Kantorovich distance can be considered as an optimal transportation problem, which is computable within a strong polynomial time $\mathcal{O}(|S|^2\log(|S|))$, where $|S|$ is the total number of possible distributions [165]. Meanwhile, the Hausdorff metric can be solved in a nearly linear time [166], which can be neglected compared with the complexity of Kantorovich distance. Based on equation (7.26), the total complexity of knowledge transfer in ADTRL is $\mathcal{O}(|N_e||A_l||S|^2\log(S))$, where $|A_l|$ is the action set of the learner. In summary, these analyses show that our knowledge transfer process can be efficiently computed, and the complexity is linearly related to the number of experts.

Algorithm 7.1 QDTRL-based joint resource allocation

```

1: Initialize: Wireless and QDTRL parameters
2: for  $TTI = 1$  to  $T^{total}$  do
3:   for Each BS do
4:     With probability  $\epsilon$ , selecting actions randomly; otherwise, choosing actions by  $a_l = \arg \max_a Q(s'_l, a, w)$ .
5:     GRM implements inter-slice resource allocation.
6:     SRMs distribute radio resource to UEs by proportional fairness, and replace cached content items.
7:     Updating system state, and saving  $(s_l, a_l, r_l, s'_l)$  to the experience pool. Every  $C$  TTIs, sampling a minibatch from the experience pool randomly.
8:     Find  $\mathcal{F}(s_l)$  and  $\mathcal{F}'(a_l)$  for any  $s_l$  and  $a_l$  in the minibatch.
9:     Generating target Q-values  $Q^{Tar}(s_l, a_l) =$ 

$$\begin{cases} r_l & \\ \sigma_1 Q^E(\mathcal{F}(s_l), \mathcal{F}'(a_l)) + r_l + \gamma Q(s'_l, \arg \max_a Q(s'_l, a, w), w') & \end{cases}$$

10:    Updating  $w$  using gradient descent by minimizing the loss  $Loss(w) = Er(Q^{Tar}(s_l, a_l) - Q(s_l, a_l, w))$ .
11:    Copying  $w$  to  $w'$  after several training.
12:   end for
13: end for
14: Output: Network and learning algorithm performance.

```

Algorithm 7.2 ADTRL-based joint resource allocation

```

1: Initialize: Wireless and ADTRL parameters
2: for  $TTI = 1$  to  $T^{total}$  do
3:   for Each BS do
4:     With probability  $\epsilon$ , selecting action  $a_l$  by:

```

$$Pr(a_l|s_l) = \frac{Sig(O_l(s_l, a_l))}{\sum_{a \in A_l} O_l(s_l, a)}$$

```

5:   Otherwise, choosing  $a_l$  by  $\arg \max_a Q(s'_l, a, w)$ .
6:   GRM implements inter-slice resource allocation.
7:   SRMs distribute radio resource to UEs by proportional fairness, and replace cached content items.
8:   Updating system state, and saving  $(s_l, a_l, r_l, s'_l)$  to the experience pool. Every  $C$  TTIs, sampling a minibatch from the experience pool randomly.
9:   Generating target Q-values  $Q^{Tar}(s_l, a_l) =$ 

$$\begin{cases} r_l & \text{if done} \\ r_l + \gamma Q(s'_l, \arg \max_a Q(s'_l, a, w), w') & \text{else} \end{cases}$$

10:  Updating  $w$  using gradient descent by minimizing the loss  $Loss(w) = Er(Q^{Tar}(s_l, a_l) - Q(s_l, a_l, w))$ .
11:  Copying  $w$  to  $w'$  after several training.
12:  end for
13: end for
14: Output: Network and learning algorithm performance.

```

Algorithm 7.3 EB-DQN-based joint resource allocation

- 1: **Initialize:** Wireless and EB-DQN parameters
 - 2: **for** $TTI = 1$ to T^{total} **do**
 - 3: **for** Each BS **do**
 - 4: With probability ϵ , choose actions randomly; otherwise, choosing actions by $a_l = \arg \max_a Q(s'_l, a, w)$.
 - 5: GRM implements inter-slice resource allocation.
 - 6: SRMs distribute radio resource to UEs by proportional fairness, and replace cached content items by TTL.
 - 7: Updating system state, and saving (s_l, a_l, r_l, s'_l) to the experience pool. Every C TTIs, sampling a minibatch from the experience pool randomly.
 - 8: Generating target Q-values $Q^{Tar}(s_l, a_l) =$

$$\begin{cases} r_l + \frac{\Psi}{\sqrt{\psi(s,a)}} \\ r_l + \frac{\Psi}{\sqrt{\psi(s,a)}} + \gamma \arg \max_a Q(s'_l, a, w') \end{cases}$$
 - 9: Updating w using gradient descent by minimizing the loss $Loss(w) = Er(Q^{Tar}(s_l, a_l) - Q(s_l, a_l, w))$.
 - 10: Copying w to w' after several training.
 - 11: **end for**
 - 12: **end for**
 - 13: **Output:** Performance of the network and the learning algorithm.
-

Algorithm 7.4 PPF-TTL based joint resources allocation

- 1: **Initialize:** Wireless networks parameters
 - 2: **for** $TTI = 1$ to T^{total} **do**
 - 3: **for** Each BS **do**
 - 4: **for** Each RB **do**
 - 5: Calculating the estimated transmission rate of UEs in the queue.
 - 6: Calculating proportional fairness metric.
 - 7: Transmitting URLLC packets with the highest proportional fairness. If no URLLC packet, then transmitting eMBB packets.
 - 8: **end for**
 - 9: if *done* BS replaces cached content items by time-to-live rule.
 - 10: **end for**
 - 11: **end for**
 - 12: **Output:** Performance of the network.
-

7.3 Performance Evaluation

7.3.1 Parameter Settings

In this section, we consider six different cases: expert 1, expert 2, learner 1 (QDTRL), learner 2 (ADTRL), baseline 1 (EB-DQN) and baseline 2 (PPF-TTL). We include 6 adjacent gNBs, and each algorithm is implemented in one gNB randomly. Each gNB contains an eMBB slice and a URLLC slice. The eMBB slice has 5 UEs, while the URLLC slice has 10 UEs. The radius of each gNB is 300 meters, and the distance between two adjacent

Table 7.2: Parameters settings of 5G networks, cache capacity, and learning algorithm

5G settings	Cache settings
Bandwidth: 20MHz 3GPP urban macro network Number of RBs: 100 Subcarriers in each RB: 12 Subcarrier bandwidth: 15kHz Transmission power: 40 dBm (uniform distributed) TTI size: 2 OFDM symbols Tx/Rx antenna gain: 15 dB.	Caching capacity: 20 contents TTL value: 50 TTIs Contents library size: 40/slice Content popularity: Zipf
	Traffic model
	URLLC/ eMBB traffic: Poisson distribution Packet size: 36 Bytes
	Learning settings
Retransmission settings	Network layers: 4
Max number of retransmissions: 1 Round trip delay: 4 TTIs Hybrid automatic repeat request.	2 LSTM network hidden layers hidden layer has 35 nodes Initial learning rate: 0.005 Experience pool size: 150
UE and gNBs	Training frequency: 30 TTIs
25 eMBB UE, 50 URLLC UE UE random distribution Number of gNBs: 5 Inter-gNB distance: 500m	Minbatch size: 30 Discount factor: 0.5 Epsilon value: 0.05 Reward weight: 0.5
Propagation model	Transfer learning rate: 0.7 Ψ value for EB-DNQ: 0.5
128.1+37.6log(distance(km)) Log-Normal shadowing: 8 dB.	

gNBs is 600 meters. Each gNB has 100 RBs in total, divided into 13 RBGs. We assume the caching capacity is reallocated every 50 TTIs because replacing the cached content items takes time. The experience of experts is presumed to be existing knowledge for learners.

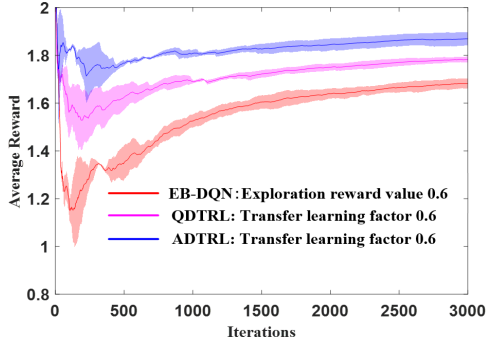
We deploy LSTM networks with 30 nodes as hidden layers for the target and main networks in DTRL and EB-DQN. The network learning rate and the number of layers are selected by the grid search method. We try different parameter combinations and find the best performance accordingly. The simulations include 3000 TTIs, where the first 1500 TTIs are the exploration period, and the remaining TTIs are the exploitation period. The simulations are implemented in MATLAB 5G library with 15 runs and a 95% confidence interval. Other 5G and learning parameters are shown in Table 7.2.

7.3.2 Performance Analyses of Various Learning Parameters

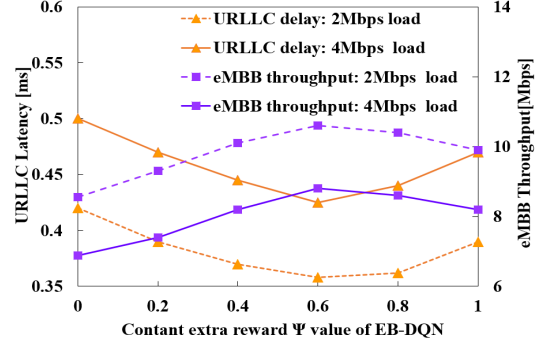
In this section, we analyze the algorithm performance under diverse learning parameters. Fig.7.4 (a) shows the convergence performance of QDTRL, ADTRL and EB-DQN, which is a critical metric for learning algorithms. ADTRL has the fastest convergence, which can be explained by the improved action selection strategy. ADTRL takes advantage of the action selection policies of experts, which indicates actions with higher potential rewards. It proves that ADTRL applies a more efficient exploration strategy and performs better. QDTRL also presents a better convergence performance than EB-DQN. In QDTRL, the Q-values of experts are extracted as extra rewards for action selections. It assumes that actions with higher Q-values in experts can also bring higher rewards to learners, and the exploration is accelerated. However, other actions can still be randomly selected for exploration, lowering the exploration efficiency. On the contrary, in EB-DQN, the agent has no prior knowledge about target tasks. The agent starts from scratch to explore its tasks, which leads to a longer convergence time and a lower average reward.

Then, Fig.7.4 (b) shows the network performance of EB-DQN against the extra exploration reward Ψ (shown in equation (7.29)). A higher Ψ value will encourage more explorations, while a lower value means more exploitation. The simulations demonstrate that a higher Ψ may hamper the network performance by over-exploration, and a lower Ψ also degrades the URLLC delay and eMBB throughput by under-exploration. Therefore, an appropriate Ψ value is critical to balance the exploration and exploitation.

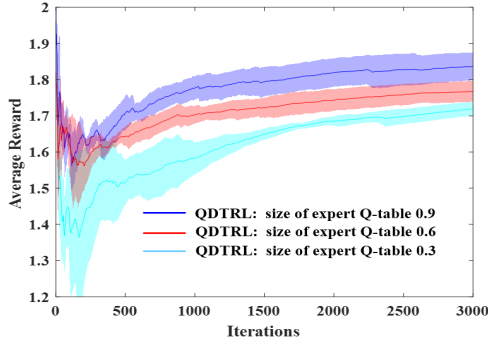
To investigate how the knowledge transfer can contribute to the learner agent performance, Fig. 7.4 (c) shows the QDTRL performance under various expert Q-table sizes. In particular, a lower value such as 0.3 means that the learner agent only has 30% of the expert Q-tables as prior knowledge. Fig. 7.4 (c) demonstrates that more prior knowledge can



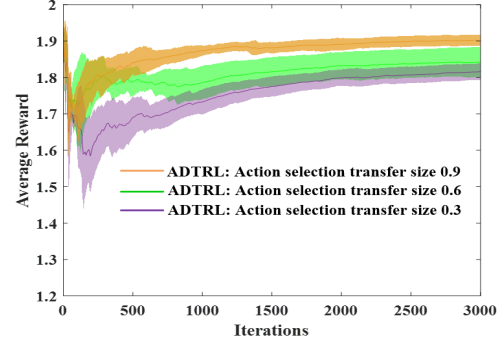
(a) Convergence performance comparison of EB-DQN, QDTRL and ADTRL



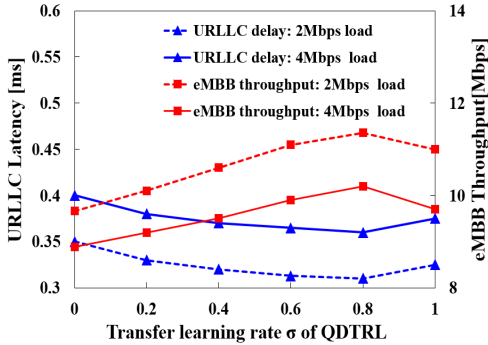
(b) Network performance of EB-DQN under various extra exploration rewards Ψ .



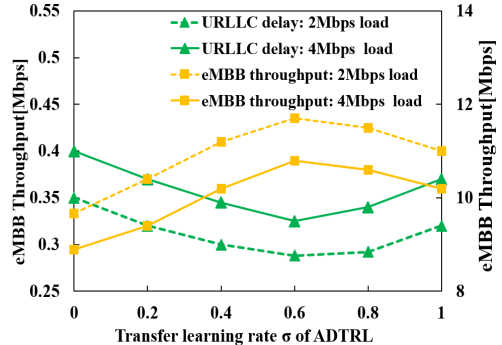
(c) Convergence of QDTRL under various Q-table sizes of expert agents



(d) Convergence of ADTRL under various action selection knowledge sizes of expert agents



(e) Network performance of QDTRL under various transfer learning rates



(f) Network performance of ADTRL under various transfer learning rates

Figure 7.4: Convergence and network performance comparison.

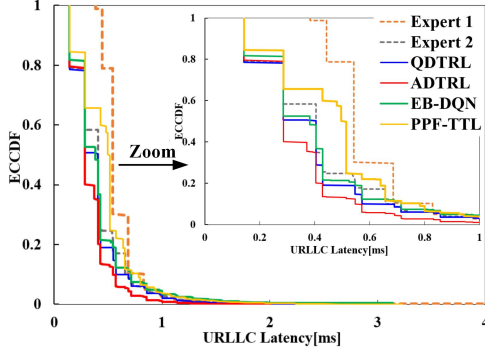
bring better performance for the learner agent, while partial prior knowledge may produce lower average rewards. Similarly, Fig. 7.4 (d) presents the ADTRL performance using different action selection transfer sizes. Specifically, a lower value indicates that the learner agent only has part of the action selection knowledge of expert agents. Consequently, one can observe that more transferred knowledge can improve exploration efficiency and produce a higher average reward for the learner agent.

Finally, note that we have defined transfer learning rates when introducing QDTRL and ADTRL, representing the importance of transferred knowledge. The network performance of QDTRL and ADTRL under different transfer learning rates is investigated here. In QDTRL, the transfer learning rate is indicated by the σ_1 in equation (7.19). Fig.7.4 (c) and (d) show that a higher transfer learning rate may lead to better network performance, which is indicated by a lower URLLC delay and a higher eMBB throughput. However, a very high transfer learning rate may affect the exploration of the agent itself, and it leads to sub-optimal results such as higher delays and lower throughput. A similar trend can be observed in Fig.7.4 (e) and (f) for ADTRL, in which the transfer learning rate is indicated by variable σ_2 in equation (7.27). A higher σ_2 value significantly reduces the exploration complexity and brings better network performance. However, when $\sigma_2 \geq 0.6$, the transferred knowledge dominates the learning process, resulting in performance degradation in terms of latency and throughput.

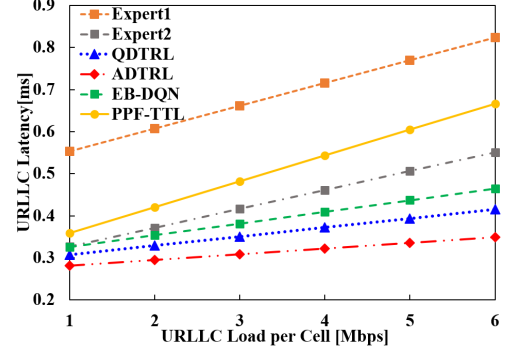
7.3.3 Network Performance Analyses

In this section, we compare the network performance of different algorithms under various traffic loads and backhaul capacities. The eMBB traffic is fixed to 1 Mbps per cell, and the URLLC traffic ranges from 1 to 6 Mbps. We first present the results, then explain the performance of each algorithm.

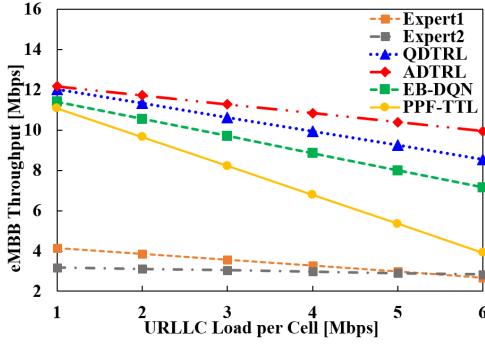
Fig.7.5 (a) shows the Empirical Complementary Cumulative Distribution Function (EC-CDF) of the URLLC latency with 2 Mbps URLLC traffic, which presents empirical distributions of packet delays. We zoom the area where the ECCDF value is higher than 0.1 and the URLLC delay is lower than 1 ms to better show the results. Expert 1 and PPF-TTL method present the highest delay, which is indicated by the high delay distribution in 0.1-1 interval of the ECCDF axis. By contrast, expert 2 has a lower delay distribution than expert 1. Meanwhile, EB-DQN and ADTRL show comparable delay performance for the URLLC slice. Finally, ADTRL achieves the best delay performance than other algorithms. Fig.7.5 (b) and (c) present average URLLC slice delay and eMBB slice throughput against traffic loads. It shows that both ADTRL and QDTRL maintain lower delay and higher



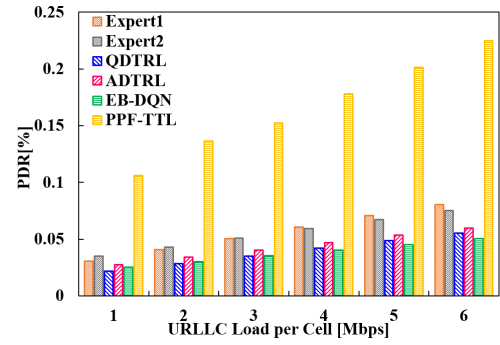
(a) ECDF of URLLC latency (1 Mbps eMBB traffic, 2 Mbps URLLC traffic)



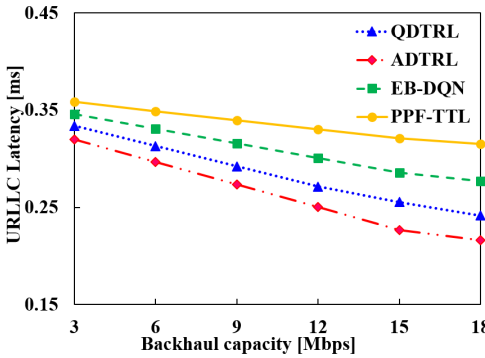
(b) URLLC latency against traffic load



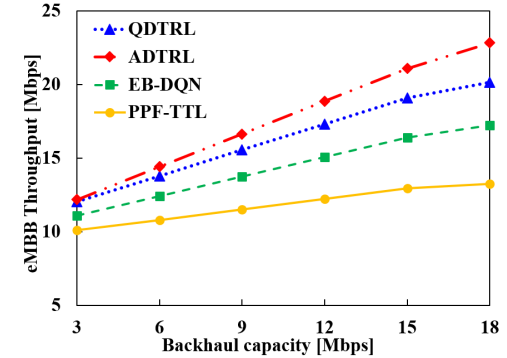
(c) eMBB throughput against traffic load



(d) Packet drop rate comparison against traffic load



(e) URLLC latency against backhaul capacity



(f) eMBB throughput against backhaul capacity

Figure 7.5: Performance comparison under various traffic load and backhaul capacity.

throughput than experts and baseline algorithms under various traffic loads. Expert 2 shows a lower URLLC delay than PPF-TTL and expert 1, but its eMBB throughput is much lower than any other algorithm. In Fig.7.5 (d), all algorithms have comparable packet drop rate except the PPF-TTL. Following, we will explain the results of each algorithm.

We first analyze the experts' performance. Expert 1 shows a high delay and a low throughput, because it has no caching capability. All packets required by UEs have to be processed by the core network, and the constant backhaul delay leads to a high URLLC delay and a low eMBB throughput. On the contrary, expert 2 has a lower delay because we apply a fixed RB allocation strategy. In particular, RBs are distributed according to the UE numbers in each slice; thus, the URLLC slice always has more RBs, which leads to a low URLLC delay. But the eMBB slice is affected by a low eMBB throughput.

For the baseline algorithms, EB-DQN outperforms the PPF-TTL method because it jointly learns the radio and cache resource allocation. The eMBB throughput of PPF-TTL method decreases significantly with the increasing URLLC load, which results from the priority settings in this method. In the PPF method, whenever a new URLLC packet arrives, it is directly scheduled over the eMBB packets, which unavoidably degrades the eMBB throughput.

Finally, the proposed QDTRL and ADTRL have the best overall performance. Due to the novel DTRL-based scheme, they can leverage the knowledge of experts and further improve their performance on new tasks. When the URLLC traffic is 4 Mbps, ADTRL presents a 21.4% lower URLLC delay and a 22.4% higher eMBB throughput than EB-DQN. A 40.8% lower URLLC delay and a 59.8% higher eMBB throughput are also observed compared with PPF-TTL method. The simulations show that the proposed DTRL-based solutions achieve more promising results than the baseline algorithms.

Moreover, backhaul capacity is one of the main bottlenecks of 5G RAN. Here we investigate the network performance under various backhaul capacities, and the results are shown in Fig.7.5 (e) and (f). Note that the basic performance of experts has been shown in former results, and here we mainly compare the DTRL-based solutions with baseline algorithms.

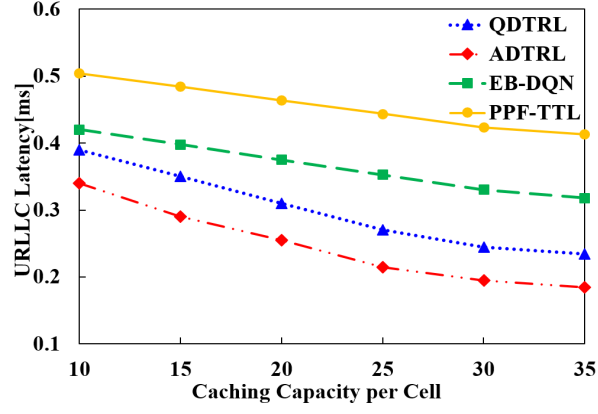
As expected, all algorithms achieve lower delays for URLLC slice and higher throughput for eMBB slice with increasing backhaul capacity, because higher capacity will reduce the backhaul delay. Moreover, QDTRL and ADTRL still achieve lower URLLC delays and higher eMBB throughput. Compared with EB-DQN, the satisfactory performance of QDTRL and ADTRL can still be explained by their knowledge transfer strategy. Meanwhile, the worst performance of PPF-TTL shows that learning-based methods outperform model-based algorithms by the superior learning capability. When the backhaul capacity

is 15 Mbps, ADTRL presents an 18.9% lower URLLC delay and a 24.2% higher eMBB throughput than EB-DQN. Compared with the PPF-TTL method, a 24.7% lower URLLC delay and a 54.3% higher eMBB throughput are observed.

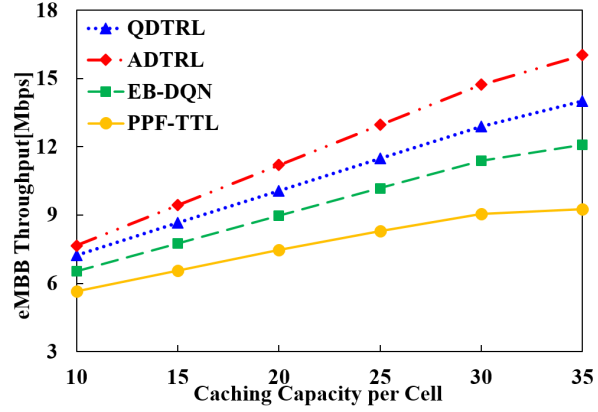
7.3.4 Content Caching Performance Analyses

In this section, we compare different algorithms under various caching capacities, indicated by the maximum number of content items stored in gNBs. As shown in Fig.7.6 (a) and (b), a higher caching capacity will reduce the URLLC latency and increase the eMBB throughput. It is because a higher caching capacity means that more items can be cached in gNBs, and the average backhaul delay will be reduced. The simulations show that ADTRL and QDTRL have the best overall performance. EB-DQN still achieves a lower URLLC delay and a higher eMBB throughput than PPF-TTL algorithm.

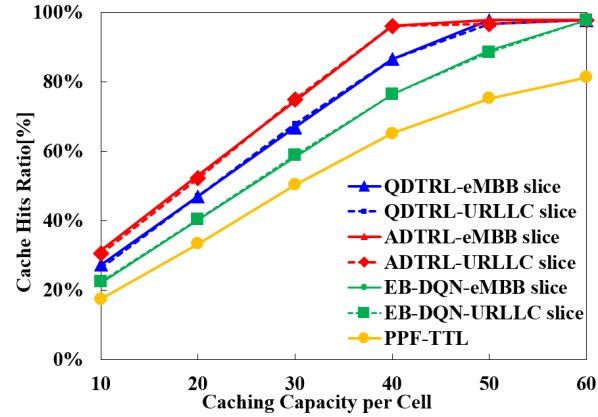
Furthermore, we present the cache hit ratio of eMBB and URLLC slices in Fig.7.6 (c). The cache hit ratio represents the proportion of packets that can be found in the cache server when they are required. A higher cache hit ratio usually indicates better network performance, which is affected by the caching capacity and the content replacement strategy. With the increasing caching capacity, cache hit ratios naturally increase for all algorithms. As expected, the cache hit ratio of the eMBB and URLLC slices are well maintained in QDTRL and ADTRL. Compared with ADTRL, the ratios in EB-DQN and PPF-TTL methods are 19.8% and 31% lower. Note that PPF-TTL method only has one curve because we assume there is no slicing in PPF-TTL algorithm. Finally, ADTRL, QDTRL and EB-DQN have similar cache hit ratios when the caching capacity is 60. It means that most required items can be cached with this capacity, and then a high cache hit ratio is observed.



(a) URLLC latency against caching capacity



(b) eMBB throughput against caching capacity



(c) Cached hit ratio against caching capacity

Figure 7.6: Network performance comparison under various caching capacities.

Chapter 8

Cooperative Hierarchical Deep Reinforcement Learning based Joint Sleep, Power, and RIS Control for Energy-Efficient HetNet

EE is one of the most important metrics for 5G and future 6G networks to reduce energy costs and control carbon footprint. Sleep control, as a cost-efficient approach, can significantly lower power consumption by switching off network devices selectively. Meanwhile, the RIS has emerged as a promising technique to enhance the EE of 5G beyond and 6G networks. In this work, we jointly consider sleep and transmission power control for RIS-aided energy-efficient HetNet. In particular, we first propose a FP method for RIS phase-shift control, which aims to maximize the sum-rate under given transmission power levels. Then, considering the timescale difference between sleep control and power control, we introduce a Co-HDRL algorithm, including a cross-entropy enabled meta-controller for sleep control, and correlated equilibrium-based sub-controllers for power control. Moreover, we proposed a surrogate optimization method as one baseline for RIS control, and conventional HDRL as another baseline for sleep and power control. Finally, simulations show that the RIS-assisted sleep control can achieve more than 16% lower energy consumption and 30% higher energy efficiency than baseline algorithms.

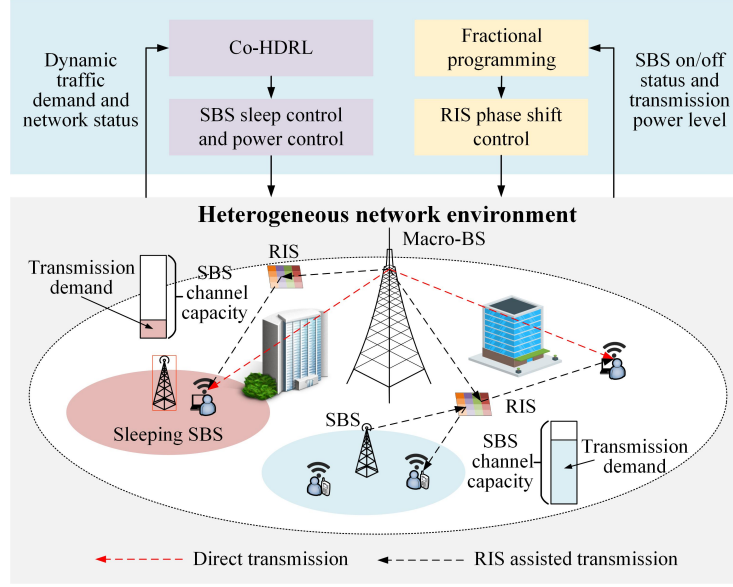


Figure 8.1: RIS-aided heterogeneous network.

8.1 Network and System Model

As shown in Fig. 8.1, we consider a multi-BS and multi-RIS HetNets environment. Firstly, the small base station (SBS)s can enter sleep mode when the transmission demand drops, which will reduce energy consumption and increase overall energy efficiency. Then the macro base station (MBS) will take over the active UEs that are previously covered by sleeping SBSs. Meanwhile, SBS can adjust its transmission power dynamically to achieve the desired SINR for attached UEs and reduce interference on other SBSs. This work proposes a Co-HDRL method for joint sleep and power control of SBSs.

On the other hand, the direct transmission between BSs and UEs may suffer high penetration loss due to dense buildings. Then, RISs are deployed to reshape the signal transmission path from BS to UEs and increase the SINR. Note that one RIS can be shared by several BSs, which means the RIS control will simultaneously affect the performance of several BSs, and we propose an FP-based algorithm for the RIS phase-shift control. In this work, we first apply Co-HDRL method to decide the on/off status and transmission power of SBSs, and then the transmission power will be used by FP-based algorithm for RIS phase-shift optimization.

8.1.1 RIS-Assisted Channel Model

The direct transmissions between BS and UEs are usually considered as non-line-of-sight (NLOS) with a high penetration loss caused by dense buildings and complicated propagation environments, which is a commonly considered assumption due to the RIS application environment. Consequently, we assume UEs mainly receive signals by the indirect RIS-assisted transmission that consists of BS-RIS and RIS-UE links [167].

RIS elements are usually deployed on the surface of high buildings, and then the BS-RIS link is considered as a line-of-sight (LOS) link:

$$\mathbf{H}_{b,m} = g_{b,m}[\mathbf{h}_{b,m,1}, \dots, \mathbf{h}_{b,m,n}, \dots, \mathbf{h}_{b,m,|\mathcal{N}_m|}], \quad (8.1)$$

where $g_{b,m}$ is the path loss from b^{th} BS to m^{th} RIS, $\mathcal{N}_m$ is the total number of elements of m^{th} RIS. The phase difference $\mathbf{h}_{b,m,n}$ is given by $\mathbf{h}_{b,m,n} = \exp\left(\frac{-2j\pi d_{b,m,n}}{\lambda}\right)$, where j is the imaginary unit, $d_{b,m,n}$ is the distance from b^{th} BS to m^{th} RIS, λ is the signal wavelength.

Then, the signal will be reflected by RIS to UEs by a phase shift matrix:

$$\mathbf{\Theta}_m = \omega_m \text{diag}(\theta_{m,1}, \dots, \theta_{m,n}, \dots, \theta_{m,\mathcal{N}_m}), \quad (8.2)$$

where diag indicates the diagonal operator, ω_m is the amplitude reflection coefficient of m^{th} RIS, and $\omega_m \in [0, 1]$. $\theta_{m,n} = e^{j\theta'_{m,n}}$ is the phase shift of n^{th} RIS elements with $\theta'_{m,n} \in \left\{0, \frac{2\pi}{2^\mu}, \dots, \frac{2\pi(2^\mu-1)}{2^\mu}\right\}$, and μ is the resolution of the RIS element's phase shifter.

Considering the complex environment on the UE side, the RIS-UE link follows the Rician fading, which is a combination of LOS and NLOS transmissions [168]:

$$\mathbf{G}_{m,k} = \sqrt{\frac{\tau_{m,k}}{\tau_{m,k} + 1}} \bar{\mathbf{G}}_{m,k} + \sqrt{\frac{1}{\tau_{m,k} + 1}} \tilde{\mathbf{G}}_{m,k}, \quad (8.3)$$

where $\tau_{m,k}$ is the Rician factor between m^{th} RIS and k^{th} UE. $\bar{\mathbf{G}}_{m,k}$ is the deterministic LOS component, which can be calculated in a similar way as (8.1). $\tilde{\mathbf{G}}_{m,k}$ is the NLOS component, which is given by independent and identically distributed complex Gaussian distribution.

For a downlink transmission between BS j and UE k , the achievable rate is:

$$C_{b,k} = b_k \log \left(1 + \frac{P_b \left| \sum_{m \in \mathcal{M}} \mathbf{H}_{b,m} \mathbf{\Theta}_m \mathbf{G}_{m,k}^\dagger \right|^2}{\sum_{b' \in \mathcal{B}_{-b}} P_{b'} \left| \sum_{m' \in \mathcal{M}} \mathbf{H}_{b',m'} \mathbf{\Theta}_{m'} \mathbf{G}_{m',k}^\dagger \right|^2 + N_0^2} \right), \quad (8.4)$$

where b_k is the bandwidth allocated to UE k , $\mathbf{G}_{m,k}^\dagger$ represents the conjugate transpose of vector $\mathbf{G}_{m,k}$, N_0^2 is the noise power, and P_b is the transmission power of b^{th} BS. $\mathcal{B}_{-b}$ denotes the set of BSs except b^{th} BS, and $\mathcal{M}$ is the set of RISs in the environment. There have been a few advanced methods to estimate the channel state information (CSI) in RIS-aided networks, e.g., sending pilot signals and using sensing devices at RISs. Therefore, here we assume perfect CSI sharing in the network, which is a widely used assumption in many existing works.

8.1.2 Energy Consumption Model

We assume only SBS can enter the sleep mode, and the energy consumption of the b^{th} SBS is [169]:

$$E_b = \begin{cases} P_{active} + \delta_{BS}P_b, & \text{if } q_b = 1; \\ P_{sleep}, & \text{if } q_b = 0, \end{cases} \quad (8.5)$$

where $q_b = 1$ indicates active mode, and $q_b = 0$ means sleep mode. P_{active} is the constant energy consumption of SBS in active mode, δ_{BS} is the slope of load-dependent energy consumption factor, P_b is the transmission power, and P_{sleep} is the SBS energy consumption in sleep mode.

8.1.3 Problem Formulation

The overall objective is to maximize the overall EE, and the problem formulation is given by:

$$\max_{q_b, P_b, \Theta_m} \frac{\sum_{b \in \mathcal{B}} \sum_{k \in \mathcal{K}_b} D_{b,k}}{\sum_{b \in \mathcal{B}} E_b} - \phi \sum_{b \in \mathcal{B}} n_b \quad (8.6)$$

$$\text{s.t.} \quad E_b = \begin{cases} P_{active} + \delta_{BS}P_b, & \text{if } q_b = 1; \\ P_{sleep}, & \text{if } q_b = 0, \end{cases} \quad (8.6a)$$

$$|\theta_{m,n}|^2 = 1, m \in \mathcal{M}, n \in \mathcal{N}_m, \quad (8.6b)$$

$$P_b \leq P_{max}. \quad (8.6c)$$

where $D_{b,k}$ is the throughput of k^{th} UE associated with b^{th} BS. Note that the throughput level $D_{b,k}$ depends on the user traffic demand and available channel capacity, which means that BS can adjust the transmission power dynamically to save energy consumption when user's traffic demand is off-peak. Therefore, such definitions can better adapt to wireless

network dynamics. n_b is a binary variable: $n_b = 1$ means the b^{th} BS is overloaded; otherwise $n_b = 0$. Here, overloading indicates the transmission demand has exceeded the BS channel capacity, and consequently attached UEs may experience a long queuing delay. We define ϕ as a penalty factor to avoid overloading and guarantee network performance. A lower ϕ value means that the system may degrade the user experience to save energy consumption and improve EE, while a higher ϕ value indicates more power consumption to guarantee service quality. Therefore, the value of ϕ depends on the network operator's preference and specific environment. Finally, The power consumption of RIS elements is not included here because it is much lower than BS power in practice [167].

The defined problem (8.6) includes three control variables q_b , P_b , and Θ_m . First, the binary variable q_b in equation (8.5) indicates the sleep control of SBSs, which will affect the energy consumption $\sum_{b \in \mathcal{B}} E_b$ in the objective. Second, the BS transmission power level P_b in equation (8.4) will affect both energy consumption and channel capacity. Finally, the channel capacity also depends on the RIS phase shift Θ_m , which is given by equation (8.4). In summary, we aim to jointly consider the sleep control, transmission power control, and RIS phase-shift control to optimize the objective defined in equation (8.6). Problem (8.6) is highly non-convex and non-linear, and the binary sleep control decision q_b indicates integer control variables, which may be NP-hard to solve. In addition, the sleep control decision has to consider dynamic traffic load level and channel conditions, and such uncertainty increases the difficulty of solving problem (8.6) directly. Therefore, we decouple this problem into two sub-problems, RIS phase-shift control and joint sleep and transmission power control, which will be introduced in the following sections.

8.2 Fractional Programming based RIS Phase-shift Control

In this section, we introduce the proposed FP-based RIS phase control. Based on equation (8.4), the SINR of UE k is:

$$\psi_k = \frac{P_b \left| \sum_{m \in \mathcal{M}} \mathbf{H}_{b,m} \Theta_m \mathbf{G}_{m,k}^\dagger \right|^2}{\sum_{b' \in \mathcal{B}_{-b}} P_{b'} \left| \sum_{m' \in \mathcal{M}} \mathbf{H}_{b',m'} \Theta_{m'} \mathbf{G}_{m',k}^\dagger \right|^2 + N_0^2} \quad (8.7)$$

The objective of RIS phase-shift control is to maximize the total data rate for all UEs,

which is given by:

$$\max_{\mathbf{\Theta}_m} \quad f_1(\mathbf{\Theta}_m) = \sum_{k \in \mathcal{K}} b_k \log(1 + \psi_k) \quad (8.8)$$

$$\text{s.t.} \quad |\theta_{m,n}|^2 = 1, m \in \mathcal{M}, n \in \mathcal{N}_m, \quad (8.8a)$$

where $\mathcal{K}$ is the set of UEs, and b_k is the bandwidth allocated to UE k . In equation (8.8), note that the control variable is $\mathbf{\Theta}_m$, and the BS transmission power P_b depends on the Co-HDRL algorithm, which will be introduced in the next section.

Proposition 1: The problem (8.8) is equivalent to [170]:

$$\begin{aligned} \max_{\mathbf{\Theta}_m, \boldsymbol{\beta}} f_2(\mathbf{\Theta}_m, \boldsymbol{\beta}) &= \sum_{k \in \mathcal{K}} \left(b_k \log(1 + \beta_k) - b_k \beta_k + \frac{b_k(1 + \beta_k)\psi_k}{1 + \psi_k} \right) \\ \text{s.t.} & (8.8a), \end{aligned} \quad (8.9)$$

where $\boldsymbol{\beta} = [\beta_1, \beta_2, \dots, \beta_{|\mathcal{K}|}]$ is the auxiliary variables given by Lagrangian dual transform.

Proof: See Appendix A.3.

Given equation (8.9), we take the ratio term out of the logarithm by Lagrangian transform. To solve the problem (8.9), we apply an iterative method to update the $\mathbf{\Theta}_m$ and $\boldsymbol{\beta}$ alternatively. First, given $\mathbf{\Theta}_m$, setting $\frac{\partial f_2}{\partial \beta_k} = 0$ claims $\beta_k^* = \psi_k$. Then, given β_k^* , optimizing $\mathbf{\Theta}_m$ means:

$$\begin{aligned} \max_{\mathbf{\Theta}_m} \quad f_3(\mathbf{\Theta}_m) &= \sum_{k \in \mathcal{K}} \frac{b_k(1 + \beta_k)\psi_k}{1 + \psi_k} \\ &= \sum_{k \in \mathcal{K}} \frac{b_k(1 + \beta_k)P_b \left| \sum_{m \in \mathcal{M}} \mathbf{H}_{b,m} \mathbf{\Theta}_m \mathbf{G}_{m,k}^\dagger \right|^2}{\sum_{b' \in \mathcal{B}} P_{b'} \left| \sum_{m' \in \mathcal{M}} \mathbf{H}_{b',m'} \mathbf{\Theta}_{m'} \mathbf{G}_{m',k}^\dagger \right|^2 + N_0^2} \\ \text{s.t.} \quad & (8.8a), \end{aligned} \quad (8.10)$$

For ease of notation, we define $\hat{\boldsymbol{\theta}}_m$ by:

$$\hat{\boldsymbol{\theta}}_m = [\theta_{m,1}, \theta_{m,2}, \dots, \theta_{m,N_m}] \in \mathbb{C}^{1 \times N}, \quad (8.11)$$

and $\sqrt{P_b} \mathbf{H}_{b,m} \mathbf{\Theta}_m \mathbf{G}_{m,k}^\dagger$ can be easily transformed to $\omega_m \hat{\boldsymbol{\theta}}_m \text{diag}(\mathbf{H}_{b,m}) \mathbf{G}_{m,k}^\dagger \sqrt{P_b}$. For notation brevity, we further define $\mathbf{v}_{b,m,k} = \omega_m \text{diag}(\mathbf{H}_{b,m}) \mathbf{G}_{m,k}^\dagger \sqrt{P_b}$. Then, equation (8.10) can

be rewritten by:

$$\begin{aligned}
\max_{\hat{\Theta}} f_3(\hat{\Theta}) &= \sum_{k \in \mathcal{K}} \frac{b_k(1 + \beta_k) \left| \sum_{m \in \mathcal{M}} \hat{\theta}_m^\dagger \mathbf{v}_{b,m,k} \right|^2}{\sum_{b' \in \mathcal{B}} \left| \sum_{m' \in \mathcal{M}} \hat{\theta}_{m'}^\dagger \mathbf{v}_{b',m',k} \right|^2 + N_0^2} \\
&= \sum_{k \in \mathcal{K}} \frac{b_k(1 + \beta_k) |\hat{\Theta} \mathbf{V}_{b,k}|^2}{\sum_{b' \in \mathcal{B}} |\hat{\Theta} \mathbf{V}_{b',k}|^2 + N_0^2} \\
&\text{s.t. (8.8a),}
\end{aligned} \tag{8.12}$$

where $\hat{\Theta} = [\hat{\theta}_1^\dagger, \hat{\theta}_2^\dagger, \dots, \hat{\theta}_{|\mathcal{M}|}^\dagger]$ represents all the RIS phase control variables, and $\mathbf{V}_{b,k} = [\mathbf{v}_{b,1,k}, \mathbf{v}_{b,2,k}, \dots, \mathbf{v}_{b,|\mathcal{M}|,k}]$.

Proposition 2: Based on quadratic transformation, equation (8.12) is equivalent to [170]:

$$\begin{aligned}
\max_{\hat{\Theta}, \eta_k} f_4(\hat{\Theta}, \eta_k) &= \sum_{k \in \mathcal{K}} (2\sqrt{b_k(1 + \beta_k)} \text{Re}\{\eta_k^\dagger (\hat{\Theta} \mathbf{V}_{b,k})\} - \eta_k^\dagger (\sum_{b' \in \mathcal{B}} |\hat{\Theta} \mathbf{V}_{b',k}|^2 + N_0^2) \eta_k) \\
&\text{s.t. (8.8a),}
\end{aligned} \tag{8.13}$$

where $\boldsymbol{\eta} = [\eta_1, \eta_2, \dots, \eta_{|\mathcal{K}|}]$ a collection of auxiliary variables, and $\text{Re}\{\}$ refers to the real number part of a complex number.

Proof: See Appendix A.4.

Given proposition 2, we rewrite equation (8.13) to optimize $\hat{\Theta}$:

$$\begin{aligned}
\max_{\hat{\Theta}} f_5(\hat{\Theta}) &= 2\text{Re}\{\hat{\Theta}^\dagger \sum_{k \in \mathcal{K}} 2\sqrt{b_k(1 + \beta_k)} \eta_k^\dagger \mathbf{V}_{b,k}\} - \hat{\Theta}^\dagger \sum_{k \in \mathcal{K}} |\eta_k|^2 \sum_{b' \in \mathcal{B}} |\mathbf{V}_{b',k}|^2 \hat{\Theta} + \sum_{k \in \mathcal{K}} |\eta_k|^2 N_0^2 \\
&= -\hat{\Theta}^\dagger \Lambda \hat{\Theta} + 2\text{Re}\{\hat{\Theta}^\dagger \Psi\} + \sum_{k \in \mathcal{K}} |\eta_k|^2 N_0^2 \\
&\text{s.t. (8.8a),}
\end{aligned} \tag{8.14}$$

where $\Lambda = \sum_{k \in \mathcal{K}} |\eta_k|^2 \sum_{b' \in \mathcal{B}} |\mathbf{V}_{b',k}|^2$ and $\Psi = \sum_{k \in \mathcal{K}} 2\sqrt{b_k(1 + \beta_k)} \eta_k^\dagger \mathbf{V}_{b,k}$ for ease of notations.

Note that equation (8.14) is a quadratically constrained quadratic programming problem, but the phase shift constraint $|\theta_{m,n}|^2 = 1$ is non-convex. Therefore, we relax this constraint by allowing $|\theta_{m,n}|^2 \leq 1$ for convexity, which means:

$$\begin{aligned}
\max_{\hat{\Theta}} f_6(\hat{\Theta}) &= -\hat{\Theta}^\dagger \Lambda \hat{\Theta} + 2\text{Re}\{\hat{\Theta}^\dagger \Psi\} \\
&\text{s.t. } |\theta_{m,n}|^2 \leq 1, m \in \mathcal{M}, n \in \mathcal{N}_m,
\end{aligned} \tag{8.15}$$

which is an optimization problem with strong convexity. The Lagrange dual of this problem can be written as an unconstrained problem:

$$\min_{\hat{\Theta}, \hat{\sigma}} f_7(\hat{\Theta}, \hat{\sigma}) = -\hat{\Theta}^\dagger \Lambda \hat{\Theta} + 2\text{Re}\{\hat{\Theta}^\dagger \Psi\} - \sum_{m \in \mathcal{M}} \sum_{n \in \mathcal{N}_m} \sigma_{m,n} (|\theta_{m,n}|^2 - 1), \quad (8.16)$$

where $\sigma_{m,n}$ is the Lagrange dual variable for each constraint with $\hat{\sigma} = [\sigma_1, \dots, \sigma_m, \dots, \sigma_{\mathcal{M}}]$, and each $\sigma_m = [\sigma_{m,1}, \dots, \sigma_{m,n}, \dots, \sigma_{m,|\mathcal{N}_m|}]$.

Proposition 3: the optimal value of equation (8.15) can be achieved by updating optimal $\hat{\Theta}$ and $\hat{\sigma}$ iteratively.

Proof: Setting $\frac{\partial f_7}{\partial \hat{\Theta}} = 0$, then we have the optimal $\hat{\Theta}^*$ by:

$$\hat{\Theta}^* = \frac{\Psi}{\Lambda + \text{diag}}(\hat{\sigma}). \quad (8.17)$$

Then, given $\hat{\Theta}^*$, we differentiate (8.16):

$$\begin{aligned} \frac{\partial f_7}{\partial \sigma_{m,n}} &= \frac{\partial (\sum_{m \in \mathcal{M}} \sum_{n \in \mathcal{N}_m} \sigma_{m,n} (1 - |\theta_{m,n}|^2))}{\partial \sigma_{m,n}} \\ &= 1 - \frac{\partial (\hat{\Theta} \hat{\Theta}^\dagger \text{diag}(\hat{\sigma}))}{\partial \sigma_{m,n}} \\ &= 1 - |\theta_{m,n}|^2 \end{aligned} \quad (8.18)$$

Setting $\frac{\partial f_7}{\partial \sigma_{m,n}} = 0$ will bring $|\theta_{m,n}|^2 = 1$, which means each element in $\hat{\Theta}^*$ will satisfy the constraints (8.8a).

Finally, substituting (8.17) into (8.16) claims:

$$f_8 = \frac{\Psi \Psi^\dagger}{\Lambda + \text{diag}}(\hat{\sigma}) + \text{tr}(\text{diag}(\hat{\sigma})), \quad (8.19)$$

which can be solved by the well-known Schur complement:

$$\begin{aligned} \max_{\hat{\sigma}, \kappa} f_9(\hat{\sigma}, \kappa) &= \kappa - \text{tr}(\text{diag}(\hat{\sigma})) \\ \text{s.t.} \quad &\begin{bmatrix} \Lambda + \text{diag}(\hat{\sigma}) & \Psi \\ \Psi^\dagger & -\kappa \end{bmatrix} \succeq 0 \end{aligned} \quad (8.20)$$

Finally, equation (8.20) can be efficiently solved as a semidefinite programming (SDP) problem, and we summarize the FP-based RIS phase control in Algorithm 1, which applies an alternative method for optimization.

Algorithm 8.1 Fractional programming based RIS phase control

- 1: **Initialize:** Wireless channel parameters and BS transmission powers. Setting iteration number $y = 1$, selecting a feasible initial value for $\Theta^{(y)}$ based on the constraints.
 - 2: **repeat**
 - 3: Given $\Theta^{(y)}$, updating auxiliary variable $\beta_k^{*(y+1)} = \psi_k^{(y)}$ by equation (8.7).
 - 4: Calculating $\hat{\Theta}^{(y+1)}$ and $\mathbf{V}_{b,k}^{(y+1)}$ which has been defined in equation (8.12).
 - 5: Calculating $\eta^{*(y+1)}$ as in Appendix A.4, and generating $\Lambda^{(y+1)}$ and $\Psi^{(y+1)}$ accordingly.
 - 6: Calculating $\hat{\sigma}^{(y+1)}$ by solving problem (8.20).
 - 7: Updating $\hat{\Theta}^{(y)}$ by equation (8.17), transforming $\hat{\Theta}^{(y)}$ to $\Theta^{(y)}$.
 - 8: $y = y + 1$
 - 9: **until** Problem (8.9) converges, or the iteration number y reaches the predefined maximum value.
 - 10: **Output:** Maximum data transmission rate and optimal RIS phase shifts under given transmission power of BSs.
-

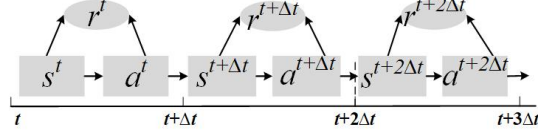
8.3 Cooperative Hierarchical Deep Reinforcement Learning for Sleep Control and Power Control

We introduce the proposed Co-HDRL algorithm in this section. Co-HDRL applies a hierarchical architecture for decision-making, including a meta-controller for sleep control, and multiple sub-controllers for transmission power control.

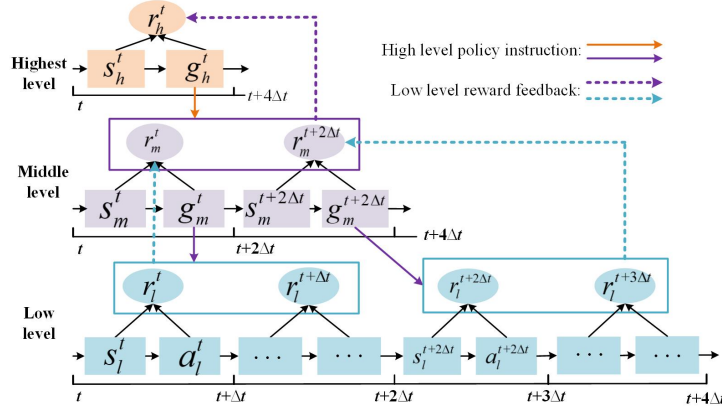
8.3.1 Hierarchical SMDP, Meta-controller and Sub-controller Definitions

In this section, we introduce a hierarchical semi Markov decision process (SMDP), which is used to transform our defined problem formulation into MDP context.

A conventional MDP scheme can be defined by a tuple $\langle S, A, T, R \rangle$, where S is the state set, A is the action set, T is the transition probability, and R is the reward function, respectively. As shown in Fig. 8.2a, the agents select an action a^t based on current state s^t , then it receives the reward r^t and moves to the next state $s^{t+\Delta t}$. However, conventional MDP has difficulty in defining more complicated and realistic problems such as tasks with sparse rewards and different timescale actions. For example, sleep control is obviously a long-term decision, since turning on/off SBSs will greatly affect the network performance. By contrast, power control is a delicate and short-term decision to change the SINR of



(a) Conventional MDP architecture



(b) SMDP architecture.

Figure 8.2: Comparisons between MDP and SMDP.

UEs. Therefore, we include a novel hierarchical SMDP framework to better handle the timescale difference between sleep control and power control.

Compared with conventional MDP, the main difference of SMDP lies in the hierarchical architecture. The SMDP here can be defined by a tuple $\langle S, A, T, R, \mathcal{G} \rangle$, where $\mathcal{G}$ is the set of goals. For example, Fig. 8.2b includes three agents in different levels, and each agent has its own MDPs. Given current state s_h^t , the highest level agent will first select a goal g_h^t at time slot t , and this goal serves as a high-level policy instruction for middle-level agent from time slot t to $t + 4\Delta t$. Similarly, g_m^t , which is the goal selected by middle-level agent, can affect the low-level action selection from t to $t + 2\Delta t$. On the other hand, low-level agents will provide reward feedback to higher-level agents, which can be used to evaluate higher-level goals. For instance, r_l^t and $r_l^{t+\Delta t}$, which are low-level rewards, may affect the r_m^t calculation, i.e. the middle-level reward.

Moreover, these agents can apply different timescales for decision making: e.g., the highest level selects g_h^t every $4\Delta t$, the middle level chooses g_m^t every $2\Delta t$, and the low level makes decisions every Δt . This hierarchical timescale setting enables higher flexibility for SMDP in more complicated and realistic environments, which makes it an ideal model for describing the joint sleep control and power control problem. Meanwhile, Fig. 8.2b reveals that the high-level agents rely on the reward feedback of lower levels for goal evaluation. Unstable feedback from low-level agents can mislead the high-level controllers, and sub-

optimal policy may be selected, which will finally degrade the overall system reward. Note that SMDP has many variants, and here we define our SMDP to solve the hierarchical control problem [171]. We use three layers as an example in Fig. 8.2b, but the proposed SMDP scheme is compatible with any number of layers.

Based on Fig. 8.2b, we define an SMDP with two layers. MBS is defined as a meta-controller for sleep control, which will decide the long-term on/off status of attached SBSs. Then, each SBS is considered as a sub-controller to adjust its short-term transmission power level. Sleep control is refereed to high-level policy instruction for SBSs, and SBSs can provide reward feedback to the meta-controller to evaluate the sleep control policies.

For the SBS sub-controller, we define the state, action and reward by:

- **State:** For the b^{th} SBS, the state s_{sub} is defined by the total transmission demand level of attached UEs:

$$s_{sub} = \frac{\sum_{k \in \mathcal{K}_b} W_{b,k}}{W_b^{max}}, \quad (8.21)$$

where $W_{b,k}$ represents the transmission demand of UE k , $\mathcal{K}_b$ represents the set of UEs that are associated with b^{th} SBS. W_b^{max} is the max transmission demand of b^{th} SBS, which is referred to as a constant value to normalize transmission demand in the current time slot. It is assumed the UE daily transmission demand follows specific patterns as in [172].

- **Action:** The SBS sub-controller can adjust its transmission power level P_b to satisfy the transmission demand in the current time slot. Therefore, the action of b^{th} SBS sub-controller is $a_{sub} = \{P_b\}$.
- **Low-level reward:** The low-level reward of sub-controller is:

$$r_{sub} = \begin{cases} \frac{\sum_{k \in \mathcal{K}_b} W_{b,k}}{E_b} - \phi n_b, & \text{if } q_b = 1 \\ 0, & \text{if } q_b = 0 \end{cases} \quad (8.22)$$

where ϕ and n_b are overloading penalty factor and overload indicator, respectively. The overload definition has been given in equation (8.6). q_b has been defined in equation (8.5) as the sleep control indicator. r_{sub} aims to maximize the EE of b^{th} SBS, and we assume the r_{sub} is 0 in sleep mode.

The MBS is defined as a meta-controller, which will produce high-level sleep control policies for the SBS sub-controllers:

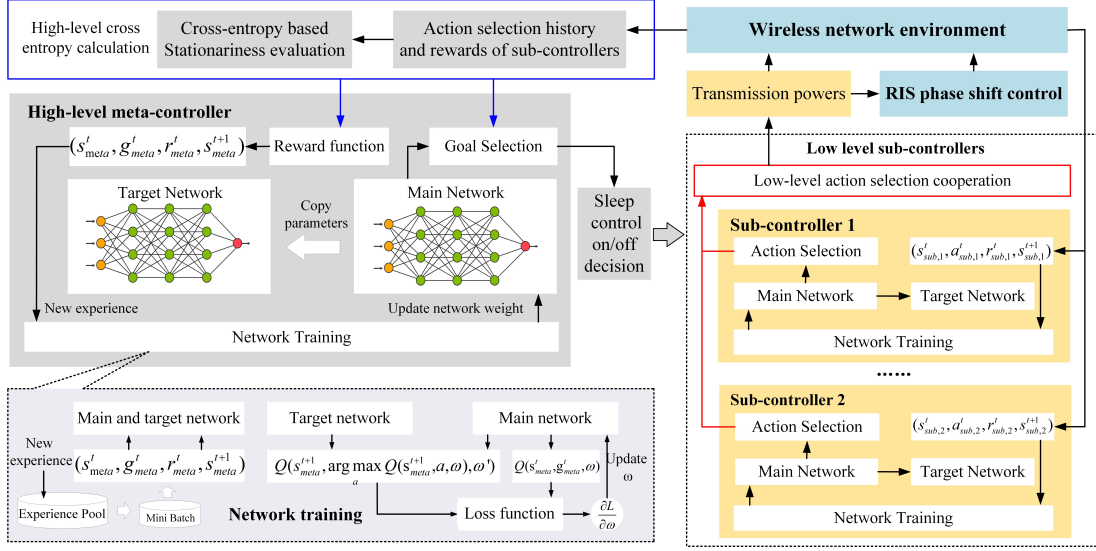


Figure 8.3: Overall architecture of the proposed Co-HDRL.

- **State:** The state of the MBS meta-controller includes the transmission demand level of all BSs:

$$s_{meta} = \left[\frac{\sum_{k \in \mathcal{K}_1} W_{1,k}}{W_1^{max}}, \frac{\sum_{k \in \mathcal{K}_2} W_{2,k}}{W_2^{max}}, \dots, \frac{\sum_{k \in \mathcal{K}_{|\mathcal{B}|}} W_{|\mathcal{B}|,k}}{W_{|\mathcal{B}|}^{max}} \right], \quad (8.23)$$

where $\mathcal{B}$ is the set of all BSs.

- **High-level goals:** Given the transmission demand level of all BSs, the meta-controller can produce sleep control decisions for SBS sub-controllers:

$$g_{meta} = [q_1, q_2, \dots, q_b, \dots, q_{|\mathcal{B}_{SBS}|}], \quad (8.24)$$

where q_b has been defined in equation(8.5) as sleep mode indicator of b^{th} SBS, and $\mathcal{B}_{SBS}$ is the set of SBSs.

- **High-level reward:** The meta-controller is responsible for the overall performance of all BSs. Accordingly, the high-level reward is given by the objective function we have defined in the problem formulation:

$$r_{meta} = \frac{\sum_{b \in \mathcal{B}} \sum_{k \in \mathcal{K}_b} D_{b,k}}{\sum_{b \in \mathcal{B}} E_b} - \phi \sum_{b \in \mathcal{B}} n_b, \quad (8.25)$$

8.3.2 Co-HDRL Framework and Network Training

The overall framework of our proposed Co-HDRL algorithm is given in Fig. 8.3, which includes one meta-controller and multiple sub-controllers. The goal selected by the meta-controller will decide the on/off status of sub-controllers.

In each controller, we deploy a double deep Q-learning (DDQN) scheme for the Q-value prediction, which has been introduced in former Chapters. The DDQN scheme is proposed to separate the action selection and evaluation by:

$$L(w) = Er(r^t + \gamma Q(s^{t+1}, \arg \max_a Q(s^{t+1}, a, w), w') - Q(s^t, a^t, w)), \quad (8.26)$$

where s^t , a^t and r^t are the state, action and reward at time slot t , respectively. w and w' are the weight of the main and target networks. γ is the discount factor ($0 < \gamma < 1$). $Q(s^t, a^t, w)$ is the current Q-value that is predicted by the main network. $\arg \max_a Q(s^{t+1}, a, w)$ indicates the main network selects the next action, and the target network evaluates the action by $Q(s^{t+1}, \arg \max_a Q(s^{t+1}, a, w), w')$. Decoupling the action selection and evaluation can provide a more accurate target for the main network training, which will further reduce the Q-value prediction error.

In Co-HDRL, first, the meta-controller selects a high-level goal g_{meta} , which indicates the sleep control decisions on sub-controllers. g_{meta} is temporarily fixed in the following several time slots, and sub-controllers select actions, receive rewards, and train their networks accordingly. The transmission power of sub-controllers is sent to the RIS control block for phase shift optimization. Based on the long-term performance of sub-controllers, the meta-controller receives an average reward r_{meta} from the wireless environment and moves to the next state s_{meta} . The new experience tuple $\langle s_{meta}^t, g_{meta}^t, r_{meta}^t, s_{meta}^{t+1} \rangle$ will be sent to the experience pool (shown as the left bottom block in Fig. 8.3). Then the agent will sample a random mini batch from the experience pool, and train the main network as equation (8.26). The target network will copy the main network weight after several training, which guarantees a stable target for the main network training.

In Fig. 8.3, it is worth noting that we define a high-level cross-entropy calculation module (shown by the blue block on the top), and a low-level action selection cooperation module (indicated by the red block on the right). In HRL, the meta-controller produces goals to instruct sub-controllers, and sub-controllers provide reward feedback to the meta-controller to evaluate the high-level goal. However, although the high-level goal g_{meta} is temporarily fixed, the action of sub-controllers can be constantly changing during this period. Indeed, the sub-controllers' exploration increases the uncertainty of the reward feedback to the meta-controller. Therefore, this feedback uncertainty can mislead the goal evaluation of meta-controller [173, 174]. To this end, we proposed a cross-entropy enabled

policy for high-level meta-controller, and a correlated equilibrium-based cooperation strategy for low-level sub-controllers. Following, we will introduce two techniques in detail.

8.3.3 High-level Cross-entropy enabled Meta-controller Policy

In this section, we propose a cross-entropy enabled policy for meta-controller. More specifically, we use the cross-entropy as a metric to evaluate the stationarity of sub-controllers' actions, then the defined metric is used for high-level goal exploration. For example, goals with high reward feedback but low stationarity may be unreliable, since the low stationarity metric means the sub-controllers are still exploring action combinations. On the other hand, goals that bring high stationarity can be less visited in the exploration phase, because the sub-controllers have entered a stable status and it requires less training [175]. Following, we first present the cross-entropy metric definition. We apply this metric to evaluate the policy changes between the current and previous time slots, which represent the stationarity of sub-controllers' policy. Specifically, it shows the distribution difference between current and historical action selections.

Given a random variable x , and N^X is the total number of possible outcomes of x , then the entropy of x in set X is defined by:

$$I(X) = - \sum_{i=1}^{N^X} pr(x_i) \log(x_i), \quad (8.27)$$

where $pr(x_i)$ is the probability of x_i in set X , and $\sum_{i=1}^{N^X} pr(x_i) = 1$. $I(X)$ indicates the average level of information and uncertainty of variable x_i in set X .

Then we introduce the Kullback–Leibler divergence to define the relative entropy from one distribution X to another distribution Y of variable x [176] :

$$D_{KL}(X||Y) = \sum_{i=1}^{N^X} pr(x_i) \log \frac{pr(x_i)}{pr'(x_i)}, \quad (8.28)$$

where $pr'(x_i)$ is the probability of x_i in set Y . $D_{KL}(X||Y)$ indicates the difference of set X and Y in terms of probability distribution of x .

In this work, we consider X as the set of action selection history of sub-controllers, and Y is the set of actions selected in the current time interval Δt . Consequently, we can apply $D_{KL}(X||Y)$ to measure the stationarity of the low-level action selection policy. The idea

is that a stationary policy will produce similar action selections in different time intervals, which is indicated by a lower $D_{KL}(X||Y)$ value. By contrast, a higher $D_{KL}(X||Y)$ means sub-controllers require more training to stabilize the low-level policies.

Proposition 4: The Kullback–Leibler divergence has the minimum value when $pr(x_i) = pr'(x_i)$.

Proof 4: Based on Gibbs' inequality [177]:

$$\begin{aligned}
D_{KL}(X||Y) &= \sum_{i=1}^{N^X} pr(x_i) \log \frac{pr(x_i)}{pr'(x_i)} \\
&= - \sum_{i=1}^{N^X} pr(x_i) \log \left(\frac{pr'(x_i)}{pr(x_i)} \right) \quad (\text{Using } \log(x) < x - 1 \text{ for all } x > 0) \\
&\geq - \sum_{i=1}^{N^X} pr(x_i) \left(\frac{pr'(x_i)}{pr(x_i)} - 1 \right) \\
&= \sum_{i=1}^{N^X} pr(x_i) - \sum_{i=1}^{N^X} pr'(x_i) \\
&= 0.
\end{aligned} \tag{8.29}$$

$D_{KL}(X||Y) = 0$ if and only if X and Y has the same distribution $pr(x_i) = pr'(x_i)$.

Moreover, we rewrite equation (8.28) by:

$$\begin{aligned}
D_{KL}(X||Y) &= \sum_{i=1}^{N^X} pr(x_i) \log \frac{pr(x_i)}{pr'(x_i)} \\
&= \sum_{i=1}^{N^X} pr(x_i) \log(pr(x_i)) - \sum_{i=1}^{N^X} pr(x_i) \log(pr'(x_i)) \\
&= -I(X) - \sum_{i=1}^{N^X} pr(x_i) \log(pr'(x_i)),
\end{aligned} \tag{8.30}$$

where $I(X)$ is the entropy of the action selection distribution history, and Y is the action selection distribution in the current time interval. The history distribution X is generally more stable than the current distribution Y . It means $\sum_{i=1}^{N^X} pr(x_i) \log(pr'(x_i))$ item contributes more to the uncertainty, which is known as the cross-entropy [178]. In this work, we use cross entropy to represent the stationarity of the sub-controllers.

We include multiple sub-controller in this work, and the cross-entropy of b^{th} sub-controller under high-level goal g_{meta} is:

$$I(X_{b,t-1}, Y_{b,t}|g_{meta}) = - \sum_{i=1}^{A_{g_{meta}}} pr(a_i) \log(pr'(a_i)), \quad (8.31)$$

where $X_{b,t-1}$ is the accumulated action selection distribution of sub-controllers in former $t-1$ time slots under high-level goal g_{meta} , $Y_{b,t-1}$ is the action selection distribution under g_{meta} in current time slot, and $A_{g_{meta}}$ is the action set of sub-controllers under g_{meta} . The cross-entropy metric $I(X_{b,t-1}, Y_{b,t}|g_{meta})$ defines the stationary of sub-controller action selections under g_{meta} .

Finally, in the exploration phase, the meta-controller selects high-level goals by:

$$pr(g_{meta}|s_{meta}) = \frac{\tanh}{(\sum_{b \in \mathcal{B}_{-M}} I(X_{b,t-1}, Y_{b,t}|g_{meta}))} \sum_{g_{meta} \in \mathcal{G}} \tanh(\sum_{b \in \mathcal{B}_{-M}} I(X_{b,t-1}, Y_{b,t}|g_{meta})), \quad (8.32)$$

where $\mathcal{B}_{-M}$ indicates the set of controllers except the MBS meta-controller. The $\tanh$ indicates the Tanh function, which is applied to normalize all the cross-entropy values.

In summary, equation (8.32) shows that goals that lead to lower stability will be more frequently selected in the exploration phase, which is indicated by a high cross-entropy value. It guarantees that sub-controllers can provide reliable and stable feedback to the meta-controller for goal evaluation. The meta-controllers and sub-controllers are updated by Fig. 8.4. The meta-controller will first select a goal g_{meta}^t at time t , which is temporarily fixed in the next $N_T \Delta t$ time slot. Then, sub-controllers select the actions in each Δt under current state s_{sub} and high-level goal g_{meta}^t . Finally, the actions and rewards of sub-controllers from t to $t+N_T \Delta t$ are collected by meta-controllers for cross-entropy calculation and goal selection as shown by equation (8.32).

8.3.4 Low-level Sub-controllers Cooperation

We propose a correlated equilibrium-based cooperation strategy for low-level sub-controllers. Correlated equilibrium is proposed as a multi-agent collaboration strategy [179], and we assume all sub-controllers will share true information with each other. Compared with existing methods such as Nash equilibrium, the existence and convergence of correlated equilibrium can be better guaranteed by linear programming solutions. Indeed, cooperative action selections can bring higher stability than independent action selections. With

Algorithm 8.2 Co-HDRL based joint sleep and power control

- 1: **Initialize:** Wireless network and Co-HDRL parameters
 - 2: **for** every $N_T \Delta t$, the MBS **do**
 - 3: Updating the cross-entropy metric under g_{meta} based on the action selection history of last $N_T \Delta t$.
 - 4: Calculating the average EE metric of former $N_T \Delta t$ as high-level reward r_{meta} by equation (8.6).
 - 5: Updating new state s_{meta} , and saving $(s_{meta}, g_{meta}, r_{meta}, s'_{meta})$ to the experience pool.
 - 6: Sampling a mini-batch from the experience pool randomly. Generating target Q-values

$$Q^{Tar}(s_{meta}, g_{meta}) = \begin{cases} r_{meta} & \text{if done} \\ r_{meta} + \gamma Q(s'_{meta}, \arg \max_g Q(s'_{meta}, g, w), w') & \text{else} \end{cases}$$
 - 7: Updating w using gradient descent by minimizing the loss $L(w) = Er(Q^{Tar}(s_{meta}, g_{meta}) - Q(s_{meta}, g_{meta}, w))$.
 - 8: Copying w to w' after several training.
 - 9: Selecting a high-level goal g_{meta} of the next $N_T \Delta t$ by following method:
 - 10: with probability ϵ , selecting high-level goal g_{meta} by equation (8.32); otherwise, choosing goal g_{meta} by greedy policy $\arg \max_a Q(s_{meta}, g_{meta}, w)$.
 - 11: **end for**
 - 12: **for** every Δt , active SBSs **do**
 - 13: Selecting the transmission power level a_{sub} by:
 - 14: with probability ϵ , sub-controllers select a_{sub} randomly; otherwise, choosing joint action $\vec{a}_{sub}$ by correlated equilibrium as equation (8.33).
 - 15: Sending the transmission power to RIS control **Algorithm 8.1**, then receives the data transmission rate from **Algorithm 8.1**, and calculates the low-level reward.
 - 16: Updating new state s_{sub} , and saving $(s_{sub}, a_{sub}, r_{sub}, s'_{sub})$ to the experience pool. Each sub-controller samples a mini-batch from its experience pool, and trains the network similar to the meta-controller.
 - 17: **end for**
 - 18: **Output:** Performance of the network and the learning algorithm.
-

correlated equilibrium, the SBS sub-controllers choose actions by:

$$\max_{a_{sub} \in A_{sub}} \sum_{\vec{a}_{sub} \in A_{sub}} pr(\vec{a}_{sub} | s_{sub}) Q(s_{sub}, \vec{a}_{sub}, w) \quad (8.33)$$

$$sub.to \quad 0 \leq pr(\vec{a}_{sub} | s_{sub}) \leq 1, \quad (8.33a)$$

$$\sum_{\vec{a} \in A} pr(\vec{a}_{sub} | s_{sub}) = 1, \quad (8.33b)$$

$$\sum_{a-b \in A-b} pr(\vec{a}_{sub} | s_{sub}) (Q(s_{sub}, \vec{a}_{sub}, w) - Q(s_{sub}, \vec{a}_{-b}, a_b, w)) \geq 0, \quad (8.33c)$$

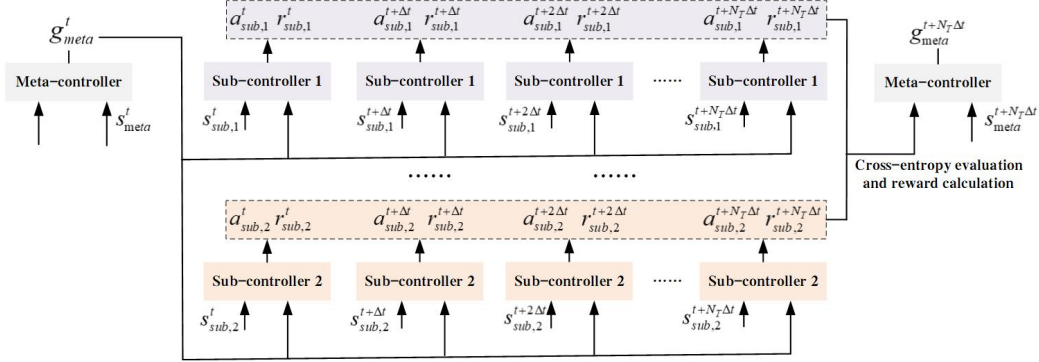


Figure 8.4: Co-HDRL system update between meta-controller and sub-controllers.

where $pr(\vec{a}_{sub}|s_{sub})$ is the probability of selecting action combination $\vec{a}_{sub} = (a_1, a_2, \dots, a_{|\mathcal{B}_{-M}|})$ under state s_{sub} , $\vec{a}_{-b}$ indicates the joint action of all sub-controllers except b^{th} sub-controller, and w is the main network weight. $\mathcal{B}_{-M}$ has been defined in equation (8.32) as the set of controllers except the MBS meta-controller, $\mathcal{A}_{-b}$ is the set of $\vec{a}_{-b}$.

Equation (8.33) uses Q-values predicted by the main network to represent the expected reward. It aims to maximize the total potential reward of all sub-controllers by finding an optimal action selection probability distribution $pr(\vec{a}_{-b}|s_{sub})$. Constraints (8.33a) and (8.33b) are probability distribution constraints. The constraint (8.33c) indicates that $\vec{a}_{-b}$ produces higher expected reward with probability $pr(\vec{a}_{-b}|s_{sub})$, which is indicated by $pr(\vec{a}_{sub}|s_{sub})(Q(s_{sub}, \vec{a}_{sub}, w) - Q(s_{sub}, \vec{a}_{-b}, a_b, w)) \geq 0$.

Finally, the proposed Co-HDRL algorithm is summarized in Algorithm 8.2.

8.3.5 Computational Complexity Analyses

In this section, we analyze the computational complexity of the proposed algorithms. For the RIS control part, the complexity lies in solving equation (8.20), which is a semidefinite programming (SDP) problem. SDP can be efficiently solved by many methods such as the interior point method, first-order method and so on. Based on the model proposed in [180], the computational complexity is given by $O(\sqrt{\rho}(c\rho^2 + c^l + \rho^l) \log(\frac{1}{\zeta}))$, where ρ is the variable size, c is the number of constraints, l is the cost of matrix multiplication, and ζ is the relative accuracy. For the Co-HDRL part, the computational complexity depends on neural network training. In this work, we deploy Long short-term memory (LSTM) networks for the Q-values prediction, which is a special recurrent neural network that can better capture the long-term dependency of input data. The computational complexity of

Algorithm 8.3 HDRL-based joint sleep and power control

- 1: **Initialize:** Wireless network and HDRL parameters
 - 2: **for** every $N_T\Delta t$, the MBS **do**
 - 3: Calculating the average EE metric of former $N_T\Delta t$ as high-level reward r_{meta} by equation (8.6).
 - 4: Updating new state s_{meta} , and saving $(s_{meta}, g_{meta}, r_{meta}, s'_{meta})$ to the experience pool.
 - 5: Sampling a mini-batch from the experience pool randomly. Generating target Q-values
$$Q^{Tar}(s_{meta}, g_{meta}) = \begin{cases} r_{meta} & \text{if done} \\ r_{meta} + \max_a \gamma Q(s'_{meta}, g_{meta}, w') & \text{else} \end{cases}$$
 - 6: Updating w using gradient descent by minimizing the loss $L(w) = Er(Q^{Tar}(s_{meta}, g_{meta}) - Q(s_{meta}, g_{meta}, w))$.
 - 7: Copying w to w' after several training.
 - 8: With probability ϵ , selecting a high-level goal randomly for the next $N_T\Delta t$; otherwise, choosing goal g_{meta} by greedy policy $\arg \max_a Q(s_{meta}, g_{meta}, w)$.
 - 9: **end for**
 - 10: **for** every Δt , active SBSs **do**
 - 11: Selecting the transmission power level a_{sub} by:
 - 12: with probability ϵ , sub-controllers select a_{sub} randomly; otherwise, choosing action a_{sub} by greedy policy.
 - 13: Sending the transmission power to RIS control **Algorithm 8.1**, then receives the data transmission rate from **Algorithm 8.1**, and calculating the low-level reward.
 - 14: Updating new state s_{sub} , and saving $(s_{sub}, a_{sub}, r_{sub}, s'_{sub})$ to the experience pool. Each sub-controller samples a mini-batch from its experience pool, and trains the network similar to the meta-controller.
 - 15: **end for**
 - 16: **Output:** Performance of the network and the learning algorithm.
-

updating LSTM networks is $O(\alpha^2\chi^2)$, where α indicates the memory block number, and χ represents the memory cell number of one block [20].

8.4 Baseline Algorithms

8.4.1 Hierarchical Deep Reinforcement Learning

To better demonstrate the capability of our proposed Co-HDRL algorithm, we propose a conventional hierarchical deep reinforcement learning (HDRL) based method as a bench-

Algorithm 8.4 Surrogate optimization based RIS phase control

- 1: **Initialize:** Wireless channel and surrogate optimization parameters, and BS transmission powers. Setting iteration number $y = 1$, and initializing the surrogate model.
 - 2: **repeat**
 - 3: Obtaining an estimated solution to equation (8.34) by optimizing the surrogate.
 - 4: Evaluating the surrogate model at the estimated solution produced in the last step.
 - 5: Updating the radial basis function-based surrogate model using the new high-fidelity model data.
 - 6: $y = y + 1$
 - 7: **until** Problem (8.34) converges, or the iteration number y reaches the predefined maximum value.
 - 8: **Output:** Maximum data transmission rate and optimal RIS phase shifts under given transmission power of BSs.
-

mark. In HDRL, both meta-controller and sub-controllers apply ϵ -greedy policy for goal and action selections, respectively, and there is no high-level cross-entropy and low-level cooperation between agents. The HDRL is given in Algorithm 8.3.

8.4.2 Surrogate Optimization based RIS phase control

This section introduces the surrogate optimization-based RIS phase-shift control as a baseline algorithm. In practice, the phase shift can only be selected from a set of fixed values, which mainly depends on the predefined resolution settings of phase shifters. Therefore, we formalize the RIS control problem to maximize the total transmission rate of all UEs by:

$$\max_{\mathcal{Z}} \quad f_{10}(\mathcal{Z}) = \sum_{k \in \mathcal{K}} b_k \log \left(1 + \frac{P_b \left| \sum_{m \in \mathcal{M}} \mathbf{H}_{b,m} \mathbf{\Theta}_m \mathbf{G}_{m,k}^\dagger \right|^2}{\sum_{b' \in \mathcal{B}_{-b}} P_{b'} \left| \sum_{m' \in \mathcal{M}} \mathbf{H}_{b',m'} \mathbf{\Theta}_{m'} \mathbf{G}_{m',k}^\dagger \right|^2 + N_0^2} \right) \quad (8.34)$$

$$\text{s.t.} \quad \theta'_{m,n} = z_{m,n} \frac{2\pi}{2^\mu} \quad (8.34a)$$

$$z_{m,n} \in \{0, 1, \dots, (2^\mu - 1)\} \quad (8.34b)$$

where $\theta'_{m,n}$ has been defined in equation (8.2) as the RIS phase-shift angle. $z_{m,n}$ belongs to the matrix $\mathcal{Z}$, which will decide the phase shift angle $\theta'_{m,n}$. Note that the decision variable $z_{m,n}$ can only be selected from fixed integer values, which means equation (8.34) is a Mixed

Integer Nonlinear Programming (MINLP) problem, then a surrogate optimization method is applied to solve this problem.

Different than the FP-based method, the surrogate optimization treats the optimization as a black-box problem [181]. It yields a reasonable and accurate approximation for the original optimization problem by a surrogate model, which avoids the complexity of designing a dedicated optimization solution. The proposed surrogate optimization-based RIS phase control is summarized in Algorithm 8.4. Note that there are many surrogate models, and here we choose the radial basis function model for its low computation complexity.

8.5 Performance Evaluation

8.5.1 Simulation Settings

We consider a HetNet that includes 1 MBS and 3 SBSs. The radius of the MBS and SBS are 400 m and 100 m, respectively. The carrier frequency for the MBS and SBS are both 4 GHz [182]. The load-dependent power consumption factor is 4.7 and 2.6 for MBS and SBS, respectively [172]. The whole cell includes 20 UEs that are randomly distributed. 6 RISs are uniformly distributed, and each contains 10 elements with 2 bits resolution. For learning settings, we apply 2 LSTM networks as hidden layers for each controller. The hyperparameters such as neural network learning rate and training frequency are selected by the grid search method, which means we try different parameter combinations and find the best results accordingly. The simulations are repeated 10 runs on MATLAB with 95% confidence interval, and other settings are summarized in Table 8.1.

8.5.2 Performance Comparison under Sleep Control and RIS

This section investigates the performance under sleep control and RIS techniques. The proposed FP and Co-HDRL methods are deployed for RIS control and sleep control, respectively. As shown in Fig. 8.5, four cases are considered by combining sleep control with RIS control.

Fig. 8.5a shows that sleep control can significantly save power consumption by switching off SBSs intelligently. Combining sleep control with RIS can achieve the lowest energy consumption. Moreover, Fig. 8.5b demonstrates that RIS can greatly increase the average throughput, which can be explained by RIS' capability to reshape the signal transmission path. Consequently, shown by Fig. 8.5c, sleep control + RIS produces the highest energy

Table 8.1: Parameters settings of wireless networks, RISs, and learning algorithms

Wireless network settings	Traffic model
3GPP urban macro network Bandwidth: 20MHz Number of RBs: 100 Subcarriers in each RB: 12 Subcarrier bandwidth: 15kHz	20% Poisson distribution 80% constant bit rate
	Learning and algorithm settings
UE and BSs settings 1 MBS, 3 SBSs MBS cell radius: 400m SBS cell radius: 100m MBS max Tx power: 20 W SBS max Tx power: 6.3 W 20 UE with random distribution	Network layers: 4 2 LSTM network as hidden layers Hidden layer has 10 nodes Experience pool size: 600 Initial network learning rate: 0.005 Minibatch size: 180 Discount factor: 0.3 Epsilon value: 0.05 Training frequency: 240 iterations Network copy: every 600 iterations Time resolution difference N_T :10 Overloading penalty factor: 0.2
RIS settings 6 RIS deployed in the cell Each RIS includes 10 elements	

efficiency, no sleep control + no RIS shows the lowest energy efficiency, and the other two combinations present a comparable performance.

To better explain how RIS can help reduce energy consumption, we compare the daily SBS on/off status of RIS and no-RIS condition in Fig. 8.5d. The vertical axis is the probability of turning on SBSs, which indicates the power consumption level during specific time slots. The blue shade represents the preset traffic load pattern. Fig. 8.5d shows that most SBSs are switched off during the low-peak period (from 3:00 to 7:00) in both RIS and no-RIS cases. However, when the traffic load increases to mid-peak and on-peak periods, no-RIS case has to turn on most SBSs to satisfy the increasing traffic demand; otherwise, the overloading may lead to a high penalty. By contrast, using RIS can greatly increase the channel capacity, which means the MBS can still switch off most SBSs to save energy without causing overloading.

In summary, the simulations in Fig. 8.5 demonstrate that combining sleep control and RIS can greatly save energy consumption, increase throughput and achieve higher energy efficiency.

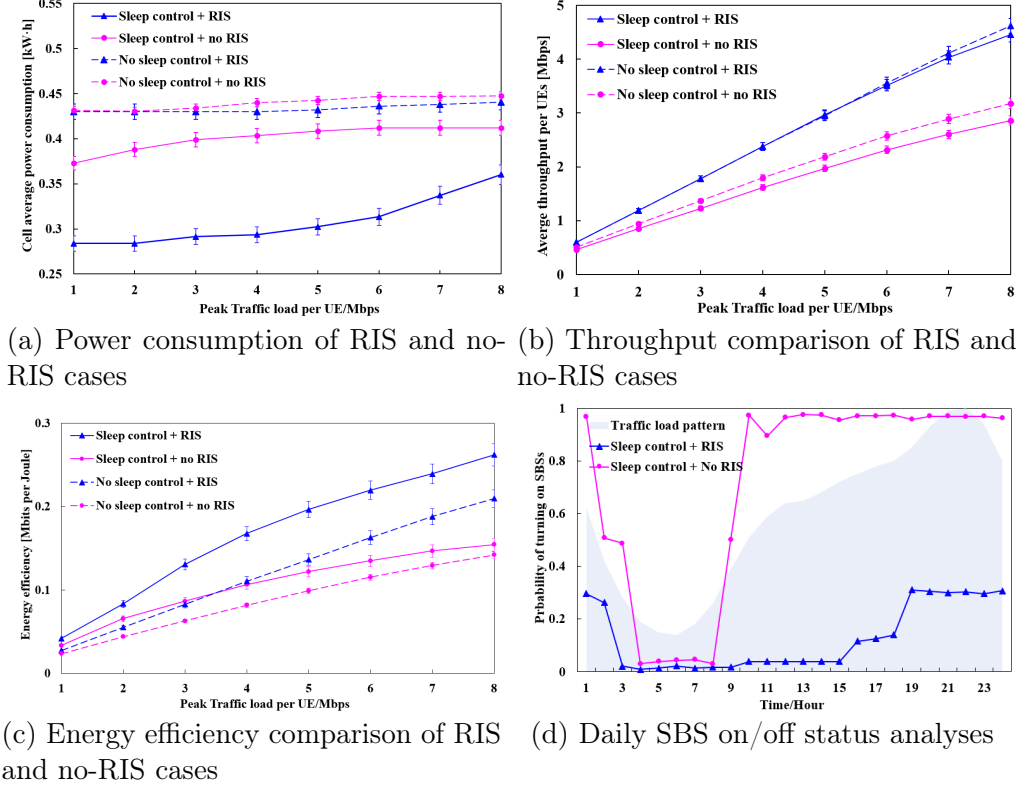


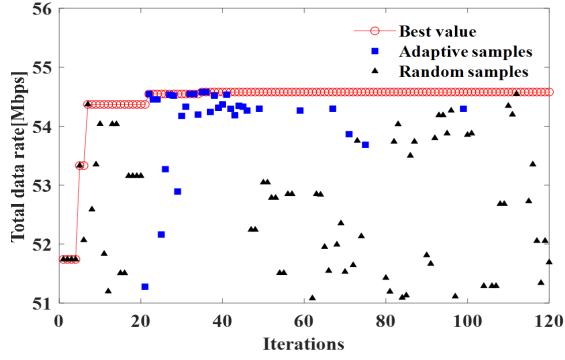
Figure 8.5: Performance comparison of sleep control and RIS.

8.5.3 RIS Control Strategy Analyses

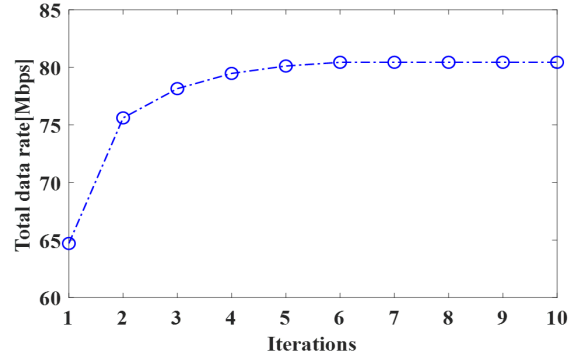
Consequently, we compare the proposed FP-based RIS control method with the surrogate optimization strategy.

Firstly, Fig. 8.6a shows the convergence performance of surrogate optimization in one episode. One can observe that surrogate optimization relies on random and adaptive samples to find better solutions. The surrogate model is constantly updated in each iteration to provide a more accurate estimation of the given problem. By contrast, in Fig. 8.6b, our proposed FP method reaches a much faster convergence. This faster convergence can be explained by the dedicated designed FP optimization model for RIS control. In the proposed FP, the alternative optimization method guarantees that the next iteration can always perform better than the former iterations.

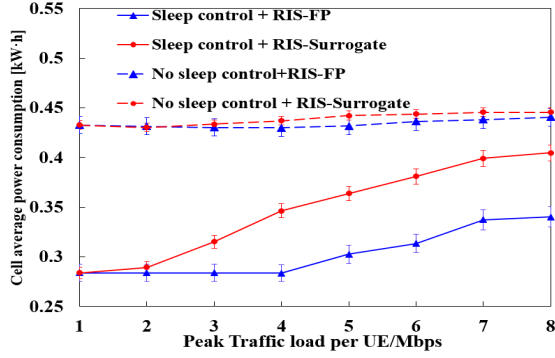
Similarly, we consider the combination of sleep control with different RIS control methods, and the power consumption, throughput and EE results are given in Fig. 8.6c, (d) and (e), respectively. Combining FP-based RIS control with sleep control can bring lower



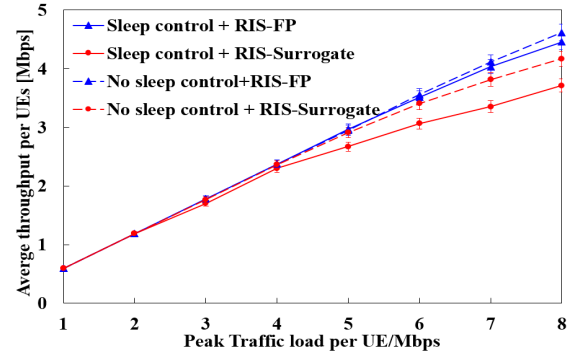
(a) Convergence performance of surrogate optimization based RIS control



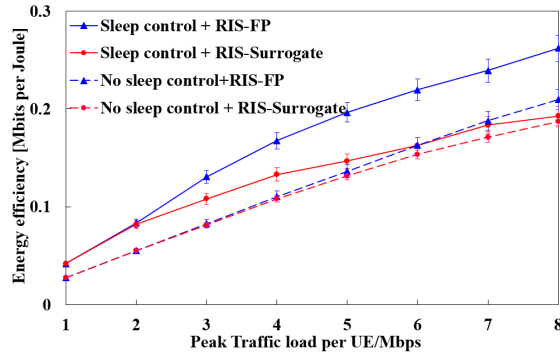
(b) Convergence performance of fractional programming (FP) based RIS control



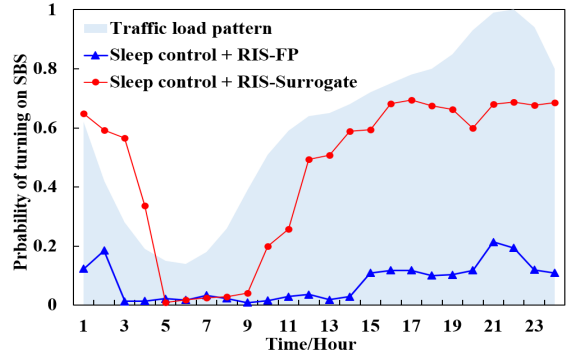
(c) Power consumption comparison of different RIS control methods



(d) Throughput consumption comparison of different RIS control methods



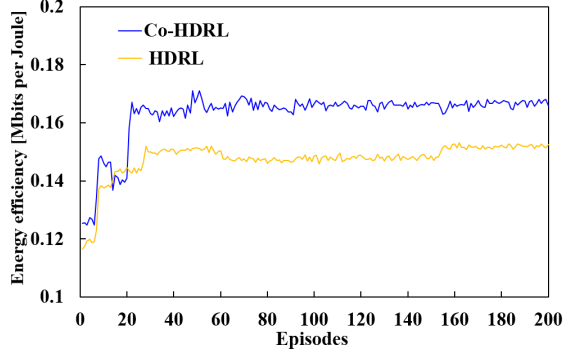
(e) Energy efficiency comparison of different RIS control methods



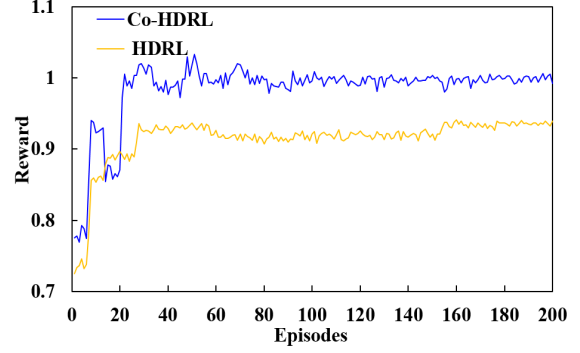
(f) Daily SBS on/off status analyses of different RIS control methods

Figure 8.6: Performance comparison of difference RIS control strategies.

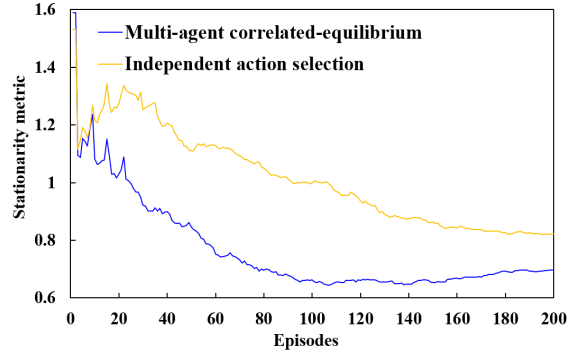
power consumption, higher throughput and EE than other cases. The proposed FP method



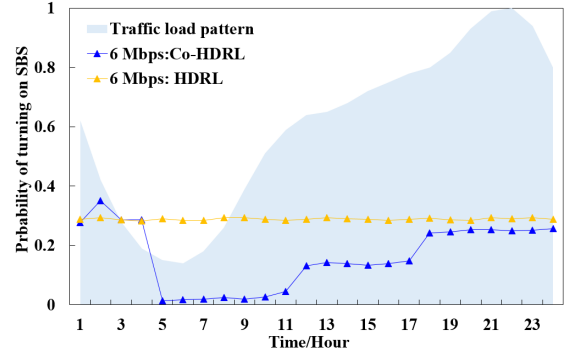
(a) Energy efficiency comparison of Co-HDRL and HDRL



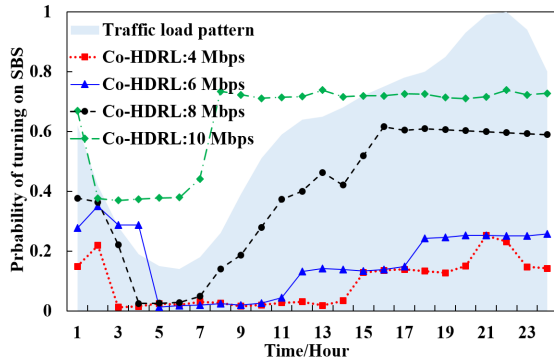
(b) Reward comparison of Co-HDRL and HDRL



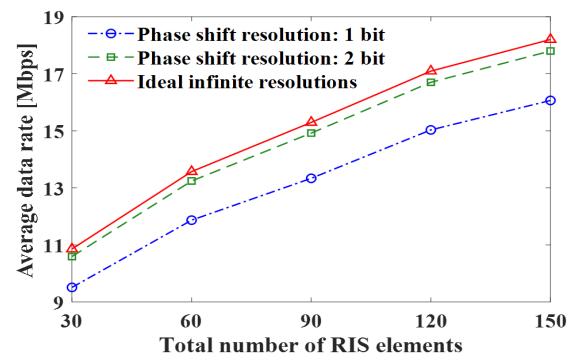
(c) Stationarity comparison of different action selection strategies



(d) Daily SBS on/off status of Co-HDRL and HDRL



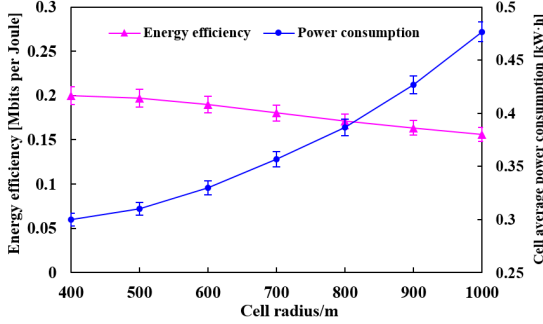
(e) Daily SBS on/off status of Co-HDRL under various traffic loads



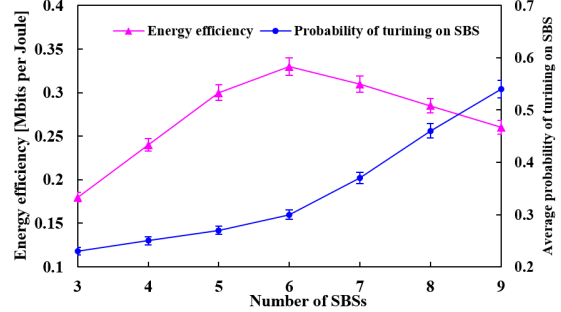
(f) Average data rate under various RIS elements numbers and phase shift resolutions

Figure 8.7: Performance analyses of Co-HDRL and HDRL.

can achieve 35% lower power consumption and 16% higher EE when the peak traffic load



(a) Energy efficiency and power consumption under various cell radius.



(b) Energy efficiency and SBS active probability under different SBS numbers.

Figure 8.8: Algorithm performance under various cell sizes and SBS numbers.

is 8 Mbps. Moreover, the SBS on/off status is shown in Fig. 8.6f. It demonstrates that FP-based RIS control can maintain low power consumption even in peak-load periods (from 17:00 to 23:00). On the contrary, surrogate-based RIS control has to turn on SBSs to adapt to the increasing transmission demand to prevent overloading penalty.

8.5.4 Machine Learning Algorithm Comparison

In this section, we compare the proposed Co-HDRL algorithm with conventional HDRL, which aims to prove that the cooperation strategy can bring better performance. The EE and reward performance of 200 episodes are given in Fig. 8.7a and (b), in which the proposed Co-HDRL algorithm achieves a higher EE and reward than the HDRL baseline. In Co-HDRL, we apply the cross-entropy-based policy for meta-controller to monitor the stability of sub-controllers, and the correlated equilibrium is deployed for joint action selection of sub-controllers. These cooperation methods enable higher stability for sub-controller action selection and training, which brings better performance than conventional HDRL. In Fig. 8.7c, we further present the stationarity of various low-level action selection strategies. The proposed multi-agent correlated equilibrium enables a more stable action selection than independent action selection that is applied in HDRL, which is indicated by a lower stationarity metric. Meanwhile, Fig. 8.7c demonstrates that correlated equilibrium can reach a stable state much faster than the independent action selection method.

Meanwhile, the satisfactory performance of Co-HDRL can also be observed in Fig. 8.7d, in which the Co-HDRL switches on/off SBSs dynamically during on-peak and off-peak periods. By contrast, HDRL produces the same sleep control decision regardless of different

traffic load conditions (traffic patterns shown by the light blue shade). Moreover, the Co-HDRL performance from 4 to 10 Mbps is presented in Fig. 8.7e, in which the proposed Co-HDRL make decisions intelligently under various traffic load level. For example, when the traffic load level reaches 10 Mbps, the meta-controller turns on most SBSs to serve the high transmission demand. On the other hand, when the traffic load is 4 Mbps, most SBSs are switched off to reduce power consumption.

We show the average data rate under various RIS element numbers and phase shift resolutions in Fig. 8.7f. It demonstrates that increasing the number of RIS elements can improve the system data rate. Meanwhile, higher phase shift resolutions can achieve a higher data rate, because the phase can be more accurately shifted.

8.5.5 Performance under various cell sizes and SBS numbers.

Finally, this subsection investigates the algorithm performance with different cell sizes and numbers of SBSs. Specifically, Fig. 8.8a shows that increasing cell size will lower EE and increase power consumption, which can be explained by lower SINR due to a larger cell radius. Meanwhile, the EE in Fig. 8.8b increases first and then decreases. In the increasing phase, the MBS has enough channel capacity to support the UEs that are previously associated with the SBSs in sleep mode. However, when the number of SBSs keeps increasing, the probability of turning on SBSs increases as illustrated in Fig. 8.8b. The main reason is that MBS cannot serve all the UEs when SBSs enter sleep mode, and some SBSs must keep active to fulfill the traffic demand. Consequently, the energy efficiency becomes lower with the increasing number of SBSs.

Chapter 9

Conclusion

ML offers promising opportunities for improving the performance of highly-dynamic large-scale systems such as smart MG and 5G RAN. In this thesis, we have developed multiple state-of-the-art ML algorithms for various applications in energy management and trading, 5G resource allocation, network slicing, HetNet, etc. The proposed algorithms can overcome environmental uncertainties, and the simulations demonstrate the capabilities of ML techniques to reduce system cost, enhance operation efficiency, and improve key performance indicators. In the following, we summarize the main findings in detail.

9.1 Multi-agent CEQ for Decentralized MG Energy Management

We propose a multi-agent CEQ algorithm for MG energy management. The simulations show that CEQ can successfully coordinate the operation of multiple agents. Compared with the Q-learning without correlation, we find that the proposed correlated Q-learning scheme could save at most 19.3% cost for the DSM agent, and earn at most 44.2% more benefit for the ESS agent. In addition, CEQ maintains the same total benefit for the agents compared with the centralized approach.

9.2 Multi-agent BA-DRL for Microgrid Energy Management under Communication Failures

MG energy management, i.e., the CEQ algorithm proposed in Chapter 3, is vulnerable to communication failures, which may degrade the MG system performance. To this end, in Chapter 4, we proposed a BA-DRL algorithm for MG energy management under imperfect communications, where a multi-agent POMDP is defined, and the Bayesian method is applied for action estimation and belief update. The proposed method is compared with Nash-DQN and ADMM, and a higher profit for PV and ESS agents as well as a lower cost for the DSM agent are observed under different communication failure probabilities.

9.3 Multi-agent CEQ for 5G RAN Slicing

5G and beyond 5G networks will serve heterogeneous users of multiple slices, which calls for novel schemes of network slicing and resource allocation. ML techniques provide promising alternatives to the existing schemes. Chapter 5 proposed a RAN slicing-based resource allocation method for 5G, namely COQRA, to allocate radio resources to eMBB and URLLC slices. The proposed algorithm is compared with the Nash Q-learning, LRTQ method and PPF algorithm. Simulation results show that the proposed COQRA scheme achieves the best overall performance.

9.4 TRL-based Radio and Computation Resource Allocation in 5G RAN Slicing

Compared with single radio resource allocation in Chapter 5, the evolving network architecture requires more efficient solutions for joint resource allocation. Although RL techniques are widely used to improve network performance, they suffer from a long convergence time and lack of generalization. Therefore, Chapter 6 proposed a KTRA method for joint radio and computation resource allocation. Compared with existing works, the main difference is that the proposed KTRA algorithm has a knowledge transfer capability. The KTRA method is compared with Q-learning-based resource allocation, and KTRA presents an 18.4% lower URLLC delay, 30.1% higher eMBB throughput, and faster convergence.

9.5 DTRL-based Joint Radio and Cache Resource Allocation in 5G RAN Slicing

The proposed KTRA algorithm in Chapter 6 is limited in tabular-based learning, and we further explore the knowledge transfer for DRL. Chapter 7 introduces two novel DTRL-based solutions for the joint radio and cache resource allocation, namely ADTRL and QDTRL. They can utilize the former experience of experts to improve the exploration efficiency on target tasks. The proposed algorithms have been compared with two baseline algorithms via simulations. The results show that the proposed methods achieve better network performance and faster convergence speeds than these other benchmarks.

9.6 Co-HDRL based Joint Sleep, Power, and RIS Control for Energy-Efficient HetNet

RISs are considered to be a promising technique for envisioned 6G networks. Chapter 8 proposed a Co-HDRL algorithm for RIS-assisted energy efficient RAN, including a cross-entropy enabled policy for the meta-controller and cooperative action selections for sub-controllers. Besides, an FP-based RIS control strategy is introduced for phase-shift design. The proposed architecture enables joint sleep, power, and RIS control with different timescales, increasing the flexibility for RAN management. This is particularly important for 6G and O-RAN where intelligence will be more abundant and control will span several time scales. The proposed method is compared with conventional hierarchical deep reinforcement learning and surrogate optimization-based RIS control, and the simulations show that the proposed algorithms achieve better energy efficiency.

9.7 Challenges and Open Issues

Machine learning has achieved great success in many fields. It is believed that newly-emerged machine learning techniques, such as federated learning and graph learning, will further contribute to the evolution of the smart grid and intelligent wireless communications. However, there are still some challenges and open issues for research efforts.

1) Hyperparameter selection and tuning

Hyperparameter selection and tuning are one of the most critical issues of machine learning techniques, which are closely related to algorithm design and training. For instance,

in DQN, the network learning rate, training frequency, and experience pool size require sophisticated design and tuning. The difficulty of selecting these hyperparameters may prevent applying advanced machine learning algorithms to smart grid and wireless communications. The grid search method is widely used for hyperparameter tuning, but it relies on the researcher's experience and lacks generalization capabilities. The bayesian method is considered a more advanced approach for hyperparameter tuning, which applies Bayesian rules to update the next candidate model, but the complexity can be an issue.

2) Standardization of machine learning algorithm applications

There have been many existing studies that apply various ML techniques to smart grid and wireless communications. However, these studies are mostly simulation-based, and a true gap exists between simulations and real applications. Standardization is important to fill this gap and advance real-environment applications. For example, the O-RAN alliance defined an intelligent radio controller for RAN, and 3GPP introduced the network data analytics function to incorporate intelligence into resource management. However, these schemes are limited to system levels, and no algorithm-level details are involved.

3) User intelligence in future wireless communications

In most existing studies, user devices act passively to receive the control command from network operators. Such centralized management can be efficient, but it neglects the specific demand of various user devices. By contrast, user intelligence indicates that user devices can propose heterogeneous service requirements to the network operator. On the one hand, this user-centric management architecture can greatly improve the user experience by providing highly user-customized services. Each user device can intelligently require a specific service level based on its unique demand. On the other hand, user intelligence relies on computational resources to implement machine learning algorithms for decision-making. Unfortunately, the limited computational capability of end users' devices may prevent deploying advanced machine learning algorithms.

4) Knowledge transfer for machine learning algorithms

Knowledge transfer is considered an alternative approach to ease the algorithm training efforts. It utilizes the existing well-trained algorithms' knowledge to achieve a jump-start in target algorithm training. Consequently, transfer learning can significantly reduce the algorithm training samples by reusing prior knowledge. For instance, one microgrid can learn the energy trading knowledge of another microgrid to shorten the exploration period, and one base station can extract the knowledge of other base stations to adapt to the dynamic wireless environment. However, existing knowledge cannot be directly applied to target algorithms due to task differences, and mapping functions become the most critical part of transfer learning. We proposed multiple manually defined mapping functions, but a unified knowledge mapping architecture is still an open challenge.

Appendix A

Mathematical Proofs

A.1 Proof of LSTM network complexity

The computational complexity of BA-DRL is dominated by the LSTM complexity. Based on the LSTM architecture defined in [183], let M , C , I and K represent the numbers of memory blocks, memory cells in each block, input nodes, and output nodes, respectively. The computational complexity per iteration is given by:

$$\begin{aligned} O(T_{LSTM}) &= O(T_r + T_i + T_o) \\ &= O_r((M(C + 2C + 1))(M(C + 2)) + M(2C + 1)) \\ &\quad + O_i((MC + 1)K) + O_o((M(C + 2C + 1)I) \\ &= O_r(3M^2C^2 + 7M^2C + 2M^2 + 2MC + M) \\ &\quad + O_i(MCK + K) + O_o(3MCI + MI) \\ &= O_r(3M^2C^2) + O_i(MCK) + O_o(3MCI) \\ &= O((MC)^2), \end{aligned}$$

where T_r denotes the run time for recurrent connections and bias. T_i and T_o denote the updating time of input and output nodes, respectively.

A.2 Proof of cache hit ratio

In the following we prove equation (10). Recalling equation (7.6) and (7.7)

$$h_{j,n,g} = \sum_{m \in \mathcal{M}_{j,n}} \phi_{j,n,m,g} \mathcal{T}_{j,n,m,g},$$

$$\sum_{g \in \mathcal{G}_n} \sum_{m \in \mathcal{M}_n} h_{j,n,m,g} = \frac{C_{j,n}}{C_{j,T}}.$$

Then we have

$$h_{j,n,g} \sum_{g \in \mathcal{G}_n} \sum_{m \in \mathcal{M}_n} h_{j,n,m,g} = \frac{C_{j,n}}{C_{j,T}} \sum_{m \in \mathcal{M}_{j,n}} \phi_{j,n,m,g} \mathcal{T}_{j,n,m,g},$$

which can be easily transformed to

$$\begin{aligned} h_{j,n,g} &= \frac{C_{j,n}}{C_{j,T}} \frac{\sum_{m \in \mathcal{M}_{j,n}} \phi_{j,n,m,g} \mathcal{T}_{j,n,m,g}}{\sum_{g \in \mathcal{G}_n} \sum_{m \in \mathcal{M}_n} h_{j,n,m,g}} \\ &= \frac{C_{j,n}}{C_{j,T}} \frac{\mathcal{T}_{j,n,g} \sum_{m \in \mathcal{M}_{j,n}} \phi_{j,n,m,g}}{\sum_{g \in \mathcal{G}_n} \sum_{m \in \mathcal{M}_n} h_{j,n,m,g}} \\ &\quad \text{(using } \mathcal{T}_{j,n,m,g} = \mathcal{T}_{j,n,g} \text{ assumption)} \\ &= \frac{C_{j,n}}{C_{j,T}} \frac{\mathcal{T}_{j,n,g} \sum_{m \in \mathcal{M}_{j,n}} \phi_{j,n,m,g}}{\sum_{g \in \mathcal{G}_n} \sum_{m \in \mathcal{M}_n} \phi_{j,n,m,g} \mathcal{T}_{j,n,m,g}} \\ &\quad \text{(using } h_{j,n,m,g} = \phi_{j,n,m,g} \mathcal{T}_{j,n,m,g}) \\ &= \frac{C_{j,n}}{C_{j,T}} \frac{\sum_{m \in \mathcal{M}_{j,n}} \phi_{j,n,m,g}}{\sum_{g \in \mathcal{G}_n} \sum_{m \in \mathcal{M}_n} \phi_{j,n,m,g}} \\ &= \frac{C_{j,n}}{C_{j,T}} \frac{\sum_{m \in \mathcal{M}_{j,n}} \phi_{j,n,m,g}}{\sum_{m \in \mathcal{M}_n} \phi_{j,n,m}}, \end{aligned}$$

which is the equation (7.8).

A.3 Proof of Proposition 1

In equation (8.9), when ψ_k is fixed, note that f_2 is a differentiable concave function for β with $\frac{\partial^2 f_2}{\partial \beta^2} < 0$. Then we have:

$$\begin{aligned} \frac{\partial f_2(\Theta_m, \beta)}{\partial \beta_k} &= \frac{\partial \left(b_k \log(1 + \beta_k) - b_k \beta_k + \frac{b_k(1 + \beta_k)\psi_k}{1 + \psi_k} \right)}{\partial \beta_k} \\ &= \frac{b_k}{1 + \beta_k} - b_k + \frac{b_k \psi_k}{1 + \psi_k} \\ &= b_k \left(\frac{\psi_k}{1 + \psi_k} - \frac{\beta_k}{1 + \beta_k} \right) \\ &= b_k \frac{\psi_k - \beta_k}{(1 + \psi_k)(1 + \beta_k)} \end{aligned}$$

Setting $\frac{\partial f_2}{\partial \beta_k}$ and obviously $\beta_k^* = \psi_k$, where β_k^* is the optimal β_k . Substituting β_k^* back to equation (8.9) will recover (8.8). As such, the optimal values of these two equations are equivalent.

A.4 Proof of Proposition 2

In equation (8.12), for ease of notations, we use A_k to represent $\sqrt{b_k(1 + \beta_k)}(\hat{\Theta} \mathbf{V}_{b,k})$, and B_k to represent $\sum_{b' \in \mathcal{B}} |\hat{\Theta} \mathbf{V}_{b',k}|^2 + N_0^2$. Then, we can rewrite equation (8.13) as $2\text{Re}\{\eta_k^\dagger A_k\} - \eta_k^\dagger B_k \eta_k$, which can be reformulated by $\eta_k^\dagger A_k + A_k^\dagger \eta_k - \eta_k^\dagger B_k \eta_k$. Then we rewrite it as $A_k^\dagger B_k^{-1} A_k - (\eta_k^\dagger - B_k^{-1} A_k^\dagger) B_k (\eta_k - B_k^{-1} A_k)$. It is obvious that the optimal value is $\eta_k^* = B_k^{-1} A_k$ which means:

$$\eta_k^* = \frac{\sqrt{b_k(1 + \beta_k)} \hat{\Theta} \mathbf{V}_{b,k}}{\sum_{b' \in \mathcal{B}} |\hat{\Theta} \mathbf{V}_{b',k}|^2 + N_0^2}.$$

Then substituting η_k^* back to equation (8.13) will get the optimal value as $\sum_{k \in \mathcal{K}} \frac{b_k(1 + \beta_k) |\hat{\Theta} \mathbf{V}_{b,k}|^2}{\sum_{b' \in \mathcal{B}} |\hat{\Theta} \mathbf{V}_{b',k}|^2 + N_0^2}$, which is exactly the objective of equation (8.12). As such, the equivalence is demonstrated.

Bibliography

- [1] M. I. Jordan and T. M. Mitchell, “Machine learning: Trends, perspectives, and prospects,” *Science*, vol. 349, pp. 255–260, Jun. 2015.
- [2] Y. Sun, M. Peng, Y. Zhou, Y. Huang, and S. Mao, “Application of machine learning in wireless networks: Key techniques and open issues,” *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3072–3108, 2019.
- [3] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, “Machine learning for wireless networks with artificial intelligence: A tutorial on neural networks,” *arXiv preprint arXiv:1710.02913*, vol. 9, 2017.
- [4] Y. Yoldas, A. Onen, S. Mueen, A. Vasilakos, and I. Alan, “Enhancing smart grid with microgrids: Challenges and opportunities,” *Renewable and Sustainable Energy Reviews*, vol. 72, pp. 205–214, May. 2017.
- [5] E. Mengelkamp, J. Gärttner, K. Rock, S. Kessler, L. Orsini, and C. Weinhardt, “Designing microgrid energy markets: A case study: The brooklyn microgrid,” *Applied Energy*, vol. 210, pp. 870–880, Jan. 2018.
- [6] C. Zhang, J. Wu, Y. Zhou, M. Cheng, and C. Long, “Peer-to-peer energy trading in a microgrid,” *Applied Energy*, vol. 220, pp. 1–12, Jun. 2018.
- [7] L. Mariam, M. Basu, and M. F. Conlon, “Microgrid: Architecture, policy and future trends,” *Renewable and Sustainable Energy Reviews*, vol. 64, pp. 477–489, Oct. 2016.
- [8] G. Chen, , and Q. Yang, “An ADMM-based distributed algorithm for economic dispatch in islanded microgrids,” *IEEE Transaction on Industrial Informatics*, vol. 14, pp. 3892–3903, Sep. 2018.

- [9] A. Kantamneni, L. E. Brown, G. Parker, and W. W. Weaver, "Survey of multi-agent systems for microgrid control," *Engineering Applications of Artificial Intelligence*, vol. 45, pp. 192–203, Oct. 2015.
- [10] T. Morstyn and M. D. McCulloch, "Multiclass energy management for peer-to-peer energy trading driven by prosumer preferences," *IEEE Transaction on Power Systems*, vol. 34, pp. 4005–4014, Sep. 2019.
- [11] C. Tu, X. He, Z. Shuai, and F. Jiang, "Big data issues in smart grid – A review," *Renewable and Sustainable Energy Reviews*, vol. 79, pp. 1099–1107, Nov. 2017.
- [12] X. Xu, Y. Jia, Y. Xu, Z. Xu, S. Chai, and C. S. Lai, "A multi-agent reinforcement learning-based data-driven method for home energy management," *IEEE Trans. on Smart Grid*, vol. 11, pp. 3201–3211, Jul. 2020.
- [13] X. Lu, X. Xiao, L. Xiao, C. Dai, M. Peng, and H. V. Poor, "Reinforcement learning-based microgrid energy trading with a reduced power plant schedule," *IEEE Internet of Things Journal*, vol. 6, pp. 10728–10737, Dec. 2019.
- [14] G. Gao, Y. Wen, and D. Tao, "Distributed energy trading and scheduling among microgrids via multiagent reinforcement learning," *IEEE Transactions on Neural Networks and Learning Systems (Early Access)*, May 2022.
- [15] M. Kuzlu, M. Pipattanasomporn, and S. Rahman, "Communication network requirements for major smart grid applications in HAN, NAN and WAN," *Computer Networks*, vol. 67, p. 74–88, Jul. 2014.
- [16] X. Yang, Y. Zhang, H. Wu, and H. He, "An event-driven ADR approach for residential energy resources in microgrids with uncertainties," *IEEE Transaction on Industrial Informatics*, vol. 66, pp. 5275–5288, Jul. 2019.
- [17] S. Marzal, R. Salas, R. G. Medina, G. Garcerá, and E. Figueres, "Current challenges and future trends in the field of communication architectures for microgrids," *Renewable and Sustainable Energy Reviews*, vol. 82, pp. 3610–3622, Feb. 2018.
- [18] H. Zhou and M. Erol-Kantarci, "Decentralized microgrid energy management: A multi-agent correlated q-learning approach," in *Proceedings of 2020 IEEE conference on Smart Grid Communication*, pp. 1–7, Nov. 2020.
- [19] H. Zhou and M. Erol-Kantarci, "Correlated deep q-learning based microgrid energy management," in *Proceedings of 2020 IEEE International Workshop CAMAD*, pp. 1–6, Sep. 2020.

- [20] H. Zhou, A. Aral, I. Brandic, and M. Erol-Kantarci, “Multi-agent bayesian deep reinforcement learning for microgrid energy management under communication failures,” *IEEE Internet of Things Journal (Early Access)*, Dec. 2021.
- [21] M. Shafi, A. Molisch, P. Smith, T. Haustein, P. Zhu, P. Silva, F. Tufvesson, A. Benjebbour, and G. Wunder, “5g: A tutorial overview of standards, trials, challenges, deployment, and practice,” *IEEE Journal on Selected Areas in Communications*, vol. 35, pp. 1201–1221, Jun. 2017.
- [22] M. Shafi, A. F. Molisch, P. J. Smith, T. Haustein, P. Zhu, P. De Silva, F. Tufvesson, A. Benjebbour, and G. Wunder, “5G: A tutorial overview of standards, trials, challenges, deployment, and practice,” *IEEE journal on selected areas in communications*, vol. 35, no. 6, pp. 1201–1221, 2017.
- [23] V. W. Wong, R. Schober, D. W. K. Ng, and L.-C. Wang, *Key technologies for 5G wireless systems*. Cambridge university press, 2017.
- [24] 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. 55765–55779, Sep 2018.
- [25] H. Xiang, M. Peng, Y. Sun, and S. Yan, “Mode selection and resource allocation in sliced fog radio access networks: A reinforcement learning approach,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4271–4284, 2020.
- [26] Y. Hua, R. Li, Z. Zhao, X. Chen, and H. Zhang, “GAN-powered deep distributional reinforcement learning for resource management in network slicing,” *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 334–349, 2020.
- [27] A. Ksentini and N. Nikaein, “Toward enforcing network slicing on RAN: Flexibility and resources abstraction,” *IEEE Communications Magazine*, vol. 55, pp. 102–108, Jun. 2017.
- [28] C. Jiang, H. Zhang, Y. Ren, Z. Han, K.-C. Chen, and L. Hanzo, “Machine learning paradigms for next-generation wireless networks,” *IEEE Wireless Communications*, vol. 24, pp. 98–105, Apr. 2017.
- [29] ETSI, “Artificial intelligence and future directions for ETSI,” *ETSI White Paper*, Jun. 2020.

- [30] 3GPP, “5G system; Network data analytics services; Stage 3 (Release 16),” *Technical specification 29.520*, Aug. 2020.
- [31] R. Shafin, L. Liu, V. Chandrasekhar, H. Chen, J. Reed, and J. Zhang, “Artificial intelligence-enabled cellular networks: A critical path to beyond-5G and 6G,” *IEEE Wireless Communications*, vol. 27, pp. 212–217, Apr. 2020.
- [32] C. Wang, M. D. Renzo, S. Stanczak, S. Wang, and E. G. Larsson, “Artificial intelligence enabled wireless networking for 5G and beyond: Recent advances and future challenges,” *IEEE Wireless Communications*, vol. 27, pp. 16–23, Feb. 2020.
- [33] D. M. Gutierrez-Estevez, M. Gramaglia, A. D. Domenico, G. Dandachi, S. Khatibi, D. Tsolkas, I. Balan, A. Garcia-Saavedra, U. Elzur, and Y. Wang, “Artificial intelligence enabled wireless networking for 5G and beyond: Recent advances and future challenges,” *IEEE Wireless Communications*, vol. 26, pp. 134–141, Oct. 2019.
- [34] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, p. 1345–1359, Oct. 2010.
- [35] H. Zhou, M. Elsayed, and M. Erol-Kantarci, “Ran resource slicing in 5G using multi-agent correlated q-learning,” in *Proceedings of 2021 IEEE conference on PIMRC*, pp. 1–6, Sep. 2021.
- [36] H. Zhou and M. Erol-Kantarci, “Knowledge transfer based radio and computation resource allocation for 5G RAN slicing,” in *Proceedings of 2020 IEEE CCNC*, pp. 1–6, Jan. 2022.
- [37] H. Zhou, M. Erol-Kantarci, and V. Poor, “Learning from peers: Deep transfer reinforcement learning for joint radio and cache resource allocation in 5G RAN slicing,” *IEEE Transactions on Cognitive Communications and Networking (Early Access)*, pp. 1–17, Sep. 2022.
- [38] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 2018.
- [39] V. Mnih, K. Kavukcuoglu, D. Silver, and et al., “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, p. 529–533, Jan. 2015.
- [40] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “A brief survey of deep reinforcement learning,” *arXiv preprint arXiv:1708.05866*, 2017.

- [41] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, “An introduction to deep reinforcement learning,” *arXiv preprint arXiv:1811.12560*, 2018.
- [42] S. Chen, Z. Yao, X. Jiang, J. Yang, and L. Hanzo, “Multi-agent deep reinforcement learning based cooperative edge caching for ultra-dense next-generation networks,” *IEEE Transactions on Communications*, vol. 59, pp. 2441–2456, Apr 2021.
- [43] L. Busoniu, R. Babuska, and B. D. Schutter, “A comprehensive survey of multiagent reinforcement learning,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, pp. 156–172, Mar. 2008.
- [44] A. Kaur and K. Kumar, “Energy-efficient resource allocation in cognitive radio networks under cooperative multi-agent model-free reinforcement learning schemes,” *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1337–1348, 2020.
- [45] H. Shayeghi, E. Shahryari, M. Moradzadeh, and P. Siano, “A survey on microgrid energy management considering flexible energy sources,” *Energies*, vol. 12, p. 1–26, May. 2019.
- [46] M. Sadeghi and M. Erol-Kantarci, “Power loss minimization in microgrids using bayesian reinforcement learning with coalition formation,” in *Proc. of IEEE Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 1–6, Sep. 2019.
- [47] T. Chen and S. Bu, “Realistic peer-to-peer energy trading model for microgrids using deep reinforcement learning,” in *Proc. of IEEE PES Innovative Smart Grid Technologies Europe*, pp. 1–5, Sep. 2019.
- [48] L. Yu, W. Xie and D. Xie, Y. Zou, D. Zhang, Z. Sun, L. Zhang, Y. Zhang, and T. Jiang, “Deep reinforcement learning for smart home energy management,” *IEEE Internet of Things Journal*, vol. 7, pp. 2751–2762, Dec. 2019.
- [49] V. Bui, A. Hussain, and H. Kim, “Double deep q-learning-based distributed operation of battery energy storage system considering uncertainties,” *IEEE Trans. Smart Grid*, vol. 11, p. 457 – 469, Jan. 2020.
- [50] J. Vardakas, I. Zenginis, N. Zorba, C. Echave, M. Morató, and C. Verikoukis, “Electrical energy savings through efficient cooperation of urban buildings: the smart community case of superblocks’ in barcelona,” *IEEE Communications Magazine*, vol. 56, p. 102 – 109, Nov. 2018.

- [51] E. Foruzan, L. Soh, and S. Asgarpour, "Reinforcement learning approach for optimal distributed energy management in a microgrid," *IEEE Transaction on Power System*, vol. 33, pp. 5749–5758, Sep. 2018.
- [52] W. Liu, P. Zhuang, H. Liang, J. Peng, and Z. Huang, "Distributed economic dispatch in microgrids based on cooperative reinforcement learning," *IEEE Transaction on Neural Networks and Learning Systems*, vol. 29, pp. 2192–2203, Mar. 2018.
- [53] I. Zenginis, J. S. Vardakas, J. Abadal, C. Echave, M. Morat 'o, and C. Verikoukis, "Optimal power equipment sizing and management for cooperative buildings in microgrids," *IEEE Transaction on Industrial Informatics*, vol. 15, pp. 158–172, Jan. 2019.
- [54] M. Sadeghi, S. Mollahasani, and M. Erol-Kantarci, "Power loss-aware transactive microgrid coalitions under uncertainty," *Energies*, vol. 13, pp. 1–12, Oct. 2020.
- [55] E. Foruzan, L. Soh, and S. Asgarpour, "Reinforcement learning approach for optimal distributed energy management in a microgrid," *IEEE Transaction on Power Systems*, vol. 33, pp. 5749–5758, Sep. 2018.
- [56] P. Kofinas, A. I. Dounisb, and G. A. Vourosa, "Fuzzy Q-learning for multi-agent decentralized energy management in microgrids," *Applied Energy*, vol. 219, pp. 53–67, Jun. 2018.
- [57] W. Liu, P. Zhuang, H. Liang, J. Peng, and Z. Huang, "Distributed economic dispatch in microgrids based on cooperative reinforcement learning," *IEEE Transaction on Neural Networks and Learning Systems*, vol. 29, pp. 2192–2203, Mar. 2018.
- [58] L. Xiao, X. Xiao, C. Dai, M. Peng, L. Wang, and H. Poor, "Reinforcement learning-based energy trading for microgrids," *arXiv:1801.06285 [cs.SY]*, pp. 1–8, Jan. 2018.
- [59] B. Huang, L. Liu, H. Zhang, Y. Li, and Q. Sun, "Distributed optimal economic dispatch for microgrids considering communication delays," *IEEE Transaction on Systems, Man, and Cybernetics: Systems*, vol. 49, pp. 1634–1642, Aug. 2019.
- [60] G. Chen and Z. Zhao, "Delay effects on consensus-based distributed economic dispatch algorithm in microgrid," *IEEE Transaction on Power systems*, vol. 33, pp. 602–612, Jan. 2018.
- [61] S. Manap, K. Dimyati, M. N. Hindia, M. S. A. Talip, and R. Tafazolli, "Survey of radio resource management in 5G heterogeneous networks," *IEEE Access*, vol. 8, pp. 131202–131223, 2020.

- [62] H. Chergui and C. Verikoukis, "Offline SLA-constrained deep learning for 5G networks reliable and dynamic end-to-end slicing," *IEEE Journal on Selected Areas in Communications*, vol. 38, pp. 350–360, Feb. 2020.
- [63] R. C. Staudemeyer and E. R. Morris, "Understanding LSTM-a tutorial into long short-term memory recurrent neural networks," *arXiv preprint arXiv:1909.09586*, 2019.
- [64] M. A. Nielsen, *Neural networks and deep learning*, vol. 25. Determination press San Francisco, CA, USA, 2015.
- [65] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: LSTM cells and network architectures," *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.
- [66] J. Cui, Z. Ding, P. Fan, and N. Al-Dhahir, "Unsupervised machine learning-based user clustering in millimeter-wave-NOMA systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 11, pp. 7425–7440, 2018.
- [67] M. Elsayed, M. Erol-Kantarci, and H. Yanikomeroglu, "Transfer reinforcement learning for 5G new radio mmwave networks," *IEEE Transactions on Wireless Communications*, vol. 20, pp. 2838–2849, May. 2021.
- [68] Y. Zhou, F. Tang, Y. Kawamoto, and N. Kato, "Reinforcement learning-based radio resource control in 5G vehicular network," *IEEE Wireless Communications Letters*, vol. 9, no. 5, pp. 611–614, 2019.
- [69] H. H. Yang, Z. Liu, T. Q. S. Quek, and H. V. Poor, "Scheduling policies for federated learning in wireless networks," *IEEE Transactions on Communications*, vol. 68, pp. 317–333, Jan. 2020.
- [70] H. Wang, Y. Wu, G. Min, and W. Miao, "A graph neural network-based digital twin for network slicing management," *IEEE Transactions on Industrial Informatics*, vol. 18, pp. 1367–1376, Feb. 2022.
- [71] N. Zhao, Y. Liang, D. Niyato, Y. Pei, M. Wu, and Y. Jiang, "Deep reinforcement learning for user association and resource allocation in heterogeneous cellular networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 11, pp. 5141–5152, 2019.

- [72] X. Liao, J. Shi, Z. Li, L. Zhang, and B. Xia, "A model-driven deep reinforcement learning heuristic algorithm for resource allocation in ultra-dense cellular networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 1, pp. 983–997, 2020.
- [73] G. Pocovi, K. Pedersen, and P. Mogensen, "Joint link adaptation and scheduling for 5G ultra-reliable low-latency communications," *IEEE Access*, vol. 6, pp. 28912–28922, May. 2018.
- [74] M. Alsenwi, N. Tran, M. Bennis, A. Bairagi, and C. Hong, "eMBB-URLLC resource slicing: A risk-sensitive approach," *IEEE Communications Letters*, vol. 23, pp. 740–743, Apr. 2019.
- [75] S. R. Pandey, M. Alsenwi, Y. K. Tun, and C. S. Hong, "A downlink resource scheduling strategy for URLLC traffic," in *Proc. of IEEE Int. Conf. on Big Data and Smart Computing*, pp. 1–6, Feb. 2019.
- [76] M. Maule, P. Mekikis, K. Ramantas, J. Vardakas, and C. Verikoukis, "Dynamic partitioning of radio resources based on 5G RAN slicing," in *Proc. of 2020 IEEE Global Communications Conference*, pp. 1–7, Dec. 2020.
- [77] M. Elsayed and M. Erol-Kantarci, "AI-enabled radio resource allocation in 5G for URLLC and eMBB users," in *Proc. of 2019 IEEE 2nd 5G World Forum (5GWF)*, pp. 590–595, Sep. 2019.
- [78] I. Comsa, S. Zhang, M. Aydin, P. Kuonen, Y. Lu, R. Trestian, and G. Ghinea, "Towards 5G: A reinforcement learning-based scheduling solution for data traffic management," *IEEE Trans. on Network and Service Management*, vol. 15, pp. 1661–1675, Dec. 2018.
- [79] J. Zhang, X. Xu, K. Zhang, B. Zhang, X. Tao, and P. Zhang, "Machine learning based flexible transmission time interval scheduling for eMBB and uRLLC coexistence scenario," *IEEE Access*, vol. 7, pp. 65811–65820, May. 2019.
- [80] Y. Shi, Y. Sagduyu, and T. Erpek, "Reinforcement learning for dynamic resource optimization in 5G radio access network slicing," in *Proc. of 2020 IEEE International Workshop on CAMAD*, pp. 1–6, Sep. 2020.
- [81] T. Li, X. Zhu, and X. Liu, "An end-to-end network slicing algorithm based on deep q-learning for 5G network," *IEEE Access*, vol. 8, pp. 122229–122240, Jul. 2020.

- [82] Y. Huang, S. Li, C. Li, Y. Hou, and W. Lou, "A deep-reinforcement-learning-based approach to dynamic eMBB/URLLC multiplexing in 5G NR," *IEEE Internet of Things Journal*, vol. 7, pp. 6439–6456, Jul. 2020.
- [83] X. Chen, Y. Cai, L. Li, M. Zhao, B. Champagne, and L. Hanzo, "Energy-efficient resource allocation for latency-sensitive mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2246–2262, 2020.
- [84] Q. Pham, H. T. Nguyen, Z. Han, and W. Hwang, "Coalitional games for computation offloading in NOMA-enabled multi-access edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 1982–1993, 2020.
- [85] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, pp. 450–465, Feb. 2018.
- [86] J. Ren, G. Yu, Y. Cai, and Y. He, "Latency optimization for resource allocation in mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 17, no. 8, pp. 5506–5519, 2018.
- [87] A. Huang, Y. Li, Y. Xiao, X. Ge, S. Sun, and H. Chao, "Distributed resource allocation for network slicing of bandwidth and computational resource," in *Proceedings of 2020 IEEE International Conference on Communications*, pp. 1–6, Jun. 2020.
- [88] B. Xiang, J. Elias, F. Martignon, and E. D. Nitto, "Joint network slicing and mobile edge computing in 5G networks," in *Proceedings of 2019 IEEE International Conference on Communications*, pp. 1–6, Jul. 2019.
- [89] W. Wu, N. Chen, C. Zhou, M. Li, X. Shen, W. Zhuang, and X. Li, "Dynamic RAN slicing for service-oriented vehicular networks via constrained learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, pp. 2076–2089, Jul. 2021.
- [90] Q. Liu, T. Han, and E. Moges, "Edgeslice: Slicing wireless edge computing network with decentralized deep reinforcement learning," in *Proceedings of 2020 IEEE International Conference on Distributed Computing Systems*, pp. 1–11, Dec. 2020.
- [91] Q. Ye, W. Shi, K. Qu, H. He, W. Zhuang, and X. Shen, "Joint RAN slicing and computation offloading for autonomous vehicular networks: A learning-assisted hierarchical approach," *IEEE Open Journal of Vehicular Technology*, vol. 2, pp. 272–288, Jun. 2021.

- [92] J. A. Ayala-Romero, A. Garcia-Saavedra, M. Gramaglia, X. Costa-Perez, A. Banchs, and J. J. Alcaraz, “vrAIn: Deep learning based orchestration for computing and radio resources in vRANs,” *IEEE Transactions on Mobile Computing*, vol. 21, pp. 2652–2670, Dec. 2020.
- [93] M. Erol-Kantarci, “Cache-At-Relay: Energy-efficient content placement for next-generation wireless relays,” *International Journal of Network Management*, vol. 25, pp. 454–470, Nov./Dec. 2015.
- [94] P. Yang, N. Zhang, S. Zhang, L. Yu, J. Zhang, and X. Shen, “Content popularity prediction towards location-aware mobile edge caching,” *IEEE Transactions on Multimedia*, vol. 21, pp. 915–929, Apr. 2019.
- [95] J. Kwak, Y. Kim, L. B. Le, and S. Chong, “Hybrid content caching in 5G wireless networks: Cloud versus edge caching,” *IEEE Transactions on Wireless Communications*, vol. 17, pp. 3030–3045, May. 2018.
- [96] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, “Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning,” *IEEE Internet of Things Journal*, vol. 6, pp. 4005–4018, Jun. 2019.
- [97] W. Lei, Y. Ye, and M. Xiao, “Deep reinforcement learning-based spectrum allocation in integrated access and backhaul networks,” *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, pp. 970–979, Sep. 2020.
- [98] H. Albinsaid, K. Singh, S. Biswas, and C. Li, “Multi-agent reinforcement learning based distributed dynamic spectrum access,” *IEEE Transactions on Cognitive Communications and Networking*, vol. 8, pp. 1174–1185, Oct. 2021.
- [99] J. Li, X. Zhang, J. Zhang, J. W. Q. Sun, and Y. Xie, “Deep reinforcement learning-based mobility-aware robust proactive resource allocation in heterogeneous networks,” *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, pp. 408–421, Mar. 2020.
- [100] C. Zhang, M. Dong, and K. Ota, “Fine-grained management in 5G: DQL based intelligent resource allocation for network function virtualization in C-RAN,” *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, pp. 428–435, Jun. 2020.

- [101] 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 Veh. Technol. Mag.*, vol. 14, pp. 28–41, Sep. 2019.
- [102] E. Basar, M. D. Renzo, J. D. Rosny, M. Debbah, M.-S. Alouini, and R. Zhang, “Wireless communications through reconfigurable intelligent surfaces,” *IEEE Access*, vol. 7, pp. 116753–116773, Aug. 2019.
- [103] E. Björnson, H. Wymeersch, B. Matthiesen, P. Popovski, L. Sanguinetti, and E. de Carvalho, “Reconfigurable intelligent surfaces: A signal processing perspective with wireless applications,” *IEEE Signal Process. Mag.*, vol. 39, pp. 135–158, Feb. 2022.
- [104] M. D. Renzo, A. Zappone, M. Debbah, M.-S. Alouini, C. Yuen, J. de Rosny, and S. Tretyakov, “Smart radio environments empowered by reconfigurable intelligent surfaces: How it works, state of research, and the road ahead,” *IEEE J. Sel. Areas Commun.*, vol. 38, pp. 2450–2525, Jul. 2020.
- [105] R. Su, L. Dai, J. Tan, M. Hao, and R. MacKenzie, “Capacity enhancement for irregular reconfigurable intelligent surface-aided wireless communications,” in *Proc. IEEE Global Commun. Conf.*, pp. 1–6, Dec. 2020.
- [106] C. Pan, H. Ren, K. Wang, W. Xu, M. ElKashlan, A. Nallanathan, and L. Hanzo, “Multicell MIMO communications relying on intelligent reflecting surfaces,” *IEEE Trans. Wireless Commun.*, vol. 19, pp. 5218–5233, Oct. 2020.
- [107] S. Zhang and R. Zhang, “Capacity characterization for intelligent reflecting surface aided MIMO communication,” *IEEE J. Sel. Areas Commun.*, vol. 38, pp. 1823–1838, Jun. 2020.
- [108] Y. Zhang, C. Zhong, Z. Zhang, and W. Lu, “Sum rate optimization for two way communications with intelligent reflecting surface,” *IEEE Commun. Lett.*, vol. 24, pp. 1090–1094, Mar. 2020.
- [109] M.-M. Zhao, Q. Wu, M.-J. Zhao, and R. Zhang, “Intelligent reflecting surface enhanced wireless networks: Two-timescale beamforming optimization,” *IEEE Trans. Wireless Commun.*, vol. 20, pp. 2–17, Jan. 2021.
- [110] J. Yuan, Y.-C. Liang, J. Joung, G. Feng, and E. G. Larsson, “Intelligent reflecting surface-assisted cognitive radio system,” *IEEE Trans. Commun.*, vol. 69, pp. 675 – 687, Jan. 2021.

- [111] B. Di, H. Zhang, L. Li, L. Song, Y. Li, and Z. Han, "Practical hybrid beamforming with finite-resolution phase shifters for reconfigurable intelligent surface based multi-user communications," *IEEE Trans. Vehi. Technol.*, vol. 69, pp. 4565 – 4570, Apr. 2020.
- [112] H. Guo, Y.-C. Liang, J. Chen, and E. G. Larsson, "Weighted sum-rate maximization for reconfigurable intelligent surface aided wireless networks," *IEEE Trans. Wireless Commun.*, vol. 19, pp. 3064–3076, May 2020.
- [113] C. You, B. Zheng, and R. Zhang, "Intelligent reflecting surface with discrete phase shifts: Channel estimation and passive beamforming," in *Proc. IEEE Int. Conf. Commun. (ICC)*, pp. 1–6, Jun. 2020.
- [114] X. Mu, Y. Liu, L. Guo, J. Lin, and N. Al-Dhahir, "Exploiting intelligent reflecting surfaces in NOMA networks: Joint beamforming optimization," *IEEE Trans. Wireless Commun.*, vol. 19, pp. 6884–6898, Oct. 2020.
- [115] Y. Chen, Y. Wang, and L. Jiao, "Robust transmission for reconfigurable intelligent surface aided millimeter wave vehicular communications with statistical CSI," *IEEE Trans. Wireless Commun.*, vol. 21, pp. 928–944, Aug. 2021.
- [116] G. Yang, X. Xu, and Y.-C. Liang, "Intelligent reflecting surface assisted non-orthogonal multiple access," in *Proc. IEEE Wireless Commun. Netw. Conf (WCNC)*, pp. 1–6, May. 2020.
- [117] J. Dai, Y. Wang, C. Pan, K. Zhi, H. Ren, and K. Wang, "Reconfigurable intelligent surface aided massive MIMO systems with low-resolution DACs," *IEEE Commun. Lett.*, vol. 25, pp. 3124–3128, Sep. 2021.
- [118] K. Zhi, C. Pan, H. Ren, and K. Wang, "Power scaling law analysis and phase shift optimization of RIS-aided massive MIMO systems with statistical CSI," *IEEE Trans. Commun.*, vol. 70, pp. 3558–3574, May 2022.
- [119] K. Zhi, C. Pan, H. Ren, and K. Wang, "Statistical CSI-based design for reconfigurable intelligent surface-aided massive MIMO systems with direct links," *IEEE Wireless Commun. Lett.*, vol. 10, pp. 1128–1132, May 2021.
- [120] Z. Yang, M. Chen, W. Saad, W. Xu, M. Shikh-Bahaei, H. V. Poor, and S. Cui, "Energy-efficient wireless communications with distributed reconfigurable intelligent surfaces," *IEEE Trans. Wireless Commun.*, vol. 21, pp. 665–679, Jul. 2021.

- [121] S. Atapattu, R. Fan, P. Dharmawansa, G. Wang, J. Evans, and T. A. Tsiftsis, "Reconfigurable intelligent surface assisted two-way communications: Performance analysis and optimization," *IEEE Trans. Commun.*, vol. 68, pp. 6552–6567, Oct. 2020.
- [122] W. Ni, X. Liu, Y. Liu, H. Tian, and Y. Chen, "Resource allocation for multi-cell RIS-aided NOMA networks," *IEEE Trans. Wireless Commun.*, vol. 20, pp. 4253–4268, Jul. 2021.
- [123] A. Taha, Y. Zhang, F. B. Mismar, and A. Alkhateeb, "Deep reinforcement learning for intelligent reflecting surfaces: Towards standalone operation," in *Proc. Int. Workshop Signal Process. Adv. Wireless commun. (SPAWC)*, pp. 1–5, Aug. 2020.
- [124] K. Feng, Q. Wang, X. Li, and C.-K. Wen, "Deep reinforcement learning based intelligent reflecting surface optimization for MISO communication systems," *IEEE Wireless Commun. Lett.*, vol. 9, pp. 745–749, May 2020.
- [125] H. Yang, Z. Xiong, J. Zhao, D. Niyato, L. Xiao, and Q. Wu, "Deep reinforcement learning-based intelligent reflecting surface for secure wireless communications," *IEEE Trans. Wireless Commun.*, vol. 20, pp. 375–388, Jan. 2021.
- [126] Z. Yang, Y. Liu, Y. Chen, and J. T. Zhou, "Deep reinforcement learning for RIS-aided non-orthogonal multiple access downlink networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, pp. 1–6, Dec. 2020.
- [127] X. Guo, Y. Chen, and Y. Wang, "Learning-based robust and secure transmission for reconfigurable intelligent surface aided millimeter wave UAV communications," *IEEE Wireless Commun. Lett.*, vol. 10, pp. 1795–1799, Aug. 2021.
- [128] C. Huang, Z. Yang, G. C. Alexandropoulos, K. Xiong, L. Wei, C. Yuen, and Z. Zhang, "Hybrid beamforming for RIS-empowered multi-hop terahertz communications: A DRL-based method," in *Proc. IEEE Global Commun. Conf. (GLOBECOM Workshops)*, pp. 1–6, Dec. 2020.
- [129] G. C. Alexandropoulos, S. Samarakoon, M. Bennis, and M. Debbah, "Phase configuration learning in wireless networks with multiple reconfigurable intelligent surfaces," in *Proc. IEEE Global Commun. Conf. (GLOBECOM Workshops)*, pp. 1–6, Dec. 2020.
- [130] X. Hu, C. Masouros, and K.-K. Wong, "Reconfigurable intelligent surface aided mobile edge computing: From optimization-based to location-only learning-based solutions," *IEEE Trans. Commun.*, vol. 69, pp. 3709–3725, Mar. 2021.

- [131] N. T. Nguyen, L. V. Nguyen, T. Huynh-The, D. H. Nguyen, A. L. Swindlehurst, and M. Juntti, "Machine learning-based reconfigurable intelligent surface-aided MIMO systems," in *Proc. Int. Workshop Signal Process. Adv. Wireless commun. (SPAWC)*, pp. 101–105, Nov. 2021.
- [132] J. Gao, C. Zhong, X. Chen, H. Lin, and Z. Zhang, "Unsupervised learning for passive beamforming," *IEEE Commun. Lett.*, vol. 24, pp. 1052–1056, May 2020.
- [133] J. López, E. Pouresmaeil, C. Cañizares, K. Bhattacharya, A. Mosaddegh, and B. Solanki, "Smart residential load simulator for energy management in smart grids," *IEEE Transactions on Industrial Electronics*, vol. 66, pp. 1443–1452, Mar. 2018.
- [134] W. El-Baz and P. Tzscheutschler, "Short-term smart learning electrical load prediction algorithm for home energy management systems," *Applied energy*, vol. 147, pp. 10–19, Jun. 2015.
- [135] "European network of transmission system operators for electricity (ENTSO-E)." Available:<https://transparency.entsoe.eu/dashboard/show>.
- [136] Z. Wang and T. Hong, "Generating realistic building electrical load profiles through the generative adversarial network (GAN)," *Energy and Buildings*, vol. 224, pp. 1–15, Oct. 2020.
- [137] M. Razghandi, H. Zhou, M. Erol-Kantarci, and Turgut, "Smart home energy management: Sequence-to-Sequence load forecasting and q-learning," in *Proc. of IEEE Globecom*, pp. 1–6, Dec. 2021.
- [138] L. Tightiz, H. Yang, and M. J. Piran, "A survey on enhanced smart micro-grid management system with modern wireless technology contribution," *Energies*, vol. 13, pp. 1–21, May. 2020.
- [139] K. Shimotakahara, M. Elsayed, K. Hinzer, and M. Erol-Kantarci, "High-reliability multi-agent Q-learning-based scheduling for D2D microgrid communications," *IEEE Access*, vol. 7, pp. 74412–74421, 2019.
- [140] M. Elsayed, M. Erol-Kantarci, B. Kantarci, L. Wu, and J. Li, "Low-latency communications for community resilience microgrids: A reinforcement learning approach," *IEEE Transactions on Smart Grid*, vol. 11, no. 2, pp. 1091–1099, 2020.
- [141] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet of Things Journal*, vol. 4, pp. 1125–1142, Oct. 2017.

- [142] W. Li, T. Logenthiran, V.-T. Phan, and W. L. Woo, "Implemented iot-based self-learning home management system (SHMS) for singapore," *IEEE Internet of Things Journal*, vol. 5, pp. 2212–2219, Jun. 2018.
- [143] Y. Wu, X. Tan, L. Qian, D. H. K. Tsang, W. Song, and L. Yu, "Optimal pricing and energy scheduling for hybrid energy trading marketing future smart grid," *IEEE Transactions on Industrial Informatics*, vol. 11, pp. 1585–1596, Dec. 2015.
- [144] P. Shamsi, H. Xie, A. Longe, and J. Joo, "Economic dispatch for an agent-based community microgrid," *IEEE Transaction on Smart grid*, vol. 7, pp. 2317–2324, Oct. 2015.
- [145] L. Kaelbling, M. Littman, and A. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, pp. 99–134, May. 1998.
- [146] B. A. Frigyik and M. R. Gupta, "Introduction to the dirichlet distribution and related processes," tech. rep., University of Washington, 2010.
- [147] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, p. 1735–1780, Nov. 1997.
- [148] C.-X. Wang, F. Haider, X. Gao, X.-H. You, Y. Yang, D. Yuan, H. M. Aggoune, H. Haas, S. Fletcher, and E. Hepsaydir, "Cellular architecture and key technologies for 5G wireless communication networks," *IEEE communications magazine*, vol. 52, no. 2, pp. 122–130, 2014.
- [149] A. Gupta and R. K. Jha, "A survey of 5G network: Architecture and emerging technologies," *IEEE access*, vol. 3, pp. 1206–1232, 2015.
- [150] X. Xiong, K. Zheng, L. Lei, and L. Hou, "Resource allocation based on deep reinforcement learning in IoT edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1133–1146, 2020.
- [151] M. A. Qureshi and C. Tekin, "Fast learning for dynamic resource allocation in AI-enabled radio networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 1, pp. 95–110, 2020.
- [152] F. Zhou and R. Q. Hu, "Computation efficiency maximization in wireless-powered mobile edge computing networks," *IEEE Transactions on Wireless Communications*, vol. 19, no. 5, pp. 3170–3184, 2020.

- [153] J. Yan, S. Bi, Y. J. Zhang, and M. Tao, "Optimal task offloading and resource allocation in mobile-edge computing with inter-user task dependency," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 235–250, 2020.
- [154] Y. K. Tun, D. H. Kim, M. Alsenwi, N. H. Tran, Z. Han, and C. S. Hong, "Energy efficient communication and computation resource slicing for eMBB and URLLC coexistence in 5G and beyond," *IEEE Access*, vol. 8, pp. 136024–136035, Jul. 2020.
- [155] W. Zhan, C. Luo, G. Min, C. Wang, Q. Zhu, and H. Duan, "Mobility-aware multi-user offloading optimization for mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 3, pp. 3341–3356, 2020.
- [156] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, pp. 450–465, Feb. 2018.
- [157] P. L. Vo, M. N. Nguyen, T. A. Le, and N. H. Tran, "Slicing the edge: Resource allocation for RAN network slicing," *IEEE Wireless Communications Letters*, vol. 7, pp. 970–973, Dec. 2018.
- [158] N. C. Fofack, P. Nain, G. Neglia, and D. Towsley, "Analysis of ttl-based cache networks," in *Proceedings of the 6th International ICST Conference on Performance Evaluation Methodologies and Tools*, pp. 1–10, Oct. 2012.
- [159] T. Han and N. Ansari, "Network utility aware traffic load balancing in backhaul-constrained cache-enabled small cell networks with hybrid power supplies," *IEEE Transactions on Mobile Computing*, vol. 16, pp. 2819–2832, Oct. 2017.
- [160] M. E. Taylor, P. Stone, and Y. Liu, "Transfer learning via inter-task mappings for temporal difference learning," *Journal of Machine Learning Research*, vol. 8, pp. 2125–2167, Sep. 2007.
- [161] A. L. Gibbs and F. E. Su, "On choosing and bounding probability metrics," *International Statistical Review*, vol. 70, pp. 419–435, Dec. 2002.
- [162] P. S. Castro and D. Precup, "Using bisimulation for policy transfer in MDPs," in *Proceedings of 24th AAAI Conference on Artificial Intelligence*, pp. 1–6, Jul. 2010.
- [163] J. J. Taylor, D. Precup, and P. Panangaden, "Bounding performance loss in approximate MDP homomorphisms," in *Proceedings of Advances in Neural Information Processing Systems*, pp. 1–8, Dec. 2008.

- [164] M. James, *Topology*. Prentice Hall, 1999.
- [165] N. Ferns, P. Panangaden, and D. Precup, “Metrics for finite Markov decision processes,” in *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pp. 162–169, Jul. 2004.
- [166] A. A. Taha and A. Hanbury, “An efficient algorithm for calculating the exact hausdorff distance,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, pp. 2153–2163, Nov. 2015.
- [167] C. Huang, A. Zappone, G. C. Alexandropoulos, M. Debbah, and C. Yuen, “Reconfigurable intelligent surfaces for energy efficiency in wireless communication,” *IEEE Transactions on Wireless Communications*, vol. 18, pp. 4157–4170, Aug. 2019.
- [168] H. Zhang, B. Di, L. Song, and Z. Han, “Reconfigurable intelligent surfaces assisted communications with limited phase shifts: How many phase shifts are enough?,” *IEEE Transactions on Vehicular Technology*, vol. 69, pp. 4498–4502, Feb. 2020.
- [169] P. Ren and M. Tao, “A decentralized sleep mechanism in heterogeneous cellular networks with QoS constraints,” *IEEE Wireless Communications Letters*, vol. 3, pp. 509–512, Oct. 2014.
- [170] K. Shen and W. Yu, “Fractional programming for communication systems—part i: Power control and beamforming,” *IEEE Transactions on Signal Processing*, vol. 66, pp. 2616–2630, May 2018.
- [171] S. Pateria, B. Subagdja, A. Tan, and C. Quek, “Hierarchical reinforcement learning: A comprehensive survey,” *ACM Computing Surveys*, vol. 54, pp. 1–35, Jun. 2022.
- [172] G. Auer, V. Giannini, C. Desset, and et.al, “How much energy is needed to run a wireless network?,” *IEEE Wireless Commun.*, vol. 18, pp. 40–49, Oct. 2011.
- [173] M. H. Buysse, K. Mets, and S. Latré, “Hierarchical reinforcement learning: A survey and open research challenges,” *Machine Learning and Knowledge Extraction*, vol. 4, pp. 172–221, Feb. 2022.
- [174] O. Nachum, S. Gu, H. Lee, and S. Levine, “Data-efficient hierarchical reinforcement learning,” in *Proc. of Advances in Neural Information Processing Systems 31*, pp. 1–11, Dec. 2018.

- [175] R. Wang, R. Yu, B. An, and Z. Rabinovich, “I2HRL: Interactive influence-based hierarchical reinforcement learning,” in *Proc. of International Joint Conference on Artificial Intelligence*, pp. 3131–3138, Jan. 2021.
- [176] T. V. Erven and P. Harremos, “Rényi divergence and kullback-leibler divergence,” *IEEE Transactions on Information Theory*, pp. 3797–3820, Jul. 2014.
- [177] D. MacKay, *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.
- [178] Z. Zhang and M. Sabuncu, “Generalized cross entropy loss for training deep neural networks with noisy labels,” in *Proc. of Advances in Neural Information Processing Systems*, p. 8792–8802, Dec. 2018.
- [179] A. Greenwald and K. Hall, “Correlated Q-learning,” in *Proc. of International Conference on Machine Learning*, p. 84–89, Aug. 2003.
- [180] H. Jiang, T. Kathuria, Y. T. Lee, S. Padmanabhan, and Z. Song, “A faster interior point method for semidefinite programming,” in *Proc. of IEEE Annual Symposium on Foundations of Computer Science*, p. 1–9, Nov. 2020.
- [181] N. V. Queipo, R. T. Haftka, R. V. W. Shyy, T. Goel, and P. K. Tucker, “Surrogate-based analysis and optimization,” *Progress in Aerospace Sciences*, pp. 1–28, Jan. 2005.
- [182] M. Elsayed and M. Erol-Kantarci, “Reinforcement learning-based joint power and resource allocation for URLLC in 5G,” in *Proc. of IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, Dec 2019.
- [183] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with LSTM,” *Neural Computation*, vol. 12, p. 2451–2471, Oct. 2020.