University of Ottawa's Ph.D. Thesis AI-enabled System Optimization with Carrier Aggregation and Task Offloading in 5G and 6G

by

Fahimeh Khoramnejad

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

Declaration of Authorship

I hereby certify that this thesis is entirely my own original work except where otherwise indicated. I am aware of the University's regulations concerning plagiarism, including those concerning consequent disciplinary actions. Any use of the works of any other author, in any form, is properly acknowledged at their point of use.

Abstract

Fifth-Generation (5G) and sixth-Generation (6G) are new global wireless standards providing everyone and everything, machines, objects, and devices, with massive network capacity. The technological advances in wireless communication enable 5G and 6G networks to support resource and computation-hungry services such as smart agriculture and smart city applications. Among these advances are two state-of-the-art technologies: Carrier Aggregation (CA) and Multi Access Edge Computing (MEC).

CA unlocks new sources of spectrum in both the mid-band and high-band radio frequencies. It provides the unique capability of aggregating several frequency bands for higher peak rates, and increases cell coverage. The latter is obtained by activating the Component Carriers (CC) in low-band and mid-band frequency (below 7 GHz) while 5G high-band (above 24GHz) delivers unprecedented peak rates with poorer Uplink (UL) coverage.

MEC provides computing and storage resources with sufficient connectivity close to end users. These execution resources are typically within/at the boundary of access networks providing support for application use cases such as Augmented Reality (AR)/Virtual Reality (VR). The key technology in MEC is task offloading, which enables a user to offload a resource-hungry application to the MEC hosts to reduce the cost (in terms of energy and latency) of processing the application.

This thesis focuses on using CA and task offloading in 5G and 6G wireless networks. These advanced infrastructures are an enabler for many broader use cases, e.g., autonomous driving and Internet of Things (IoT) applications. However, the pertinent problems are the high-dimensional ones with combinatorial characteristics. Furthermore, the time-varying features of the 5G/6G wireless networks, such as the stochastic nature of the wireless channel, should be concurrently met. The above challenges can be tackled by using data-driven techniques and Machine Learning (ML) algorithms to derive intelligent and autonomous resource management techniques in the 5G/6G wireless networks. The resource management problems in these networks are sequential decision-making problems, additionally with conflicting objectives. Therefore, among the ML algorithms, the ones based on the Reinforcement Learning (RL), constitute a promising tool to make a trade-off between the conflicting objectives of the resource management problems in the 5G/6G wireless networks, are used.

This research considers the objective of maximizing the achievable rate and minimizing the users' transmit power levels in the MEC-enabled network. Additionally, we try to simultaneously maximize the network capacity and improve the network coverage by activating/deactivating the CCs. Compared with the derived schemes in the literature,

our contributions are two folded: deriving distributed resource management schemes in 5G/6G wireless networks to efficiently manage the limited spectrum resources and meet the diverse requirements of some resource-hungry applications, and developing intelligent and energy-aware algorithms to improve the performance in terms of energy consumption, delay, and achievable rate.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Professor Melike Erol-Kantarci, for her invaluable guidance, support, and encouragement throughout my Ph.D. Her expertise, patience, and insightful feedback have been instrumental in shaping my research and helping me navigate through the challenges. I am grateful for her unwavering commitment to my success and for being an exceptional mentor and role model. I would also like to thank the members of my thesis committee for their valuable feedback and suggestions. Finally, I would like to acknowledge the support of my family and friends, without whom this journey would not have been possible.

Dedication

To my beloved husband, Ahmad, who has been my steadfast support throughout this journey. His unwavering support, encouragement, and belief in me have been the driving force behind my success. Thank him for being my partner in every sense of the word and for sharing this incredible experience with me. I dedicate this thesis to him with all my heart.

Table of Contents

List of Tables xi

List of Figures xii

List of Abbreviations xv

1 Introduction 1

1.1 Emerging Technologies in Wireless Communications 1

1.2 Application of AI in the Resource Management of the Next-generation Wireless Networks 2

1.3 Contributions 4

1.3.1 Learning at Edge 5

1.3.2 Intelligent Delay-Aware and Energy-Efficient Carrier Aggregation . 6

1.3.3 Deep Reinforcement Learning for Decision Making 7

1.4 Conclusion 9

2 Background and Literature Review 11

2.1 Background 11

2.1.1 Reinforcement Learning 11

2.1.2 Deep Reinforcement Learning 14

2.1.3 Convolutional Graph Neural Networks 19

2.2 Literature Review 20
2.2.1 Resource Allocation and Edge Computing 20
2.2.2 Delay-Aware and Energy-Efficient Carrier Aggregation 28

3 Intelligent Resource Management and Offloading in MEC-enabled Wireless Networks 34

3.1 Introduction 34
3.2 On Joint Offloading and Resource Allocation: A Double Deep Q-Network Approach 35
3.2.1 System Model and Problem Formulation 36
3.2.2 JROM Algorithm 44
3.2.3 Deep RL for Joint Resource Allocation and Offloading 48
3.2.4 Complexity Analysis 52
3.2.5 Simulation Results 53
3.3 Distributed Multi-Agent Learning for Service Function Chain Partial Offloading at the Edge 61
3.3.1 System model and Preliminaries 61
3.3.2 Problem Formulation 64
3.3.3 The Proposed MARL-based Method 66
3.3.4 Simulation Results 67
3.4 Energy and Delay Aware General Task Dependent Offloading in UAV-Aided Smart Farms 71
3.4.1 System Model and Problem Formulation 72
3.4.2 The CA2C-based Algorithm for Joint Offloading and Power Allocation 78
3.4.3 Dis-joint Offloading and Power Control: DDQN-based Offloading and Game-theory-based power control 84
3.4.4 Simulation Results 87

4 Intelligent Energy-efficient Resource Management in the Wireless Networks with CA Functionality **94**

4.1 Introduction . . . 94

4.2 Delay-Aware and Energy-Efficient Carrier Aggregation in 5G using Double Deep Q-Networks . . . 96

4.2.1 System Model and Problem Statement . . . 96

4.2.2 Multi-agent RL-based CC Management . . . 102

4.2.3 Complexity Analysis . . . 109

4.2.4 Simulation Results . . . 110

4.3 On Intelligent Energy-Aware Carrier Aggregation in 5G with Dual Connectivity . . . 115

4.3.1 System Model and Problem Statement . . . 115

4.3.2 Intelligent CA and UL Power-Sharing: A Compound-Action Actor-Critic Approach . . . 121

4.3.3 Complexity Analysis . . . 127

4.3.4 Simulation Results . . . 129

4.3.5 Convergence Behaviour of JPSCA Algorithm . . . 131

4.3.6 Performance Evaluation . . . 131

5 Conclusion **136**

5.1 Intelligent Resource Allocation at Edge . . . 136

5.2 Energy and Delay Aware Carrier Aggregation in 5G Networks . . . 137

5.3 Future Work . . . 137

A **139**

A.1 Proof of Theorem 3.1 . . . 139

A.2 Unit price for consumed energy . . . 141

A.3 PORA algorithm . . . 141

A.4 Updating Rule for the Weights of Actor and Critic DNNs . . . 143

A.5 Dis-joint Priority-based Power and Offloading Scheme (DPPOS) . . . 144

A.6 Proof of Lemma 3.1 . . . 145

B **148**

B.1 Optimal CC Management (OCCM) 148

B.2 Proof of Lemma 4.1 152

B.3 The DDQN-based UL power-sharing and CA 153

List of Tables

2.1 Resource allocation and task offloading in the next-generation wireless networks . . . 21

2.2 Resource allocation in the next-generation wireless networks with CA functionality . . . 30

3.1 Network parameters and hyperparameters for the proposed resource allocation and offloading algorithms . . . 54

3.2 Network parameters and hyperparameters for the SFC placement and offloading algorithm . . . 69

3.3 Simulation parameters . . . 89

4.1 Network parameters and hyper-parameters for the CC management algorithms 111

4.2 Table of notations . . . 116

4.3 Simulation parameter and hyper-parameters for the learning algorithms . . . 130

List of Figures

1.1 Advent of new technologies and applications in the next-generation wireless networks and deployment of AI to handle the resource management problem. 3

2.1 Interaction between the agent and the environment in RL. 12

2.2 Reinforcement learning with deep Q-Network (DQN). 15

2.3 Actor-critic-based method for training the DNNs used to approximate Q^* and $\boldsymbol{\nu}^*$ for the agent. 17

3.1 The message passing between the BS and user m for offloading a task and adjusting the CPU frequency and UL transmit power level for the user at a given time t 52

3.2 Average reward in terms of computation efficiency versus the number of episodes for different learning rates. 55

3.3 Average total computed bits versus the number of users for the scenario with fixed users where $2 \leq M \leq 30$. 56

3.4 Average total transmit power level versus the number of users for the scenario with fixed users where $2 \leq M \leq 30$. 56

3.5 Average total computation efficiency versus the number of users for the scenario with fixed users where $2 \leq M \leq 30$. 57

3.6 Average total computed bits versus the number of users for the scenario with fixed users where $32 \leq M \leq 50$. 57

3.7 Average total computation efficiency versus the number of users for the scenario with fixed users where $32 \leq M \leq 50$. 58

3.8 Average total computed bits versus the number of users for the scenario with mobile users. 59

3.9 Average total transmit power level versus the number of users for the scenario with mobile users. 59

3.10 Average total computation efficiency versus the number of users for the scenario with mobile users. 60

3.11 Convergence analysis of average end-to-end delay. 70

3.12 Convergence analysis of average consumed energy. 70

3.13 Average end-to-end delay for each user versus the number of users. 71

3.14 Average consumed energy for each user versus the number of users. 71

3.15 Actor-Critic framework employed by each UAV for joint power control and offloading. The GCN captures each task's features and the DAG structure. 73

3.16 The message passing between a UAV u and other UAVs to update the offloading scheme for the tasks and the adjustment of transmit power level. . 83

3.17 Convergence behaviour for the CA2C-based algorithm **IJPOC** 90

3.18 Convergence behaviour for the DDQN-based algorithm **IDPOC** 90

3.19 Performance evaluation for CA2C-based algorithm **IJPOC**, DDQN-based algorithm **IDPOC**, and priority-based algorithm **DPPOC** with a **fixed number of UAVs** ($U = 3$) vs. number of tasks in the application in terms of average delay for each UAV (a), average each UAV's energy consumption (b), and average ETC for each UAV (c). 91

3.20 Performance evaluation for CA2C-based algorithm **IJPOC**, DDQN-based algorithm **IDPOC**, and priority-based algorithm **DPPOC** with a **fixed number of tasks** in the application ($I_u = 6,\ \forall u \in \mathcal{U}$) vs. number of UAVs in terms of average delay for each UAV (a), average each UAV's energy consumption (b), and average ETC for each UAV (c). 92

4.1 Network and the proposed optimization-based method (at top) and RL-based strategy (at the bottom) for CC management. For the RL-based strategy, the state vector corresponds to if the QoS requirement for each UE is provided or not. Action corresponds to which CCs are activated for the UE. The reward for the UE is given in terms of the reciprocal of the delay and the power consumption for the UE. 97

4.2 The message passing between the gNB and UE k for CA at a given time t . 109

4.3 Convergence behavior for the reward function in (4.12) in **DCCM** algorithm vs number of episodes. 112

4.4 Average achievable rate per UE and average delay per UE versus the number of UEs. 113

4.5 Average consumed power per UE and average number of assigned CCs per UE versus the number of UEs. 113

4.6 Intelligent CA and UL power allocation in the network with DC technology. 117

4.7 Actor-critic-based method for training the DNNs used to approximate Q_b^* and $\boldsymbol{\nu}_b^*$ for an agent b. 124

4.8 The message passing between a gNB b and its assigned UEs in $\mathcal{K}_b$ for CA and DPS at a given time t 128

4.9 Convergence behavior for CA2C-based (JPSCA) algorithm. 131

4.10 Average rate per UE for CA2C-based (JPSCA) algorithm vs. all CCs activated DDQN-based (ACPS) algorithm, and all CCs activated static power-sharing (AEPS) algorithm. 132

4.11 Average delay per UE for CA2C-based (JPSCA) algorithm vs. all CCs activated DDQN-based (ACPS) algorithm, and all CCs activated static power-sharing (AEPS) algorithm. 132

4.12 Average number of activated CCs per UE for CA2C-based (JPSCA) algorithm vs. all CCs activated DDQN-based (ACPS) algorithm and all CCs activated static power-sharing (AEPS) algorithm. 133

4.13 Average UL power consumption per UE for CA2C-based (JPSCA) algorithm vs. all CCs activated DDQN-based (ACPS) algorithm, and all CCs activated static power-sharing (AEPS) algorithm. 133

4.14 Average UL transmit power level per UE to gNB (a), and average UL transmit power level per UE to eNB (b) for CA2C-based (JPSCA) algorithm vs. all CCs activated DDQN-based (ACPS) algorithm and all CCs activated static power-sharing (AEPS) algorithm. 134

List of Abbreviations

Acronym	Meaning
3GPP	3rd Generation Partnership Project
5G	Fifth-Generation
6G	sixth-Generation
AI	Artificial Intelligence
AP	Access Point
AR	Augmented Reality
BSs	Base Stations
CA	Carrier Aggregation
CA2C	Compound-Action Actor-Critic
CC	Component Carriers
CE	Computation Efficiency
CGNN	Convolutional Graph Neural Networks
CI	Confidence Interval
CL	Continual Learning
CNN	Convolutional Neural Network
CQI	Channel Quality Indicator
CT	Cyber Twin
DAG	Directed Acyclic Graphs
DC	Dual Connectivity
DCCM	DDQN-based CC Management
DDPG	Deep Deterministic Policy Gradient
DDQN	double Deep Q-Network
DJROM	Deep Reinforcement Learning for Joint Resource Allocation and Offloading
DL	Downlink
DNN	Deep Neural Network

DPS	Dynamic Power-Sharing
DQL	Deep Q-learning
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
DTs	Digital Twins
eMBB	Enhanced Mobile Broadband
EN-DC	E-UTRAN New Radio – Dual Connectivity
eNB	E-UTRAN NodeB
EPS	Equal Power-Sharing
ETC	Energy-Time-Cost
GCN	Graph Convolutional Network
gNB	Next Generation NodeB
GNN	Graph Neural Networks
IoE	Internet of Everything
IoT	Internet of Things
JPSCA	Joint Power-Sharing and Carrier Aggregation
KKT	Karush–Kuhn–Tucker
LTE-A	LTE-Advanced
MARL	Multi-Agent Reinforcement Learning
MARL-DQN	Multi-Agent Deep Q-Network
MARL-DQL	Multi-Agent Deep Q-learning
MC	Multiple Connectivity
MCS	Modulation and Coding Scheme
MDP	Markov Decision Process
MEC	Multi Access Edge Computing
ML	Machine Learning
NE	Nash Equilibrium
NFV	Network Function Virtualization
NN	Neural Network
NOMA	Non-Orthogonal Multiple Access
NR	New Radio
NSA	Non-Standalone
PCC	Primary Component Carrier
PORA	Pratially Offloading Resource Allocation
PT	Physical Twin
QoS	Quality of Service
RB	Resource Block
RL	Reinforcement Learning

RR	Round-Robin
S2S	Sequence-to-Sequence
SC	Sub-carrier
SCC	Secondary Component Carrier
SFC	Service Function Chains
SINR	Signal-to-Interference-Noise-Ratio
SPO	SFC Partial Offloading
TDMA	Time Division Multiple Access
TL	Transfer Learning
UAV	Unmanned Aerial Vehicles
UL	Uplink
URLLC	Ultra Reliable and Low Latency Communications
VM	Virtual Machine
VMs	Virtual Machines
VNF	Virtual Network Functions
VR	Virtual Reality

Chapter 1

Introduction

1.1 Emerging Technologies in Wireless Communications

In 5G and 6G wireless networks, applications with various services, ranging from the Enhanced Mobile Broadband (eMBB) applications to the Ultra Reliable and Low Latency Communications (URLLC), are being supported. To fulfill a variety of Quality of Service (QoS) requirements and the immense computational needs for these applications, modern wireless technologies, MEC, CA and Dual Connectivity (DC), have recently been employed in the wireless networks.

MEC is a network architecture that enables cloud computing capabilities and an IT service environment at the edge of the mobile network. MEC is designed to bring computing, storage, and networking resources closer to the end-users or devices, reducing the latency and improving the quality of service. MEC enables the deployment of applications and services at the edge of the network, closer to the users and devices, which can significantly reduce the network latency and improve the user experience. The architecture allows for computing and storage resources to be dynamically allocated and scaled according to the user's needs, providing a flexible and efficient IT service environment. In MEC, mobile devices can offload the computation-intensive tasks to a nearby MEC server and augment their computing capability [1–4]. The MEC server or servers can be set up in an office, on a vehicle, in a base station cabinet, or another set of devices in MEC-enabled networks. MEC servers are accessed through a wireless network considering that lightweight mobile devices are the users that need the MEC capability. There are two possible offloading approaches; partial or binary offloading. In the former, the computation task is divided into two parts, executed locally on the mobile device and offloaded to the MEC server.

Meanwhile, in the latter, binary offloading, the whole task is performed either locally at the device or completely offloaded to a MEC server [1, 4–7].

On the other hand, CA is a technique used in wireless communication systems to increase the available bandwidth and improve network performance. It allows for the aggregation of multiple CCs in the same or different frequency bands, which can be used for data transmission. By combining multiple carriers, the overall bandwidth can be increased, allowing for higher data rates and better user experience. CA is used in various wireless communication systems, including LTE and 5G mobile networks. In LTE, CA was introduced with the LTE-Advanced (LTE-A) release, and it enables the aggregation of up to five component carriers in the same or different frequency bands. In the wireless networks with CA technology, the UE would achieve higher throughput and lower transmission delay [8–10]. While, DC, or generally called Multiple Connectivity (MC), allows operators to simultaneously provide LTE and 5G connectivity in the same spectrum. CCs can be activated and deactivated in CA depending on multiple factors, e.g., users' energy consumption and QoS demand. Additionally, the UEs are configured to access more CCs depending on their QoS requirements, traffic load, and channel quality [8, 10]. To increase the coverage area, system throughput, load balancing and mobility robustness, the UEs can enormously benefit from employing joint CA and DC technologies in wireless networks [9, 11].

1.2 Application of AI in the Resource Management of the Next-generation Wireless Networks

With the emerging technologies in communication, the resource management problem in the next generation wireless networks are combinatorial. Additionally, the next-generation wireless networks have multi-dimensional, and dynamic characteristics. Considering the features above, the resource management optimization problem can be stated as follows,

Optimize	Long-term reward (cost) for each UE	(1.1a)
s.t.	QoS requirement for each UE	(1.1b)
	Resource constraint	(1.1c)
var.	Decision-making indicators and resource variables.	(1.1d)

In (1.1), the objective function in (1.1a), the constraints in (1.1b) and (1.1c), and the optimizing variables would be characterized based on the network infrastructure. Additionally, in a time-varying environment such as 5G/6G networks, employing the existing traditional

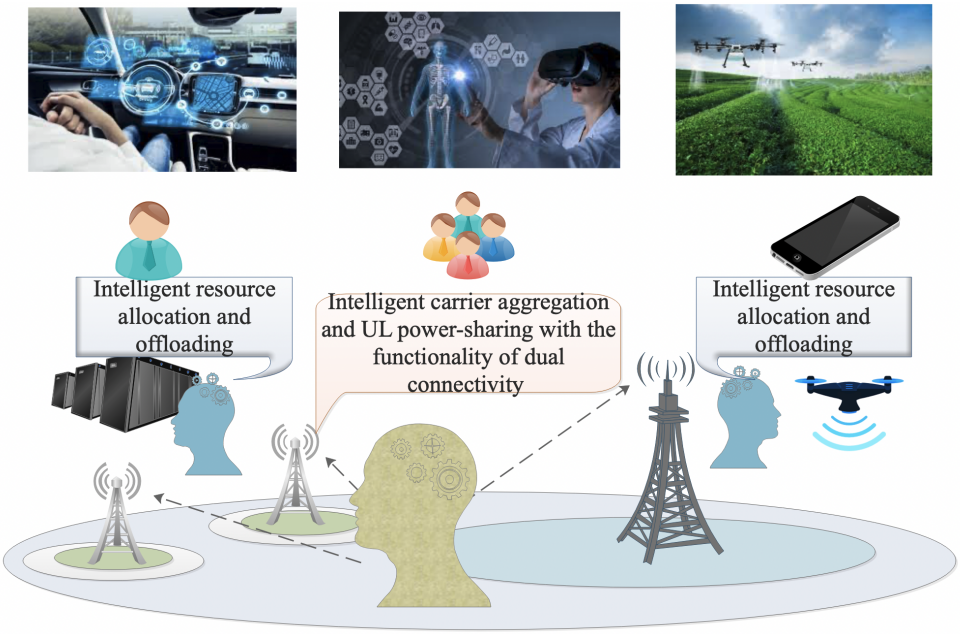

Figure 1.1: Advent of new technologies and applications in the next-generation wireless networks and deployment of AI to handle the resource management problem.

approaches to address (1.1) will jeopardize scalability and flexibility. For instance, game theory approaches (e.g., in [12–15]) and optimization methods (e.g., in [4–7, 16–30]) require complete information, which, in most cases, would not be available in 5G/6G wireless networks. Therefore, promising emerging AI techniques as in [31–53] have been recently used to overcome some of these challenges. Fig. 1.1 illustrates that by the advent of new technologies and leveraging Artificial Intelligence (AI), rate and computation-hungry applications can be supported in the next-generation wireless networks.

This thesis endeavors to investigate radio resource management in the 5G/6G wireless networks by employing the MEC, CA and DC as the emerging technologies in wireless communications[1]. In MEC-assisted networks, UEs can offload the resource-hungry ap-

[1]Based on the title of this study, the resource management schemes are proposed in 5G/6G wireless networks because, while the specific details of channel models for 5G and 6G may differ from previous generations of wireless communication, it is likely that some general characteristics will remain the same. For example, in all wireless communication systems, signals can be subject to attenuation, interference, and other types of distortion as they travel through the wireless channel. This can be influenced by factors such as distance, the presence of obstacles or other signals, and atmospheric conditions. Additionally, it is likely that certain types of channel models, such as those based on statistical models or physical

plications/tasks to the MEC hosts and take advantage of the computation and storage resources of the servers. These cutting-edge technologies eventually reduce the processing cost of the application/tasks in terms of energy and latency. On the other hand, CA and DC technologies can provide the UEs with improved coverage area, system throughput, load balancing and mobility robustness. Based on already mentioned infrastructures, the contributions, formal statement of the resource allocation problem (1.1) and the proposed RL-based schemes, are given in what follows.

1.3 Contributions

The next-generation wireless networks with advanced infrastructures, e.g., MEC, CA and DC, have multi-dimensional and dynamic characteristics which have become increasingly difficult to model. In this thesis, distributed algorithms are used for resource allocation in 5G/6G wireless networks with MEC, CA and DC technologies to efficiently manage the limited spectrum resources and meet the diverse requirements of resource-hungry applications. In this study, the main contributions correspond to the ways in which we apply distributed algorithms. They can be stated as follows,

- Decentralized resource allocation: distributed algorithms can be used to allocate resources based on the local information available at each node without the need for centralized coordination. We model the resource allocation problems as stochastic games which can be expressed as Markov Decision Process (MDP)s. MDP is a mathematical framework used for decision-making in situations where outcomes are partly random and partly under the control of a decision-maker. Specifically, the state spaces, action spaces, transition probabilities, and reward functions are defined for the MDPs.

- Intelligent resource allocation: to meet the time-varying features of the 5G/6G wireless networks in real-time, we resort to data-driven techniques and ML algorithms to deal with resource management. Since the resource management problems in the current and next-generation wireless networks are sequential decision-making problems, we focus on using RL methods which were studied in [32, 54–57]. Model-free RL[2] is employed to provide an effective approach to solving MDPs without explicitly

propagation models, will continue to be used in 5G and 6G.

[2]In RL, the agent finds optimal actions to maximize its long-term reward using model-free or model-based methods. In model-based RL methods, the agent knows the system dynamics, i.e., how the system

modeling the transition probabilities. These algorithms can learn from experience and converge to the (near) optimal policy.

- Energy-aware resource allocation: energy consumption is a critical factor in 5G/6G wireless networks, as it can significantly impact the battery life of connected devices. Distributed algorithms are developed to allocate resources in an energy-aware manner by minimizing the energy consumption of the users while meeting the desired performance requirements.

The effectiveness and performance of the RL-based schemes (e.g., achievable rate and delay) are studied and compared to the optimization and game theory methods. The following section presents our contributions of RL usage in dealing with radio resource management problems in wireless networks.

1.3.1 Learning at Edge

With MEC, mobile devices can offload their computationally heavy tasks to a nearby server, a simple node at a base station, a vehicle, or other devices. With the increasing number of devices, slices and multiple radio access technologies, the problem of task offloading has grown to be increasingly complex. Thus, traditional approaches endure limitations while machine learning algorithms have unfolded to be promising methods indeed.

In [1], [2] and [3], we have used optimization problem (1.1) and formulated the problem of joint offloading and resource allocation in MEC-enabled networks as a multi-objective optimization problem given by,

$$\begin{array}{lll} \textbf{Optimize} & \text{Long-term reward (cost) for each agent} & \forall m \in \mathcal{M}, \\ \textbf{s.t.} & \text{Target rate requirement for each agent} & \forall m \in \mathcal{M}, \\ & \text{Constraint of transmit power level and the CPU frequency} & \\ \textbf{var.} & \text{Offloading indicators, transmit power level,} & \\ & \text{and the CPU frequency,} & \end{array} \tag{1.2}$$

where $\mathcal{M}$ is the set of UEs.

transits from one state to another and how rewards are generated. This approach is not always feasible, especially in complex systems where the agent has limited knowledge about its environment. So, we employ model-free techniques where an agent learns optimal strategies without any knowledge about the system.

In the above optimization problem, the agents set out to maximize their long-term reward concurrently; therefore, (1.2) is a multi-objective optimization problem. In 5G environment, the problem (1.2) can be equivalently expressed as a multi-agent model-free RL system. In [1], for a given UE, the long-term reward is the weighted sum of the immediate reward which is given as the difference between the number of computed bits and the energy consumption of the UE. This corresponds to the computation efficiency of the UE. By considering both binary offloading and partial offloading schemes, the optimizing variables are: UL transmit power level of the UEs, the CPU frequency, and offloading scheme decision. The problem is time-dependent and non-convex with combinatorial characteristics. Therefore, it is hard to solve in polynomial time. To solve this problem, as meticulously presented in Chapter 3.2, the multi-agent Deep Reinforcement Learning (DRL) is used. We employ double Deep Q-Network (DDQN) [58] and propose a multi-agent DDQN-based scheme. DDQN is chosen to mitigate over-optimistic estimation of the Q-value function by using different values to select and evaluate an action. In [2], the objective is to make a trade-off between the energy consumption and delay to perform a Service Function Chains (SFC) request. The DDQN algorithm is used to partially offload a chain of services. Each Virtual Network Functions (VNF) of the SFC request can be performed locally or offloaded onto a MEC server. The details are given in Chapter 3.3. The problem of task-dependent offloading is studied in [3]. In an Unmanned Aerial Vehicles (UAV) and MEC-assisted smart farm, each UAV tries to perform a complex application comprising some interdependent tasks. So, we seek to simultaneously minimize the Energy-Time-Cost (ETC) for each UAV while considering the topology of the UAV applications and their energy budget. The problem is modeled as a multi-agent RL system with compound action space. As explained in Chapter 3.4, we make use of the combination of DRL and Graph Convolutional Network (GCN) to obtain the (near) optimal policy to offload a task and (near) optimal strategy to adjust the UAV transmit power levels.

1.3.2 Intelligent Delay-Aware and Energy-Efficient Carrier Aggregation

As one of the critical technologies in 5G networks, CA is studied in [8] and [9][3]. In CA, CCs can be activated and deactivated depending on multiple factors, e.g., the energy consumption of the UEs and the Quality of Service (QoS) demanded by them. By using

[3]CA can be applied for both UL and Downlink (DL) schemes. The constraint corresponding to transmit power level in the UL scheme should be considered. The derived scheme in [8] addresses the problem of CA in the DL scheme. In [9], the problem of joint UL power sharing and CA is tackled.

(1.1), the problem is formally expressed as,

Min	Average delay		(1.3a)
Min	Average consumed power		(1.3b)
s.t.	Number of active CCs	$\forall m \in \mathcal{M}$,	(1.3c)
	QoS requirement	$\forall m \in \mathcal{M}$,	(1.3d)
var.	CC and resource assignment indicators,		(1.3e)

In [8], just as discussed explicitly in Chapter 4.2, we not only develop a semi-distributed solution by modeling the problem into a stochastic game [59] but also propose a multi-agent DDQN-based CC management algorithm to solve the game.

CA and DC concurrently transmitting to two different Base Stations (BSs) in the same spectrum are considered in [9]. In the wireless networks with DC and CA, the performance of the network can be boosted by dynamically adjusting the UL transmit power level for the UEs to the BSs and properly activating/deactivating the CCs. In [9], we study the problem of joint dynamic power-sharing[4] and CC management in the networks with DC and CA. The objective is to minimize the UEs' delay and power consumption simultaneously. The pertinent problem is formally expressed as a multi-objective optimization problem with both discrete and continuous variables making it hard to solve. To tackle the problem, we employ the Compound-Action Actor-Critic (CA2C) algorithm in [60] to find the optimal policy and jointly perform UL power control and CC management. The details are provided in Chapter 4.3.

1.3.3 Deep Reinforcement Learning for Decision Making

In the next generation wireless networks, ML can be used to optimize the allocation of radio resources, including frequency bands and power levels, based on the network conditions and traffic demand. This can lead to better spectrum utilization and increased capacity, which can improve the user experience and reduce the overall cost of the network.

Recently, Deep Neural Network (DNN)s have been adopted to approximate the optimal solution for complex problems in the wireless networks in three settings: supervised ML, unsupervised ML and RL. Supervised ML showed promising results in several wireless

[4] In an Non-Standalone (NSA) 5G network, the UL power sharing process typically involves coordinating between the UL transmit power levels for the user to the 4G LTE network and the 5G New Radio (NR) network so as to ensure efficient use of network resources. This coordination is necessary because the user device may need to use both networks simultaneously to establish a connection.

problems. However, a large amount of prior labeled training and testing data should be available which is not easy to secure in real-life scenarios.

On the other hand, RL can be used to optimize the performance of the network by enabling the agent to learn from experience and make decisions that maximize a reward function. Some reasons why RL is used in wireless communications can be expressed as follows,

- Dynamic environment: wireless communication networks are highly dynamic and complex, with many variables that can affect performance, such as time-varying channels. RL can adapt to these changing conditions and optimize performance in real-time.
- Optimization of complex systems: RL can be used to optimize complex systems that have many interacting components. It can find optimal solutions by considering the interactions between the components, which can be difficult to achieve using traditional optimization methods. Hence, RL can be used to optimize the allocation of network resources, such as power, bandwidth, and frequency channels. It can learn to allocate resources based on the current network conditions and traffic demand, which can improve the overall network performance.
- Self-learning: RL can enable the network to learn from experience and adapt to changing conditions over time. This can lead to more efficient and effective network operations and maintenance.

However, 5G/6G wireless communication networks are becoming increasingly complex, with many variables. In recent years, DRL has gained a lot of attention due to its ability to solve complex optimization problems and achieve better performance compared to traditional RL methods. The reasons why DRL is effective in solving complex optimization problems in the next generation wireless networks can be stated as follows,

- Complex decision-making: DRL can handle complex decision-making problems by using DNNs to approximate the optimal decision-making policies and Q-value function.
- Large state and action spaces: RL methods can struggle when dealing with large state and action spaces, which are common in 5G/6G wireless communications. DRL can use DNNs to approximate the value or policy functions for large state and action spaces, making it more effective in optimizing the network performance.

- Better learning and generalization: DRL can learn more quickly and generalize better than RL because it can extract more meaningful features from the raw data using deep neural networks. This can lead to better performance in complex and dynamic environments such as 5G/6G wireless networks.

- Data efficiency: DRL can be more data-efficient than RL because it can extract more information from the data by using DNNs. This can reduce the amount of training data required and lead to faster convergence to optimal solutions.

It is noteworthy that the training phase of DNNs requires a considerable amount of computation power which would entail the use of GPUs and high-performance CPU clusters.

1.4 Conclusion

In this thesis, multi-agent DRL is used to address the formulated resource management problems (1.2) and (1.3). These problems are high-dimensional with combinatorial characteristics. Additionally, 5G and 6G wireless networks have dynamic features yet complete knowledge about the networks is not necessarily available, making them increasingly difficult to model. We address the challenges of applying RL to handle radio resource management in the MEC-assisted wireless networks and the networks with CA and DC technologies, such as the design of state space, action space, reward function and the compound action space. To solve the RL system with discrete action and state spaces, the DDQN algorithm is used. DDQN is a well-known algorithm developed in [58] to solve the RL systems with discrete action and state spaces. Some of the benefits of the DDQN algorithm can be given in terms of reduced overestimation bias, improved stability, faster convergence, and increased performance [58]. In the RL system with compound action space composed of both discrete and continuous actions, the CA2C is used. The CA2C algorithm is an actor-critic DRL algorithm that is developed by the authors in [60] to address the RL systems with compound action space. The benefits of CA2C algorithm can be enumerated as improved learning efficiency, enhanced exploration, improved performance, and generalization [60]. The outcomes of this thesis are:

[J01] F. Khoramnejad and M. Erol-Kantarci, "On Joint Offloading and Resource Allocation: A Double Deep Q-Network Approach," in IEEE Transactions on Cognitive Communications and Networking, vol. 7, no. 4, pp. 1126-1141, Dec. 2021, doi: 10.1109/TCCN.2021.3116251.

[J02] F. Khoramnejad, R. Joda, A. Bin. Sediq, H. Abou-zeid, R. Atawia, G. Boudreau, and M. Erol-Kantarci, "Delay-Aware and Energy-Efficient Carrier Aggregation in 5G using Double Deep Q-Networks," in IEEE Transactions on Communications, vol. 70, no. 10, pp. 6615-6629, Oct. 2022, doi: 10.1109/TCOMM.2022.3204846.

[J03] F. Khoramnejad, R. Joda, A. B. Sediq, G. Boudreau, M. Erol-Kantarci, "AI-enabled Energy-aware Carrier Aggregation in 5G New Radio with Dual Connectivity," *Submitted at* IEEE Access.

[J04] F. Khoramnejad, A. Syed, W. S. Kennedy and M. Erol-Kantarci, "Energy and Delay Aware General Task Dependent Offloading in UAV-Aided Smart Farms," *Submitted at* IEEE Transactions on Network and Service Management.

[C01] F. Khoramnejad, R. Joda and M. Erol-Kantarci, "Distributed Multi-Agent Learning for Service Function Chain Partial Offloading at the Edge," International Conference on Communications Workshops (ICC Workshops), 2021, doi: 10.1109/ICCWorkshops50388.2021.9473554.

Chapter 2

Background and Literature Review

2.1 Background

2.1.1 Reinforcement Learning

RL is an area of ML where an intelligent agent learns how to map situations to actions in order to maximize its long-term reward, the weighted sum of its immediate reward. At a given time, the immediate reward for the agent is a numerical signal, and the agent tries different actions to discover which yields the most immediate reward. These actions would affect the following situations and all subsequent immediate rewards. The characteristics above are the two most essential features of RL: trial-and-error search and delayed reward [61].

Using the dynamical system theory idea, the problem of RL can be formally expressed as an incompletely-known MDP. Specifically, interacting over time with its environment, the agent senses the state of the environment and takes action affecting the state. The agent has a goal related to the state and action. A well-suited RL method uses dynamic programming techniques to solve an RL problem.

Based on what was discussed above, RL is different from supervised and unsupervised learning. Since RL is an interactive problem, it is often impractical to obtain examples of desired behavior that are both correct and representative of all the situations in which the agent has to act. Thus, an agent must be able to learn from its own experience. Furthermore, in RL, although it would be worthwhile to uncover the structure in an agent's experience, this would not tackle the RL problem of maximizing a reward signal by itself.

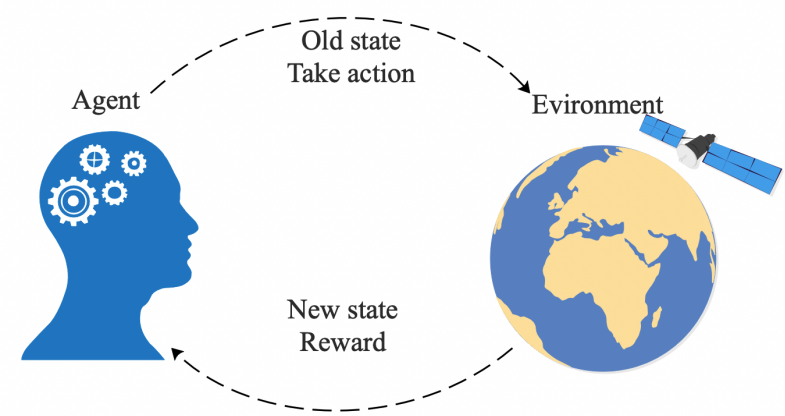

Figure 2.1: Interaction between the agent and the environment in RL.

Accordingly, the authors in [61] have considered the RL as a third ML paradigm alongside supervised and unsupervised learning.

MDPs can be used to model the learning problems interacting with the environment to achieve a goal, e.g., RL problems. The decision maker is called the agent that interacts with the environment. Fig. 2.1 illustrates the interaction between the agent and the environment. Given $\mathcal{S}$ and $\mathcal{A}$ as the state space and action space, respectively, in a sequence of discrete time steps $t = 1, 2, 3, \ldots$, at each time step t, based on the representation from the environment's old state, i.e., $S_t \in \mathcal{S}$, the agent selects an action $A_t \in \mathcal{A}$. In one time step later, $t+1$, as a consequence of action A_t, the agent receives a numerical reward R_{t+1}, and the environment presents a new state S_{t+1} to the agent with the transition probability of $p(s' \mid s, a) = Pr\{S_{t+1} = s' \mid S_t = s, A_t = a\}$. This rises to the following sequence or trajectory given by,

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots \tag{2.1}$$

In an RL system, the discounted cumulative reward (long-term reward) for an agent is defined as,

$$G_t = R_{t+1} + \lambda R_{t+2} + \lambda^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \lambda^k R_{t+k+1} \tag{2.2}$$

where λ is a parameter called discount factor, and $\lambda \in [0, 1]$. Additionally, a policy is defined as a mapping from states to probabilities of selecting each possible action. That is, at a given time t and state $S_t = s$, when the agent is following policy π, $\pi(a \mid s)$ is the probability that $A_t = a$. Under a policy π and a given state s, the value function is

denoted by $v_\pi(s)$ and formally stated by [61],

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\Big[\sum_{k=0}^{\infty} \lambda^k R_{t+k+1} \;\Big|\; S_t = s\Big],\ \forall s \in \mathcal{S} \tag{2.3}$$

where $\mathbb{E}_\pi[.]$ is the expected value for a random variable when the agent follows policy π. The value function is the expected return when starting in s and following π thereafter. To estimate how good it is for an agent to adopt an action a in a given state s under policy π, the action-value function $q_\pi(s, a)$ is formally defined as,

$$q_\pi(s,a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi\Big[\sum_{k=0}^{\infty} \lambda^k R_{t+k+1} \;\Big|\; S_t = s, A_t = a\Big],\ \forall s \in \mathcal{S},\ \forall a \in \mathcal{A} \tag{2.4}$$

To solve an RL system, the optimal policy, a policy that maximizes the long-term reward for the agent, should be discovered. In other words, π^* is the optimal policy if and only if $v_{\pi^*}(s) \geq v_\pi(s)$ for any given policy π and for all $s \in \mathcal{S}$. Correspondingly, the optimal value function and optimal action-value function are defined by (2.5) and (2.6), respectively,

$$v_*(s) = \max_\pi v_\pi(s), \forall s \in \mathcal{S} \tag{2.5}$$

$$q_*(s,a) = \max_\pi q_\pi(s,a),\ \forall s \in \mathcal{S},\ \forall a \in \mathcal{A} \tag{2.6}$$

The optimal action-value function can be given in terms of optimal value function as,

$$q_*(s,a) = \mathbb{E}\Big[R_{t+1} + \lambda v_*(S_{t+1}) \;\Big|\; S_t = s, A_t = a\Big]. \tag{2.7}$$

To approximate the optimal action-value function, $q_*(s, a)$, Q-learning is one of the early algorithms in RL developed by the authors in [62] and defined by,

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \delta\Big[R_{t+1} + \lambda \max_{a \in \mathcal{A}} Q(S_{t+1}, a) - Q(S_t, A_t)\Big]. \tag{2.8}$$

The convergence behavior of the Q-learning algorithm has been studied in [62], and it has been shown that if all state-action pairs continue to be updated, Q converges with probability 1 to Q^*. Accordingly, as the dimensions for either state space or action space grow, Q-learning no longer provide sufficient performance to address the computational needs of the RL system.

2.1.2 Deep Reinforcement Learning

In many practical decision-making problems with high-dimensional state space or action space, the traditional RL algorithm, e.g., Q-learning, cannot be applied to solve the problem. DRL algorithms incorporate deep learning and use DNN for either value function approximation or policy approximation. The details for two well-known DRL algorithms used in this thesis are as follows.

Double-Deep Q-Network Algorithm

In high-dimensional problems (problems with either huge state space or huge action space) with discrete action space, the non-linear function approximation methods, including neural networks, can be employed to obtain the value functions. For instance, when the number of states increases, we can learn the parametrized action-value function for a given user (agent) to figure out all action values for all states. Let $Q(s, a, \boldsymbol{\theta})$ denote the parametrized action-value function. Given $[S_t, A_t, R_{t+1}, S_{t+1}]$, by using a DNN to approximate the action-value function, the Deep Q-learning (DQL) algorithm to update parameter $\boldsymbol{\theta}$ is given by [61],

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \zeta(Y^{\mathrm{Q}} - Q(s, a, \boldsymbol{\theta}))\nabla_{\boldsymbol{\theta}} Q(s, a, \boldsymbol{\theta}). \tag{2.9}$$

In (2.9), ζ is a scalar step size, and the target for the user is given by,

$$Y^{\mathrm{Q}} = R_{t+1} + \lambda \max_{a' \in \mathcal{A}} Q(S_{t+1}, a', \boldsymbol{\theta}). \tag{2.10}$$

To improve the performance and conquer the learning instability, the Deep Q-Network (DQN) [63] can be engaged to update the DNN parameters. In DQN, a *target network*, *online network* and the *experience replay* are used to update the parameters. Let $\boldsymbol{\theta}^-$ and $\boldsymbol{\theta}$ be the parameters for the target and online networks, respectively. The parameters of the online network are copied after every N step to the target network so that $\boldsymbol{\theta}^- = \boldsymbol{\theta}$. Accordingly, the target used by DQN is,

$$Y^{\mathrm{DQN}} = R_{t+1} + \lambda \max_{a' \in \mathcal{A}} Q(S_{t+1}, a', \boldsymbol{\theta}^-), \tag{2.11}$$

and by replacing Y^{DQN} in (2.9),

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \zeta(Y^{\mathrm{DQN}} - Q(s, a, \boldsymbol{\theta}))\nabla_{\boldsymbol{\theta}} Q(s, a, \boldsymbol{\theta}). \tag{2.12}$$

is gained. As depicted in Fig. 2.2, the target network is used to achieve $\mathrm{Y}^{\mathrm{DQN}}$ [which is noted in equation (2.11)]. Also, as mentioned in reference [33, 58], the loss function

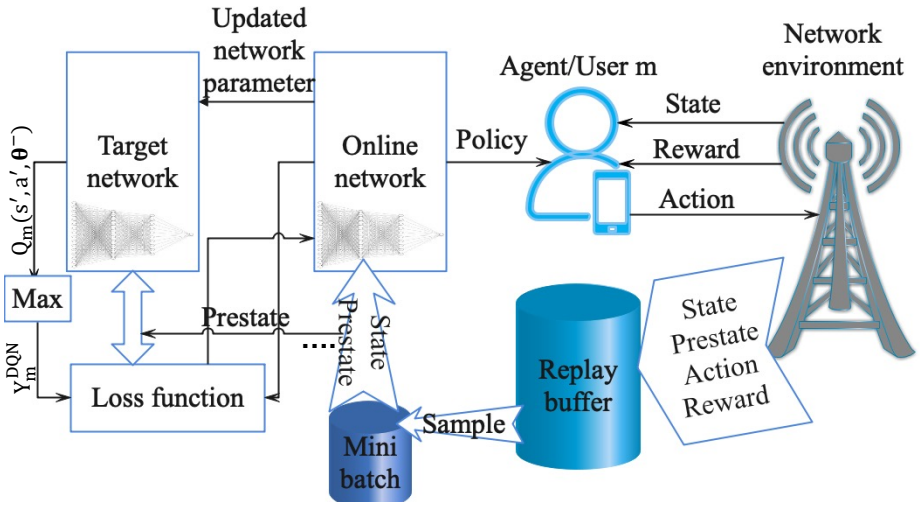

Figure 2.2: Reinforcement learning with deep Q-Network (DQN).

is given in terms of the difference between $\mathrm{Y}^{\mathrm{DQN}}$ and the Q-value obtained from the online network. To optimize the loss function, the parameters for the online network are updated using the updating function in equation (2.12). Additionally, the observed transitions $[S_t, A_t, R_{t+1}, S_{t+1}]$ which are stored in the experienced memory (replay buffer) $\mathcal{D}$, are sampled uniformly (as mini-batches) to train the online network, conquering the learning instability and preventing the optimal policy from being driven to a local minima [33].

In both Y^{Q} and Y^{DQN}, the max operator results in overoptimistic estimates. The DDQN algorithm is proposed in [58] to resolve this issue. In the DDQN algorithm, the selection is decoupled from the evaluation [58]. Specifically, by using the online network with parameter $\boldsymbol{\theta}$ and target network with parameter $\boldsymbol{\theta}^-$ the target for DDQN is given by,

$$Y^{\mathrm{DDQN}} = R_{t+1} + \lambda Q(S_{t+1}, \arg\max_{a' \in \mathcal{A}} Q(S_{t+1}, a', \boldsymbol{\theta}), \boldsymbol{\theta}^-). \tag{2.13}$$

By replacing Y^{DDQN} in (2.9), the updating function for the online network parameters is given by,

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \zeta(Y^{\mathrm{DDQN}} - Q(s, a, \boldsymbol{\theta}))\nabla_{\boldsymbol{\theta}} Q(s, a, \boldsymbol{\theta}). \tag{2.14}$$

Compound-Action Actor-Critic Algorithm

In some cases, the decision problems have the action space with both continuous and discrete actions, i.e.,

$$\mathcal{A} = \{\mathbf{a} = (\boldsymbol{\alpha}, \mathbf{r}) \mid \boldsymbol{\alpha} \text{ is discrete action and } \mathbf{r} \text{ is continuous action}\} \tag{2.15}$$

Taking advantage of the Deep Deterministic Policy Gradient (DDPG) algorithm [64] and DQN algorithm [63], the authors in [60] have derived the CA2C method to learn the optimal policy to handle both continuous and discrete action. Let $\boldsymbol{\pi}$ be the policy function for the agent; therefore, $\boldsymbol{\pi}$ is a function from a given state s to action $\mathbf{a}$, i.e., $\boldsymbol{\pi} : \mathcal{S} \mapsto \mathcal{A}$. The policy $\boldsymbol{\pi}$ corresponds to the behavior patterns of the agent at the different states of the environment. Thus, it should be optimized to maximize the long-term reward for the agent. To find the optimal policy, i.e., $\boldsymbol{\pi}^*$, both current and future rewards for the agent should be considered. For a given state s and action $\mathbf{a}$, let $Q(\boldsymbol{\pi}, s, \mathbf{a})$ be the Q-function for the agent under the policy $\boldsymbol{\pi}$. For ease of reference, from here on, we use the notation $Q(s, \mathbf{a})$ for the agent's Q-function. $Q(s, \mathbf{a})$ is defined in terms of the expectation of the weighted sum of the short-term reward for that agent [61]. Based on what is discussed in [60] and [1], the optimal policy for the agent is the policy under which the Q-function is maximized, i.e.,

$$\mathbf{a}^* = \pi^*(s) = \arg\max_{\mathbf{a}_b \in \mathcal{A}_b} Q^*(s, \mathbf{a}_b), \qquad \forall s \in \mathcal{S}. \tag{2.16}$$

In the above problem, the action space is composed of continuous and discrete actions. The CA2C algorithm is proposed in [60] to attain the optimal policy. The details are marked in what follows.

Let us decompose the optimal policy $\boldsymbol{\pi}^*$ into two optimal policies corresponding to continuous and discrete actions. Given state s and discrete action $\boldsymbol{\alpha}$, let $\boldsymbol{\nu}^*(s, \boldsymbol{\alpha})$ denote the optimal policy corresponding to continuous action. The best discrete action is obtained as follows [60],

$$\alpha^* = \arg\max_{\boldsymbol{\alpha} \in \mathcal{A}} Q^*\Big(s, (\boldsymbol{\alpha}, \boldsymbol{\nu}^*(s, \boldsymbol{\alpha}))\Big) \tag{2.17}$$

In (3.58), finding the exact value for Q^* and $\boldsymbol{\nu}^*$ is challenging because the action and state spaces are high dimensional. To address this issue, the DNN can be used to approximate them. The CA2C algorithm in [60] employs the DQN and DDPG algorithms to train the corresponding DNNs separately. The details are presented below.

As illustrated in Fig. 2.3, two separated DNNs, actor DNN with parameter $\boldsymbol{\theta}$ and critic DNN with parameter $\mathbf{w}$, are used to approximate the Q^* and $\boldsymbol{\nu}^*$, respectively. Let us denote the parametrized Q-function and the parametrized policy pertinent to continuous

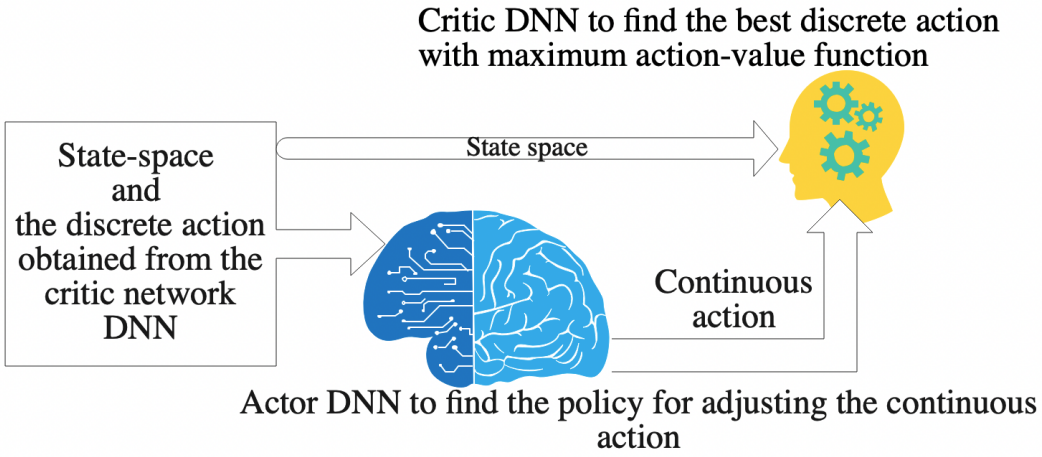

Figure 2.3: Actor-critic-based method for training the DNNs used to approximate Q^* and $\boldsymbol{\nu}^*$ for the agent.

action by $Q(s, \mathbf{a}, \mathbf{w})$ and $\boldsymbol{\nu}(s, \boldsymbol{\alpha}, \boldsymbol{\theta})$, respectively. The state s and discrete action $\boldsymbol{\alpha}$ are first given as the input to the actor DNN. The actor DNN approximates $\boldsymbol{\nu}(s, \boldsymbol{\alpha}, \boldsymbol{\theta})$ to obtain the continuous action $\mathbf{r}(t)$. Then, given the state s and $\mathbf{r}(t)$ to the critic network, the Q-function for the agent, $Q(s, \mathbf{a}, \mathbf{w})$, is approximated. Based on (3.58), $Q(s, \mathbf{a}, \mathbf{w})$, corresponds to discrete action.

Training procedure: To train the actor and critic DNNs, the DDPG and DQN algorithms in [64] and [63] are used, respectively [60]. In both algorithms above, the target and online network concept is employed to prevent over-optimism and instability. We denote the parameters for the actor and critic target networks by $\boldsymbol{\theta}^-$ and $\mathbf{w}^-$, respectively. Further, the parameters for the actor and critic online networks are $\boldsymbol{\theta}$ and $\mathbf{w}$, respectively. At each step, the soft update method is used to update the parameters for the target networks, i.e., $\boldsymbol{\theta}^- = \tau\boldsymbol{\theta}^- + (1-\tau)\boldsymbol{\theta}$, and $\mathbf{w}^- = \tau\mathbf{w}^- + (1-\tau)\mathbf{w}$ where τ is the fixed parameter. The replay buffer is used to train the actor and critic online DNNs (i.e., setting the parameters $\boldsymbol{\theta}$ and $\mathbf{w}$, respectively). At a given time t, let us denote the tuple of current space, current action, next state and reward as the experience $\mathbf{e}^t = [S_t, \mathbf{a}_t, S_{t+1}, R_t]$. The replay buffer for the agent is denoted by $\mathcal{D} = \{\mathbf{e}^t\}$. Additionally, we denote the functions evaluated by the actor and critic networks by $J_b(\boldsymbol{\theta})$ and $L(\mathbf{w})$, respectively. The function $J(\boldsymbol{\theta})$ is given in terms of the average Q-function for the agent, and $L(\mathbf{w})$ is given in terms of the average difference between the Q-function and target value for the agent. The details for functions $J(\boldsymbol{\theta})$ and $L(\mathbf{w})$ and the corresponding target value are given in [60]. We refer the readers for more details to this reference and mention the updating function for $\boldsymbol{\theta}$ and

$\mathbf{w}$ as follows [60],

$$\boldsymbol{\theta} = \boldsymbol{\theta} + \zeta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}), \tag{2.18a}$$

$$\mathbf{w} = \mathbf{w} - \zeta \nabla_{\mathbf{w}} L(\mathbf{w}), \tag{2.18b}$$

where ζ is the learning rate. By following the procedure illustrated in Fig. 2.3 and using (A.9a) and (A.9b), the algorithm to train the actor and critic networks are provided in **Algorithm 2.1** and **Algorithm 2.2**, respectively [60].

Algorithm 2.1 Critic Network Training Algorithm

Input: A sample experience $\mathbf{e}^t = [S_t, \mathbf{a}, R_{t+1}, S_{t+1}]$, online network parameter $\boldsymbol{\theta}$, target network parameter $\boldsymbol{\theta}^-$, τ.

Output: Updated parameters for critic online networks and critic target networks.

1: **procedure**
2: Given S_{t+1} and $\boldsymbol{\alpha}$ (discrete action), obtain the output for the actor online network, i.e., $\mathbf{r}$ (continuous action);
3: Given $\mathbf{r}$ and S_{t+1} as input to critic target network, obtain the Q-function and update $\boldsymbol{\alpha}$ based on (3.58);
4: Given $\boldsymbol{\alpha}$ and S_{t+1} as input to actor target network, update $\mathbf{r}$;
5: Obtain the target value for the critic and the Q-function calculated by the critic target network, which is multiplied by the discount factor;
6: Employing critic optimizer (e.g., Adam algorithm), update the parameters for the critic online network using (A.9b);
7: Update the parameter for critic target network as $\mathbf{w}^- = \tau \mathbf{w}^- + (1 - \tau)\mathbf{w}$.
8: **end procedure**

Explainability

RL algorithms like many ML ones suffer from explainability. RL applications need a model that can explain their decisions and actions to human users. Note that DRL models rely on many factors, e.g., environment and deep learning models and the algorithm used to train, and hence are complex to debug. Some classification and assessment of current explainable RL methods are available in [65].

As mentioned in [65], explainable AI methods can be intrinsic or post-hoc. In the former, the ML model is constructed to be inherently self-explanatory at the time of training which is obtained by restricting the complexity of the model. Decision tree can

Algorithm 2.2 Actor Network Training Algorithm

Input: A sample experience $\mathbf{e}^t = [S_t, \mathbf{a}, R_{t+1}, S_{t+1}]$, online network parameter $\mathbf{w}$, target network parameter $\mathbf{w}^-$, τ.

Output: Updated parameters for critic online networks and critic target networks.

1: **procedure**
2: Calculate the gradient of function $J(\boldsymbol{\theta})$ with respect to continuous action in all sampled experience form replay buffer $\mathcal{D}_b$ (by using equation (28) in [60]);
3: Using the actor optimizer (e.g., Adam algorithm), update actor online parameter based on (A.9a);
4: Update the parameter for actor target network as $\boldsymbol{\theta}^- = \tau\boldsymbol{\theta}^- + (1-\tau)\boldsymbol{\theta}$.
5: **end procedure**

be given as a good example of the intrinsic method. On the contrary, in the latter, the post-hoc method, a second simpler model is created to analyze and provide explanation for the original model after training [65]. DDQN and CA2C algorithms are based on the DQN algorithm. To study the DQN explainability, a post-hoc method has been developed in [66]. It shows that state aggregation in DQN algorithm gives the agent the ability to learn specific policies for the different regions, thus provides an explanation for the success of the agents.

2.1.3 Convolutional Graph Neural Networks

Taking advantage of DNNs and generating data in the Euclidean space, several ML tasks such as natural language and image processing, have recently been revolutionized by deep learning. However, the number of applications with data generated from non-Euclidean domains is increasing. Graphs can represent these applications with complex relationships and interdependency between objects. Graph Neural Networks (GNN)s provide graph data with extending deep learning approaches. The state-of-the-art for GNNs has been studied in [67]. The Convolutional Graph Neural Networks (CGNN) is employed in this thesis and briefly reviewed in what follows.

In the CGNN, the notion of convolution used in Convolutional Neural Network (CNN)s is employed for graph data. Particularly, it generalizes the convolution operation from grid data to graph data. By stacking multiple graph convolutional layers with different weights, a node in the graph is represented by its features and neighbors' features. For instance, to extract the overall graph's structure, the GCN is proposed in [68] and used by the authors in [69] to tackle the problem of offloading in 5G networks. In [68], the GCN algorithm,

the combination of deep Q-learning algorithm in RL and graph embedding, is proposed to approach the NP-hard graph optimization problems. Precisely, each node feature is iteratively updated using a DNN. To train the DNN, the deep Q-learning algorithm has been used. The authors in [69] further extended this method where an actor-critic algorithm is used to train the model (i.e., actor and critic DNNs and the embedding layers) and extract each node feature. Specifically, by using the Spatial-based approach in CGNN, where the graph convolutions are defined by propagating the information [67], the network layer-wise propagation rules for a given node v_i are given by,

$$H_i^{l+1} = \sigma\Big(\sum_{v_j \in \mathbf{succ}(v_i)} W^l H_j^l + H_i^l \Big), \tag{2.19}$$

where W^l is the weight for the lth layer in GCN, $\mathbf{succ}(v_i)$ is the set of successor nodes of the v_i, and $\sigma(.)$ is a nonlinear activation function, e.g., Sigmoid and Tanh. Given $O^i = H_i^L$ where L is the number of layer for the GCN, the embedding for the node v_i is denoted by E_{v_i} and calculated as follows,

$$E_{v_i} = \sigma\Big(\sum_{v_j \in \mathbf{succ}(v_i)} O^j \Big). \tag{2.20}$$

2.2 Literature Review

2.2.1 Resource Allocation and Edge Computing

Innovative applications for smart UEs such as smart cities applications, AR and VR are increasing, and they demand intensive computing and storage resources. Due to the high cost of running these resource-intensive applications, e.g., time cost and energy consumption, MEC is proposed. The computing and storage resources from a cloud would be shifted to the MEC hosts (e.g., BSs) at the edge of networks using MEC. This technology provides the UEs with closer resources, which subsequently reduces service latency, and alleviates network congestion.

In MEC, the key technology is task offloading, which enables a UE to offload a resource-hungry application to the MEC hosts so as to reduce the cost in terms of energy and latency and process the application. Specifically, in this technology, considering the task profile, CPU cycles and the size of input/output data, the UE should either perform the task locally or offload it to the MEC hosts. In literature, the offloading problem has been studied from two viewpoints: offloading an application with independent tasks and

Table 2.1: Resource allocation and task offloading in the next-generation wireless networks

Ref.	Task Dep.	Problem Formulation			Offloading Strategy		Solution Method
		Objective		Variable			
		Latency	Energy		Binary	Partial	
[70]	✓	✗	✓	Offloading	✓	✗	Heuristic
[71]	✓	✓	✗	Offloading Scheduling	✓	✗	Heuristic
[72]	✓	✓	✗	Offloading	✓	✗	Heuristic
[16]	✓	✓	✗	Offloading Service cashing	✓	✗	Optimization theory
[73]	✓	✓	✓	Offloading	✓	✗	DRL with S2S
[69]	✓	✓	✓	Offloading	✓	✗	DRL with GCN
[12]	✗	✓	✗	Offloading	✓	✗	Game theory
[13]	✗	✓	✓	Offloading SC allocation	✓	✗	Game theory
[4]	✗	✗	✓	Harvesting and offloading time, transmit power, CPU frequency	✓	✓	Optimization theory
[5]	✗	✓	✗	Task time, task partitioning	✓	✓	Optimization theory
[6]	✗	✗	✗	Transmit power, offloading ratio	✓	✓	Optimization theory
[7]	✗	✗	✓	Offloading, CPU frequency, transmit power	✓	✓	Optimization theory
[17]	✗	✓	✓	Offloading, CPU frequency	✓	✗	Optimization theory
[18]	✓	✓	✓	Offloading, CPU frequency, transmit power	✓	✗	Optimization theory
[19]	✗	✓	✗	Offloading, computation resource	✓	✗	Optimization theory
[74]	✗	✓	✓	Offloading	✓	✗	RL
[75]	✗	✗	✗	Offloading	✓	✗	RL

offloading an application with interdependent tasks. In the former, each task has its input and output, and the problem corresponds to offloading an *atomic task*, and it is a binary integer programming optimization problem. However, most of the applications in emerging wireless networks are *interdependent tasks*. They consist of several tasks, and the outputs of some of the tasks are the input for the others. Therefore, the inherent dependency among them should be regarded while studying the problem of resource management and offloading. Ergo, this problem is an NP-hard problem.

Based on Table 2.1, the works in [4–7, 12, 13, 17–19, 69–78] tackle the offloading problem by using different methods such as optimization theory, game theory, heuristics and RL. The details are provided in the following sections.

Atomic task offloading

In an atomic task application, each task has its input and output. In Table 2.1, a detailed classification of the proposed schemes for offloading an atomic task is given based on various features. The studies differ in their objective, problem formulation and solution approach. In terms of methodology, several works have used game theory approaches ([12, 13]), whereas some others brought optimization-based solutions into play to address this problem ([4–7, 17–19, 77]). In addition, recently learning-based methods have been explored ([74, 75]). The details are given in what follows.

Game theory has been employed in [12, 13] to address the problem of task offloading. Only the binary/full offloading scheme is considered in these works. A task can be performed locally or offloaded onto a MEC server in the binary offloading strategy. In [12], considering a MEC-enabled vehicular network, a non-cooperative game is proposed to address the problem of task offloading. Each vehicle can offload its task to the MEC server or execute it locally. So, in the game, the strategy space for each player corresponds to its task offloading probability to the MEC server by considering the set of vehicles as the set of players. The payoff function for each vehicle (player) is the difference between its utility and cost functions. The former is given in terms of the execution time of the player task. At the same time, the price function for a vehicle is given in terms of the available computation and storage resources in the MEC server. Particularly, the more the available resources in the MEC, the less the price for the player. It has been proven that a unique Nash Equilibrium (NE) is the solution for the proposed offloading non-cooperative game if the players offload their task based on the probability obtained from their best responses. Taking advantage of Non-Orthogonal Multiple Access (NOMA) in MEC-aided networks, the authors in [13] have proposed a joint offloading and Sub-carrier (SC) allocation to

minimize the computation overhead, i.e., the weighted sum of the latency and energy consumption to execute a task (either locally or by offloading it onto the MEC server). The problem is mixed-integer programming. The authors have restated it as a coalition game where the UEs are the players. Since the UEs can employ each SC to offload their task, each coalition in the game corresponds to a set of UEs using an SC to offload their tasks. The computation gain for each coalition is defined as the gain obtained from offloading the tasks. It is given in terms of the difference between the local computation and offloading computation overheads.

Optimization theory is the methodology which has been used by the authors in [4–7, 17–19, 77] to see to the offloading problem. In [4–7, 77], the partial offloading scheme for each task was taken into consideration. This scheme divides the computation task into two parts, executed locally on the mobile device and offloaded to the MEC server. Defining Computation Efficiency (CE) as the ratio of the total computed bits to the consumed energy, the authors in [4] developed the algorithms in four different scenarios, NOMA-based MEC-assisted network with binary and partial offloading schemes and the mentioned schemes for a MEC network with Time Division Multiple Access (TDMA). In [4], a non-linear function is used to model energy harvesting. Subjected to the constraint of minimum computed bits requirement, energy harvesting requirement (i.e., the energy consumption for a UE should be lower than the harvested energy), time for harvesting energy and offloading, the objective is to maximize the minimum CE for the UEs. The optimizing variables are the UL transmit power level, harvesting and offloading time, and the CPU frequency for the UEs. The iterative algorithms are derived to address the non-convex problems. For a TDMA-based network, three different scenarios for computation have been considered in [5], local computation, cloud computation and partial offloading computation. By viewing the fairness, the objectives for the problems in all scenarios above are to minimize the weighted sum of the delay for the tasks in different time slots. The problems above are convex optimization problems, and the optimal solutions are pertinent to the optimal time required to perform each task. The optimizing variables for the partial offloading scenario are the variables that correspond to the time slot for each task and its partitioning. The closed-form expression for the optimal task segmentation is obtained given the task interval. Using the closed-form expression in the optimization problem for the partial offloading scenario, the problem was reduced to a tractable and convex problem. Subjected to the constraint of delay budget for two fixed UEs, the objective in [6] is to maximize the successful computation probability. The successful probability is defined in terms of offloading ratio for two users, power allocation for UL NOMA, the time for task offloading and task executing, and the distance between the UEs and the MEC server. The optimizing variables are UL power and offloading ratio for the UEs. The optimal

solution for the problem reveals that the offloading ratio corresponds to the difference between the computation resources at the MEC and those of the edge UE. Also, the UE locations play a significant role in the UL power allocation ratio. In [7], for each offloading scheme, partial offloading and binary offloading, the problem of resource management and offloading has been separately tackled. Subjected to the constraint of latency requirement for the UEs, transmission quality, computational budget, and transmit power, the objective of the optimization problems is to minimize the sum of energy consumption for the UEs. For each UE, the energy consumption is the summation of energy for performing a task locally, the energy for transferring a task to the MEC server and the energy during the standby mode. The optimizing variables are the offloading indicators, the transmit power, and the UE's CPU frequency. For partial offloading, the offloading indicator is between zero and one. Meanwhile, it is a binary variable in the binary offloading scenario. To address the problems, the binary offloading variable in the binary offloading scenario is transformed into a continuous one. Additionally, for the partial offloading method, the l_0-norm in the constraint is approximated by using the linearization. The problems are addressed using the concave-convex procedure that handles the highly coupled and joint optimization parameters.

Considering the mobility of the UEs and the complete offloading method, the authors in [17] aim to solve the problem of offloading decisions and frequency allocation. The objective is to make a tradeoff between the latency for the tasks and their energy consumption. The problem is a mixed-integer and non-linear optimization problem, which is solved using a heuristic. Specifically, the main problem is decomposed into local convex optimization problems and non-linear integer programming sub-problems. The former problems are solved numerically, and their optimal solutions correspond to the local frequency for the tasks. Meanwhile, the latter problems are for offloading decisions and are solved heuristically. In [18], for given two UEs, it has been assumed that the output of the application (task) running by one UE is the input for that of another UE. By considering only two UEs, a scheme for task offloading, power control and local CPU frequency allocation has been derived. Subjected to the constraint of UL transmit power level for each UEs and their local frequency, the objective is to minimize the weighted sum for the total energy consumption and latency. Given the optimal solution for the offloading scheme adopted by each UE, the problem has been reduced to a convex problem through which closed-form expressions for the optimal transmit power level of the UEs are obtained. By using the closed-form expressions and a bisection method, the authors derived the optimal value for the local CPU frequency and transmit power level. To continue, during a time interval, it has been shown that it is optimal for a UE to offload its task onto a MEC server at most once, and wherefore, by using the Gibbs sampling method, the optimal offloading

decision for each UE is obtained. In a software-defined Ultra-dense network, an offloading scheme has been derived in [19] to minimize total UE delays subject to the constraint of each UE battery life. The authors decompose the formulated NP-hard problem into two offloading and computation resource allocation problems using a transformation method. The offloading is linearized and solved by using the $\epsilon-$constraint method. Meanwhile, the latter, a convex problem, has been solved numerically.

In vehicular networks assisted by edge computing, in [74, 75], to effectively model the impact of every UE action in a dynamic environment, RL has been used to find the optimal offloading policy. In [74], they attempt to efficiently support the resource-hungry applications running by vehicles. Thus, the weighted sum of task latency and energy consumption has been taken as the long-term cost. By considering the binary offloading scheme, the action space corresponds to performing the tasks locally or remotely; additionally, a task in the queue can be held idle. The state space is given in terms of the task features, e.g., task generation time and input data size, the location of the vehicles, the remaining cycles for local CPUs and MEC server to complete the tasks, and the remaining data of the tasks should be transmitted to the MEC server. A DRL-based method is employed to solve this RL system with massive action and state spaces. DNNs used to estimate the policy and value functions share the same input layer consisting of the CNN layers for the queue state information and the other data in the state space. The proximal policy optimization algorithm is used to train the DNNs. In [75], the utility for a task is given in terms of the saved time in executing the task compared to the maximum delay tolerance. Subjected to the delay constraint of tasks, the objective is to maximize the total task utilities. The problem is modeled as an RL system. The state space is the vector of the computation required to perform the tasks at each server. The action space is given in terms of the offloading strategy for the tasks. The deep Q-learning algorithm has been used to find the optimal policy in order to offload the task.

Offloading the applications with general task dependency

In a task-dependent application, the outputs of some tasks are the input for the others. In particular, an interdependent task application is represented by a Directed Acyclic Graphs (DAG). In works reviewed in the previous section, the task dependency has been ignored, which may have a detrimental effect on provisioning the QoS and using the edge resources. This drives the authors in [69–73, 78] to face the problem of resource management and offloading of the applications with general task dependency. The details are given in what follows.

To address the NP-hard general task-dependent offloading problem, heuristic algorithms

have been derived in [70–72]. In [70], the network comprises several UEs, MEC hosts and a remote cloud. The UEs can be idle, the application is neither run nor initiated. The offloading strategy is running an application locally or offloading it onto an idle UE, a MEC host and a cloud server. The authors derive the cost function for the system in terms of energy consumption and processor utilization in MEC hosts and cloud server. Subjected to the constraint of the energy consumption of each task, the task dependency in the applications and the complete time for the application execution, the objective is to minimize the cost. The formulated problem is an NP-hard problem. A heuristic algorithm for offloading a task has been proposed to address the problem sub-optimally.

Specifically, first, the tasks are arranged based on their ready time and complete time. Then, according to the required energy consumption for execution, they are assigned to a device namely another UE, a MEC host or the cloud server. Finally, the tasks will be rearranged to minimize the cost. Similar to the system model in [70], the authors in [71] aim to maximize the number of completed applications subjected to the constraint of the applications target delay. The variable of interest in the formulated resource management problem corresponds to offloading a task onto an edge host or the cloud and scheduling the offloaded task. A heuristic has been derived to tackle the problem wherein the priority of a task is estimated and used to offload and schedule the task. In the MEC-Cloud network in [72], the cloud offers a limited number of Virtual Machines (VMs), and the edge hosts have limited computation resources. The edge hosts cover the UEs, and each UE uses its corresponding local edge host to access the whole system. Since the UEs have the same network topology, the offloading problem is studied for the network with only one UE for more convenience. For a given application run by the UE, the makespan is defined as the difference between the initiation and the completion time of the application. Subjected to the constraint of completing each application before its deadline, the objective is to minimize the average makespan of the applications. First, the probabilistic bottom level is defined as the data transfer time between the two dependent tasks to address the problem sub-optimally. Using this parameter, a heuristic algorithm is proposed to derive the expected completion time of each task and prioritize the tasks accordingly. Based on the priority for each task, it is performed locally by the UE or assigned to a proper server.

The authors have used optimization theory in [16] to address the problem of offloading the interdependent task applications in MEC-Cloud networks. In [16], it is assumed that every task in an application is offloaded, and the edge hosts have limited service cashing. Subjected to the constraint of available services at each edge host, task dependency in the application, and offloading a task onto the only server, the objective is to minimize the execution time for the application. This problem is an NP-hard problem. To address it, two different scenarios have been considered: heterogeneous and homogeneous scenarios.

In the first scenario, the processing delay of the same job will vary on different edge nodes because the edge devices have other hardware specifications. Meanwhile, in the latter, the edge node hardware specifications are similar. Considering the first scenario, the authors use some concepts in convex optimization theory, relax the NP-hard formulated problem and solve it with CPLEX solver. In the second scenario, the formulated problem is relaxed so as to find which two consecutive tasks should be offloaded onto the same edge host. Then, the tasks are offloaded based on their starting time.

The authors in [69, 73, 78] have employed DRL to tackle the problem of offloading applications with interdependent tasks in a time-varying network. In [69, 73] the network consists of various UEs and is assisted by some MEC hosts and a cloud server. Each MEC host runs multiple VMs, and the cloud is used to process the computation-intensive and resource-hungry jobs. A UE application has multiple tasks with general dependency represented by a DAG. Using offloading scheduler, a UE makes the offloading decision for its application. The problem of task offloading is modeled as a MDP. In [73], the state space at a given time t is defined in terms of the DAG information of the application and the offloading plan for the tasks in the application from the beginning to time t. The action space represents the local execution or offloading. The reward function is a decreasing function of the energy consumption and the latency for a task in the DAG. To solve the MDP, a DRL-based algorithm has been proposed. The DRL based algorithm uses Sequence-to-Sequence (S2S) Neural Network (NN). The input for the S2S NN is the DAG, and its output is the policy for offloading the tasks. To efficiently train the S2S NN using the RL, the replay experience buffer gathers the experiences, each of which is a tuple of the current state, current action, reward and the next state. Additionally, to avoid a large policy update, the final training function is given in terms of the ratio of the current policy to the old policy, an entropy bonus and a squared error loss function. In [69], the state space is given in terms of the MEC environment status information, namely edge server status and user status, and the state of tasks. To reduce the dimension of the information above, the MEC environment status information is given as input to a NN to obtain the MEC embedding. For the latter, the task's state information, a GCN is used to obtain the task embedding. The state space is composed of task embedding and MEC embedding. The action space corresponds to performing a task either locally, or offloading it to a MEC host, or sending it to the cloud server. The reward function is given in terms of the average energy-time cost for all UEs. An actor-critic framework is derived to solve the proposed RL system, i.e., find the near-optimal policy to offload a task in each application of UE.

In [78], the MEC-assisted network consists of a UE and an Access Point (AP), the gateway for the edge cloud. The UE has a computationally intensive application with interdependent tasks that can be performed locally or offloaded onto the AP. A DAG

represents the dependency for the tasks. The objective is to minimize the weighted sum of application execution time and the UE's energy consumption subjected to the constraint of CPU frequency for the UE. This problem is an optimization problem with combinatorial characteristics. To address it, an actor-critic learning method is used. The offloading policy is obtained through the actor NN. Specifically, the output for the actor NN is quantized to explore the action space more efficiently. For each quantization level, the reward (cost) given in terms of the objective function for the optimization problem is calculated, and the action corresponding to the minimum value for the reward is considered. Given the offloading strategy for a task, the optimization problem is solved to simplify the output for the critic network. A closed-form expression for optimal CPU frequency of the UEs is obtained.

2.2.2 Delay-Aware and Energy-Efficient Carrier Aggregation

Spectrum aggregation or CA, introduced by the 3rd Generation Partnership Project (3GPP) in its new LTE-A standards, is a candidate technology to design the next-generation wireless networks. This concept has been deployed in HSPA cellular systems where the two adjacent carriers in the same spectrum bands are aggregated in DL and UL transmissions. In LTE-A systems, however, the CCs in the non-continuous spectrums in different bandwidths can be aggregated up to 100 MHz with five CCs of 20 MHz. This supports wider transmission bandwidth between E-UTRAN NodeB (eNB) and UE. In an LTE-A system, as the radio resource control connection is established by a UE, one CC is activated for the UE in the primary serving cell. This CC is referred to as always activated Primary Component Carrier (PCC). Depending on the QoS requirements for the UEs and the traffic pattern, one or more additional temporary CCs in the secondary serving cells are activated for each UE, which are called Secondary Component Carrier (SCC). The eNB is responsible for activating/deactivating an SCC for the UE. Additionally, it would not necessarily activate the same CC as the PCC for the different UEs, i.e., a CC can be configured as a PCC for one UE and activated as an SCC for another UE.

Despite its advantages, CA causes UEs to consume more energy because the UEs monitor a large aggregated spectrum than they would perform with a single carrier allocation [21]. UE power consumption affects a mobile device's battery life, so it must be carefully managed. In the UE receiver, the Radio Frequency module and Base Band processing part consume energy to receive the signal and decode information [21]. Therefore, the power consumed by each UE grows as the number of CCs and the rate increases. Hence, to save the UE power consumption in CA mode while enhancing the throughput and delay, dynamic management of activation and deactivation of CCs plays a prominent

role. In Table 2.2, to support the CA functionality, the authors in [14, 15, 20–28, 31, 79–85] have studied the radio resource management problem in the next-generation wireless networks. The proposed schemes are classified based on the various aspects, such as problem statement, solution method, objective and applications. The details are given in what follows.

Carrier aggregation and energy efficiency

The authors in [20] propose the optimum joint CC selection and resource block (RB) allocation scheme while satisfying the QoS requirements of the users with a full buffer traffic model. They also considered the latency for activating and deactivating the CCs and control channel overhead for switching CCs. While minimizing the UE power consumption to monitor the activated CCs, joint optimum CC selection and RB allocation problem is considered in [21], and it is assumed that the CC management can be applied frequently. In [31], the problem of energy-efficient CA is modeled as a multi-agent RL system. To consider the energy efficiency, the reward function for each UE (as an agent) is given in terms of the number of activated CCs and the achievable rate for the UE, and the action space for each UE corresponds to activating/deactivating a CC. The state space for a UE is the tuple of an indicator for useful action, an indicator of meeting the target rate requirement for the UE and the required number of CCs. The Q-learning algorithm is used to solve the proposed RL system, finding the optimal policy to activate and deactivate the SCCs for each UE.

In [79], it is assumed that all CCs are assigned to a UE, which can result in an increased UL overhead, e.g., Channel Quality Indicator (CQI) feedback. Therefore, strategies in different layers are described to reduce the UL overhead. In [22], the objective is to maximize the weighted sum of the utilities for the UEs subjected to the constraint of the number of activated CCs for each UE. Using the auxiliary variables and solving a sequence of successive geometric programming approximations of the problem, a CA and RB allocation algorithm is derived to address the problem sub-optimally. It is shown that the converges point of the proposed algorithm meets the Karush–Kuhn–Tucker (KKT) conditions of the problem. Considering the proportional fairness in the LTE-Advanced system, the authors in [23] intend to maximize the sum of the logarithmic average throughput for the UEs. A linear optimization problem relaxes the mentioned integer linear programming problem, and a resource management scheme is derived for CA, modulation and coding scheme assignment and RB allocation.

In [80], the bursty traffic in an LTE-Advanced system is modeled by the birth-death process (the special case of the continuous-time Markov process). For the UEs sharing a

Table 2.2: Resource allocation in the next-generation wireless networks with CA functionality

Ref.	Technology		Resource management problem			Methodology			
	CA	DC	Objective	EE	Var.	ML	Opt. theory	Game theory	Heuristic
[20]	✓	✗	Max. Total utility (in terms of rate for UE)	✗	RB and CC indicator	✗	✓	✗	✗
[21]	✓	✗	Min. Power consumption	✓	RB and CC indicator	✗	✓	✗	✗
[31]	✓	✗	Max. The rate for each UE Min. Number of activated SCC	✓	CC indicator	✓	✗	✗	✗
[22]	✓	✗	Max. Total utility (in terms of rate for UE)	✗	RB and CC indicator	✗	✓	✗	✗
[23]	✓	✗	Max. Sum of logarithmic average throughput for UEs	✗	RB and CC indicator and MCS	✗	✓	✗	✗
[80]	✓	✗	Find UE with max. value of fairness per RB on a CC	✗	Packet scheduling	✗	✗	✗	✓
[24]	✓	✗	Max. Total utility for UEs (logarithmic and sigmoidal-like function over throughput)	✗	CC indicator	✗	✓	✗	✗
[25] [26]	✓	✗	Max. Total throughput for UEs	✗	CC indicator	✗	✓	✗	✗
[14]	✓	✗	Max. System throughput	✗	CC indicator	✗	✗	✓	✗
[83]	✓	✗	Study different traffic on power consumption	✓	CC indicator	✗	✗	✗	✓
[84]	✓	✗	Max. Spectrum utilization Min. UE wake-up time	✓	CC indicator	✗	✗	✗	✓
[85]	✓	✗	Study DRX configuration on power consumption	✓	DRX parameters	✗	✗	✗	✓
[15]	✓	✗	Max. System throughput	✗	CC indicator	✗	✗	✓	✗
[27]	✓	✗	Min. Consumed power	✓	CC, UL pwr.	✗	✓	✗	✗
[82]	✓	✓	Load balancing	✗	CC and BS assign. indicator	✗	✗	✗	✓
[28]	✓	✓	Max. Spectrum efficiency Max. Energy efficiency	✓	DL power, CC, and BS assign. indicator	✗	✓	✗	✗

CC, the fairness metric for a UE is defined in terms of the ratio of the estimated instantaneous throughput of the UE to its average delivered throughput on the CC. The objective of the load balancing problem is to find the UE with the maximum fairness metric per RB on a CC. By considering two load balancing algorithms in LTE-Rel'8, i.e., Round Robin (RR) and Mobile Hashing (MH) balancing, a packet scheduling algorithm is proposed to improve the coverage performance and fairness among the users. In [24], the proportional fairness utility for each UE is given as the logarithmic and sigmoidal-like function of the achievable rate for the UE on the CCs. The objective is to maximize the total utility subjected to the constraint of the maximum achievable rate through each CC. The authors are derived a robust distributed resource allocation algorithm with CA functionality and provide optimal rates in high-traffic and low-traffic situations.

In a network where the millimeter-wave frequencies are used for higher data rate, the authors in [25, 26] employ CA to improve the performance network in terms of the total throughput. In [14], in a heterogeneous network with multi-flow CA, a hierarchical game theoretic-based algorithm is proposed to maximize the system throughput. The effect of CA on the application level throughput and delays of LTE networks are studied in [81]. Subjected to the constraint of high data rate requirements for the device-to-device (D2D) links, the problem of minimizing the total consumed power is addressed in [30]. Specifically, the CA technology is developed in D2D-enabled 5G networks to satisfy the rate requirements for the D2D links. The resource management problem is formulated as a mixed-integer optimization problem. The transformation and variable substitution are used to address the problem. A two-layer algorithm is proposed for joint UL power allocation and carrier aggregation.

Recently, the impact of UE power savings has also been considered in the resource management schemes proposed for 5G. Thus, based on the current standardization progress in 5G, the authors in [86] provide an overview of power-saving techniques. Considering the energy consumption for dynamically activating and deactivating an SCC, the authors in [21] and [31] studied the energy-aware resource allocation in the networks with CA functionality. In [84], a heuristic algorithm is proposed to maximize spectrum utilization and minimize UE wake-up time concurrently. In [87], the details for the power saving potential in radio resource control are studied by considering only one UE in a network without CA. In [83], for different traffic models, e.g., FTP and web browsing, the effect of the CC activation on UE power consumption is studied. In the power saving mechanism proposed in [84, 85, 88, 89], a UE can stop monitoring the Physical Downlink Control Channel (PDCCH) for a while (i.e., the DRX mechanism is used). Specifically, in [85], the duration and the frequency of the non-monitoring periods are optimized based on the QoS requirements for the applications used by the UE. The authors in [88] propose a power

and delay scheduling scheme for DL transmission with bursty packet data traffic. Meanwhile, in [89], the efficiency of using the DRX mechanism to prolong battery life is studied by measuring the power consumption, cell bandwidth, screen and CPU power consumption. However, DRX-based energy saving is challenging in practice due to the need for coordination between multiple activated CCs.

UL power-sharing and dual connectivity

This subsection briefly reviews the most recently proposed resource management schemes in the 5G networks with CA and DC. Some resource management schemes in the networks with CA technology have been derived in [15, 27]. In [15], it is assumed that more than one cell operator can serve the mobile devices. Specifically, a two-layer interacting game is formulated to maximize the system throughput. The upper layer comprises a Stackelberg game and is responsible for adjusting the spectrum price. Meanwhile, the lower game contains a bargaining game, and it allocates the spectrum resources to the users.

DC technology was first initiated in standard 3GPP Released 12 for the LTE networks. To employ the non-standalone 5G networks to simultaneously connect to the 5G and LTE, the standard 3GPP Released 15 has been developed, which is analyzed in [90]. In the networks with DC technology, some resource management problems have been tackled by the authors in [91–94]. To maximize energy efficiency in a heterogeneous network (HetNet) with DC, joint power control and traffic offloading scheme is derived in [91]. Specifically, an ergodic iterative search method is used for adaptive connectivity. Then, the authors used Lagrangian function and the gradient descent method to derive an optimal resource management scheme. In [92], in a MEC-enabled network with DC technology, the problem of minimizing the total energy consumption is formulated as a mixed-integer non-linear programming (MINLP) problem. The MINLP problem is first addressed by using some concepts from optimization theory. Then, deep learning is used to derive an intelligent offloading scheme. In a MEC-enabled HetNet with DC technology, to provide the edge devices with adequate resources, sub-6 GHz and mmWave BSs are employed in [93], and the benefits of using the corresponding links for the reliable delivery of the VR traffic are studied. In [94], the DC is used in a UAV-assisted HetNet. The UAVs are responsible for controlling reconfigurable intelligent surfaces (RISs) which provides a strong line-of-sight (LOS) connection with the ground users by operating on microwave channels in the sky. By considering orthogonal multiple access (OMA) over the microwave channel for the macro BSs and non-orthogonal multiple access (NOMA) over the mmWave channel for the small BSs, the authors formally express the problem of minimizing the total DL transmit power level for the macro and small BSs. The problem is decomposed into two sub-problems. The

former is solved by deriving an intelligent dueling deep Q-Network-based algorithm, and the latter sub-problem is tackled by using successive convex approximation in optimization theory.

In an E-UTRAN New Radio – Dual Connectivity (EN-DC) network consisting of one eNB and one gNB, to tackle the problem of UL power-sharing, the authors in [95] quantize the continuous UL transmit power levels for the users (i.e., by approximating power levels with amplitudes restricted to a prescribed set of values). Then the Q-learning algorithm [62] was used to address the problem of dynamic power-sharing. In [95], the CA technology has not been considered. Additionally, the Q-learning-based algorithm may not perform efficiently as the action space grows, i.e., by increasing the quantization levels or expanding the number of users. To increase the system throughput and coverage, DC/MC and CA technology have been used in the wireless networks [28, 82]. For instance, in [82], the authors derive a heuristic UE-BS association and CA scheme for the load (i.e., the number of assigned UEs to the BSs) balancing in the networks. Specifically, a UE selects its serving primary cell based on either received reference signal power or received reference signal quality. Then, a CC management scheme is applied for CA and secondary cell selection. In the LTE HetNets, the problem of jointly maximizing spectrum and energy efficiency is formulated as a bi-objective optimization problem in [28]. Using some optimization theory concepts, the authors derive a resource management scheme for CA, DL power control, and user association.

Chapter 3

Intelligent Resource Management and Offloading in MEC-enabled Wireless Networks

3.1 Introduction

In 5G and 6G networks, the fast growth of applications with immense computational needs, such as streaming video analysis, AR/VR mobile games, navigation of self-driving cars and drones, call for higher computational capacity than what a smart device can accommodate on board. Therefore, leveraging advanced wireless communications and MEC technologies to support advanced applications become inevitable. In MEC, mobile devices can offload the computation-intensive tasks to a nearby MEC server and augment their computing capability [4]. The MEC server or servers can be in an office, on a vehicle, in a base station cabinet, or even be another set of devices. MEC servers are accessed through a wireless network considering that lightweight mobile devices are the users that need the MEC capability.

In MEC-enabled networks, there are two possible offloading approaches; partial or binary offloading. In the former, the computation task is divided into two parts, executed locally at the mobile device and offloaded to the MEC server. For the latter, i.e., binary offloading, the whole task is performed either locally at the device or completely offloaded to a MEC server [4]. Task offloading in MEC has been studied widely in the literature. Due to the multi-dimensional and dynamic characteristics of the wireless networks, the task offloading problem is challenging. In next-generation wireless networks, with the increasing

number of users, user types, slices, servers and multi-RAT technologies, traditional algorithms based on optimization and game theory will suffer from scalability and flexibility. Specifically, the above methods would require complete information, which, in most cases, would not be available. Therefore, emerging AI techniques are promising to overcome some of these challenges.

In this part, we propose machine learning approaches to jointly take offloading decisions in MEC and allocate resources in the wireless network. In Section 3.2, by considering both partial offloading and binary offloading schemes in MEC-assisted networks, we propose a multi-agent DRL-based algorithm to tackle the problem of maximizing the computation efficiency [1]. In Section 3.3, we address the problem of partial offloading of the SFCs, meaning some VNFs are executed at the device locally and others at the servers based on UE constraints. Compared with the studies on task-dependent offloading in the literature, the problem for partial offloading of an SFC calls for consideration of constraints for the SFC placement. Therefore, in [2] we consider SFC placement constraints and task offloading constraints. In Section 3.4, we consider a UAV-MEC-assisted smart farm. Focusing on the topology of the complex application performed by the UAVs, we employ the combination of DRL and GCN, and derive an intelligent joint offloading and power control algorithm in [3].

3.2 On Joint Offloading and Resource Allocation: A Double Deep Q-Network Approach

In this section, we consider binary and partial offloading problems. We seek to find optimal decisions for jointly offloading and resource allocation, maximizing the number of computed bits while at the same time minimizing the energy consumption [1]. This allows improved usage of UL transmit power and local CPU resources. We propose the Deep Reinforcement Learning for Joint Resource Allocation and Offloading (DJROM) algorithm that uses the double deep Q-network approach and models UEs as agents [1]. We compare the proposed method with two other machine learning-based techniques, namely, Multi-Agent Deep Q-learning (MARL-DQL) and Multi-Agent Deep Q-Network (MARL-DQN) under fixed and mobile scenarios. Our results show that the DJROM scheme enhances the efficiency better than the other compared algorithms.

3.2.1 System Model and Problem Formulation

System Model and Assumptions

We consider a MEC-enabled wireless network consisting of a single base station (BS), a set of M users, $\mathcal{M} = \{1, \ldots, M\}$, and a single MEC server. We consider these devices can use both partial and binary computation offloading schemes. In the partial offloading scheme, the computation tasks can be divided into two parts, one for local computing and the other for offloading. For the binary offloading scheme, the computation tasks can be performed either entirely at the local device or at the MEC server.

For the offloading process, users employ the NOMA technique to improve the offloading throughput by concurrently offloading their tasks on a frequency band (with a limited number of users). The available NOMA techniques can broadly be divided into two major categories, i.e., power-domain NOMA and code-domain NOMA which attain multiplexing in power domain and multiplexing in code domain, respectively. Here the typical UL power-domain NOMA in single cell scenario is used [96–99]. To perform successive interference cancellation (SIC), we use a similar approach as in [4] where the users' channel gain determine the decoding order [96–99], i.e., using the simple decoding order based on the order of the channel [4]. In doing so, without loss of generality, similar to [4], it is assumed that the channel gains between the users (employing NOMA protocols) and the BS have an ascending order, i.e., $h_1 < \ldots < h_M$, and based on it, the users are decoded in increasing order. Thus, user m sees the interference based on the decoding order from the others, i.e., $I_m = \sum_{i>m}^{M} p_i h_i$ [4, 98, 99]. Therefore, considering the limited number of users in the frequency band with NOMA (which is used for task offloading), we use the Signal-to-Interference-Noise-Ratio (SINR) and rate formulation in [4]. This rate and SINR formulations are used in [98, 99] as well. We also assume that each device has a single antenna.

For a given time t, let us define $\alpha_m(t)$ and $\beta_m(t)$ as offloading indicators for partial and binary offloading modes, respectively. Based on the offloading mode chosen by user m, we define:

$$\alpha_m(t) = \begin{cases} 1 & \text{if user } m \text{ chooses partial offloading,} \\ 0 & \text{if user } m \text{ chooses binary offloading.} \end{cases}$$

Additionally, if user m chooses binary offloading, we define:

$$\beta_m(t) = \begin{cases} 1 & \text{if user } m \text{ performs computation} \\ & \text{completely at MEC server,} \\ 0 & \text{if user } m \text{ completely performs} \\ & \text{computation locally.} \end{cases}$$

In the following model, we use the above binary indicators to define the SINR for a given user m. Specifically, let $\mathbf{p}(t) = [p_i(t)]^{\mathrm{T}}_{i\in\mathcal{M}}$ be the UL transmit power vector for the users to offload their computation task. The UL transmit power level for the users are limited, i.e., $\mathbf{0} \leq \mathbf{p}(t) \leq \overline{\mathbf{p}}$ where $\overline{\mathbf{p}} = [\overline{p}_i]^{\mathrm{T}}_{i\in\mathcal{M}}$ and $\overline{p}_i$ is maximum UL transmit power level for user m to offload a computation task at the MEC server. Let h_m be the channel gain between the mth user and the BS at duration T (T is the frame duration for the users during which the channel power gain for the users are assumed to be fixed [4][1]). At time t, let $\boldsymbol{\alpha}(t) = [\alpha_m(t)]_{\forall m\in\mathcal{M}}$, and $\boldsymbol{\beta}(t) = [\beta_m(t)]_{\forall m\in\mathcal{M}}$. The UL SINR for user m is denoted by $\Gamma_m[\mathbf{p}(t), \boldsymbol{\alpha}(t), \boldsymbol{\beta}(t), \mathbf{h}]$. For more convenience, instead of using $\Gamma_m[\mathbf{p}(t), \boldsymbol{\alpha}(t), \boldsymbol{\beta}(t), \mathbf{h}]$, we apply the brief notation $\Gamma_m(t)$ for the SINR of user m and express it as,

$$\Gamma_m(t) = \frac{p_m(t)h_m}{\sum\limits_{i>m}^{M} (\alpha_i(t) + \beta_i(t))p_i(t)h_i + \sigma_0^2}, \tag{3.1}$$

where σ_0^2 is the noise. Here, for any given user m, to offload a task, we can express the number of computed bits and consumed energy. The details for both offloading schemes, partial and binary offloading, are given as follows.

Partial offloading: In this offloading mode, as mentioned before, for a given user m, the computation tasks can be divided into two parts, one for local computing and the other for offloading. For the local computation, let us assume C is the number of cycles required for computing one bit of raw data at the user side CPU. Additionally, at a given time t, let $f_m(t)$ be the CPU frequency for user m. $f_m(t)$ is limited to the maximum CPU frequency $\overline{f}_m$, i.e., $0 \leq f_m(t) \leq \overline{f}_m$. Therefore, the number of computed bits which is locally performed by user m and the consumed energy for the user are $\frac{Tf_m(t)}{C}$ (equation (4b) in [4]) and $T\gamma_c f_m^3(t)$, respectively, where γ_c is the effective capacitance coefficient of the processor's chip. It is dependent on the chip architecture [4]. Additionally, by using

[1]In [1], in each episode t in the proposed RL-based algorithms, the channel power gain for each user is considered to be fixed. Accordingly, the notation t corresponds to episode t in proposed RL-based algorithms in the following equations.

NOMA protocol for offloading, the number of computed bits which is offloaded onto the MEC server is $\frac{B\tau}{\nu_m}\log\Big(1+\Gamma_m(t)\Big)$ (equation(6b) in [4]). Accordingly, for the offloading process, the total number of computed bits, i.e., $R_m(t)$, and the total consumed energy (energy spent for computation and offloading), i.e., $E_m(t)$, can be expressed as [4]:

$$R_m(t) = \frac{Tf_m(t)}{C} + \frac{B\tau}{\nu_m}\log\Big(1+\Gamma_m(t)\Big), \tag{3.2}$$

and,

$$E_m(t) = \tau_0 P_{r,m} + \epsilon\tau(p_m(t) + P_{c,m}) + T\gamma_c f_m^3(t), \tag{3.3}$$

In the above equations, offloading task for user m consists of both raw data and communication overhead. Additionally, ν_m is the communication overhead for user m, ϵ is the amplifier coefficient, $P_{r,m}$ is the received power for the received signal processing during the transmission, and $P_{c,m}$ is the constant circuit power consumption for user m during the computation offloading.

Binary offloading: For a given user $m \in \mathcal{M}$ that completely performs its computation task locally, the total number of locally computed bits and energy consumption for computation are given as [4]:

$$R_m(t) = \frac{Tf_m(t)}{C}, \tag{3.4}$$

and,

$$E_m(t) = \tau_0 P_{r,m} + T\gamma_c f_m^3(t), \tag{3.5}$$

For a user $m \in \mathcal{M}$ that completely offloads its computation task at the MEC server, we have [4],

$$R_m(t) = \frac{B\tau}{\nu_m}\log\Big(1+\Gamma_m(t)\Big), \tag{3.6}$$

and,

$$E_m(t) = \tau_0 P_{r,m} + \epsilon\tau(p_m(t) + P_{c,m}), \tag{3.7}$$

Let $R_m^{\min}$ be the minimum number of computed bits required by a given user m. We say that the QoS for user m is satisfied if we have,

$$R_m(t) \geq R_m^{\min}. \tag{3.8}$$

It can be said the network is feasible, if (3.8) is satisfied for all users (i.e., $R_m \geq R_m^{\min}$, $\forall m \in \mathcal{M}$). Therefore, the network feasibility depends on some parameters, e.g., the number of users, offloading schemes chosen by the users, path gain for the users, their CPU frequency,

$$\eta_m(t) = \alpha_m\Big[\frac{Tf_m(t)}{C} + \frac{B\tau}{\nu_m}\log\Big(1+\Gamma_m(t)\Big) - w_m\Big(\tau_0 P_{r,m} + \epsilon\tau(p_m(t)+P_{c,m}) + T\gamma_c f_m^3(t)\Big)\Big]$$
$$+ (1-\alpha_m)\Big[\beta_m\Big(\frac{B\tau}{\nu_m}\log\Big(1+\Gamma_m(t)\Big) - w_m\Big(\tau_0 P_{r,m} + \epsilon\tau(p_m(t)+P_{c,m}\Big)\Big)$$
$$+ (1-\beta_m)\Big(\frac{Tf_m(t)}{C} - w_m\Big(\tau_0 P_{r,m} + T\gamma_c f_m^3(t)\Big)\Big)\Big]. \tag{3.9}$$

UL transmit power level for the users, and their QoS requirements[2]. Note that, here, our QoS definition only covers the computation rate. Other QoS metrics such as latency may also be likely. In (3.8), based on the offloading mode selected by a user m, $R_m(t)$ is obtained from equations either (3.2), (3.4), or (3.6).

At a given time t, let $\eta_m(t)$ denote the immediate reward function for user m in terms of computed bits and cost. Specifically, computed bits is the most commonly used metric to formally express the objective function, e.g., computation efficiency. For instance, in reference [4], the objective is maximizing the computation efficiency, defined as the ratio of computed bits to the consumed energy. In [1], we also define the immediate reward for a given user in terms of the computed bits, i.e., the difference between the computed bits and consumed energy for the users. To formally express the immediate reward function for the user, we use the equations for the computed bits and consumed energy in reference [4] (i.e., equations (3.2),(3.3), (3.4), (3.5), (3.6), (3.7)). This reward function corresponds to the computation efficiency. So, we can express,

$$\eta_m(t) = R_m(t) - w_m E_m(t),$$

where w_m is the unit price of energy which is consumed by user m for offloading[3]. By employing the offloading indicators, $\boldsymbol{\alpha}(t)$ and $\boldsymbol{\beta}(t)$, we extend the above equation and restate it in (3.9).

[2]In [1], the *main objective* is to maximize the long-term reward for the users (which is given in terms of the computation efficiency). At the same time, we aim to satisfy their QoS requirements. However, there is no strict guarantee for satisfying QoS for all users. We assume users are admitted as long as QoS can be satisfied.

[3]For a given user m, w_m can be set by using (A.4) . The details are discussed in **Appendix A.2**.

Problem Formulation: Joint Resource Allocation and Offloading Management

To formulate our problem, we use the immediate reward for a given user m stated in equation (3.9). In doing so, let Φ_m denote the long-term reward for user m. Φ_m is given (in 3.10) as the weighted sum of the immediate rewards for user m over the finite time T as follows,

$$\Phi_m = \sum_{t=0}^{T-1} \lambda^t \eta_m(t), \tag{3.10}$$

where $\lambda \in [0, 1)$ is the discount factor to determine the weight of the future reward. In (3.10), if $\lambda = 0$, only the immediate reward is taken into account. Meanwhile, when $\lambda < 1$, the weighted immediate reward (i.e., $\lambda^t \eta_m(t)$) in the future is less than that in the earlier periods. Accordingly, to concurrently maximize the long-term reward for each user m, we formulate the problem of joint resource allocation and offloading management (**JROM**) as an optimization problem. That is,

$$\max_{\mathbf{p},\mathbf{f},\boldsymbol{\alpha},\boldsymbol{\beta}} \quad \Phi_m \qquad \forall m \in \mathcal{M}. \tag{3.11}$$

In the optimization problem (3.11), the users concurrently try to maximize their long-term rewards. Based on (3.9) and (3.10), in optimization problem (3.11), when a user m offloads its computation task either partially or completely at the MEC server, maximizing the long-term reward for user m corresponds to maximizing the long-term number of computed bits for that user while minimizing the consumed energy for offloading, and computing. Minimizing the consumed energy for offloading corresponds to minimizing the interference imposed on the user.

Note that the optimization problem in (3.11) is a multi-objective non-convex optimization problem with combinatorial characteristics; therefore, obtaining a globally optimal solution for (3.11) is challenging. Furthermore, employing the traditional method in optimization theory for addressing (3.11) would require nearly complete information about the environment. In what follows, we use RL, particularly the Multi-Agent Reinforcement Learning (MARL) method, to concurrently optimize the users' long-term rewards. We assume that the users can get the global state information with message exchanges.

To simultaneously optimize the long-term reward for the users, we first formulate a stochastic game. Then, we employ MARL and propose a method to solve it. The obtained equilibrium would be an optimal solution for the optimization problem in (3.11).

Stochastic Game Formulation

We assume that the users do not have complete information about the network environment; additionally, they are selfish and rational. Each user sets its own UL transmit power level and CPU frequency for offloading to maximize the long-term reward. At any given time t, the immediate reward for each user is given in terms of the current state of the network environment and other users' action in the environment. The next state is influenced by the current state and the actions selected by all users. Therefore, this optimization issue can be formulated as a stochastic game $\mathcal{G} = \langle \mathcal{M}, \mathcal{S}, \mathcal{A}_m, P_{ss'}, \eta_m \rangle$ [59] where,

- $\mathcal{M}$ is the set of users,
- $\mathcal{S}$ is the set of possible states,
- $\mathcal{A}_m$ is the set of actions for user m,
- P is the transition probability function, and
- η_m is the reward function for user m.

Given a time t, let $s(t)$ be the state indicating whether any user $m \in \mathcal{M}$ meets its own QoS requirement or not. That is, $s(t) = [s_m(t)]^{\mathrm{T}}_{m \in \mathcal{M}}$, where $s_m(t) \in \{0, 1\}$, $\forall m \in \mathcal{M}$. In more details, for a given user m, if (3.8) holds true, then $s_m(t) = 1$; otherwise, $s_m(t) = 0$. Accordingly, the number of possible states is 2^{M}, and it will increase exponentially as M grows.

At time t, user m must choose the offloading mode. Therefore, the action space for a given user m is the set of all offloading modes that the user can opt. By using the values for α_m and β_m (the offloading indicators for user m), we define $\mathcal{A}_m = \{a_m | a_m = \alpha_m \beta_m\}$ where $\alpha_m \beta_m \in \{1x, 00, 01\}$. For user m, $\alpha_m \beta_m = 1x$ corresponds to partial offloading mode, $\alpha_m \beta_m = 00$ is for binary offloading mode with completely local computation, and $\alpha_m \beta_m = 01$ is for binary offloading mode where computation is wholly performed at MEC server. Accordingly, the number of possible actions for user m is 3.

We define the set $\mathcal{A}$ as the Cartesian product of the action space for the users, i.e., $\mathcal{A} = \mathcal{A}_1 \times \ldots \times \mathcal{A}_M$. Let $\mathbf{a} = (a_1, \ldots, a_M)$ and $\mathbf{a} \in \mathcal{A}$ is the action vector (action profile) for the users where $a_m \in \mathcal{A}_m$. The state transition probability function is a function from $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$ to $[0, 1]$ (i.e., $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$). Specifically, the state space is given in terms of the number of computed bits for each user and the minimum QoS requirements. Based on the former computed bits and the equations (3.2), (3.4), and (3.6), the state is

a function over CPU frequency, UL transmit power level and the path gain for the users. Additionally, the action space is given in terms of the offloading scheme for the users. Accordingly, the transmission probability is related to minimum QoS requirements, CPU frequency, UL transmit power level, the path gain and the offloading scheme for the users. Additionally, $P_{ss'}(\mathbf{a})$ is the probability of transition to state s' if the action profile $\mathbf{a}$ is employed by the users in state s. Given $\mathbf{a}_{-m} = (a_1, \ldots, a_{m-1}, a_{m+1}, \ldots, a_M)$, the reward for user m, i.e., $\eta_m(t)$ can be given by $\eta_m(t) = \eta_m(s(t), a_m, \mathbf{a}_{-m})$.

Let $(a_m^*, \mathbf{a}_{-m}^*)$ be a possible solution for the game. We say $(a_m^*, \mathbf{a}_{-m}^*)$ is a Nash equilibrium (NE) , if, for any given $s \in \mathcal{S}$ and user $m \in \mathcal{M}$, the following inequality holds true [100],

$$\eta_m(s, a_m^*, \mathbf{a}_{-m}^*) \geq \eta_m(s, a_m, \mathbf{a}_{-m}^*) \quad \forall a_m \in \mathcal{A}_m. \tag{3.12}$$

As shown in [100], for user m, the action a_m^* can be regarded as the best response to the others, and no user can benefit from unilateral deviation.

This stochastic game is episodic, where the state is reset at the end of each episode to start a new one. Additionally, for any episode, the policy is composed of states, actions, and rewards which can be also used to obtain accumulative rewards from the environment. Therefore, the information about the reward function for each user and state transition are required to find the NE. However, the mentioned information is unknown to each user, creating the main challenge in solving the game. To deal with this challenge, an RL-based method obtains a NE point at each state s. The details are given in what follows.

Multi-agent RL Method for Addressing JROM Problem

Markov properties are used in this part to find a NE for the stochastic game $\mathcal{G}$. To describe this stochastic game, we introduce a finite Markov decision process (MDP) and propose a multi-agent Q-learning method to solve it.

To describe the MDP, we use the discrete state space $\mathcal{S}$ for the environment states, the discrete action space $\mathcal{A}_m$ for the possible actions of any user m, $\eta_1, \ldots, \eta_M$ for the reward functions for the users, and $P_{ss'}(\boldsymbol{\pi})$ for the state transition probability under the policy vector $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_M)$.

Given an unknown stochastic environment, to maximize the long-term reward for each user, the RL methods have been used in [61] to find the optimal policy. In [1], based on its distributed characteristic, we employ collaborative MARL with local states. Let π_m^* be the optimal policy for a user m which is a function from state space $\mathcal{S}$ to the action space for the user, i.e., $\pi_m^* : \mathcal{S} \to \mathcal{A}_m$. Note that the number of computed bits for each user

should be known in order to choose the next state, based on (3.8). Therefore, at any state $s \in \mathcal{S}$, to maximize its state-value function, each user tries to learn $a_m^* = \pi_m^*(s)$ where $\pi_m^*(s) \in \mathcal{A}_m$. The state-value function for user m is the accumulative expected discounted reward for the user from a given state.

For a given policy vector $\boldsymbol{\pi} = (\pi_m, \boldsymbol{\pi}_{-m})$, let $V_m(s, \pi_m, \boldsymbol{\pi}_{-m})$ denote the state-value function for user m at state s. Given state s and policy vector $\boldsymbol{\pi}$, the state-value function for user m, i.e., $V_m(s, \pi_m, \boldsymbol{\pi}_{-m})$, is defined as [61],

$$\begin{aligned} V_m(s, \pi_m, \boldsymbol{\pi}_{-m}) = E\Big[\sum_{t=0}^{T-1} \lambda^t \eta_m \Big(s(t), \pi_m(t), \boldsymbol{\pi}_{-m}(t) \Big) \\ \mid s(0) = s \Big], \end{aligned} \tag{3.13}$$

where $E[.]$ denotes the expectation operator. By extending (3.13), we have [33],

$$\begin{aligned} V_m(s, \pi_m, \boldsymbol{\pi}_{-m}) = u_m(s, \pi_m, \boldsymbol{\pi}_{-m}) \\ + \lambda \sum_{s' \in \mathcal{S}} P_{ss'}(\boldsymbol{\pi}) V_m(s', \pi_m, \boldsymbol{\pi}_{-m}), \end{aligned} \tag{3.14}$$

where $u_m(s, \pi_m, \boldsymbol{\pi}_{-m}) = E[\eta_m(s, \pi_m, \boldsymbol{\pi}_{-m})]$. Additionally, given action $a_m \in \mathcal{A}_m$ and state $s \in \mathcal{S}$, let $Q_m(s, a_m)$ denote the action-value function for user m under policy $\boldsymbol{\pi}$[4]. The action-value function for user m is stated as:

$$\begin{aligned} Q_m(s, a_m) = E\Big[\sum_{t=0}^{T-1} \lambda^t \eta_m \Big(s(t), \pi_m(t), \boldsymbol{\pi}_{-m}(t) \Big) \\ \mid s(0) = s, a_m(0) = a_m \Big], \end{aligned} \tag{3.15}$$

As mentioned in [61], for an agent, the optimal state-value function and action-value function are obtained under the optimal policy. Let $\boldsymbol{\pi}^* = (\pi_m^*, \boldsymbol{\pi}_{-m}^*)$ denote the optimal policy vector for the users. We say $\boldsymbol{\pi}^* = (\pi_m^*, \boldsymbol{\pi}_{-m}^*)$ is a NE, if for any given user m and any π_m, the following inequality holds true at each state $s \in \mathcal{S}$,

$$V_m(s, \pi_m^*, \boldsymbol{\pi}_{-m}^*) \geq V_m(s, \pi_m, \boldsymbol{\pi}_{-m}^*). \tag{3.16}$$

To find the NE $(\pi_m^*, \boldsymbol{\pi}_{-m}^*)$, it is required to address a MDP problem which is given by [33],

$$V_m(s, \pi_m^*, \boldsymbol{\pi}_{-m}^*) = \max_{a_m \in \mathcal{A}_m} Q_m^*(s, a_m), \tag{3.17}$$

[4]For more convenience, given state s and action a_m for a user m, we use the notation $Q_m(s, a_m)$ instead of $Q_m(s, a_m, \pi_m, \boldsymbol{\pi}_{-m})$ for action value function for that user under policy $\boldsymbol{\pi}$.

wherein for a given state s and action $a_m \in \mathcal{A}_m$, $Q_m^*(s, a_m)$ is the optimal action-value function for user m, and it is obtained from the following optimal Bellman equation,

$$Q_m^*(s, a_m) = u_m(s, a_m, \boldsymbol{\pi}_{-m}^*) + \lambda \sum_{s' \in \mathcal{S}} P_{ss'}(a_m, \boldsymbol{\pi}_{-m}^*) V_m(s', \pi_m^*, \boldsymbol{\pi}_{-m}^*).$$

In the above equation, for a MEC-enabled wireless network, it is challenging to receive the information about the state transition probability [i.e., $P_{ss'}(a_m, \boldsymbol{\pi}_{-m})$]. We use the Q-learning method proposed in [62] to deal with this challenge. In the Q-learning method, the optimal policy for a user m (i.e., π_m^*) can be found recursively by using the available information $[s, a_m, s', u_m(s, a_m, \boldsymbol{\pi}_{-m})]$. Based on [62], at a state $s \in \mathcal{S}$, for a given action $a_m \in \mathcal{A}_m$, the action-value function for a user m is updated as follows,

$$\begin{aligned} Q_m(s, a_m) \leftarrow Q_m(s, a_m) + \delta[u_m(s, a_m, \boldsymbol{\pi}_{-m}) \\ + \lambda \max_{a'_m \in \mathcal{A}} Q_m(s', a'_m) - Q_m(s, a_m)], \end{aligned} \tag{3.18}$$

where δ is the learning rate to determine the update of $Q_m(s, a_m)$.

3.2.2 JROM Algorithm

Up to now, to simultaneously maximize the long-term reward for the users, as provided in optimization problem (3.11), in an unknown stochastic environment, we have first expressed the stochastic game $\mathcal{G} = \langle \mathcal{M}, \mathcal{S}, \mathcal{A}_m, P_{ss'}, \eta_m \rangle$ in Subsection 3.2.1. We then used the Q-learning MARL method to find a NE for the game. In this section, we complete our discussion and derive an algorithm for solving the JROM problem.

Offloading Mode Selection

One of the challenges in RL is the tradeoff between exploration and exploitation. To balance the ratio of exploration and exploitation, ϵ-greedy is a promising policy for action adoption. In ϵ-greedy policy, at each state, an action is selected randomly with the probability of ϵ, and the best one (the action that has the maximum action-value function) is chosen with the probability of $1 - \epsilon$. In [1], we use the ϵ-greedy policy for action selection mechanism where at each state s, given $Q_m(s, a_m), \forall a_m \in \mathcal{A}_m$ for user m, the action, i.e., offloading scheme, is chosen by using ϵ-greedy policy stated as,

$$a_m = \begin{cases} \arg\max_{a \in \mathcal{A}_m} Q_m(s, a) & \text{with probability of } 1 - \epsilon, \\ \text{randomly selected from } \mathcal{A}_m & \text{with probability of } \epsilon. \end{cases} \tag{3.19}$$

$$\eta_m(\mathbf{p}, f_m) = \begin{cases} \dfrac{Tf_m}{C} + \dfrac{B\tau}{\nu_m} \log\Big(1 + \dfrac{h_m p_m}{\sum\limits_{i>m}^{M} (\alpha_i + \beta_i) h_i p_i + \sigma_0^2}\Big) \\ -w_m\Big(\tau_0 P_{r,m} + \epsilon\tau(p_m + P_{c,m}) + T\gamma_c f_m^3\Big) & \text{if } a_m = 1x, \\ \dfrac{Tf_m}{C} - w_m\Big(\tau_0 P_{r,m} + T\gamma_c f_m^3\Big) & \text{if } a_m = 00, \\ \dfrac{B\tau}{\nu_m} \log\Big(1 + \dfrac{h_m p_m}{\sum\limits_{i>m}^{M} (\alpha_i + \beta_i) h_i p_i + \sigma_0^2}\Big) \\ -w_m\Big(\tau_0 P_{r,m} + \epsilon\tau(p_m + P_{c,m})\Big) & \text{if } a_m = 01\ . \end{cases} \tag{3.20}$$

Resource Management: Power Control and Frequency Selection

Given the action chosen by the users in a state, adequate resources should be allocated to the users for performing their tasks. Specifically, for a user m, given state $s \in \mathcal{S}$, and given action $a_m \in \mathcal{A}_m$ (i.e., given α_m, β_m), the immediate reward for the user is given in (3.20). In equation (3.20), as explained before in Subsection 3.2.1, each member in action space for user m corresponds to offloading scheme (given by α_m and β_m). Specifically, the binary sequence $\alpha_m\beta_m = 1x$ corresponds to partial offloading mode at the MEC server, $\alpha_m\beta_m = 00$ is for the binary offloading mode where computation is completely performed at the user side, and $\alpha_m\beta_m = 01$ corresponds to the binary offloading mode where computation is completely performed at MEC server.

As mentioned before, an environment state s indicates whether the QoS requirement for any user is satisfied or not. Therefore, the number of computed bits for each user should be obtained to choose the next state, based on (3.8). In doing so, the transmit power level and the CPU frequency for each user m (i.e., p_m and f_m, respectively) should be set. The users try to simultaneously maximize their immediate reward (based on their current state and their chosen action). Let us call the above problem as joint power control and frequency selection (**PCFS**). The problem of **PCFS** can be expressed as follows,

$$\max_{\mathbf{0} \le \mathbf{p} \le \overline{\mathbf{p}}, 0 \le f_m \le \overline{f}_m} \quad \eta_m(\mathbf{p}, f_m) \qquad \forall m \in \mathcal{M}. \tag{3.21}$$

In the following, $\mathcal{M}_1$ denotes the set of users either partially offloading their computation task or completely performing it locally, i.e., $\mathcal{M}_1 = \{m \in \mathcal{M} \mid \alpha_m\beta_m = 1x \text{ or } \alpha_m\beta_m = 00\}$, $\mathcal{M}_2$ denotes the set of users either partially or completely offloading their computation

task onto MEC server, i.e., $\mathcal{M}_2 = \{m \in \mathcal{M} \mid \alpha_m\beta_m = 1x \text{ or } \alpha_m\beta_m = 01\}$. The following theorem can obtain an optimal solution for the above optimization problem.

Theorem 3.1. *Given action vector* $\mathbf{a} = [a_m]^{\mathrm{T}}_{m\in\mathcal{M}}$ *for the users, let* $(\mathbf{p}^*, \mathbf{f}^*)$ *be an optimal solution for* (3.21)*, where* $\mathbf{f}^* = [f^*_m]^{\mathrm{T}}_{m\in\mathcal{M}}$ *and* $\mathbf{p}^* = [p^*_m]^{\mathrm{T}}_{m\in\mathcal{M}}$*. Then, we have,*

$$f^*_m = \sqrt{(3\gamma_c C w_m)^{-1}} \qquad \text{if } m \in \mathcal{M}_1 \tag{3.22}$$

and

$$p^*_m = \frac{B}{\epsilon\nu_m w_m} - \frac{I_m(\mathbf{p}^*)}{h_m} \qquad \text{if } m \in \mathcal{M}_2. \tag{3.23}$$

In the above equations, $I_m(\mathbf{p}^*) = \sum_{i>m, i\in\mathcal{M}_2} h_i p^*_i + \sigma_0^2$ *and* h_i *is the path gain between user* i *and the BS.*

Proof. See **Appendix A.1**. □

Based on the above theorem, for any user $m \in \mathcal{M}_1$, the CPU frequency would be conveniently set to the value given in (3.22). For a user $m \in \mathcal{M}_2$, the UL transmit power level for their offloading task can be obtained by using the closed-form expression in (3.23). Let $\mathbf{p}_2 = [p_j]^{\mathrm{T}}_{j\in\mathcal{M}_2}$ denote the UL transmit power vector for the users in $\mathcal{M}_2$, offloading their tasks onto MEC server (either partially or completely). By performing some matrix operations at the BS, the UL transmit power vector for users, i.e., $\mathbf{p}_2$, are obtained by solving the following equation,

$$\mathbf{p}_2 = \mathbf{D}^{-1}\mathbf{\Gamma}, \tag{3.24}$$

where $\mathbf{\Gamma} = \left[\frac{B}{\epsilon\nu_m w_m} - \frac{\sigma_0^2}{h_m}\right]^{\mathrm{T}}_{m\in\mathcal{M}_2}$ and $\mathbf{D} = [d_{im}]$ is a $\mid \mathcal{M}_2 \mid \times \mid \mathcal{M}_2 \mid$ matrix where $d_{im} = \frac{h_i}{h_m}$ if $i \geq m$; otherwise, $d_{im} = 0$. It is worth mentioning that in equation (3.24) and (3.22), for a given user m, the transmit power level and the CPU frequency for that user are limited to the values $\overline{p}_m$ and $\overline{f}_m$, respectively. Therefore, for any given user $m \in \mathcal{M}$, the unit price of consumed energy for offloading should be set properly, which is discussed in detail in **Appendix A.2**.

Based on the above discussion, we propose the **JROM** algorithm in **Algorithm 3.1**.

In the **JROM** algorithm, at each step for a given episode, first, a user m chooses its action by using the ϵ-greedy algorithm, (3.19). The selected action a_m corresponds to the offloading mode chosen by user m to perform its computation either partially at the MEC server, completely at the MEC server, or completely locally at the user side. Then, given

Algorithm 3.1 Proposed **JROM** algorithm

Input: The action set for each user (i.e., $\mathcal{A}_m, \forall m \in \mathcal{M}$) consisting of actions (partial offloading, binary offloading with completely local computation, binary offloading with complete computation at MEC), duration of offloading at MEC (i.e., τ), QoS requirement for each user (i.e., $R_m^{\min}, \forall m \in \mathcal{M}$);

Output: Optimal policy (i.e., the optimal sequence of actions) for satisfying the QoS requirement for all users (if the network is feasible) or a subset of users (if the network is infeasible);

Initialize: For any given state $s \in \mathcal{S}$, any user $m \in \mathcal{M}$, and action $a_m \in \mathcal{A}_m$ set action-value $Q_m(s, a_m)$ to an arbitrary value;

procedure
  for each episode **do**
    Initialize the network state s;
    Initialize the unit price for consumed energy (i.e., $w_m, \forall m \in \mathcal{M}$) by using (A.4);
    for each step **do**
      At state s, each user $m \in \mathcal{M}$ chooses a_m by employing the ϵ-greedy policy in (3.19);
      Based on the selected action a_m, the CPU frequency and uplink transmit power level for each user $m \in \mathcal{M}$ (i.e., f_m and p_m) are obtained by using (3.22) and (3.24), respectively;
      Given $\mathbf{p}, \mathbf{f}, \mathbf{a}$, each user $m \in \mathcal{M}$ checks if its QoS requirement is satisfied or not by using (3.8);
      All users obtain the next state s' through the message passing. Set $s \leftarrow s'$;
      Each user $m \in \mathcal{M}$ updates $Q_m(s, a_m)$ by using (3.18);
      if the QoS for each user is satisfied; i.e., $s = \mathbf{1}$ **then**
        break;
      end if
    end for
  end for
end procedure

a_m, user m obtains the CPU frequency and UL transmit power level for offloading onto the MEC server (i.e., f_m and p_m, respectively). Now, the users are ready to obtain the next state s' by checking if their QoS requirements are satisfied or not. Having the next state s', each user $m \in \mathcal{M}$ updates its action-value based on the updating function in (3.18). Obviously, when equation (3.8) holds true for all users (i.e., for all $m \in \mathcal{M}$, we have $s_m(t) = 1$ and thus $s = \mathbf{1}$), the current episode terminates; otherwise, the task for updating the action-value of the users continues for T steps. When the algorithm stops, if the network is feasible, the optimal policy corresponding to the sequence of offloading modes selected by the users at each step is obtained. Additionally, the QoS requirements for the users are satisfied. Otherwise, if the network is infeasible, the obtained optimal policy would not result in satisfying the QoS requirements for all users.

3.2.3 Deep RL for Joint Resource Allocation and Offloading

In the previous sections, we used a Q-learning-based MARL method to address the **JROM** problem, and we proposed **Algorithm 3.1**. Note that in our proposed algorithm, by increasing the number of users, the number of states would increase exponentially; thus, the employed Q-learning-based method would not perform effectively in finding the optimal policy. In this section, we resort to deep learning and use the DRL-based form in [58] to deal with the mentioned issue.

In high-dimensional problems (problems with either huge state space or huge action space), the non-linear function approximation methods, e.g., neural networks, can be employed to obtain the value functions. For instance, in our problem, when the number of users, and consequently the number of states, increase to learn all action values for all states, we can learn the parametrized action-value function for a given user. Let us denote $Q_m(s, a_m, \boldsymbol{\theta})$ as the parametrized action-value function for a user $m \in \mathcal{M}$. Given $[s, a_m, u_m(s, a_m), s']$, by using a DNN to approximate the action-value function for a user, the DQL algorithm for updating parameter $\boldsymbol{\theta}$ is given by [61],

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \zeta(Y_m^{\mathrm{Q}} - Q_m(s, a_m, \boldsymbol{\theta}))\nabla_{\boldsymbol{\theta}} Q_m(s, a_m, \boldsymbol{\theta}). \tag{3.25}$$

In (3.25), ζ is a scalar step size, and the target for user m is given by,

$$Y_m^{\mathrm{Q}} = u_m(s, a_m) + \lambda \max_{a' \in \mathcal{A}_m} Q_m(s', a', \boldsymbol{\theta}). \tag{3.26}$$

To improve the performance and conquer the learning instability, the DQN algorithm [63] can be employed to update DNN parameters. In DQN, a *target network*, *online*

network and the *experience replay* are used to update the parameters. Let $\boldsymbol{\theta}^-$ and $\boldsymbol{\theta}$ be the parameters for the target and online networks, respectively. The parameters of the online network are copied after every N step to the target network so that $\boldsymbol{\theta}^- = \boldsymbol{\theta}$. Accordingly, the target used by DQN is,

$$Y_m^{\mathrm{DQN}} = u_m(s, a_m) + \lambda \max_{a' \in \mathcal{A}_m} Q_m(s', a', \boldsymbol{\theta}^-), \tag{3.27}$$

and by replacing Y_m^{DQN} in (3.25), we have,

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \zeta(Y_m^{\mathrm{DQN}} - Q_m(s, a_m, \boldsymbol{\theta}))\nabla_{\boldsymbol{\theta}} Q_m(s, a_m, \boldsymbol{\theta}). \tag{3.28}$$

As depicted in Fig. 2.2, the target network is used to obtain $\mathrm{Y}_m^{\mathrm{DQN}}$ (which is stated in equation (3.27)). Also, as mentioned in reference [33, 58], the loss function is given in terms of the difference between $\mathrm{Y}^{\mathrm{DQN}_m}$ and the Q-value obtained from the online network. To optimize the loss function, the parameters for the online network are updated by using the updating function in equation (3.28). Additionally, the observed transitions $[s, a_m, u_m(s, a_m), s']$ which are stored in the experienced memory (replay buffer) $\mathcal{D}$, are sampled uniformly as mini-batches to train the online network. This would result in conquering the learning instability and preventing the optimal policy from being driven to a local minima [33].

In both Y_m^{Q} and Y_m^{DQN}, the max operator results in overoptimistic value estimation. To deal with this issue, we use double DQN (DDQN) algorithm in [58]. In the DDQN algorithm, the selection is decoupled from the evaluation [58]. By using the online network with parameter $\boldsymbol{\theta}$ and target network with parameter $\boldsymbol{\theta}^-$ the target for DDQN is given by,

$$Y_m^{\mathrm{DDQN}} = u_m(s, a_m) + \lambda Q_m(s', \arg\max_{a' \in \mathcal{A}_m} Q_m(s', a', \boldsymbol{\theta}), \boldsymbol{\theta}^-). \tag{3.29}$$

By replacing Y_m^{DDQN} in (3.25), we have,

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \zeta(Y_m^{\mathrm{DDQN}} - Q_m(s, a_m, \boldsymbol{\theta}))\nabla_{\boldsymbol{\theta}} Q_m(s, a_m, \boldsymbol{\theta}). \tag{3.30}$$

Now, based on the above discussion, we can use the DDQN algorithm to extend the **JROM** algorithm. We call the extended algorithm as **DJROM** algorithm and present it in **Algorithm 3.2**.

In the **DJROM** algorithm, in each step, for a given state s, each user $m \in \mathcal{M}$ estimates the action-value $Q_m(s, a_m, \boldsymbol{\theta}), \forall a_m \in \mathcal{A}_m$. The users use the estimated action values, and by employing the ϵ-greedy algorithm, the action a_m is selected by each user $m \in \mathcal{M}$. As mentioned before, the selected action a_m corresponds to the offloading mode chosen by

Algorithm 3.2 Proposed **DJROM** algorithm

Input: The action set for each user, i.e., $\mathcal{A}_m, \forall m \in \mathcal{M}$ consisting of actions (partial offloading, binary offloading with completely local computation, binary offloading with complete computation at MEC), and duration of offloading at MEC (i.e., τ);

Output: Optimal policy (i.e., the optimal sequence of actions) for satisfying the QoS requirement for all users (if the network is feasible) or a subset of users (if the network is infeasible);

Initialize: Initialize experience memory $\mathcal{D}$, neural network parameters $\boldsymbol{\theta}$, and the target network replacement frequency N; $C_U \leftarrow 0$; Initialize both online network and target network with weights $\boldsymbol{\theta}$;

procedure
  for each episode **do**
    Initialize the network state s;
    Initialize the unit price for consumed energy (i.e., $w_m, \forall m \in \mathcal{M}$) by using (A.4);
    for each step **do**
      Any given user $m \in \mathcal{M}$ approximates action-value $Q_m(s, a_m, \boldsymbol{\theta}), \forall a_m \in \mathcal{A}_m$ by using online network;
      At state s, each user $m \in \mathcal{M}$ chooses a_m by employing the ϵ-greedy policy in (3.19) where $Q_m(s, a) \leftarrow Q_m(s, a, \boldsymbol{\theta}), \forall a \in \mathcal{A}_m$;
      Based on the selected action a_m, the CPU frequency and uplink transmit power level for each user $m \in \mathcal{M}$ (i.e., f_m and p_m) are obtained by using (3.22) and (3.24), respectively;
      Given $\mathbf{p}, \mathbf{f}, \mathbf{a}$, each user $m \in \mathcal{M}$ checks if its QoS requirement is satisfied by using (3.8);
      All users obtain the next state s' through the message passing. Set $s \leftarrow s'$;
      Each user $m \in \mathcal{M}$ stores the transition $(s, a_m, u_m(s, a_m), s')$ in $\mathcal{D}$;
      Each user $m \in \mathcal{M}$ samples random mini-batch of transitions $(s, a_m, u_m(s, a_m), s')$ from $\mathcal{D}$;
      The parameters for online network, $\boldsymbol{\theta}$, is updated using (3.30);
      $C_U \leftarrow C_U + 1$;
      if $C_U == N$ **then**
        $\boldsymbol{\theta}^- = \boldsymbol{\theta}$; $C_U \leftarrow 0$;
      end if
      if the QoS for each user is satisfied, i.e., $s = \mathbf{1}$ **then**
        break;
      end if
    end for
  end for
end procedure

the user m to perform its computation either partially at the MEC server, or completely at the MEC server, or locally at the user side. Then, given a_m, user m obtains the CPU frequency and uplink transmit power level for offloading at the MEC server, f_m and p_m, respectively. When the users obtain the next state s' (by checking if their QoS requirements are satisfied or not), and their immediate reward (i.e., $u_m(s, a_m), \forall m \in \mathcal{M}$), the transition $[s, a_m, u_m(s, a_m), s']$ would be stored in the replay memory $\mathcal{D}$. What's more, the users randomly sample mini-batch from memory $\mathcal{D}$ and update the parameters for the online network, i.e., $\boldsymbol{\theta}$, by using the update function in (3.30). When equation (3.8) holds true for all users (i.e., for all $m \in \mathcal{M}$, we have $s_m(t) = 1$ and thus $s = \mathbf{1}$), the current episode terminates; otherwise, the task continues for T steps. When the algorithm stops, if the network is feasible, the optimal policy corresponding to the sequence of offloading modes selected by the users at each step is obtained. Additionally, the QoS requirements for the users are satisfied. Otherwise, if the network is infeasible, the obtained optimal policy would not result in satisfying the QoS requirements for all users.

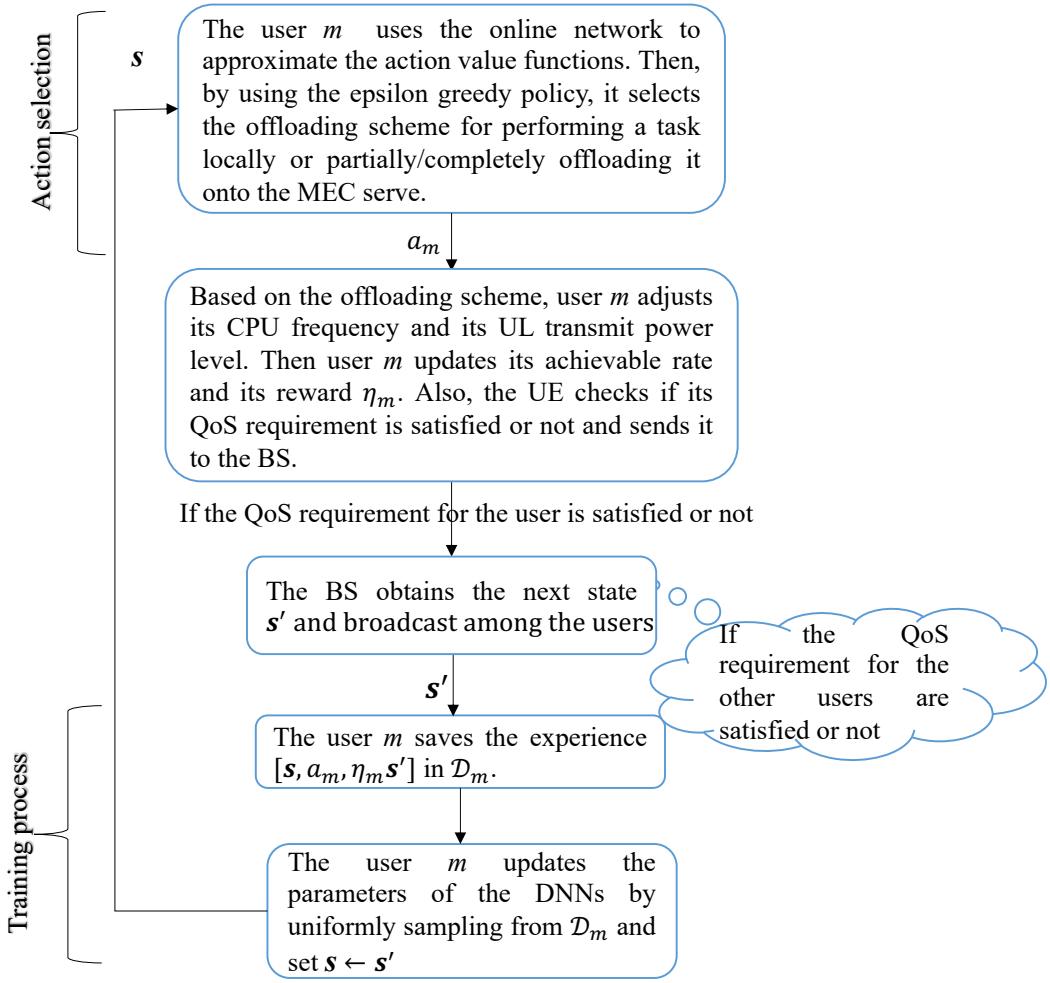

Figure 3.1: The message passing between the BS and user m for offloading a task and adjusting the CPU frequency and UL transmit power level for the user at a given time t .

3.2.4 Complexity Analysis

Based on **Algorithm 3.2**, the message passing between the BS and user m at a given time t is depicted in **Fig. 3.1**. Seen are two main parts: action selection and training the online DNN. The complexity analyses for each part are provided in what follows.

Action selection complexity: For a given user m, based on **Algorithm 3.2**, the

action selection corresponds to offloading scheme in order to perform a task locally or to partially/completely execute it at the MEC server. This would be the most complicated part of the algorithm.

Given the fully connected online DNN with a fixed number of hidden layers and neurons in each of them, for a given input, the computational complexity is pertinent to the sum of input and output sizes [101]. The input size for the online DNN corresponds to the state vector dimension, and it equals M. So, the complexity of estimating the Q-value is $O(M)$. Besides, each user m should select the Q-value corresponding to three offloading schemes; therefore, the computational complexity for action selection is $O(M + 3) = O(M)$.

Training process complexity: To obtain the complexity for training, the forward and backward propagation complexities should be considered. Let T_m be the training batch used to update the weights for the online DNN. For a given user m, based on equations (3.29) and (3.30), the Q-value function for updating the weights of the online DNN is needed. In the previous part, for a fixed number of hidden layers and neurons, the complexity of the Q-values was obtained. By considering the training batch, the forward propagation complexity would be $O(T_m \times M)$. Additionally, the complexity of the backward propagation algorithm corresponds to the product of the size of the input and output layers [101]. For a given user m, the size for the output of online DNN is 3. So, the backward propagation complexity is $O(T_m \times 3 \times M)$. As a result, the computational complexity for training procedure corresponds to $O(T_m \times 3 \times M)$.

3.2.5 Simulation Results

In the simulations, we consider a single BS, a single MEC server and M number of users, which varies between 2 and 30. The users are uniformly distributed through the coverage area, which is a 200×200 square unit. The simulation parameters are given in Table 3.1. The structure of the DNN is composed of an input layer (the number of users), one hidden layer with 256 neurons and an output layer with 3 neurons. Additionally, ReLU function is used as the activation function. The Adam optimization approach is used in the weight-updating process. It is worth mentioning that the hyper-parameters for our training method such as learning rate (i.e., λ), ϵ-greedy, the number of hidden layers and the number of neurons in the hidden layers, have been tuned through the simulations. Due to space limitations, those results are not presented here.

We first consider the scenario with the fixed location of the users. We compare the performance of DDQN learning method to DQN in (3.27) and DQL in (3.26). We call these approaches MARL-DQN and MARL-DQL, respectively. Then, we consider the scenario

Table 3.1: Network parameters and hyperparameters for the proposed resource allocation and offloading algorithms

Parameters	Value
The communication bandwidth (B) [4]	2 MHz
The noise power (σ_0^2)	10^{-9} Watt
The number of cycles for one bit (C) [4]	1000
The capacitance coefficient (γ_c) [4]	10^{-28}
The maximum power for each user m ($\overline{p}_m$) [4]	0.005 Watt
The maximum frequency of CPU for each user m ($\overline{f}_m$)	2 GHz
The received power (P_{rk}) [4]	0.0032 Watt
The constant circuit power (P_{ck}) [4]	0.0032 Watt
The amplifier coefficient (ϵ) [4]	3
The communication overhead for user m (ν_m) [4]	1.5
Number of episodes [33]	500
Number of steps [33]	500
Discount factor (λ) [33]	0.9
ϵ-greedy [33]	0.1
Learning rate (δ)	0.01
Replay memory ($\mathcal{D}$) size [33]	500
Optimizer [33]	Adam
Activation function [33]	ReLU

wherein the users are mobile and repeat the simulations[5]. In both scenarios, we compare the performance of the algorithms above with that of a brute-force-based algorithm wherein an exhaustive search obtains the offloading scheme of each user at each state through all possible offloading schemes. In the brute force-based algorithm, the computational complexity increases exponentially with an increase in the number of users. Therefore, the simulation results are reported for the maximum number of 10 users. Note that, we only evaluate the performance of **DJROM**, since **JROM** version suffers from state-action pair explosion for a large number of users. Additionally, in each scenario above, by considering 95% confidence level, the range of plausible values [Confidence Interval (CI)] for the parameters used to evaluate the algorithm performances are very small. All the

[5]In the first scenario where the users are fixed, the path gain for the users are given before and will not change for each episode. In the second scenario with moving users, the path gain for the users are given before; however, they will change for each episode.

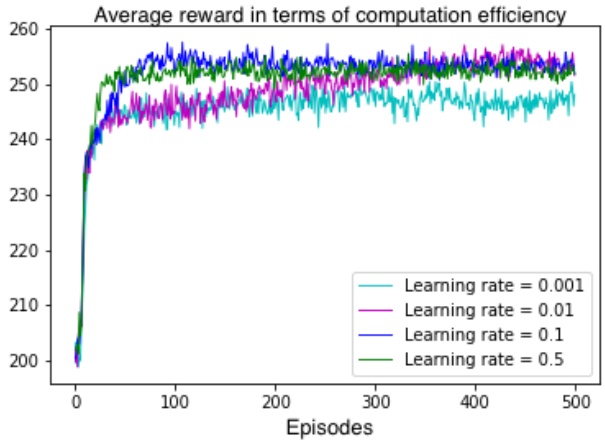

Figure 3.2: Average reward in terms of computation efficiency versus the number of episodes for different learning rates.

simulation results are obtained through 500 independent simulation runs. In each scenario, the number of users is increased as long as the user QoS requirements are met.

Training Evaluation with Different Learning Hyperparameters

In this subsection, **JROM** algorithm which is based on the DDQN algorithm, is evaluated by four different values for the learning rates. As illustrated in Fig. 3.2, for the last three learning rates, i.e., $\delta = 0.01$, $\delta = 0.1$ and $\delta = 0.5$, fewer training steps would be required to converge than those required for $\delta = 0.001$. We consider $\delta = 0.01$.

Performance Evaluation with Fixed Devices

In this subsection, we consider the users with fixed locations. We evaluate the performance of **DJROM** algorithm with those of MARL-DQN, MARL-DQL algorithms, and Pratially Offloading Resource Allocation (PORA) algorithm in **Appendix A.3**, respectively. Both binary and partial offloading are considered in the deep Q-learning-based algorithms. The objective is to simultaneously maximize the long-term reward for the users. However, in the PORA algorithm, only partial offloading is considered, and the objective is to simultaneously maximize the immediate reward.

For each user $m \in \mathcal{M}$, we have $0 \leq R_m^{\min} \leq \overline{R}$ where $R_m^{\min}$ is its maximum QoS requirement. Additionally, $\overline{R} = 0.3$ Mbps, is the maximum assigned QoS requirement value

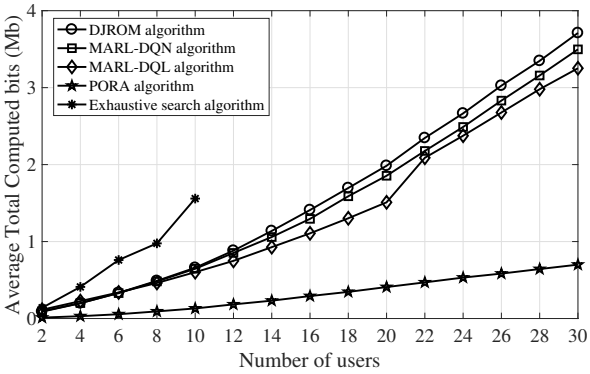

Figure 3.3: Average total computed bits versus the number of users for the scenario with fixed users where $2 \leq M \leq 30$.

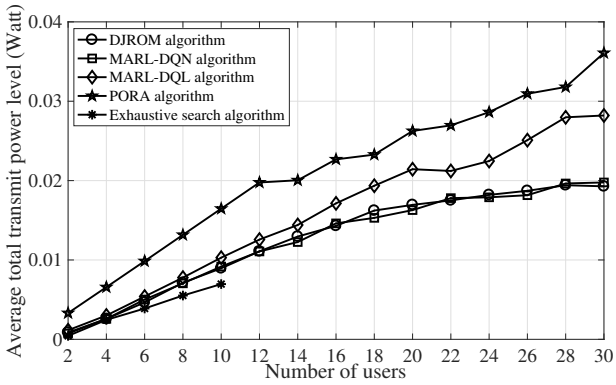

Figure 3.4: Average total transmit power level versus the number of users for the scenario with fixed users where $2 \leq M \leq 30$.

of each user. Figs. 3.3, 3.4, and 3.5 illustrate the average total computed bits, the average total uplink transmit power level for the users, and average total computation efficiency (i.e., the ratio of the computed bits to the consumed power) for the users, respectively.

As illustrated in Fig. 3.3, the total average number of computed bits for the users increases with the growth in the number of users. Moreover, as the number of users increases, **DJROM** algorithm shows better performance (in terms of the total average number of computed bits for the users) than the other MARL-based algorithms, MARL-DQN and MARL-DQL. This is because of mitigating over-optimistic estimation of Q-values in the DDQN method, which is used in **DJROM** algorithm.

Based on Fig. 3.4, as the number of users increases, the total average transmit power

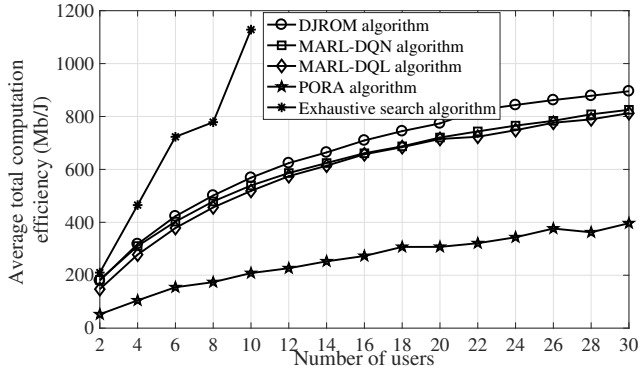

Figure 3.5: Average total computation efficiency versus the number of users for the scenario with fixed users where $2 \leq M \leq 30$.

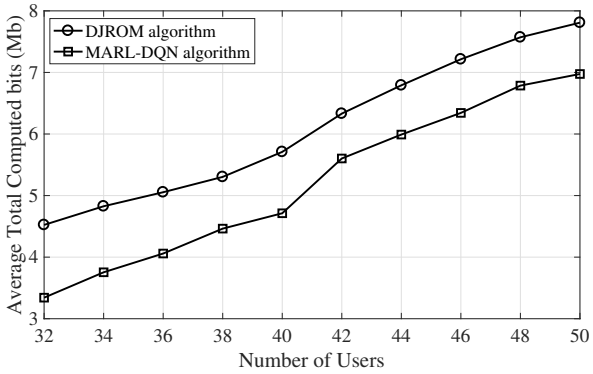

Figure 3.6: Average total computed bits versus the number of users for the scenario with fixed users where $32 \leq M \leq 50$.

level for the users grows. However, **DJROM** and MARL-DQN algorithms demonstrate better performance (in terms of the total transmit power level for the users) than the MARL-DQL algorithm.

In Fig. 3.5, it is demonstrated that the total average of computational efficiency for the users increases when the number of users grows. In **DJROM** algorithm, because of decoupling selection from evaluation [58], the performance (in terms of the total average of computational efficiency for the users) is better than those of MARL-DQN and MARL-DQL algorithms. By using **DJROM** algorithm, the performance improves by 11.27%.

In Figs. 3.3, 3.4, and 3.5, it is illustrated that the Q-learning based algorithms perform better than PORA in terms of computation efficiency, number of computed bits, and power

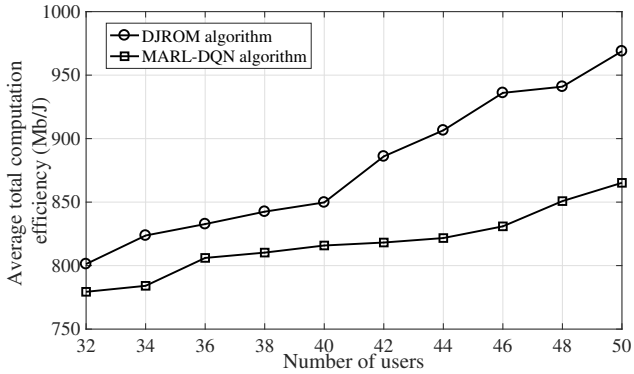

Figure 3.7: Average total computation efficiency versus the number of users for the scenario with fixed users where $32 \leq M \leq 50$.

consumption. It is likely because in PORA algorithm, only the partial offloading scheme is used which can lead to the consumption of more energy and computation resources. [102].

Moreover, we repeat the simulation for this scenario where the number of users varies between 32 and 50 and compare the performance of **DJROM** algorithm with that of the MARL-DQN algorithm in terms of the average total computation efficiency and average total computed bits. With an increase in the number of users, Figs. 3.6 and 3.7 illustrate that the performance of the **DJROM** algorithm is better than that of the MARL-DQN algorithm. For this scenario, by using **DJROM** algorithm, the performance in terms of the computation efficiency improves by 7.3%.

Performance Evaluation with Mobile Users

We consider mobile devices which move according to the random waypoint mobility model. The small-scale fading is modeled as the Rayleigh fading with unit scale. In this situation, the performance of **DJROM** algorithm and those of MARL-DQN and MARL-DQL algorithms are evaluated.

Figs. 3.8, 3.9, and 3.10 illustrate the average total computed bits, average total UL transmit power level for the users, and average total computation efficiency (i.e., the ratio of the computed bits to the consumed power for a user) for the users, respectively.

As illustrated in Fig. 3.8, the total average number of computed bits for the users increases with the growth in the number of users. As the number of users increases, **DJROM** algorithm displays slightly better performance (in terms of the total average number of computed bits for the users) than the other algorithms.

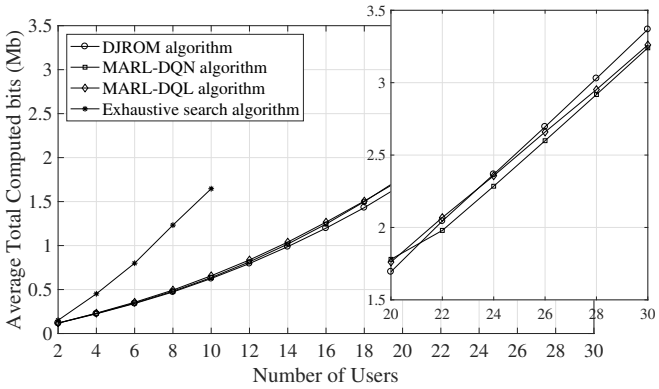

Figure 3.8: Average total computed bits versus the number of users for the scenario with mobile users.

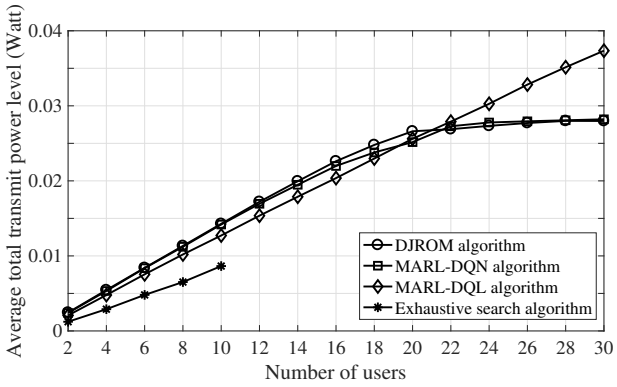

Figure 3.9: Average total transmit power level versus the number of users for the scenario with mobile users.

Based on Fig. 3.9, the total average transmit power level for the users grows as the number of users increases. However, for the situations with more than 20 users, **DJROM** algorithm and MARL-DQN algorithm show better performance (in terms of the total transmit power level for the users) than the MARL-DQL algorithm. This is expected owing to the use of the target network and the experience replay. In Fig. 3.10, it is displayed that the total average of computational efficiency for the users increases when the number of users grows. Specifically, **DJROM** algorithm performs better (in terms of the total average of computational efficiency for the users) in the networks with more than 20 users. This may be because of mitigating the over-optimistic Q-values estimation compared to the MARL-DQN and MARL-DQL algorithms. Specifically, by using **DJROM** algorithm,

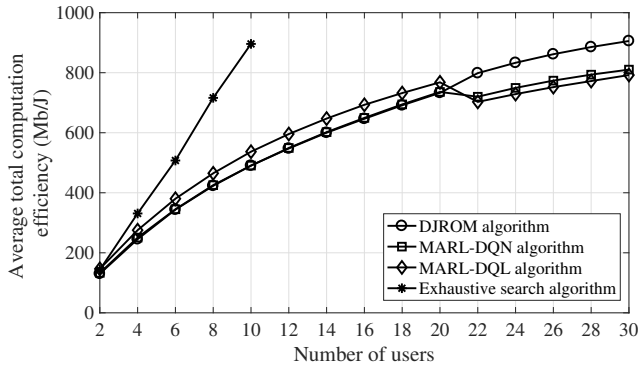

Figure 3.10: Average total computation efficiency versus the number of users for the scenario with mobile users.

the performance improves by 3.32% for 30 users.

3.3 Distributed Multi-Agent Learning for Service Function Chain Partial Offloading at the Edge

Using Network Function Virtualization (NFV), SFCs, a set of ordered VNFs can be deployed within the MEC infrastructure. The UEs can offload VNFs with intense computational load to the MEC servers with rich storage and computation resources. In this section, we address the partial offloading of a chain of services where each VNF of the SFC request can either be performed locally or offloaded onto a MEC server [2]. The objective is to concurrently minimize the long-term cost of the UEs, which is given in terms of both delay and energy consumption. This problem is highly complex and calls for distributed multi-agent learning techniques. We formulate the problem as a distributed multi-agent reinforcement learning problem and use the DDQN algorithm to solve it. Our simulation results show that the proposed DDQN-based solution has comparable results to an exhaustive search algorithm.

3.3.1 System model and Preliminaries

We consider a multi-server MEC system consisting of one AP, several MEC servers denoted by $\mathcal{M} = \{1, \ldots .M\}$ and a set of UEs denoted by $\mathcal{U} = \{1, \ldots, U\}$. The bandwidth is equally divided among the UEs. Each UE u has a SFC request denoted by V^u, which consists of several dependent VNFs. The AP is the gateway for the MEC servers in the edge cloud. Additionally, two given MEC servers in the edge cloud are assumed to be reachable to each other through the backhaul network. Let h_u, be the path-gain between UE u and AP in both UL and DL directions, p_u^U and p_u^D be the UL and DL transmit power levels for the UE, respectively. Denoting UL SINR by γ_u^U and DL SINR by γ_u^D, we have $\gamma_u^U = \dfrac{h_u p_u^U}{\sigma^2}$ and $\gamma_u^D = \dfrac{h_u p_u^D}{\sigma^2}$, where σ^2 is the noise power. Accordingly, the UL and DL rates for the UE u are given by $R_u^U = B \log_2(1 + \gamma_u^U)$ and $R_u^D = B \log_2(1 + \gamma_u^D)$, respectively, where B is the communication bandwidth.

In [2], the physical MEC infrastructure is modeled as an undirected graph, and the SFC request for each UE u is modeled as a directed acyclic graph. Let $\mathcal{G} = (\mathcal{M}, \mathcal{E}^p)$ denote the graph for the physical MEC network where $\mathcal{E}^p$ is the set of physical links in the network. Each MEC server has several Virtual Machine (VM)s, each of which supports only one VNF. We denote the remaining computing capacity and the set of remaining function type for MEC server $m \in \mathcal{M}$ by c_m and $\mathcal{F}_m = \{1, \ldots, F_m\}$, respectively. Also, the remaining

bandwidth capacity for the physical link $e^p \in \mathcal{E}^p$ is given by b_{e^p}. Moreover, we assume that the bandwidth between the VMs is sufficient to support the VNFs in an SFC [103].

For a given UE $u \in \mathcal{U}$, we model the SFC request V^u by the graph $\mathcal{G}^u = (\mathcal{V}^u, \mathcal{E}^u)$ where $\mathcal{V}^u$ and $\mathcal{E}^u$ are the set of VNFs and the set of virtual link in the SFC, respectively. Additionally, c_{v^u} is the computing demand for VNF $v^u \in \mathcal{V}^u$, F_{v^u} is the function type for VNF $v^u \in \mathcal{V}^u$, and b_{e^u} is the bandwidth demand for virtual link $e^u \in \mathcal{E}^u$. For convenience, two VNFs are considered source and destination virtual nodes for the SFC request V^u and are denoted by s^u and d^u, respectively [18]. When the VNF s^u is executed, the SFC request V^u is commenced, and it is terminated when the VNF d^u is executed.

We consider the partial offloading problem for an SFC request. In this problem, each VNF in an SFC request can be performed locally or offloaded onto a MEC server. Thus, the offloading indicator variable $a^{v^u} = \sum_{m \in \mathcal{M}} a_m^{v^u}$ is defined for a given VNF v^u, where $a_m^{v^u}$ is the indicator for offloading VNF v^u onto server m. Specifically, $a^{v^u} = \sum_{m \in \mathcal{M}} a_m^{v^u} = 0$, if VNF v^u is executed locally; otherwise, if VNF v^u is offloaded onto a MEC server m, we have $a^{v^u} = \sum_{m \in \mathcal{M}} a_m^{v^u} = 1$. It is worth mentioning that, for an SFC request V^u, the virtual VNFs s^u and d^u are executed locally, i.e., $a^{s^u} = a^{d^u} = 0$. Furthermore, we define the binary variable $\phi_{e^p}^{e^u}$ for mapping the virtual link e^u onto physical link e^p where $\phi_{e^p}^{e^u} = 1$ if virtual link e^u is mapped onto physical link e^p; otherwise, $\phi_{e^p}^{e^u} = 0$.

To obtain the consumed energy for executing an SFC request and its end-to-end delay, the energy consumption and the delay for each VNF in that request for both local and edge computing need to be determined. Hence, we first define the finish time and the ready time for a given VNF v^u of an SFC request V^u as in [18]. Then, we formulate the energy consumption and the delay of the VNF for both local and edge computing scenarios, and compute the finish time and the ready time for VFN v^u. Finally, the consumed energy and end-to-end delay for the SFC request V^u are obtained.

Definition 3.1. For VNF v^u, let $\mathrm{FT}_l^{v^u}$ and $\mathrm{FT}_c^{v^u}$ denote the finish time for local and edge computing, respectively. Both $\mathrm{FT}_l^{v^u}$ and $\mathrm{FT}_c^{v^u}$ correspond to the moment when the workload (computation demand) for VNF v^u is executed. Additionally, let $\mathrm{RT}_l^{v^u}$ and $\mathrm{RT}_c^{v^u}$ be the ready time for local and edge computing, respectively. Both $\mathrm{RT}_l^{v^u}$ and $\mathrm{RT}_c^{v^u}$ correspond to the earliest time when the necessary input data for VNF v^u, and the output data for the predecessor VFN of VNF v^u, is ready to initiate the VNF computation.

Local Computing: Let $f_l^u \leq \overline{f}_l^u$ be the CPU frequency for a given UE $u \in \mathcal{U}$ which is limited to the maximum frequency $\overline{f}_l^u$. Therefore, the local execution time and the local

energy consumption for executing VNF v^u of SFC request V^u is respectively denoted by $\tau_l^{v^u}$ and $e_l^{v^u}$ [18]:

$$\tau_l^{v^u} = \frac{c_{v^u}}{f_l^u}, \tag{3.31a}$$

$$e_l^{v^u} = \kappa c_{v^u} (f_l^u)^2 = \kappa \frac{(c_{v^u})^3}{(\tau_l^{v^u})^2}, \tag{3.31b}$$

where κ is the effective switched capacitance and depends on the chip architecture. In the sequel, we use the above equations to formulate the local computing finish time and ready time for VNF v^u.

In SFC V^u, let VNF $k \in \mathcal{V}^u$ be the predecessor for the VNF v^u. If VNF k is performed in a MEC server at edge-cloud, then its output data O_{k,v^u} should be transmitted in DL direction to UE u to execute VNF v^u locally. By using the DL rate for UE u, the transmission time to send the data O_{k,v^u} is given by $\tau_{k,v^u}^D = \dfrac{O_{k,v^u}}{R_u^D}$. Therefore, we can express the local computing ready time and finish time for VNF v^u, i.e., $\mathrm{RT}_l^{v^u}$ and $\mathrm{FT}_l^{v^u}$, respectively as follows [18]:

$$\mathrm{RT}_l^{v^u} = (1 - a^k) \times \mathrm{FT}_l^k + a^k \times (\mathrm{FT}_c^k + \tau_{k,v^u}^D), \tag{3.32a}$$

$$\mathrm{FT}_l^{v^u} = \mathrm{RT}_l^{v^u} + \tau_l^{v^u}. \tag{3.32b}$$

In the above equation, FT_c^k is the edge computing finish time for VNF k, computed in what follows.

Edge Computing: Let VNF v^u be performed at a MEC server m (i.e., $a_m^{v^u} = 1$ and thus $a^{v^u} = \sum_{m \in \mathcal{M}} a_m^{v^u} = 1$). Given f^m as the CPU frequency for MEC server m, the time for performing VNF v^u at MEC server m is denoted by $\tau_m^{v^u}$ and expressed as [18]:

$$\tau_m^{v^u} = \frac{c_{v^u}}{fm} \tag{3.33}$$

Now assume VNF k, the predecessor VNF of VNF v^u, is performed locally (i.e., $a_m^k = 0, \forall m \in \mathcal{M}$ and thus $a^k = \sum_{m \in \mathcal{M}} a_m^k = 0$). Thus, the time and consumed energy for transmitting the output data O_{k,v^u} to the MEC server m is given by $\tau_{k,v^u}^U = \dfrac{O_{k,v^u}}{R_u^U}$ and $e_{k,v^u}^U = p_u^U \tau_{k,v^u}^U$, respectively. Therefore, we can express the edge computing ready time and

finish time for VNF v^u, i.e., $\mathrm{RT}_c^{v^u}$ and $\mathrm{FT}_c^{v^u}$ as [18]:

$$\mathrm{RT}_c^{v^u} = (1 - a^k) \times (\mathrm{FT}_l^k + \tau_{k,v^u}^U) + a^k \times \mathrm{FT}_c^k, \tag{3.34a}$$

$$\mathrm{FT}_c^{v^u} = \mathrm{RT}_c^{v^u} + \sum_{m \in \mathcal{M}} a_m^{v^u} \times \tau_m^{v^u} \tag{3.34b}$$

3.3.2 Problem Formulation

In this part, we first express the constraints for the SFC placement, i.e., performing each VNF of the SFC locally or offloading onto a MEC server, and the cost function for executing the SFC. We then formulate the optimization problem to minimize the cost function.

Constraints for the SFC placement

Performing NFVs locally or on MEC servers is under the computing capacity, function types and link bandwidths constraints as follows [103]:

$$\sum_{v^u \in \mathcal{V}^u} a_m^{v^u} c_{v^u} \leq c_m \qquad \forall m \in \mathcal{M}, \tag{3.35a}$$

$$a_m^{v^u} F_{v^u} \in \mathcal{F}_m \qquad \forall m \in \mathcal{M}, \forall v^u \in \mathcal{V}^u, \tag{3.35b}$$

$$\sum_{e^u \in \mathcal{E}^u} \phi_{e^p}^{e^u} b_{e^u} \leq b_{e^p} \qquad \forall e^p \in \mathcal{E}^p. \tag{3.35c}$$

Additionally, the finish time of a given SFC u is limited by the maximum allowable delay for UE u (i.e., $\overline{T}_u$), and VNF v^u needs to be offloaded onto only one MEC server. Thus, we have;

$$\mathrm{FT}_l^{d^u} \leq \overline{T}_u \qquad \forall u \in \mathcal{U}, \tag{3.36a}$$

$$\sum_{m \in \mathcal{M}} a_m^{v^u} \leq 1 \qquad \forall m \in \mathcal{M} \tag{3.36b}$$

In (3.36a), $\mathrm{FT}_l^{d^u}$ is the finish time for the VNF d_u of SFC V^u which is in fact the finish time for that SFC request. As denoted before, VNF d_u is performed locally.

For the path constraint, let us consider two consecutive VNFs v^u and v'^u in SFC V^u. For the continuous path, these two VNFs must be placed either at the same MEC server or UE u. In addition, they can be placed at two different but reachable MEC servers or

one VNF at a MEC server and another VNF at UE u. Accordingly, the continuous path constraint can be expressed as follows:

$$\sum_{m\in\mathcal{M}'_u}\sum_{p\in\mathcal{M}'_u}\mathbb{E}_{mp}(a_m^{v^u}\times a_p^{v'^u}) = a^{v^u}\times a^{v'^u} + a^{v^u}(1-a^{v'^u}) + a^{v'^u}(1-a^{v^u}) + (1-a^{v^u})(1-a^{v'^u}), \forall v^u, v'^u \in \mathcal{V}^u, \forall u \in \mathcal{U}. \tag{3.37}$$

In the above equation, $\mathcal{M}'_u = \mathcal{M}\cup\{u\}$, i.e., each node in the set $\mathcal{M}'_u$ corresponds to either a MEC server or UE u. Additionally, $\mathbb{E}_{mp}=1$, if two nodes m and p are reachable or $m=p$; otherwise, $\mathbb{E}_{mp}=0$. We say that the path constraint for SFC request V^u is satisfied if (3.37) holds for every two consecutive VNFs in that request.

Cost function

We define the cost function for a given UE u as the weighted sum of total consumed energy for executing SFC V^u and the corresponding execution time. Specifically, let E_u denote the consumed energy to execute SFC V^u for user u. Obviously, we have $E_u = \sum_{\forall v^u\in\mathcal{V}^u}(1-a^{v^u})e_l^{v^u} + (1-a^k)a^{v^u}e^U_{k,v^u}$, where k is the predecessor VNF of the VNF v^u. Accordingly, the energy-time cost function for UE u is given as follows:

$$\chi_u = \alpha_u E_u + \beta_u F T^l_{d^u}, \tag{3.38}$$

where α_u and β_u are positive and $\alpha_u+\beta_u=1, \forall u\in\mathcal{U}$. In what follows, for UE u, we use the constraint for the placement of SFC V^u and the energy-time cost for the UE to formulate the immediate cost function for UE u.

For UE $u\in\mathcal{U}$ and time $t\in[0,T)$, let us denote the immediate cost function by $\theta_u(t)$ and define it as follows:

$$\theta_u(t) = \chi_u(t) + \omega_u(f_u^1(t)+f_u^2(t)), \tag{3.39}$$

where, $f_u^1(t) = \sum_{m\in\mathcal{M}}\sum_{v^u\in\mathcal{V}^u}\mathbb{F}_m^{v^u}a_m^{v^u}(t)c_{v^u} - c_m$, $f_u^2(t) = \sum_{e^p\in\mathcal{E}^p}\sum_{e^u\in\mathcal{E}^u}\phi_{e^p}^{e^u}(t)b_{e^u} - b_{e^p}$. Additionally, $\mathbb{F}_m^{v^u}=1$ if $F_{v^u}\in\mathcal{F}_m$; otherwise, $\mathbb{F}_m^{v^u}=0$. More specifically, $f_u^1(t)$ and $f_u^2(t)$ correspond to the penalty imposed on the cost function of UE u if the constraints (3.35a), (3.35b) and (3.35c) are not satisfied. Using the immediate cost function for UE u given in (3.39), the long-term cost for UE u is given by:

$$\Theta_u = \sum_{t=0}^{T-1}\lambda^t\theta_u(t), \tag{3.40}$$

where $\lambda\in[0,1)$.

3.3.3 The Proposed MARL-based Method

To concurrently minimize the long-term cost function for the user, we formulate the problem of SFC offloading as:

$$\min_{\phi_{e^p}^{e^u}, a_m^{v^u}} \Theta_u \quad , \forall u \in \mathcal{U} \tag{3.41}$$

In the above optimization problem, minimizing the long-term cost function for each UE corresponds to minimizing the execution energy consumption and delay for the users simultaneously. Note that (3.41) is an optimization problem with combinatorial characteristic. Additionally, solving this problem requires nearly complete information about the environment. To overcome this problem, we resort to MARL.

To describe the MARL-based system, $\mathcal{S}$ and $\mathcal{A}_u$ are used for the discrete environment state space and the discrete action space, respectively. Specifically, $\mathcal{S} = \{\mathbf{s}_u(t)\}_{\forall u \in \mathcal{U}}$ where $\mathbf{s}_u(t) = (s_u^{\text{E2E}}(t), s_u^{\text{PC}}(t))$ and $s_u^{\text{E2E}}(t)$ and $s_u^{\text{PC}}(t)$ indicate whether the delay constraint for the UE (i.e., (3.36a)) and the path constraint for the SFC request V^u (i.e. expression of (3.37) for every two consecutive VNFs in SFC request V^u), are satisfied, respectively. Therefore, $\mid \mathcal{S} \mid = 4^{|\mathcal{U}|}$. Furthermore, $\mathcal{A}_u = \{a_m^{v^u}, \phi_{e^p}^{e^u}\}, \forall v^u \in \mathcal{V}^u, e^u \in \mathcal{E}^u, m \in \mathcal{M}, e^p \in \mathcal{E}^p$ and thus $\mid \mathcal{A}_u \mid = (\mid \mathcal{M} \mid +1)^{|\mathcal{V}^u|} \times (\mid \mathcal{E}^p \mid +1)^{|\mathcal{E}^u|}$. Here, an action for user u corresponds to if a given VNF in SFC request V^u is performed locally or offloaded onto a MEC server. Additionally, it corresponds to mapping virtual links in that SFC onto either physical links or backhaul between AP and a MEC server. Also, $\eta_1, \ldots, \eta_U$ where $\eta_u = -\theta_u$, is used for the user reward functions.

Based on the above discussion, the problem is high-dimensional with a huge state and action space. In what follows, we first reduce the dimension of the action space and then use the DDQN algorithm in [58] to obtain the value functions.

To reduce the dimension of action space for a given UE/agent u, we restate it as $\mathcal{A}_u = \{a_m^{v^u}\}, \forall v^u \in \mathcal{V}^u, m \in \mathcal{M}$. Specifically, based on this action space for the agent, a given VNF of the agent's request is offloaded to a MEC server or performed locally. So we have $\mid \mathcal{A}_u \mid = (\mid \mathcal{M} \mid +1)^{|\mathcal{V}^u|}$. Additionally, let v^u and v'^u be two consecutive VNFs in SFC V^u. To map the virtual links, we follow;

$$\begin{cases} \text{Find the shortest path between} \\ \text{MEC } m \text{ and MEC } p, \quad \text{if } a_m^{v^u} = 1 \text{ and } a_p^{v'^u} = 1, \\ \text{Use the backhaul between} \\ \text{MEC } m \text{ and AP}, \qquad \text{if } a_m^{v^u} = 1 \text{ or } a_m^{v'^u} = 1. \end{cases} \tag{3.42}$$

Specifically, based on the above expression, we consider two scenarios. In the first scenario, two consecutive VNFs in a request are mapped onto two different MEC servers. In

this situation, based on the requested bandwidth for the virtual link and the remaining bandwidth of the physical links in the MEC network, the shortest path between the MEC servers is considered by using the Dijkstra algorithm. In the second scenario, only one of two consecutive VNFs in a request is offloaded onto a MEC server. In this situation, the virtual link between the two VNFs is mapped onto the backhaul link between the MEC and AP.

In our high-dimensional problem, the non-linear function approximation methods, e.g., neural networks, can be employed to obtain the value functions. Specifically, we can learn the parametrized action-value function for a given user. Let us denote $Q_u(\mathbf{s}, a_u, \boldsymbol{\theta})$ as the parametrized action-value function for a user $u \in \mathcal{U}$. In [2], the DDQN algorithm in [58] is used to approximate it. By using the DDQN algorithm, not only is the performance improved by conquering the learning instability, but the overoptimistic value estimation issue of other Q-learning-based algorithms (e.g., DQN algorithm in [63]) can also be addressed [58]. We use the DDQN method for the SFC Partial Offloading (SPO) problem and present our derived scheme in **Algorithm 3.3**.

3.3.4 Simulation Results

We consider a single AP and four MEC servers connected to the AP via physical links. Eight different network functions can be provided in the MEC network, and each MEC server chooses three different network functions from among those. The computing capacity for each MEC server is randomly chosen between 300 Mcycles and 500 Mcycles, and the bandwidth of physical links are between 300 MHz and 500 MHz. The number of UEs varies between 2 and 10. They are uniformly distributed through the coverage area which is a circular area with a radius 500m. Each UE has a SFC request consisting of $2-5$ different VNFs. The computing demand for each VNF varies between 10 Mcycles and 20 Mcycles, the bandwidth of virtual links is between 10 MHz and 20 MHz, and the output of each VNF is between 1000 Kbytes and 2000 Kbytes.

The rest of the simulation parameters are given in Table 3.2. The structure of the DNN is composed of an input layer (the number of users), one hidden layer with 256 neurons and an output layer with $(U+1)^{|\mathcal{V}^u|}$ neurons where $2 \leq \mid \mathcal{V}^u \mid \leq 5$.

We consider the scenario wherein the users are mobile and compare the performance of the **SPO** algorithm with that of the brute-force algorithm. The brute-force algorithm means either offloading or locally performing the VNFs in each SFC request at each state is obtained from exhaustive search through all possible schemes. Given a state, each user searches for all possible actions and opts the one minimizing the cost function. Accordingly,

Algorithm 3.3 Proposed **SPO** algorithm

Input: The action set for each user, i.e., $\mathcal{A}_u, \forall u \in \mathcal{U}$;

Output: Optimal policy (i.e., the optimal sequence of actions) for satisfying the end-to-end requirement and continuous path constraint for all users;

Initialize: Initialize experience memory $\mathcal{D}$, neural network parameters $\boldsymbol{\theta}$, and the target network replacement frequency N; $C_U \leftarrow 0$;
Initialize both online network and target network with weights $\boldsymbol{\theta}$;

procedure
  for each episode **do**
    Initialize the network state $\mathbf{s}$ and w_u, $\forall u \in \mathcal{U}$;
    for each step **do**
      Any given user $u \in \mathcal{U}$ approximates action-value $Q_u(\mathbf{s}, a_u, \boldsymbol{\theta}), \forall a_u \in \mathcal{A}_u$ by using online network;
      At state $\mathbf{s}$, each user $u \in \mathcal{U}$ chooses a_u by employing the ϵ-greedy policy;
      Based on a_u, each user u maps the virtual links in SFC V^u by using (3.42);
      Each user $u \in \mathcal{U}$ obtain $\mathbf{s}_u = (s_u^{\text{E2E}}, s_u^{\text{PC}})$ by using (3.36a) and (3.37);
      The next state $\mathbf{s}'$ is obtained by using $\mathbf{s}_u, \forall u \in \mathcal{U}$ through the message passing. Set $\mathbf{s} \leftarrow \mathbf{s}'$;
      Each user $u \in \mathcal{U}$ stores the transition $[\mathbf{s}, a_u, u_u(\mathbf{s}, a_u), \mathbf{s}']$ in memory $\mathcal{D}$;
      Each user $u \in \mathcal{U}$ samples random mini-batch of transitions $[\mathbf{s}, a_u, u_u(\mathbf{s}, a_u), \mathbf{s}']$ from $\mathcal{D}$;
      The parameters for online network, $\boldsymbol{\theta}$, is updated using (3.30);
      $C_U \leftarrow C_U + 1$;
      if $C_U == N$ **then**
        $\boldsymbol{\theta}^- = \boldsymbol{\theta}$; $C_U \leftarrow 0$;
      end if
    end for
  end for
end procedure

Table 3.2: Network parameters and hyperparameters for the SFC placement and offloading algorithm

Parameters	Value
Communication bandwidth (B)	2 MHz
Noise power (σ_0^2)	10^{-9} Watt
Capacitance coefficient (κ) [4]	10^{-28}
UL power for each UE u (p_u^U) [18]	0.01 Watt
DL power for each UE u ($p_u^D = p^{AP}$) [18]	1 Watt
Max. frequency of CPU for each UE u ($\overline{f}_l^u$) [18]	1.5 GHz
Max. end-to-end delay for each UE u ($\overline{T}_u$) [18]	0.9 Sec.
Max. frequency of CPU for each MEC server m ($\overline{f}^m$) [18]	50 GHz
Number of episodes [33]	500
Number of steps [33]	500
Discount factor (λ) [33]	0.9
ϵ-greedy [33]	0.1
Replay memory ($\mathcal{D}$) size [33]	500
Optimizer [33]	Adam
Activation function [33]	ReLU

the computational complexity increases exponentially with an increase in the number of users in the brute-force search algorithm. Therefore, the simulation results are reported for the maximum number of 5 users.

Considering three UEs, each of which has a SFC request with three VNFs, firstly, we study the convergence analysis for the **SPO** algorithm. The end-to-end delay and average consumed energy for each user in different episodes are illustrated in Figs. 3.11 and 3.12, respectively. Both objectives converge close to 200 episodes. Then we analyze the performance of the proposed **SPO** algorithm. Figs. 3.13 and 3.14 illustrate the performance evaluation of the **SPO** algorithm and the brute-force search algorithm in terms of the UE average end-to-end delay and the UE average energy consumption, respectively. As observed in Figs. 3.13 and 3.14, by increasing the number of UEs, UE's average end-to-end delay (for its SFC request) and the corresponding execution energy increase. On this account, for more number of UEs, more resources of the MEC networks are consumed. Furthermore, less bandwidth is allocated to each UE when the number of UEs stretches. Therefore, the UL data rate for the UEs decreases. Thus, we witness the increase of the time and consumed energy for UL transmission. The brute-force algorithm is only run up to 5 users because of the long computational time that makes it infeasible for more num-

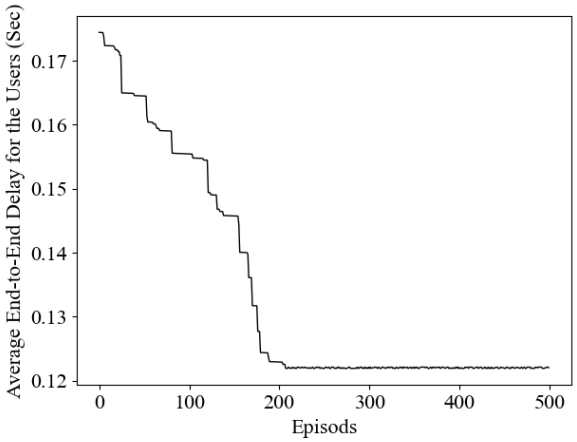

Figure 3.11: Convergence analysis of average end-to-end delay.

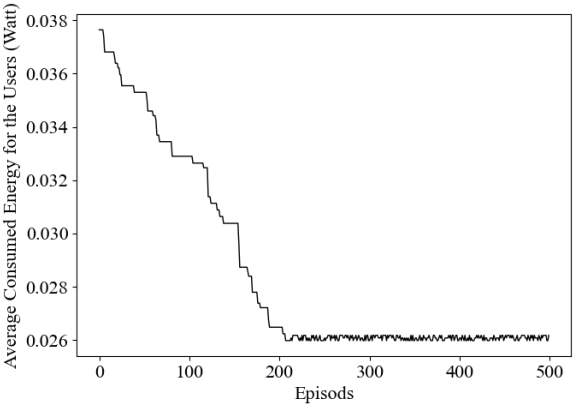

Figure 3.12: Convergence analysis of average consumed energy.

ber of users. Up until 5 users, both algorithms perform closely. Therefore, SPO displays comparable performance and is computationally more efficient.

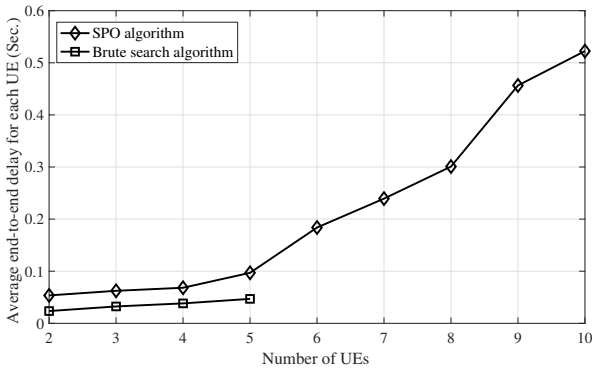

Figure 3.13: Average end-to-end delay for each user versus the number of users.

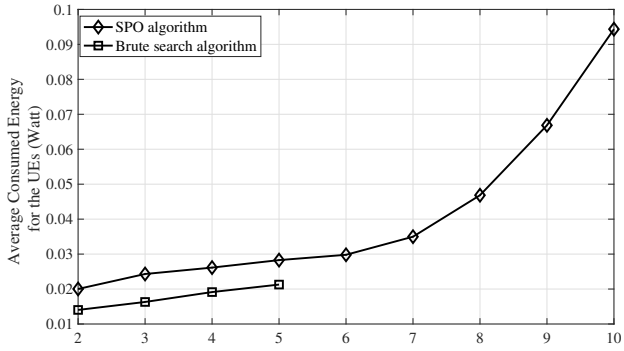

Figure 3.14: Average consumed energy for each user versus the number of users.

3.4 Energy and Delay Aware General Task Dependent Offloading in UAV-Aided Smart Farms

Motivated by edge computing as a technical solution for network reliability, deploying MEC technology and UAVs in smart agriculture has been considered a viable technology. In smart agriculture, resource-limited IoT devices need to locally monitor the environment, collect information and run energy-consuming delay-sensitive applications with complicated topologies to provide various services. To handle this issue, the power and computational resources of UAVs and those at the edge of networks in the MEC can be used to offload and run these resource-hungry tasks in UAV-MEC-assisted networks. In this chapter, the joint power allocation and the offloading problem is studied in the UAV-MEC-assisted wireless networks. Focusing on the topology for the interdependent tasks

offloaded to the UAVs, we formally express the problem of simultaneously minimizing the ETC for the UAVs as a mixed integer multi-objective optimization problem. On the one hand, we consider the topology for IoT applications, while on the other, we have two sets of discrete and continuous variables. Therefore, the problem mentioned has the combinatorial characteristic and is hard to solve in feasible time.

To tackle it, we contemplate two different perspectives in [3]: (i) sequential decision-making for partially offloading an IoT application and, at the same time, adjusting the UAVs' transmit power, and (ii) separating offloading decisions from UAVs' power allocation decisions. In both perspectives, a DAG is used to model the interdependent tasks, and the GCNs are used to characterize the DAG. Subsequently,, for the perspective, we propose an actor-critic method with embedding layers corresponding to the layers of GCNs, and we employ the CA2C algorithm proposed in [60] to train the model. For the second perspective, to separate the offloading problem from the power control, the DDQN [58] algorithm is used to train the GCN. Given the offloading plans for the tasks, we model the power allocation problem as a non-cooperative game among the UAVs.

In what follows, the system model and formal statement of the problem are given in Sub-section 4.3.1. In Sub-section 3.4.2, the joint offloading and power control problem is formally expressed as a multi-agent DRL model, and the CA2C algorithm in [60] is employed to solve the system. To perform the offloading and power control dis-jointly, the baseline algorithms, heuristic and DDQN-based algorithm for offloading, are derived, and a game-theory-based algorithm for power control is given in Sub-section 3.4.3. The simulation results are provided in Sub-sections 3.4.4.

3.4.1 System Model and Problem Formulation

This sub-section describes the details of the network, application, computation and communication models. Then, the energy and delay-aware problem formulation are given. Let us consider a real-time IoT network consisting of IoT devices, a set of UAVs, and MEC hosts. In this network, each IoT device is associated with only one UAV. The IoT devices gather information and forward the corresponding packets to the UAVs capable of performing the computing tasks. However, each UAV has limited computing capability because of its limited onboard battery capacity. Therefore, when executing an intensive computation application consisting of several dependent tasks (such as the one depicted in Fig. 3.15), a UAV would perform each computing task locally or offload it onto a MEC host or another UAV, each of which can have different computing capabilities. A typical example of a task-dependent application is video/image processing, where the outputs of some tasks are input for others.

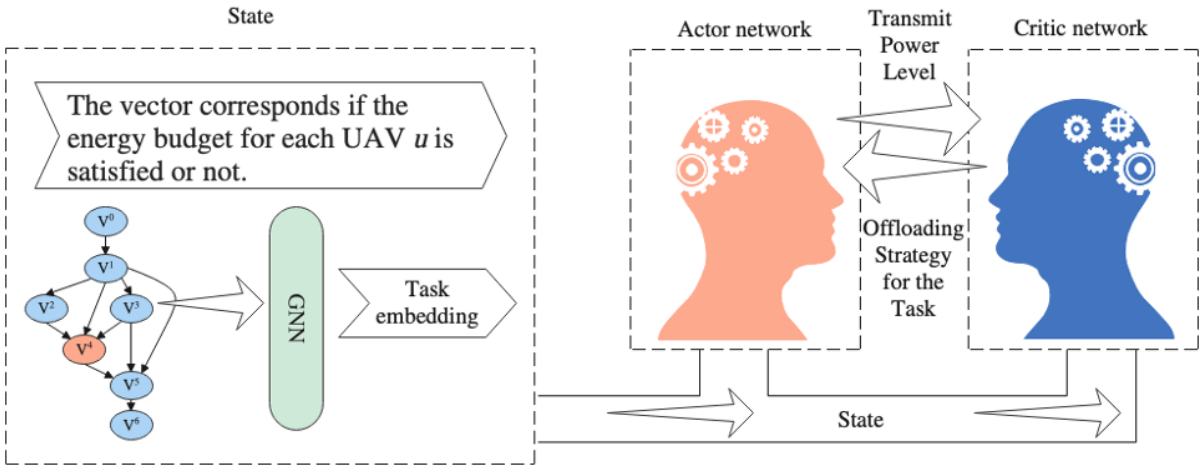

Figure 3.15: Actor-Critic framework employed by each UAV for joint power control and offloading. The GCN captures each task's features and the DAG structure.

Let us denote the set of IoT devices, UAVs, and MEC hosts by $\mathcal{N} = \{1, \ldots, N\}$, $\mathcal{U} = \{1, \ldots, U\}$ and $\mathcal{S} = \{1, \ldots, S\}$, respectively. For a given UAV $u \in \mathcal{U}$, the set of IoT devices in its coverage area is denoted by $\mathcal{N}_u$. Additionally, , we represent the set of processors except UAV u in the network (i.e., other UAVs and the MEC hosts in the system) by $\mathcal{P}_u$ and define it as $\mathcal{P}_u = \mathcal{U} \cup \mathcal{S} - \{u\}$. To efficiently use the frequency band, the UAVs employ the frequency reuse technology [104], and there is competition between the UAVs for the communication and computing resources at the network's edge.

For a given UAV u, we define the set of tasks for the application generated by that UAV as $\mathcal{I}_u = \{0, \ldots, I_u + 1\}$ where I_u is the number of tasks in that application. Let DAG $\mathcal{G}_u = (\mathcal{V}_u, \mathcal{E}_u)$ be the graph for the application in UAV u. In $\mathcal{G}_u$, the set $\mathcal{V}_u = \{v_u^i \mid i = 1, \ldots I_u\}$ is the set of computing tasks. (L_u^i, C_u^i) characterize a given task v_u^i where L_u^i and C_u^i are the workload (in terms of required CPU cycles) and processing data size (in bit) for the task, respectively, [69, 78]. In the graph, the edge $(i, j) \in \mathcal{E}_u$ represents a dependency between task i and task j. That is, task i must be completed before task j. Two virtual entry and exit tasks with indices 0 and $I_u + 1$ respectively are considered, which are performed locally, i.e., $C_u^{I_u+1} = C_u^0 = 0$ [69, 78].

As aforementioned, the UAV u can either perform the task $v_u^i \in \mathcal{V}_u$ locally or offload it onto a processor $m \in \mathcal{P}_u$, i.e., a UAV $u' \in \mathcal{U} \setminus u$, or a MEC server $s \in \mathcal{S}$. For a given task $v_u^i \in \mathcal{V}_u$, we define a_u^i as the indicator for performing the task locally (by the UAV u) or offloading it onto a processor $m \in \mathcal{P}_u$. Let $a_{u,m}^i$ indicate the binary variable corresponding to which processor the task v_u^i is offloaded. Specifically, $a_{u,m}^i = 1$ if UAV

u offloads v_u^i onto processor m; otherwise, $a_{u,m}^i = 0$. So, we have $a_u^i = \sum_{\forall m \in \mathcal{P}_u} a_{u,m}^i$. It is noteworthy that the virtual entry and exit tasks in graph $\mathcal{G}_u$ are performed locally, i.e., we have $a_u^{I_u+1} = a_u^0 = 0$. The finish time and the ready time for a given task [78] are defined in what follows. These definitions are used to describe both communication and computation models. We conclude the section with a formal statement of the problem.

Definition 3.2. *Let $RT_{u,i}^l$ and $RT_{u,i}^c$ be the ready time for the task v_u^i when it is performed locally (by UAV u) and when it is executed at a processor $m \in \mathcal{P}_u$, respectively. $RT_{u,i}^l$ and $RT_{u,i}^c$ are the earliest time when all input data for the task v_u^i has been received, and the task is ready to be executed.*

Definition 3.3. *The finish time for the task v_u^i when performed locally, i.e., $FT_{u,i}^l$, or executed at a processor, i.e., $FT_{u,i}^c$, is the time when the workload for that task, L_u^i, is performed thoroughly.*

Communication Model

The IoT-UAV-assisted networks can be used for intelligent agriculture applications such as the ones in [105–107]. Specifically, the IoT devices would gather raw data from their surrounding environment and send them to the UAVs to be processed. Furthermore, due to the limited UAV energy resources, a given UAV u would offload a part of the computing tasks in the IoT application onto a processor $m \in \mathcal{P}_u$. Therefore, there are two communication links, one communication link between the UAV u and another processor $m \in \mathcal{P}_u$, and a communication link between IoT device $n \in \mathcal{N}_u$ and the UAV u, each of which is explained in what follows.

Processor and UAV Communication: At a given time t, let UAV $u \in \mathcal{U}$ offload the task v_u^i onto processor $m \in \mathcal{P}_u$. We denote the set of UAVs offloading their tasks onto processor m by $\mathcal{U}_m$, i.e., $\mathcal{U}_m = \{m' \in \mathcal{U} \text{ so that } \alpha_{m',m}^{i'_{m'}} = 1\}$ where $i'_{m'} \in \mathcal{I}_{m'}, m' \neq m$. Given p_u as the transmit power level for UAV u to processor m, based on Shannon theorem, we have,

$$R_u^m = W \log \left(1 + \frac{p_u h_u^m}{\sum\limits_{m' \in \mathcal{U}_m, m' \neq u} p_{m'} h_{m'}^m + \sigma_0^2} \right), \tag{3.43}$$

where R_u^m is the transmission rate from UAV u to processor m. Let $v_u^k \in \mathcal{V}_u$ be the preceding task v_u^i. So, the transmission time to offload the task i onto process $m \in \mathcal{P}_u$ is

denoted by $T_{u,m}^{k,i}$ and defined by [78],

$$T_{u,m}^{k,i} = \frac{C_u^k}{R_u^m}. \tag{3.44}$$

IoT Device and UAV Communication: Let p_u^n be the transmit power level from IoT device $n \in \mathcal{N}_u$ to UAV u. Denoted by R_u^n, based on Shannon's theorem, the rate for transmitting a packet from the IoT device to the UAV is given by $R_u^n = W \log\left(1 + \frac{p_u^n h_u^n}{\sum_{n' \in \mathcal{N}_u, n' \neq n} p_u^{n'} h_u^{n'} + \sigma_0^2}\right)$, where h_u^n is the path gain between IoT device n and the UAV u, and σ_0^2 is the background noise. Accordingly, the transmitting time is denoted by T_u^n and stated as,

$$T_u^n = \frac{\Lambda_u^n}{R_u^n} + \frac{D_u^n}{C}, \tag{3.45}$$

where Λ_u^n is the size for the device packet, and D_u^n is the distance between the IoT device n and the UAV u.

Computation Model

For a given UAV u, based on the requested service for the packet sent from IoT devices to that UAV, a task in the IoT application can be performed locally or offloaded onto a processor $m \in \mathcal{P}_u$. In the sequel, we characterize the local computing and edge computing models by using the discussion above and the definitions for ready time and finish time.

Local Computing Model: Let us denote the computing capability for the UAV u in terms of CPU cycles by $f_u \leq f_u^{\text{Max}}$. We denote the time and energy consumption of UAV u for locally performing the task v_u^i by $\tau_{u,i}^l$ and $e_{u,i}^l$ and define them as $\tau_{u,i}^l = \frac{L_u^i}{f_u}$ and $e_{u,i}^l = \kappa_u L_u^i (f_u)^2$, respectively, where κ_u is effective switched capacitance, and is based on the chip architecture [78]. Let $v_u^k \in \mathcal{V}_u$ be the preceding task v_u^i. Besides, let v_u^k be executed at a processor $m \neq u$. By using equation (3.44) and the formal expression for local ready time and local finish time given in [69, 78], the ready time and finish time for locally performing v_u^i are given by (3.46) and (3.47), respectively, as follows,

$$RT_{u,i}^l = \max_{k \in \textbf{Preced(i)}} \max_{m \in \mathcal{P}_u} \{(1 - a_u^k) FT_{u,k}^l + a_{u,m}^k (FT_{u,k}^c + T_{u,m}^{k,i})\} \tag{3.46}$$

$$FT_{u,i}^l = RT_{u,i}^l + \tau_{u,i}^l \tag{3.47}$$

Edge Computing Model: Let $e^{k,i}_{u,m}$ denote the energy consumption for UAV u to offload the task v^i_u to processor $m \in \mathcal{P}_u$. By using (3.44), it is given by $e^{k,i}_{u,m} = p_u T^{k,i}_{u,m}$ [78]. Besides, given f_m as the local frequency of the CPU in destination processor m, the processing time for the task v^i_u is expressed by $\tau^c_{u,i}$ and stated as $\tau^c_{u,i} = \dfrac{L^i_u}{\sum\limits_{m \in \mathcal{P}_u} a^i_{u,m} f_m}$. By using equations from (3.44) to (3.47) and the formal expression of ready time and finish time for edge computing given in [69, 78], the ready time and finish time for executing the task v^i_u at processor $m \in \mathcal{P}_u$ are given by (3.48) and (3.49), respectively,

$$RT^c_{u,i} = \max_{k \in \textbf{Preced(i)}} \max_{m \in \mathcal{P}_u} \{(1 - a^k_u)(FT^l_{u,k} + T^{k,i}_{u,m}) + a^k_{u,m} FT^c_{m,k}\} \tag{3.48}$$

$$FT^c_{u,i} = RT^c_{u,i} + \tau^c_{u,i} \tag{3.49}$$

Problem Formulation

Based on the derived formal expression of finish time for performing task v^i_u by UAV u, i.e., equations (3.49) and (3.47), the transmit power level for UAV u to offload the task, the task's workload and the data forwarded from the UAV u to a processor $m \in \mathcal{P}_u$ may cause a severe delay in executing the IoT application (generated by UAV u). On the other hand, UAV u has limited computation and energy resources for complex IoT applications. These motivate us to make a trade-off between the delay and the UAV's energy consumption for performing the IoT application. Whereas in [3], we aim to simultaneously minimize the weighted sum of the long-term delay and energy consumption of the UAVs. The details are given in the following..

At a given time t, let E^t_u denote the cumulative energy consumption for the UAV u. Generally, for UAV u, the energy consumption is given in terms of propulsion energy, the energy for communicating, and the energy for local computing [104]. Propulsion energy depends on aircraft parameters and the trajectories, which can also be considered fixed [104]. Furthermore, the delay in executing the IoT application generated by UAV u corresponds to the finish time for the virtual exit task, i.e., $FT^l_{u,I+1}$ (as aforementioned, the virtual exit task v^{I+1}_u is performed locally).Therefore, to execute the IoT application, based on offloading strategy for each task $v^i_u \in \mathcal{V}_u$ which is given by $a^i_u = \sum\limits_{m \in \mathcal{P}_u} a^i_{u,m}$, the cumulative energy consumption for the UAV u, i.e., E^t_u, can be stated as follows [78] [104],

$$E^t_u = E^{t-1}_u + \sum_{i \in \mathcal{I}_u} \Big((1 - a^i_u) e^l_{u,i} + \sum_{k \in \textbf{Preced(i)}} a^i_u (1 - a^k_u) e^{u,m}_{k,i} \Big), \tag{3.50}$$

where $e^{u,m}_{k,i} = p_u T^{k,i}_{u,m}$ is the energy consumption to offload the task i onto processor $m \in \mathcal{P}_u$. Using (3.50) and (3.47), given time t, we denote the immediate ETC for the UAV u by $\eta_u(t)$ and define it as follows,

$$\eta_u(t) = w^e_u(E^t_u - E^{t-1}_u) + w^t_u F T^l_{u,I+1} = \\ w^e_u \sum_{i \in \mathcal{I}_u} \Big((1 - a^i_u) e^l_{u,i} + \sum_{k \in \textbf{Preced(i)}} a^i_u (1 - a^k_u) e^{u,m}_{k,i} \Big) + w^t_u \Big(R T^l_{u,I_u+1} + \tau^l_{u,I_u+1} \Big) \quad (3.51)$$

Defined as the weighted sum of the immediate cost, the long-term cost is formally expressed as,

$$\phi_u = \sum_{t=0}^{T-1} \lambda^t \eta_u(t), \quad (3.52)$$

where λ is the discount factor and $\lambda \in [0, 1)$. Let us define $\mathbf{a}_u = [a^i_{u,m}]_{i \in \mathcal{V}_u, m \in \mathcal{P}_u}$ and $\mathbf{P}_u = [p^i_u]_{i \in \mathcal{V}_u}$. Driven by the discussion above, we state the resource allocation problem as follows,

$$\begin{array}{lll} \textbf{min.} & \text{Long-term ETC for UAV } u & \forall u \in \mathcal{U} \\ \textbf{s.t.} & \text{Energy budget for UAV } u & \forall u \in \mathcal{U} \\ \textbf{var.} & \mathbf{a}_u, \mathbf{P}_u & \forall u \in \mathcal{U} \end{array} \quad (3.53)$$

In optimization problem (3.53), the long-term ETC for each UAV u is given in (3.52), and in turn, the constraint corresponding to energy budget for UAV u is given by $\lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} E^t_u \leq E^{\text{Max}}_u$ where E^{Max}_u is the maximum energy budget for the UAV u. Additionally, for the optimizing variables $\mathbf{a}_u, \mathbf{P}_u$, if $a^i_{u,m} = 0$ then $p_u = 0$. We call (3.53) the joint power and offloading control (**JPOC**) problem. This is a mixed-integer multi-objective optimization problem with combinatorial characteristics and coupling optimizing variables making it an NP-hard problem.

In [3], we have two perspectives to address the optimization problem (3.53). For the first perspective, we employ some concepts in multi-agent DRL and address the joint problem. The details are given in Sub-section 3.4.2. In the second perspective, we disjointly perform the task offloading and power control. Specifically, we derive a heuristic and a DDQN-based offloading algorithm. Given the offloading scheme for each task in the application, we employ game theory concepts to propose the power control problem. The corresponding details can be seen in Sub-section 3.4.3.

3.4.2 The CA2C-based Algorithm for Joint Offloading and Power Allocation

To resolve the multi-objective optimization problem (3.53), some methods in optimization theory or brute force search-based methods can be utilized. Using these methods in an environment with a time-varying channel would be time-consuming because the problem needs to be addressed frequently. The situation worsens when the dimensions of the problem grow, e.g., when either the number of nodes in the IoT applications or the number of UAVs increases. Recently, machine learning-based methods with more emphasis on RL-based approaches have been used as promising methodologies to study the complex resource management problems in the environment with dynamic characteristics, namely 5G and 6G wireless networks [54, 55]. To address (3.53), the leading roles of the IoT application topology and the discrete and continuous optimizing variables in (3.53) should be assessed. In [69], GCN is employed to derive the features of each node in the IoT application regarding the structure of the application. Additionally, an actor-critic method, the CA2C algorithm, is proposed in [60] to work out the DRL system with the compound-action space. In this sub-section, to approximately solve the optimization problem (3.53), we integrate the methods above and use the CA2C algorithm to train the model.

Each UAV tries to minimize its cost in optimization problem (3.53). Considering these AUVs as rational agents, the problem can be restated as a stochastic game [59]. In a dynamic environment, to find the NE for the game at each state, we can model the problem as a multi-agent and model-free RL system (similar to references [1, 33]). The details for the reward function, action, and state space are as follows.

Reward function: In the RL system, $\mathcal{U}$ is the set of agents (UAVs), and η_u is the immediate cost function for agent u which is given in (3.51).

Action space: We denote the action space for an agent u by $\mathcal{A}_u$, and we have,

$$\mathcal{A}_u = \{\boldsymbol{\alpha}_u \mid \boldsymbol{\alpha}_u = (\mathbf{a}_u, \mathbf{P}_u) \;\; \text{where } a^i_{u,m} \in \{0,1\} \text{ and } 0 \leq p_u \leq p_u^{\text{Max}}\}. \tag{3.54}$$

In (3.54), $a^i_{u,m} = 1$ if the task v^i_u is offloaded onto processor $m \in \mathcal{P}_u$; otherwise, if the task v^i_u is locally performed, we have $a^i_{u,m} = 0$, $\forall m \in \mathcal{P}_u$. Additionally, if $a^i_{u,m} = 1$, then p_u is the transmit power level from UAV u to processor $m \in \mathcal{P}_u$; otherwise, if the task v^i_u is performed locally, $p_u = 0$. Based on (3.54), the action space is composed of both continuous and discrete actions, offloading strategy for task v^i_u, and transmit power level for the UAV u if v^i_u is offloaded onto a processor $m \in \mathcal{P}_u$. Therefore, it is an RL system with compound action space.

State space: Let $\mathcal{S}_u$ be the state space for the agent (UAV) u. The state space $\mathcal{S}_u$ is given in terms of the energy budget for the UAVs, and the tasks' embedding. Let us define $\mathbf{e}^t = [e_u^t]_{\forall u \in \mathcal{U}}$ as the vector of the AUV energy budgets. We have, $e_u^t = 1$ if $E_u^t \leq E_u^{\text{Max}}$; otherwise, $e_u^t = 0$. Based on equation (3.50), the energy consumption for UAV u reflects the channel situation for the UAV to offload a task onto a processor $m \in \mathcal{P}_u$. Additionally, for a given task $v_u^i \in \mathcal{V}_u$, the embedding task vector is denoted by $\mathbf{e}_{v_u^i}$ and obtained from the output layer of a GCN. Worth noting that the state space for the UAV u is given by,

$$\mathcal{S}_u = \{\mathbf{S}_u^t \mid \mathbf{S}_u^t = (\mathbf{e}^t, \mathbf{e}_{v_u^i}), \forall v_u^i \in \mathcal{V}_u\}. \tag{3.55}$$

In what follows, we describe how the task embedding vectors are obtained. Specifically, to perform task $v_u^i \in \mathcal{V}_u$ locally or offload it onto a processor $m \in \mathcal{P}_u$, the features of that task which are given in terms of its workload and processing data, L_u^i and C_u^i, respectively, should be considered alongside those of its neighbors. That is, the structure of DAG for an IoT application generated by UAV u, $\mathcal{G}_u$, should be captured. Based on Fig. 3.15, this can be learned through the graph neural network. Our learning method is based on what is used in [68, 69].

Let us denote the matrix characterizing the features for all nodes in the IoT application DAG by $X_u = [L_u^i, C_u^i]_{i \in \mathcal{I}_u}$. The multi-layer GCN corresponding to $\mathcal{G}_u$ can be modeled by $f_u(X_u, A_u)$ where A_u is the adjacent matrix for $\mathcal{G}_u$. The layer propagation rule is given by [108],

$$H_u^{l+1} = \sigma\Big(\tilde{D}_u^{-\frac{1}{2}} \tilde{A}_u \tilde{D}_u^{-\frac{1}{2}} H_u^l W_u^l\Big) \tag{3.56}$$

where $\tilde{A}_u = A_u + I$, $\tilde{D}_u(i,i) = \sum_j \tilde{A}_u(i,j)$, W_u^l is trainable weight matrix for layer l, $\sigma(.)$ is the activation function, and $H_u^0 = X_u$. We denote the output for the GCN model by $\mathbf{O}_u = [\mathbf{o}_u^i]_{\forall v_n^i \in \mathcal{V}_u} = f_u(X_u, A_u)$ and the task embedding for v_n^i is obtained by,

$$\mathbf{e}_{v_u^i} = \sigma\Big(\sum_{k \in \mathbf{succ}(i)} \mathbf{o}_u^k\Big), \tag{3.57}$$

where $\mathbf{succ}(i)$ is the set of successors to task v_u^i [69].

Training Compound-Action RL Model with Embedding Layers

The RL system above, based on (3.54), has the compound-action space. It also has embedding layers for capturing the structure of complex IoT applications. We use the compound-action actor-critic (CA2C) algorithm proposed in [60] to solve this RL system and train the

embedding layers. CA2C algorithm is an algorithm to train the RL system with compound action space. We explain the details in what follows.

To learn the (near) optimal policy in a compound-action RL system, the CA2C algorithm is given in [60], which has actually taken advantage of the deep deterministic policy gradient (DDPG) algorithm [64], and the deep Q-network (DQN) algorithm [63]. Specifically, given $\boldsymbol{\pi} = (\boldsymbol{\pi}_1, \dots \boldsymbol{\pi}_U)$ as the policy profile for the UAVs, where $\boldsymbol{\pi}_u$ is the policy function for the UAV u. $\boldsymbol{\pi}_u$ is a function from a given state $\mathbf{S}_u$ to action $\boldsymbol{\alpha}_u$ (i.e., $\boldsymbol{\pi}_u : \mathcal{S} \mapsto \mathcal{A}_u$). Let us denote the Q-function for UAV u by $Q_u(\mathbf{S}_u, \boldsymbol{\alpha}_u)$. Q-function is defined in terms of the expectation of the weighted sum of the short-term reward for UAV u [61]. Denoted by $\boldsymbol{\pi}_u^*$, the optimal policy for the UAV u is the policy under which the Q-function for that UAV is maximized, i.e., $\boldsymbol{\alpha}_u^* = \pi_u^*(\mathbf{S}_u) = \arg \max_{\boldsymbol{\alpha}_u \in \mathcal{A}_u} Q_u^*(\mathbf{S}_u, \boldsymbol{\alpha}_u)$, $\forall \mathbf{S}_u \in \mathcal{S}_u$. In this equation, the action space is a compound. Thus, finding the optimal policy is in fact challenging. To face it, we decompose $\boldsymbol{\pi}_u^*$ into two policies [60] which are pertinent to the offloading scheme of the UAVs and adjustment of the UAV transmit power levels, respectively. Given state $\mathbf{S}_u$ and discrete action $\mathbf{a}_u$ (which corresponds to offloading the tasks), we denote the policy to adjust the UAV transmit power for offloading the tasks by $\boldsymbol{\nu}_u^*(\mathbf{S}_u, \mathbf{a}_u)$. The best discrete action is obtained by solving the following equation [60],

$$a_u^* = \arg \max_{\mathbf{a}_u \in \mathcal{A}_u} Q_u^*\Big(\mathbf{S}_u, (\mathbf{a}_u, \boldsymbol{\nu}_u^*(\mathbf{S}_u, \mathbf{a}_u))\Big). \tag{3.58}$$

The equation in (3.58) is challenging to solve. So, the Q-value function is approximated using two separated DNNs [60]. For our problem, each DNN is alongside some embedding layers for capturing the structure of the IoT application [69]. As depicted in Fig. 3.15, to train them, we use the CA2C algorithm proposed in [60], which is an actor-critic algorithm. For a given UAV u, let us denote the weights corresponding to actor and critic DNNs by $\boldsymbol{\theta}_u$ and $\mathbf{w}_u$, respectively. Actor DNN is used to approximate the optimal policy (which corresponds to adjusting the UAV's transmit power level), i.e., $\boldsymbol{\nu}_u^*$, and the critic network is used to approximate Q_u^* which in turn is used to obtain the task offloading schemes. The pertinent parameterized functions are denoted by $\boldsymbol{\nu}_u^*(\mathbf{S}_u, \mathbf{a}_u, \boldsymbol{\theta}_u)$ and $Q_u^*(\mathbf{S}_u, \boldsymbol{\alpha}_u, \mathbf{w}_u)$, respectively. The details of the training algorithm to update the parameters $\boldsymbol{\theta}_u$ and $\mathbf{w}_u$ are given in **Appendix A.4**. Using the derived training algorithms in the appendix, we propose our intelligent decision-making algorithm to offload the tasks and adjust the transmit power in the following.

The Proposed CA2C-based Algorithm

In this part, we use **Algorithm 1.1** and **Algorithm 1.2** in Appendix A.4 and drive a CA2C-based algorithm for task offloading and the adjustment of the UAV transmit power levels. We call it the intelligent joint power and offloading control (**IJPOC**) algorithm. In our proposed algorithm, for the critic side, given continuous action and transmit power level for the UAV, the $\epsilon-$greedy algorithm is used to adopt the discrete action, i.e., the tasks' offloading scheme. Therefore, for a given UAV u, we have,

$$\boldsymbol{\alpha}_u = \begin{cases} \arg\max\limits_{\mathbf{a}_u \in \mathcal{A}_u} Q_u(\mathbf{S}_u, \boldsymbol{\alpha}_u, \mathbf{w}_u) & \text{with the probability of } 1-\epsilon, \\ \text{randomly adopted from } \mathcal{A}_u & \text{with probability of } \epsilon. \end{cases} \tag{3.59}$$

Accordingly, the details for the **IJPOC** algorithm are given in **Algorithm 3.4**.

Based on **IJPOC** Algorithm, at a given state $\mathbf{S}_u$, regarding the scheme for locally performing/offloading the tasks in $\mathcal{V}_u$, UAV u uses the output of the online actor DNN. It updates the policy for its transmit power levels. Then, given $\mathbf{S}_u$ and $\mathbf{p}_u$, the task offloading schemes can be obtained by using the output of the online critic DNN. At each step, the transition of the current state, current action, reward, and the next state are captured in $\mathcal{D}_u$, experience memory for the agent (UAV) u. The experiences are uniformly chosen as mini-batch to train the actor and critic DNNs and the embedding layers.

Algorithm 3.4 The Proposed **IJPOC** Algorithm

Input: The maximum energy budget for the UAVs, E_u^{Max}, $\forall u \in \mathcal{U}$, adjacent matrix, A_u, $\forall u \in \mathcal{U}$, the features matrix X_u, $\forall u \in \mathcal{U}$;

Output: Optimal policy for locally performing/offloading the tasks in $\mathcal{V}_u$ and adjusting transmit power level for offloading a task onto a processor $m \in \mathcal{P}_u$;

Initialize: Experience memory $\mathcal{D}_u, u \in \mathcal{U}$, actor online network parameters $\boldsymbol{\theta}_u, u \in \mathcal{U}$, actor target network parameters $\boldsymbol{\theta}_u^- = \boldsymbol{\theta}_u, u \in \mathcal{U}$, critic online network parameters $\mathbf{w}_u, u \in \mathcal{U}$, critic target network parameters $\mathbf{w}_u^- = \mathbf{w}_u, u \in \mathcal{U}$, and the embedding layer parameters $\mathbf{W}_u, u \in \mathcal{U}$.;

procedure
  for each episode **do**
    Initialize $\mathbf{e}$ and $\mathbf{e}_{v_u^i}$ and set the network state $\mathbf{S}_u = (\mathbf{e}, \mathbf{e}_{v_u^i}), \forall v_u^i \in \mathcal{V}_u$;
    for each step **do**
      Given state $\mathbf{S}_u$ and the continuous action $\mathbf{p}_u$, the UAV $u \in \mathcal{U}$ uses its critic online network and approximates action-value $Q_u(\mathbf{S}_u, \boldsymbol{\alpha}_u, \boldsymbol{\theta}_u)$, $\forall \mathbf{a}_u \in \mathcal{A}_u$ and adopts $\mathbf{a}_u$ by using (4.21);
      Given state $\mathbf{S}_u$ and the discrete action $\mathbf{a}_u$ (corresponds to locally performing/offloading the tasks in $\mathcal{V}_u$), $\mathbf{p}_u$ is obtained by using the output for the actor network;
      For each UAV u, $\mathbf{e}$ is updated through the message passing and $\mathbf{e}_{v_u^i}, \forall v_u^i \in \mathcal{V}_u$ is updated through the embedding layers (by using (3.57)) and the next state $\mathbf{S}'_u$ is obtained. Set $\mathbf{S}_u \leftarrow \mathbf{S}'_u$;
      Store experience $\mathbf{x}_u = [\mathbf{S}_u, \boldsymbol{\alpha}_u, \eta_u, \mathbf{S}'_u]$ in memory $\mathcal{D}_u$ for $u \in \mathcal{U}$;
      Each UAV $u \in \mathcal{U}$ samples random mini-batch of transitions $[\mathbf{S}_u, \boldsymbol{\alpha}_u, \eta_u, \mathbf{S}'_u]$ from $\mathcal{D}_u$;
      For each UAV u, the parameters of the DNNs and the ones for embedding layers are updated by using **Algorithm 1.1** and **Algorithm 1.2** (in Appendix A.4);
    end for
  end for
end procedure

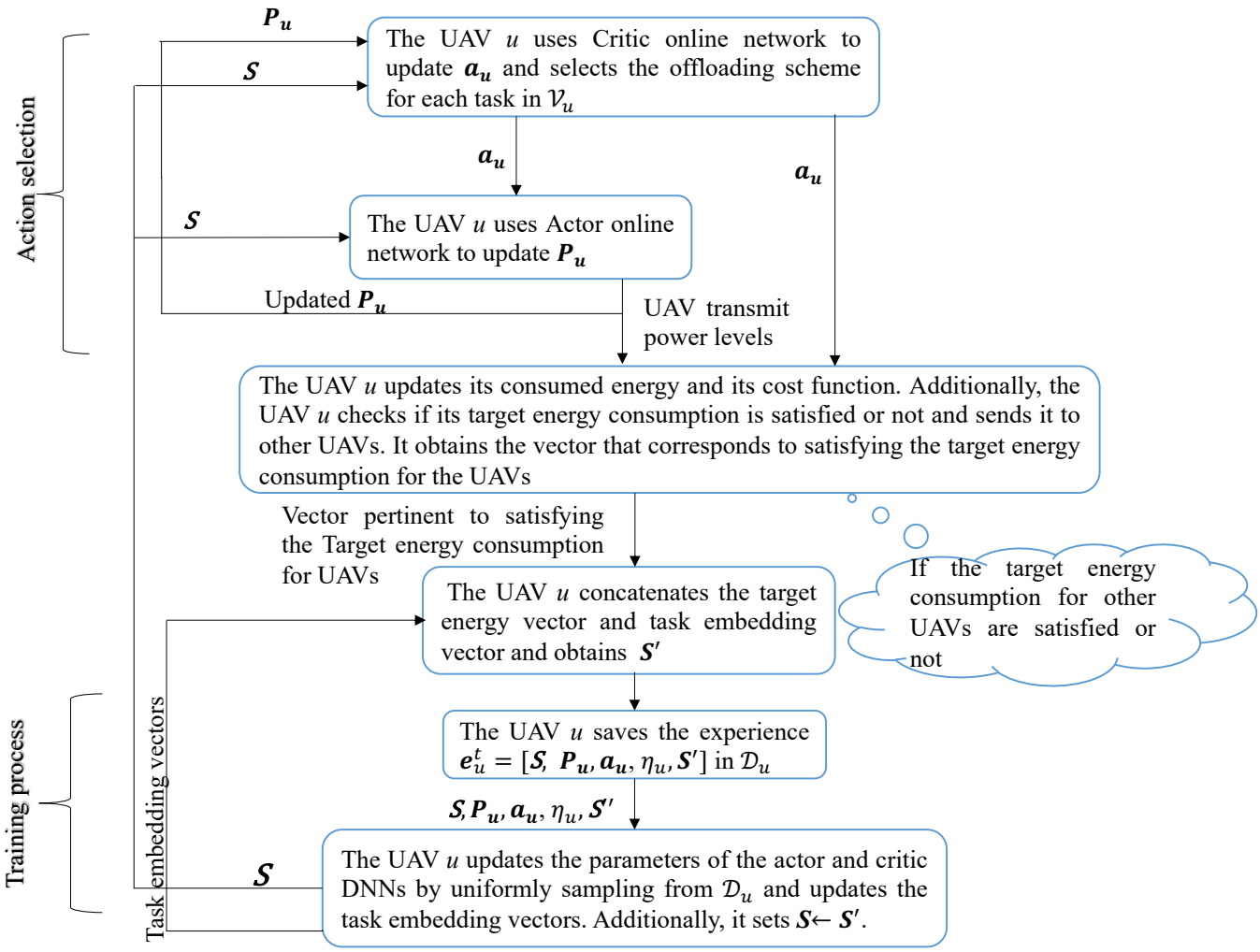

Figure 3.16: The message passing between a UAV u and other UAVs to update the offloading scheme for the tasks and the adjustment of transmit power level.

Complexity Analysis

Based on **Algorithm 3.4**, at a given time t, the message passing between a UAV u and the processors in $\mathcal{P}_u$ is depicted in **Fig. 3.16**. Note that for offloading and adjusting the transmit power for the UAV u, we have two main parts: action selection and training the actor and critic DNNs, and the embedding layers corresponding to the GCN. The complexity analyses for each part are provided in what follows.

Action selection complexity: For a given UAV u, based on **Algorithm 3.4**, the action selection corresponds to locally performing a task v_u^i or offloading it onto a processor $m \in \mathcal{P}_u$. Additionally, the action corresponds to adjusting the transmit power level for UAV u to offload the task v_u^i. This would be the most complicated part of the algorithm.

Given the fully connected actor and critic DNNs with a fixed number of hidden layers and neurons in each of them, for a given input, the computational complexity is pertinent to the sum of input and output sizes [101] for the actor DNN, critic DNN and the embedding layers. Based on equations, (3.58) and (3.55), the input size for the actor and critic DNNs are $U + 2 \times I_u + I_u \times U$ and $U + 2 \times I_u + I_u = U + 3 \times I_u$, respectively. Based on (3.56), the input layer size for the embedding layers is I_u^2. Specifically,in the actor-network, the term $I_u \times U$ corresponds to the output for the critic network, which are one-hot vectors used for offloading the tasks in $\mathcal{I}_u$. Additionally, $2 \times I_u$ corresponds to the embedding vector of the tasks. So, the complexity of obtaining the Q-value for a given state and action is $O(I^2 + U \times I_u)$. UAV u should select the Q-value among all processors in $\mathcal{P}_u$. Therefore, the computational complexity for action selection corresponds to $O(\mid \mathcal{P}_u \mid (I^2 + U \times I_u)) = O(U \times I_u^2 + U^2 \times I_u)$.

Training process complexity: In this part, both forward and backward propagation complexity should be considered. Let T_u be the training batch used to update the weights for the actor and critics DNNs. So, the forward propagation complexity corresponds to $O(T_u \times (U \times I_u^2 + U^2 \times I_u))$. The backward propagation complexity for DNNs with a fixed number of hidden layers and neurons corresponds to the product of the size of the input layer and the one for the output layer [101]. Thus, by considering the output size for the actor DNN, critic DNN and the embedding layers, i.e., I_u, $I_u \times U$ and $2 \times I_u$, respectively, the backward propagation complexity would be pertinent to $O(T_u \times (I_u^3 + U \times I_u^2))$. Accordingly, the computational complexity for the training procedure corresponds to $O(T_u \times (I_u^3 + U \times I_u^2))$.

3.4.3 Dis-joint Offloading and Power Control: DDQN-based Offloading and Game-theory-based power control

Previously, for a UAV as an agent, we employed the CA2C algorithm [60] to simultaneously offload the tasks of the IoT application (generated by the UAV) and adjust its transmit power level. In this section, we perform the task offloading and power control dis-jointly to address the optimization problem (3.53) sub-optimally. We particularly apply the DDQN algorithm [58] to derive an intelligent offloading scheme. Then, given the offloading scheme for the tasks of IoT application of each UAV, we derive a distributed and energy-efficient

power control scheme to send a task to a processor to be executed. The details are given in what follows.

DDQN-based Offloading Scheme

To derive an intelligent offloading scheme, let us define a multi-agent RL system where the UAVs are the agents. For a given agent (UAV) u, the reward function is given by equation (3.51), and the action space is defined as,

$$\mathcal{A}_u^{\text{DDQN}} = \{\mathbf{a}_u \mid \mathbf{a}_u = [a^i_{m,u}]^{i\in\mathcal{V}_u}_{m\in\mathcal{P}_u} \text{ where } a^i_{u,m} \in \{0,1\}\}. \tag{3.60}$$

Additionally, using the state space given by (3.55), we define the sate space as

$$\mathcal{S}_u^{\text{DDQN}} = \{\mathbf{S}_u^{\text{DDQN}} \mid \mathbf{S}_u^{\text{DDQN}} = (\mathbf{e}^t, \mathbf{e}^{\text{DDQN}}_{v^i_u}), \forall v^i_u \in \mathcal{V}_u\}. \tag{3.61}$$

For UAV u, as aforementioned, $\mathbf{e}^t$ corresponds to whether the energy budget for each UAV is satisfied or not, and $\mathbf{e}^{\text{DDQN}}_{v^i_u}$ is obtained from the output layer of a GCN. Here, we employ DDQN algorithm [58] to train the GCN layers. The details are given in what follows.

Based on the action space definition given above, $\mathcal{A}_u^{\text{DDQN}}$ is not only discreet but also grows exponentially as the number of UAVs and MEC servers increases. Previously, in [1,8], we employed DDQN algorithm [58] to address the RL system with huge discrete action space. DDQN algorithm is a DRL-based algorithm wherein, to prevent instability and overoptimistic value estimation, the action selection is separated from the action evaluation by using the target network with weights $\mathbf{O}^-_u, b \in \mathcal{B}$, the online network with weights $\mathbf{O}_u, b \in \mathcal{B}$ and replay buffer $\mathcal{D}^{\text{DDQN}}$. Ergo, the training procedure to update the weights for the online network can be seen below [58],

$$Y_u = U(\mathbf{S}_u^{\text{DDQN}}, \mathbf{a}_u) + \lambda Q_u^{\text{DDQN}}(\mathbf{S}'_u, \arg\max_{\mathbf{a}'\in\mathcal{A}_u^{\text{DDQN}}} Q_u^{\text{DDQN}}(\mathbf{S}'_u, \mathbf{a}', \mathbf{O}_u), \mathbf{O}^-_u), \tag{3.62}$$

$$\mathbf{O}_u \leftarrow \mathbf{O}_u + \zeta(Y_u - Q_u^{\text{DDQN}}(\mathbf{S}_u^{\text{DDQN}}, \mathbf{a}_u, \mathbf{O}_u))\nabla_{\mathbf{O}_u} Q_u^{\text{DDQN}}(\mathbf{S}_u^{\text{DDQN}}, \mathbf{a}_u, \mathbf{O}_u), \tag{3.63}$$

where $U(\mathbf{S}_u^{\text{DDQN}}, \mathbf{a}_u) = E[\eta_u(\mathbf{S}_u^{\text{DDQN}}, \mathbf{a}_u)]$, and $E[.]$ is used for the expectation operator. In this work, as already mentioned, we use GCN to derive the features for each task in the application by considering its neighborhood. We deliberately add some embedding layers to the DNNs (as input layers) and exploit the training procedure in (4.19) and (4.20) to train them. Additionally, the equations in (3.56) and (3.57) are used for propagation rule for embedding layers and obtaining $\mathbf{e}^{\text{DDQN}}_{v^i_u}$, respectively. Given the offloading scheme for tasks in IoT applications, we derive an iterative and distributed energy-efficient power control to adjust the transmit power level for the UAVs to offload a task.

Distributed Game-theory-based Power Control Scheme

Based on the objective in the resource allocation problem in (3.53), each UAV aims to simultaneously minimize its energy consumption and the delay in executing its IoT application (i.e., the UAV's ETC). As mentioned before, the delay in performing an IoT application corresponds to the finish time of the exit task of that application. Therefore, based on the formal expression for the UAV energy consumption, equation (3.50), and the finish time and ready time for performing a task in IoT application as in equations from (3.46) to (3.49), each UAV should minimize the transmit power level and maximize the rate to offload a task. This accord with maximizing the energy efficiency of the UAVs to perform the IoT application.

As aforementioned, for a given processor m, all UAVs in set $\mathcal{U}_m$ (the set of UAVs offloading their task onto the processor m, $a^i_{u,m} = 1, \forall u \in \mathcal{U}, i \in \mathcal{V}_u$) employ the frequency reuse technology. Based on the above discussion, focusing on simultaneously maximizing the energy efficiency for the UAVs to perform their complex IoT applications, we define the non-cooperative power control game as $\mathbf{G} = \langle \mathcal{U}_m, \mathbb{P}^m, U^m_u \rangle$ where the set of UAVs $\mathcal{U}_m$ is the set of players, and the strategy space is given by $\mathbb{P}^m = \bigtimes_{u \in \mathcal{U}_m} \mathbb{P}_u$ where $\mathbb{P}_u = [0, \ldots, p^{\text{Max}}_u]$. Further, the utility function for a player (UAV) u is denoted as $U^m_u(p_u, \mathbf{p}_{-u})$ and stated by,

$$U^m_u(p_u, \mathbf{p}_{-u}) = \frac{W \log_2 \left(1 + p_u H^m_u(\mathbf{p}_{-u})\right) - C_m f_m}{p_u}, \tag{3.64}$$

where $H^m_u(\mathbf{p}_{-u}) = \dfrac{h^m_u}{I^m_u(\mathbf{p}_{-u})}$, $I^m_u(\mathbf{p}_{-u}) = \sum\limits_{m' \in \mathcal{U}_m, m' \neq u} p_{m'} h^m_{m'} + \sigma^2_0$, $\mathbf{p}_{-u} = [p_{m'}]_{m' \in \mathcal{U}_m \setminus \{u\}}$, and C_m is the CPU cost for executing the offloaded task. In the above game, the utility function in (3.64) corresponds to the energy efficiency function for the UAV u. Energy efficiency is a well-known utility function previously in the non-cooperative power control games. Additionally, the strategy space $\mathbb{P}_u$ is compact, convex, and non-empty. Therefore, there is NE for the non-cooperative game $\mathbf{G}$. We say $\mathbf{p}^* = [p^*_u]_{u \in \mathcal{U}}$ is a NE for $\mathbf{G}$ if we have $U_u(p^*_u, \mathbf{p}^*_{-u}) \geq U_u(p_u, \mathbf{p}^*_{-u})$, for all UAV $u \in \mathcal{U}$. For a given UAV u, the transmit power level p^*_u is acquired through the following lemma.

Lemma 3.1. *For a given UAV u and processor $m \in \mathcal{P}_u$, let us define $\widehat{\gamma}^m_u = \exp\left(1 - b^m_u + W_L(-\dfrac{1}{\exp(1 - b^m_u)})\right) - 1$ where $b^m_u = -\dfrac{\log 2 C_m f_m}{W}$ and $W_L(.)$ is the W-Lambert function main branch and defined over $[-\dfrac{1}{e}, +\infty)$. Given $\mathbf{p}^* = [p^*_u]_{u \in \mathcal{U}_m}$ as a NE for non-cooperative power control $\mathbf{G}$, we have,*

$$p^*_u = \widehat{\gamma}^m_u \times \frac{I^m_u(\mathbf{p}^*_{-u})}{h^m_u} \tag{3.65}$$

Proof. See Appendix A.6. □

Now, using (3.65), we derive an iterative and distributed power control scheme as follows,

$$p_u^{(t+1)} = \psi_u(\mathbf{p}^t) = \min\{p_u^{\text{Max}}, \widehat{\gamma}_u^m \times \frac{I_u^m(\mathbf{p}_{-u}^t)}{h_u^m}\}, \tag{3.66}$$

where $\mathbf{p} = [p_u]_{u \in \mathcal{U}_m}$.

Lemma 3.2. *The power updating function $\psi_u(\mathbf{p}^t)$ converges to a unique fixed point.*

Proof. The power updating function (3.65) is the well-known power updating function in [109]. As shown in [109], $\psi_u(\mathbf{p}^t)$ is a standard interference function [110]. As seen in [110], the standard interference function converges to a unique fixed point. □

Proposed Intelligent Dis-joint Power and Offloading Control Algorithm

Now, based on the above discussion in the subsections 3.4.3 and 3.4.3, we derive our intelligent dis-joint power and offloading control (IDPOC) scheme in **Algorithm 3.5**. In **IDPOC** Algorithm, an agent (UAV) u, at state $\mathbf{S}_u^{\text{DDQN}}$, approximates the action value functions by using the output of its online DNN and obtaining the action, i.e., the offloading scheme for a task by employing ϵ-greedy policy in (4.21). Then, given the offloading schemes for tasks obtained by each agent, iterative and distributed scheme in (3.66) is used to adjust the transmit power level for each UAV. At each step, the transition of the current state, current action, reward, and the next state are captured in $\mathcal{D}_u^{\text{DDQN}}$, experience memory for the agent (UAV) u. The experiences are uniformly chosen as mini-batch to train the DNN and the embedding layers by using (4.19) and (4.20).

3.4.4 Simulation Results

We simulate a smart farm environment where several IoT devices ($\mid \mathcal{N} \mid = 24$) are randomly located. In this setup, we consider a MEC-UAV-assisted wireless network with one MEC server and several UAVs hovering above the farm area. The channels have both small and large fading with path loss and shadowing. Each UAV $u \in \mathcal{U}$ covers some IoT devices ($3 \leq \mid \mathcal{N}_u \mid \leq 8, \ \forall u \in \mathcal{U}$) and generates a complex IoT application consisting of dependent tasks, i.e., an application with general task dependency. To compare the performance of our proposed intelligent algorithms, the CA2C-based **IJPOC** algorithm and the DDQN-based **IDPOC** algorithm, with that of the priority-based **DPPOC** algorithm, we consider

Algorithm 3.5 The Proposed **IDPOC** Algorithm

Input: The maximum energy budget for the UAVs, E_u^{Max}, $\forall u \in \mathcal{U}$, adjacent matrix, A_u, $\forall u \in \mathcal{U}$, the features matrix X_u, $\forall u \in \mathcal{U}$;

Output: Optimal policy for locally performing/offloading the tasks in $\mathcal{V}_u$; UL transmit power level for offloading a task onto a processor $m \in \mathcal{P}_u$;

Initialize: Experience memory $\mathcal{D}_u^{\text{DDQN}}, u \in \mathcal{U}$, online network parameters $\mathbf{O}_u, u \in \mathcal{U}$, target network parameters $\mathbf{O}_u^- = \mathbf{O}_u, u \in \mathcal{U}$;

procedure
  for each episode **do**
    Set the network state $\mathbf{S}_u^{\text{DDQN}} = (\mathbf{e}, \mathbf{e}_{v_u^i}^{\text{DDQN}}), \forall v_u^i \in \mathcal{V}_u$;
    for each step **do**
      Given state $\mathbf{S}_u^{\text{DDQN}}$, each UAV $u \in \mathcal{U}$ approximates its action-value $Q_u^{\text{DDQN}}(\mathbf{S}_u^{\text{DDQN}}, \mathbf{a}_u, \mathbf{O}_u)$, $\forall \mathbf{a}_u \in \mathcal{A}_u^{\text{DDQN}}$ by using its online network;
      Given state $\mathbf{S}_u^{\text{DDQN}}$, each UAV $u \in \mathcal{U}$ employs the ϵ-greedy policy in (4.21) to set $a_{m,u}^i$ for all $i \in \mathcal{V}_u, m \in \mathcal{P}_u$;
      For each UAV u, given the offloading vector for the tasks in $\mathcal{V}_u$, obtain the transmit power level $\mathbf{P}_u$ by using the power control scheme (3.66);
      For each UAV u, $\mathbf{e}$ is updated through the message passing and $\mathbf{e}_{v_u^i}^{\text{DDQN}}, \forall v_u^i \in \mathcal{V}_u$ is updated through the embedding layers (by using (3.57)) and the next state $\mathbf{S}'_u$ is obtained. Set $\mathbf{S}_u^{\text{DDQN}} \leftarrow \mathbf{S}'_u$;
      Store experience $\mathbf{x}_u^{\text{DDQN}} = [\mathbf{S}_u^{\text{DDQN}}, \mathbf{a}_u, \eta_u, \mathbf{S}'_u]$ in memory $\mathcal{D}_u^{\text{DDQN}}$ for $u \in \mathcal{U}$;
      Each UAV $u \in \mathcal{U}$ samples random mini-batch of transitions $[\mathbf{S}_u^{\text{DDQN}}, \mathbf{a}_u, \eta_u, \mathbf{S}'_u]$ from $\mathcal{D}_u^{\text{DDQN}}$;
      For each UAV u, the parameters of the DNNs and the ones for embedding layers are updated by using (4.19) and (4.20);
    end for
  end for
end procedure

Table 3.3: Simulation parameters

Parameters	Value
Number of service types and packet length	2, and [16,8]
Computing (CPU) resources	UAV: Randomly between 0.5 and 1 GHz [104], MEC server: 2.2 GHz [78]
Maximum transmit power level	UAV-to-UAV: 10 , IoT-to-UAV: 3 W
Transmission rate	IoT-to-UAV: 2 Mbps, UAV's target rate: 4 Mbps
Number of episodes and steps	100, 100
Discount factor (λ), ϵ, learning rate (ζ), τ [33]	0.9, 0.1, 0.01, 0.01
Replay memory size for intelligent algorithms [33]	500
Optimizer and activation function [33]	Adam and ReLU
Capacitance coefficient (κ)	10^{-25}
Task features [78]	Workload: Randomly between 1 and 2 (normalized) Processing data: Randomly between 60 and 200 (normalized)
UAV's max. tolerable energy consumption (communication and computation energies) [104]	Randomly between 50 joule and 70 joule

two different scenarios. In the first scenario, the IoT devices are located close to each other, and the number of UAVs is fixed, but the number of tasks in the application varies. In the second scenario, the IoT devices are scattered in the farm area and can be located far from each other, and we consider that the number of UAVs varies and belongs to the set of $\{3, 5, 7\}$. In this scenario, the number of tasks in the application generated by each UAV is fixed. The performance is studied in terms of average delay for each UAV, energy consumption for each UAV and the ETC for each UAV. In our proposed DRL-based algorithms, i.e., **IJPOC** and **IDPOC**, the DDNs have two embedding layers, three hidden layers containing $128, 512, 1024$ neurons, respectively, and one output layer. The details for the simulation setup are given in Table 4.3.

It is noteworthy that the limitation of the number of UAVs in simulation results is due to using an actor-critic algorithm. **IJPOC** algorithm is an actor-critic algorithm that can be slow for several reasons: the complexity of the algorithms, the size of the network, and the difficulty of optimizing the network parameters. Specifically, three separate networks - the actor network, the critic network, and the embedding layers corresponding to GCN - must be trained together, which can be computationally expensive and time-consuming. In addition, optimizing the network parameters can be challenging, especially in environments with large state and action spaces.

Convergence Behaviour of IJPOC and IDPOC Algorithm

The convergence behavior of the intelligent algorithms for offloading and power control, **IJPOC** and **IDPOC**, are studied in Figs. 3.17 and 3.18, respectively. We attentively consider a scenario with five UAVs, each generating an IoT application with 6 interdependent

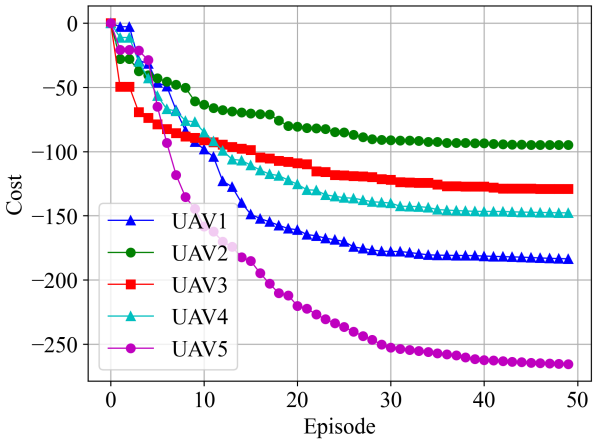

Figure 3.17: Convergence behaviour for the CA2C-based algorithm **IJPOC**

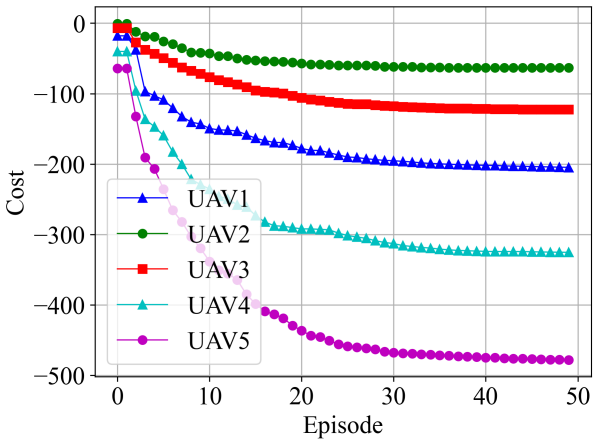

Figure 3.18: Convergence behaviour for the DDQN-based algorithm **IDPOC**

tasks. As discussed previously, the CA2C algorithm and DDQN algorithm are used to derive **IJPOC** and **IDPOC**, and their convergence behaviors have been studied in [60] and [58], respectively. The figures display that each UAV's average weighted reward function converges to a stationary point. Based on Figs. 3.17 and 3.18, the fixed point for each UAV's average weighted reward function would not be the same. This can be because each UAV employs its DNNs to offload and adjust the transmit power level in the CA2C-based algorithm and obtain the offloading schemes in the DDQN-based algorithm.

Performance Evaluation

In this subsection, the performance of our proposed intelligent algorithms, CA2C-based algorithm **IJPOC** and DDQN-based algorithm **IDPOC**, in comparison with that of our derived priority-based algorithm **DPPOC** are studied.

First, we consider the scenario where the number of UAVs is fixed, i.e., $U = 3$, and the number of tasks in the application generated by each UAV changes, $I_u \in \{3, \ldots, 11\}, \forall u \in \mathcal{U}$. In Figs. 3.19a, 3.19b, 3.19c, it can be seen that the performance of the proposed intelligent algorithms **IJPOC** and **IDPOC** in terms of average delay and average consumed energy and average ETC for each UAV is better than those for priority-based **DPPOC** algorithm. In our intelligent algorithms, by considering the embedding layers, the features of each task in the application regarding its neighborhood in DAG are used to derive the offloading scheme. Specifically, in **IJPOC** algorithm, the embedding layers are used by the actor DNNs to intelligently adjust the transmit power level for the UAV to offload a

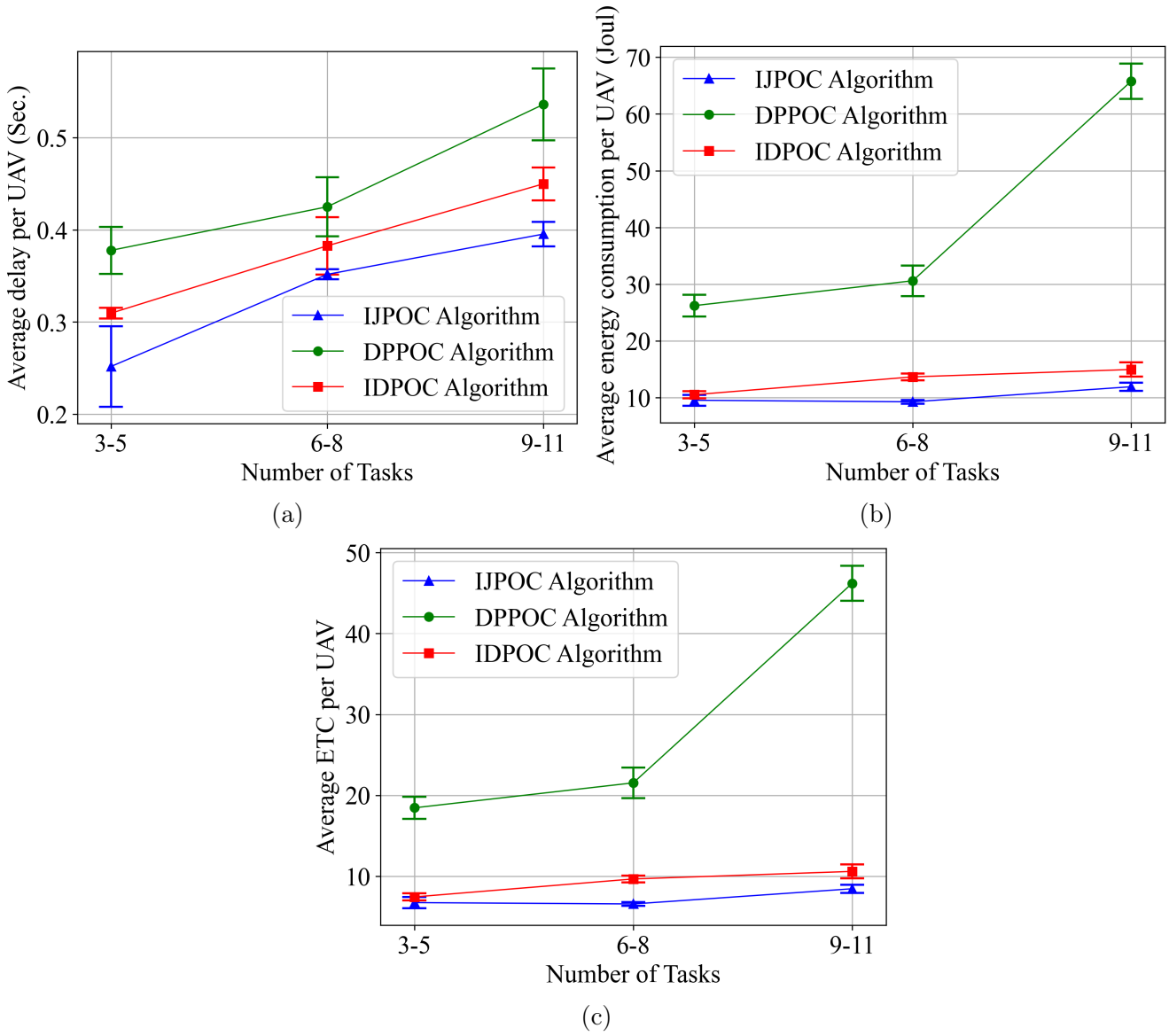

Figure 3.19: Performance evaluation for CA2C-based algorithm **IJPOC**, DDQN-based algorithm **IDPOC**, and priority-based algorithm **DPPOC** with a **fixed number of UAVs** ($U = 3$) vs. number of tasks in the application in terms of average delay for each UAV (a), average each UAV's energy consumption (b), and average ETC for each UAV (c).

task. This can result in better performance for the CA2C-based **IJPOC** algorithm than the DDQN-based **DPPOC** algorithm.

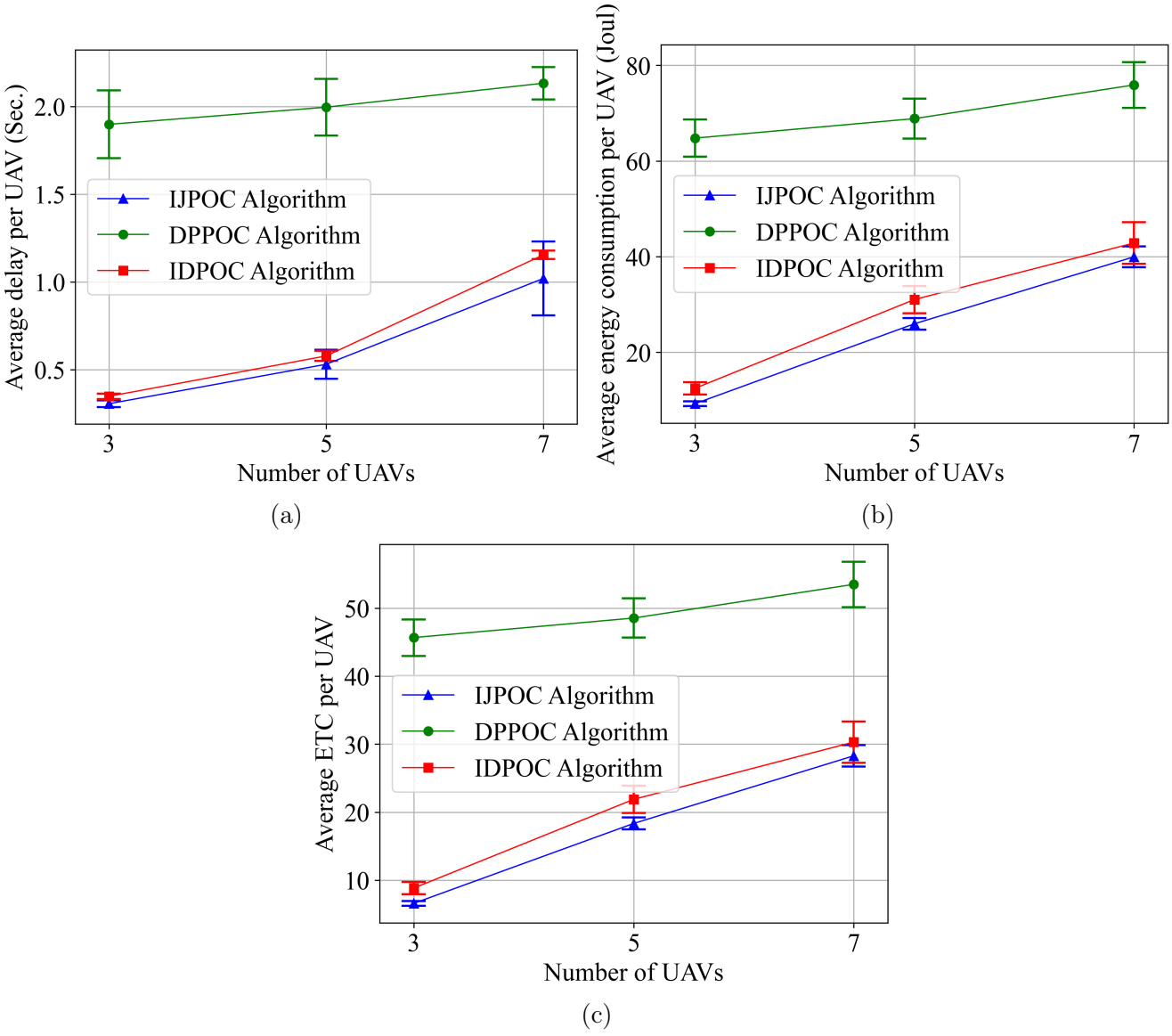

Figure 3.20: Performance evaluation for CA2C-based algorithm **IJPOC**, DDQN-based algorithm **IDPOC**, and priority-based algorithm **DPPOC** with a **fixed number of tasks** in the application ($I_u = 6,\ \forall u \in \mathcal{U}$) vs. number of UAVs in terms of average delay for each UAV (a), average each UAV's energy consumption (b), and average ETC for each UAV (c).

To continue, we assume that the number of UAVs varies, i.e., $U \in \{3, 5, 7\}$. Additionally, each IoT application is composed of 6 interdependent tasks. As depicted in Figs. 3.20a, 3.20b, 3.20c, the performance of the intelligent algorithms **IJPOC** and **IDPOC** in terms of average delay and average consumed energy and average ETC for each UAV is better than those for priority-based **DPPOC** algorithm. This is because, in the intelligent algorithms, the offloading scheme for each task in the IoT applications is intelligently derived using the DDQN algorithm. This has a substantial effect on energy consumption for executing the IoT application. It can also be observed from these figures that the performance of the CA2C-based **IJPOC** algorithm is better than that of the DDQN-based **IDPOC** algorithm. This is because, in **IJPOC** algorithm, both offloading scheme and power control are jointly performed by considering the topology for the IoT application. In contrast, in **IDPOC** algorithm, only offloading scheme is performed intelligently.

Chapter 4

Intelligent Energy-efficient Resource Management in the Wireless Networks with CA Functionality

4.1 Introduction

Fulfilling various QoS requirements including high throughput, low latency and low UE power consumption is highly challenging for 5G wireless networks [20]. In CA, a UE is served with multiple frequency blocks, referred to as CC, to achieve higher throughput and lower transmission delay. Additionally, the UEs are configured to access more CCs depending on their QoS requirements, traffic load, and channel quality [10].

Recently, 5G resorts to DC to boost its capacity and provide enhanced coverage under multiple bands. DC or, more generally, MC allow operators to simultaneously provide LTE and 5G connectivity. In 3GPP Release 15 and thereon, this technology is called EN-DC, where LTE is used as a master node, and NR is used as a secondary node. In EN-DC, UE is allowed to connect to a LTE eNB and a 5G Next Generation NodeB (gNB) simultaneously, i.e., separately transmitting/receiving LTE and 5G signals and then aggregating the streams. The UEs can enormously benefit from employing joint CA and DC technologies to increase coverage area, system throughput, load balancing and mobility robustness [82]. Meanwhile, these emerging technologies have increased complexity, and challenged deriving beneficial resource management schemes.

Despite the above advantages, employing CA and DC technologies would cause some challenges in the wireless networks. For instance, CA causes UEs to consume more energy

because the UEs monitor a large aggregated spectrum than they would perform with a single carrier allocation [21]. UE power consumption affects the battery life of a mobile device and must be carefully managed. In the UE receiver, the Radio Frequency module and Base Band processing part, which depend on the number of CCs and the rate, consume energy to receive the signal and decode information [21]. Therefore, the power consumed by each UE is intensified as the number of CCs and the rate increase. Hence, to save the UE power consumption in CA mode while enhancing the throughput, dynamic management of activation and deactivation of CCs plays a prominent role [21]. For the DC, one of the significant challenges arising in UL transmission in the EN-DC network is dividing the UL transmit power level between the eNB and gNB, subjected to the maximum power budget for the UE. In the literature, [95,111] two different schemes, Dynamic Power-Sharing (DPS) and Equal Power-Sharing (EPS), have been proposed to adjust the UL transmit power level. The UL transmit power levels to both eNB and gNB are dynamically adjusted in DPS. While, in EPS, the UL transmit power level is equally divided between the eNB and the gNB. Employing EPS may result in poor network performance because the time-varying characteristics for the channels and UE traffic in 5G networks would not be taken into consideration [95].

Furthermore, 5G has multi-dimensional and dynamic characteristics. These features has motivate us to employ RL as a valid tool for resource management in 5G networks [112] to face the above issues in 5G wireless with CA and DC technologies. One may employ some optimization and game theory methods to tackle the problem; however, this can result in a lack of scalability and flexibility.

Based on what was discussed above, in Section 4.2, to decide whether to activate or deactivate the CCs, we consider both the delay in transmitting UE data and the consumed power by the UE to monitor the activated CCs and also for based-band processing. We formulate the problem in a distributed manner where each UE (acting as an agent) can use their local knowledge about the network parameters, and learn to activate and deactivate the CCs. Accordingly, the CC management problem is formally expressed as a stochastic game and solved using a multi-agent DRL algorithm [8]. In Section 4.3, we consider an EN-DC network with CA technology. We use multi-agent RL and derive a UL resource management scheme to adjust UE UL transmit power levels to the eNB and gNBs. To the best of our knowledge, this problem has not yet been studied. By employing the CA2C method in [60], we propose an intelligent joint power control and CC management scheme to minimize the delay and the total power consumption of UEs [9]. The details are given in the following sections.

4.2 Delay-Aware and Energy-Efficient Carrier Aggregation in 5G using Double Deep Q-Networks

As a critical technology in 5G networks, CA is studied in [8]. In CA, CCs can be activated and deactivated depending on multiple factors, e.g., the energy consumption of the users and QoS demand. CC management strategies are proposed where each UE simultaneously minimizes both its average delay and power consumption while considering that CCs can be activated and deactivated only at certain times, as in real-world CA implementations. The problem can be modeled as a centralized multi-objective optimum CC management problem. Since centralized approaches would impose significant overhead on the system, we develop a semi-distributed solution by modeling the problem as a stochastic game and propose a multi-agent DDQN-based CC management algorithm to solve the stochastic game. Our derived DDQN-based algorithm is compared with the proposed approaches with single CC activation, all-CC activation, and the optimum CC activation baseline schemes [8].

4.2.1 System Model and Problem Statement

Network Model and Preliminaries

We consider the downlink transmission in a single-cell 5G network. The network consists of a gNB and a set of K UEs denoted by $\mathcal{K} = \{1, \ldots, K\}$. The channel between a gNB and a UE is a time-frequency-varying one. The channels have both small and large fading with path loss and shadowing. Additionally, the gNB uses CA, i.e., several CCs can be selected to serve the UEs. Let $\mathcal{M} = \{1, \ldots, M\}$ denote the set of M non-overlapping and orthogonal CCs. Each CC m consists a set of N_m Resource Block (RB)s denoted by $\mathcal{N}_m = \{1, \ldots, N_m\}$. Let M_k be the maximum number of CCs that can be assigned to UE k. For a UE k, M_k includes both PCC and SCCs assigned to the UE. The PCC is the main CC and is only updated when a UE is assigned/handover to a cell. In [8], to assign a PCC to a UE, the Round-Robin (RR) approach is used, and the PCC is always active for the UE as long as there is user traffic. Meanwhile, a SCC is an auxiliary CC that can be activated/deactivated anytime to boost the data rate of the UE. So, a SCC will be assigned to a UE based on traffic and QoS demand.

We assume that each RB is assigned to only one UE in a given time slot. However, several RBs in a CC supported by UE k can be assigned to the user. Additionally, each UE uses its instantaneous SINR on an RB to compute the Channel Quality Indicator

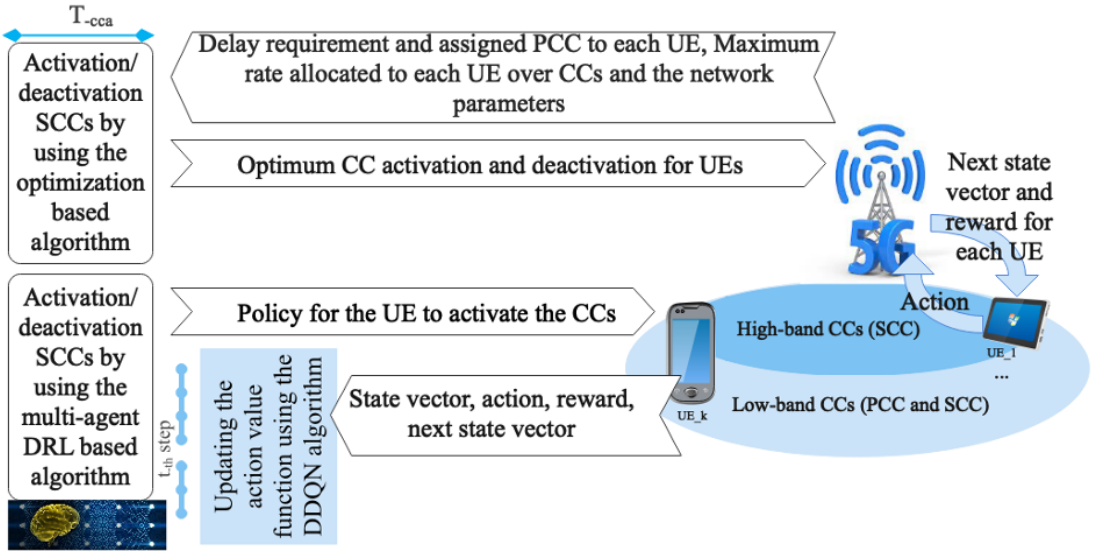

Figure 4.1: Network and the proposed optimization-based method (at top) and RL-based strategy (at the bottom) for CC management. For the RL-based strategy, the state vector corresponds to if the QoS requirement for each UE is provided or not. Action corresponds to which CCs are activated for the UE. The reward for the UE is given in terms of the reciprocal of the delay and the power consumption for the UE.

(CQI). Then, the computed CQI is sent to the gNB and mapped to the corresponding Modulation and Coding Scheme (MCS), and spectral efficiency as in the 3GPP LTE/NR standard [113].

In [8], the CC activation/deactivation and the RB allocation are performed separately. Hence, two different resource management schemes are considered. We propose DDQN-based CC management scheme for CC assignment to minimize the average delay. A CC can be shared among different UEs. Therefore, we use a RR scheme to assign the RBs in the CC to the UEs (sharing the CC). It is worth mentioning that RB allocation is performed at each time slot. Meanwhile, due to practical limitations, the CC activation/deactivation scheme is executed at every T_{cca}.

The following sub-sections briefly explain the throughput, delay and power consumption models and our CC management solution. The details for the system model and the signals transferred between a UE, gNB, and our proposed algorithm and the optimum baseline algorithm are briefly depicted in Fig. 4.1. Specifically, for CA, the details for deriving the optimum baseline algorithm are given in Subsection B.1 and the multi-agent RL-based algorithm is explained in Subsection 4.2.2. As illustrated in Fig. 4.1, in our proposed

multi-agent RL-based CA scheme, based on whether the delay requirement for each UE is satisfied or not (which corresponds to the state of the RL-based system), a given UE $k \in \mathcal{K}$ takes action to activate/deactivate a SCC. In the eNB, based on action taken by the UE, a SCC is activated/deactivated for the UE, and the reward for UE k and the state are updated and sent to the UE for getting the new action. As will be seen in Subsection 4.2.2, the reward for the UE k is given in terms of the reciprocal of the delay for the UE and its power consumption.

UE Throughput Model: Let us define $\alpha_m^k \in \{0, 1\}$ as the CC allocation indicator, where $\alpha_m^k = 1$ if CC m is allocated to UE k; otherwise, we have $\alpha_m^k = 0$. We denote the instantaneous total achievable rate for UE k by $R^k(t)$ which is expressed as follows [21],

$$R^k(t) = \sum_{m \in \mathcal{M}} \alpha_m^k R_m^k(t), \tag{4.1}$$

where $R_m^k(t)$ is the instantaneous achievable rate for UE k over CC m[1]. In what follows, we use the instantaneous throughput for a UE as given in (4.1), and express its delay model.

UE Delay Model: The CC activation/deactivation scheme is executed every $\mathrm{T_{cca}}$ time, and after the decision, the activated or deactivated CCs can no longer be changed until the next $\mathrm{T_{cca}}$ time. Thus, we have to estimate the next bursts of payloads and the arrival times during the next $\mathrm{T_{cca}}$ time to evaluate the required rate or the activated CCs for each user during this $\mathrm{T_{cca}}$ time. Considering the current duration, i.e., $[t, t + \mathrm{T_{cca}}]$, we formally express the end-to-end delay for a UE. Specifically, the delay for a UE in one-time duration would be the summation of the delay for delivering the remaining bursts of payloads in the scheduling queue from the previous scheduling occasion and the delay for delivering the estimated bursts of payloads in the scheduling queue, both of which are formulated as below.

For the given UE k, let $\overline{D}_r^k$ denote the delay in delivering the remaining bursts of payloads from the previous duration. Additionally, let $\overline{q}_r^k(t)$ and $\widehat{N}_r^k(t)$ be the average number of bits in one bursts of payloads in the previous duration and the number of remaining bursts of payloads in the scheduling queue from the previous duration, respectively. Using the instantaneous rate for UE k in (4.1), we have,

$$\overline{D}_r^k = \frac{\overline{q}_r^k(t) \times \widehat{N}_r^k(t)}{R^k(t)}. \tag{4.2}$$

[1]The achievable rate for UE k on a given CC m is the summation of the rate for that UE over its assigned RBs on CC m. That is , $R_m^k(t) = \sum_{n \in \mathcal{N}_m} \omega_{m,n}^k R_{m,n}^k(t)$ where $R_{m,n}^k(t)$ is the rate for the UE k over the RB n in CC m. As denoted, for a given CC, we employ RR scheduler to allocate the RBs to the UEs sharing that CC (i.e., for obtaining $\omega_{m,n}^k$).

Further, to estimate the delay in delivering the bursts of payloads at the current duration, we denote $\widehat{q}_c^k(t)$ and $\widehat{T}^k(t)$ as the estimated average number of bits in one burst of payloads and the estimated inter-arrival time between two consecutive bursts, respectively. Given T_f^k as the time that the last burst of payloads for user k has arrived, the estimation of the average number of bursts at the current duration is denoted by $\widehat{N}_c^k(t)$ and can be given by $\widehat{N}_c^k(t) = \Big\lfloor \frac{t + \mathrm{T_{cca}} - T_f^k}{\widehat{T}^k(t)} \Big\rfloor$. Thus, we denote the estimated average delay for transmitting the next arriving bursts of payloads in the current duration by $\overline{D}_c^k(t)$, and we have,

$$\overline{D}_c^k = \frac{\widehat{q}_c^k(t) \times \widehat{N}_c^k(t)}{R^k(t)} = \frac{\widehat{q}_c^k(t)}{R^k(t)} \times \Big\lfloor \frac{t + \mathrm{T_{cca}} - T_f^k}{\widehat{T}^k(t)} \Big\rfloor. \tag{4.3}$$

Using eq.(4.2) and (4.3), we denote the delay for UE k in current duration (i.e., $[t, t+\mathrm{T_{cca}}]$) by $\overline{D}^k(t)$ and state it as,

$$\overline{D}^k(t) = \overline{D}_r^k(t) + \overline{D}_c^k(t) = \frac{\widehat{q}^k(t)}{R^k(t)}, \tag{4.4}$$

in which

$$\widehat{q}^k(t) = \widehat{q}_c^k(t) \times \Big\lfloor \frac{T_{cca} + T_f^k}{\widehat{T}^k} \Big\rfloor + \overline{q}_r^k(t) \times \widehat{N}_r^k(t). \tag{4.5}$$

Let $\mathrm{D}_{\mathrm{QoS}}^k$ be the maximum tolerable delay for a given UE k. At a given time t, the QoS requirement for UE k is satisfied if we have,

$$\frac{\widehat{q}^k(t)}{R^k(t)} \leq \mathrm{D}_{\mathrm{QoS}}^k. \tag{4.6}$$

Additionally, we say that the vector of delay requirement for the UEs, $[\mathrm{D}_{\mathrm{QoS}}^k]_{k\in\mathcal{K}}$, is feasible if (4.6) holds for each UE. In (4.6), $\widehat{q}^k(t)$ is the average arrived number of bits per burst of payloads for UE k.

UE Power Consumption Model: We model the consumed power for UE k similar to [114]. The authors have exploited this model in [21] to formulate the consumed power for the users in 5G network with CA. The consumed power for UE k is denoted by $P^k(t)$ and given by,

$$P^k(t) = P_0 + \sum_{m=1}^{M} \alpha_m^k(t) \Big[P_{R_x,m}^k + \Big(K_r R_m^k(t) + K_{r_0} \Big) + \Big(K_s p_m^k(t) + K_{s_0} \Big) + \Big(K_{BW} BW_m + K_{B_0} \Big) \Big]. \tag{4.7}$$

In the above equation, $R_m^k(t)$ is the achievable rate for UE k on its activated CC m, $p_m^k(t)$ is received power for UE k over CC m, BW_m is the bandwidth of CC m, P_0 is the base power when the UE is in its RRC connected mode without receiving any traffic. $P_{Rx,m}^k$ is the base consumed power when CC m is activated for UE k, and finally $K_r, K_{r_0}, K_s, K_{s_0}, K_{BW}$ and K_{B_0} are the fixed parameters. Based on (4.7), the consumed power for the UE is related to each activated CC. It is a first-degree polynomial function of the received power, the bandwidth of the activated CC, and the rate achieved for the UE over this CC[2]. We use the above equations and express the resource management problem in what follows.

Resource Management Problem Formulation

In [8], we deal with the problem of CC management in 5G networks to maximize the system efficiency. Inspired by energy efficiency, the number of bits that can be sent over a unit of power consumption [116], we define the system efficiency as the ratio of the reciprocal of the delay to the consumed power for activating/deactivating the CCs. This can be used in the networks with the services sensitive to the delay and the energy. Let us denote the system efficiency by ζ and define it as follows,

$$\zeta = \frac{1}{E[D] \times E[P]}. \tag{4.8}$$

In (4.8), $E[D]$ and $E[P]$ are the long-term average delay and the average power consumption for the UEs, respectively. Since the average delay and average power are both positive, we can restate the problem of maximizing ζ as simultaneously minimizing both $E[D]$ and $E[P]$ [117]. Hence, the CC management in each duration aims to minimize the average delay given in (4.4) and simultaneously minimize the average consumed power among all users defined in (4.7) by selecting optimum CCs. However, the two objectives mentioned above, minimizing the delay and minimizing the power consumption for the UEs, are conflicting. That is, there is no point wherein the two objectives are simultaneously minimized. To handle this trade-off, we use the concept of multi-objective optimization and formulate the

[2]As can be seen in Subsection 4.2.2, in our proposed multi-agent RL-based algorithm, a DNN is used by each UE to approximate the Q-value function. Training a DNN would consume power; however, a pre-trained hidden layer structure can be used to pre-train the Q-value function and reduce the time and complexity. So, training does not need to be done frequently; thus, our power consumption model has not considered the corresponding computational power. Recently, pre-trained models have been widely used to solve complicated RL systems [115].

problem as follows,

Min	Average delay		(4.9a)
Min	Average consumed power		(4.9b)
s.t.	Number of active CCs	$\forall k \in \mathcal{K}$,	(4.9c)
	QoS requirement	$\forall k \in \mathcal{K}$,	(4.9d)
var.	CC and RB assignment indicators,		(4.9e)

where the constraint (4.9d) corresponds to the delay requirements for the UEs in (4.6). It is noteworthy that the optimal solution for problem 4.9 is the solution for the problem of maximizing the system efficiency. For the proof, the reader is referred to [117]. Problem (4.9) is a multi-objective optimization problem, and in addition, it has the combinatorial characteristics. Thus, (4.9) is hard to solve, especially for large-scale networks. As mentioned in [10], the problem of CC management and RB allocation can be sorted out jointly or dis-jointly. In [8], following the latter solution, we formulate the CC management and RB allocation in (4.9). We specifically focus on CC activation/deactivation in [8], and use RR for RB allocation.

To solve the CC activation/deactivation problem in (4.9), two viewpoints, centralized and distributed approaches, can be considered. For the former, accurate knowledge about the environment, including the UEs' CQI and parameters for computing the users' power consumption, is available at a central unit. The weighted sum of the objective functions can specifically provide a trade-off between (4.9a), and (4.9b) and a Pareto frontier for the problem can be derived. In this method, the weight for the objectives correspond to the priority for optimizing them. The more the weight for an objective, the more the priority for optimizing it. In Appendix B.1, by considering the central unit, the problem is formally expressed, and Charnes-Cooper transformation and Glover's linearization are applied to obtain equivalent LP. However, the complexity of addressing problem (4.9) in a centralized manner increases as the number of UEs grows. This thesis considers the scenario wherein each UE can use its knowledge about the network parameters, and a semi-distributed approach is derived in Subsection 4.2.2 to address (4.14). This method employs the CC activation/deactivation decision in each UE individually by utilizing a multi-agent RL-based method. The details for the action, reward and state for our proposed multi-agent RL-based method are given in Subsection 4.2.2. Thus, using the objective function in (4.9a) and (4.9b) for centralized and semi-distributed scenarios, we formulate two corresponding optimization problems (B.8) and (4.14), respectively. For the details of the derived optimum CC management algorithm (**OCCM**), we refer readers to **Appendix B.1**. The details for our derived intelligent CC management algorithm are given in the next subsection.

4.2.2 Multi-agent RL-based CC Management

First, this subsection formally expresses the CC management problem as a stochastic game. Specifically, we formally express the reward for each player (UE) in terms of the reciprocal of its delay and power consumption. To address the stochastic game, we express it as a MDP. Thus, knowledge about the UE reward functions and state transition is needed. This information would not necessarily be available for each UE in 5G network environment. To respond to this issue, we employ RL and expressed the MDP as a model-free multi-agent RL system [61]. Finally, we use the multi-agent DRL to find the NE at each state of the game. The details are given in what follows.

Stochastic Game Formulation

Given time t, let $U_k(t)$ be the utility function for UE k. In [8], each UE tries to simultaneously minimize its delay and consumed power, as mentioned before. Using equation (4.4), we define the utility function for UE k in terms of the ratio of the reverse delay for the UE to the total reverse delay for other UEs and formally express it as follows,

$$U^k(t) = \log_2(1 + \frac{\overline{D}^k(t)^{-1}}{\sum\limits_{j \in \mathcal{K}, j \neq k} \overline{D}^j(t)^{-1}}), \tag{4.10}$$

Additionally, denoting by $C^k(t)$, we use the proposed cost function in [118] and define the cost function for UE k in terms of its consumed power, i.e.,

$$C^k(t) = P^k(t), \tag{4.11}$$

where $P^k(t)$ is given in (4.7). Specifically, since we aim to minimize the delay while minimizing UE power consumption, we have chosen the UE power consumption as the cost. Accordingly, we denote the instantaneous reward function for the UE k by $\eta^k(t)$ and express it as follows,

$$\eta^k(t) = U^k(t) - \beta^k \times C^k(t) = \log_2(1 + \frac{\overline{D}^k(t)^{-1}}{\sum\limits_{j \in \mathcal{K}, j \neq k} \overline{D}^j(t)^{-1}}) - \beta^k P^k(t), \tag{4.12}$$

where β^k is the unit price of the consumed power for UE k. The immediate reward function for UE k in (4.12), i.e., $\eta^k(t)$, is used to formulate the problem. Specifically, we denote the

long-term reward for UE k by Φ^k and formally express it as,

$$\Phi^k = \sum_{t=0}^{T-1} \lambda^t \eta^k(t). \tag{4.13}$$

Based on (4.13), Φ^k is the weighted sum of $\eta^k(t)$ over the finite time T. Additionally, in (4.13), we have $0 \leq \lambda \leq 1$. λ is the discount factor used to determine the weight for the future reward. Clearly, the future reward equals to immediate reward when $\lambda = 0$; otherwise, if $\lambda < 1$, the value for $\lambda^t \eta^k(t)$ in the earlier periods is greater than its future value. Accordingly, to concurrently maximize the long-term reward for each UE k, the problem of CC management can be formally expressed as a multi-objective optimization problem as follows,

$$\max_{\boldsymbol{\alpha}^k} \quad \Phi^k \qquad \forall k \in \mathcal{K}, \tag{4.14}$$

where $\boldsymbol{\alpha}^k = [\alpha_m^k]_{m \in \mathcal{M}}$. In optimization problem (4.14), each UE maximizes its long-term reward. More specifically, based on (4.12), (4.4), and (4.7), when a CC is activated/deactivated for a UE k, maximizing the value of Φ^k corresponds to minimizing the long-term delay, and at the same time minimizing the long-term consumed power to activate/deactivate the CCs. Note that some issues will arise when finding a globally optimal solution for (4.14) because it is a multi-objective non-convex optimization problem with combinatorial characteristics. To tackle them, we first reformulate the problem in (4.14) as a stochastic game. Then, an RL-based method is proposed to find the NE at each state for the game.

Let us assume that the UEs do not have complete knowledge about the network environment. Additionally, each UE activates/deactivates the CCs to maximize its long-term reward, irrespective of other UE rewards. At time t, the instantaneous reward for a UE is given in terms of the current state of the network environment and the actions of other UEs. Also, the current state and the actions taken by all other UEs influence the next state. Hence, the optimization problem in (4.14) can be expressed as a stochastic game $\mathcal{G} = \langle \mathcal{K}, \mathcal{S}, \mathcal{A}^k, P_{ss'}, \eta^k \rangle$ [59], where, $\mathcal{K}$ is the set of UEs. The details for the state space, $\mathcal{S}$, actions set for UE k, $\mathcal{A}^k$, the transition probability function $P_{ss'}$, and the reward function for UE k, η^k, are given in what follows.

State: In the stochastic game $\mathcal{G}$, the set of possible states is defined by $\mathcal{S} = \{\mathbf{s}(t) | \mathbf{s}(t) = [s^k(t)]_{k \in \mathcal{K}}^{\mathrm{T}}\}$. Especially at a time t, the kth element in vector $\mathbf{s}(t)$ corresponds to whether the delay requirement for UE $k \in \mathcal{K}$, is met or not. That is, $s^k(t) \in \{0, 1\}$, $\forall k \in \mathcal{K}$, and $s^k(t) = 1$ if (4.6) holds true for UE k; otherwise, $s^k(t) = 0$. Accordingly, $| \mathcal{S} | = 2^{\mathrm{K}}$ and $| \mathcal{S} |$ increases exponentially as K grows.

Action: As mentioned before, a UE k should activate/deactivate a CC at a given time t. So, we define the action space for UE k as the set of all possible CCs that can be activated/deactivated by that UE, i.e., $\mathcal{A}^k = \boldsymbol{\alpha}^k = [\alpha_m^k]_{m\in\mathcal{M}}$. Specifically, $\alpha_m^k = 1$, if CC m is activated for UE k and $\alpha_m^k = 0$, otherwise. Therefore, we have $\mid \mathcal{A}^k \mid = 2^{\mathrm{M}}$, which grows exponentially by an increase in M.

Transition probability: The function for the state transition probability is defined as $P : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \rightarrow [0, 1]$. Specifically, P is a function from $\mathcal{S}\times\mathcal{A}\times\mathcal{S}$ to $[0, 1]$ where the set $\mathcal{A}$ is the Cartesian product of the action space for the UEs, i.e., $\mathcal{A} = \mathcal{A}^1 \times \ldots \times \mathcal{A}^K$. Thus, for a given action vector $\mathbf{a} \in \mathcal{A}$, we have $\mathbf{a} = (a^1, \ldots, a^K)$ where $a^k \in \mathcal{A}^k$. Furthermore, when action vector $\mathbf{a}$ is used by the UEs in state s, $P_{ss'}(\mathbf{a})$ is the probability of transition to state s'. Therefore, based on (4.2) to (4.6), the transition probability is related to the UE delay requirements, the achievable rate for the UEs, number of bits per burst of payloads and the CCs activated for the UEs.

Reward: The reward for UE k, i.e., $\eta^k(t)$, can be expressed by $\eta^k(t) = \eta^k(s(t), a^k, \mathbf{a}^{-k})$, where $\mathbf{a}^{-k} = (a^1, \ldots, a^{m-1}, a^{m+1}, \ldots, a^K)$.

Let us denote $(a^{*k}, \mathbf{a}^{*-k})$ as a solution for the game. The solution $(a^{*k}, \mathbf{a}^{*-k})$ is a NE point, if we have,

$$\eta^k(s, a^{*k}, \mathbf{a}^{*-k}) \geq \eta^k(s, a^k, \mathbf{a}^{*-k}) \quad \forall s \in \mathcal{S}, \forall k \in \mathcal{K}, \forall a^k \in \mathcal{A}^k. \tag{4.15}$$

In (4.15), the action a^{*k} for UE k would correspond to the best response to the other players (UEs). Additionally, unilateral deviation from the equilibrium point would not be beneficial for other UEs [100, 119].

The stochastic game $\mathcal{G}$ is an episodic one. Specifically, in $\mathcal{G}$, at the end of an episode, the state is reset to start a new episode. Additionally, for a given episode, the accumulative rewards can be obtained from the environment by using the policy given in terms of states, actions, and rewards. Thus, information about the UE reward functions and state transition is needed to find the NE. The mentioned information would not be necessarily available for each UE in the network environment. This is the major issue in solving the game $\mathcal{G}$ (i.e., obtaining a NE point at a given state s). We propose an RL-based method to address this issue in the following subsection.

The Proposed DDQN-based CC Management (DCCM) Algorithm

Based on the above discussion, the stochastic game $\mathcal{G}$ can be described as a finite MDP. Therefore, the NE at each state can be found by using the Markov properties. In this

subsection, we derive a multi-agent DRL-based scheme to tackle the MDP corresponding to the game $\mathcal{G}$.

We describe the MDP by using the discrete state space $\mathcal{S}$ and action space $\mathcal{A}^k, \forall k \in \mathcal{K}$, in the game $\mathcal{G}$. The former is used for the environment states, and the latter is used for the possible actions of any UE k. To describe the reward function, we should consider whether the actions are applied or not. Specifically, a not applied action corresponds to an action that a posterior action has overridden, which becomes an applied action otherwise [31]. Accordingly, we define τ_k and we have $\tau^k = 0$ if $\widehat{T}^k - T_f^k \leq \mathrm{T_{cca}}$; otherwise $\tau^k = 1$ [31]. So, we restate the reward function for UE k by,

$$\eta_k(t) = \begin{cases} c_p^k & \text{if } \tau^k(t-1) = 1, \\ U^k(t) - \beta^k \times C^k(t) & \text{otherwise}, \end{cases} \tag{4.16}$$

where c_p^k is the amount of penalty imposed on UE k if its previous action is not applied and $U^k(t)$ and $C^k(t)$ are given in (4.10) and (4.11), respectively. Finally, for the state transition probability, we use $P_{ss'}(\boldsymbol{\pi})$ where $\boldsymbol{\pi} = (\pi^1, \ldots, \pi^K)$ is the vector of policies under which the UEs take their action.

Let us assume a stochastic environment where the UEs do not know enough about the environment. Additionally, the UEs aim to maximize their long-term reward. As mentioned in [61], we can use RL to find the optimal policy for the users. In [8], based on the mentioned distributed characteristics, the collaborative multi-agent RL with local states is employed. Specifically, let π^{*k} denote the optimal policy for UE k. π^{*k} is a function from state space $\mathcal{S}$ to the action space for UE k, i.e., $\pi^{*k} : \mathcal{S} \to \mathcal{A}^k$. Based on (4.6), the delay for each UE should be known to select the next state. Let us define the state-value function for the UE k as the accumulative expected discounted reward for the UE starting from a given state [61]. Therefore, at a given state $s \in \mathcal{S}$, a UE (aiming to maximize its state-value function) should learn $a^{*k} = \pi^{*k}(s)$ where $\pi^{*k}(s) \in \mathcal{A}^k$. In what follows, we prove Lemma 4.1 to obtain the optimal policy for each UE k recursively.

Lemma 4.1. Let us denote $\boldsymbol{\pi}^* = (\pi^{*k}, \boldsymbol{\pi}^{*-k})$ as the optimal policy vector for the UEs. Additionally, let $V^k(s, \pi^{*k}, \boldsymbol{\pi}^{*-k})$ and $Q^{*k}(s, a^k)$ be the the optimal state-value function and optimal action-value function for the UE k at state s for an action a^k, respectively.

a) The NE point for the game $\mathcal{G}$ at state s is obtained by addressing the following MDP problem [33],

$$V^k(s, \pi^{*k}, \boldsymbol{\pi}^{*-k}) = \max_{a^k \in \mathcal{A}^k} Q^{*k}(s, a^k), \tag{4.17}$$

b) Based on (4.17), the optimal policy for a UE k, π^{*k}, can be found recursively as follows [62],

$$Q^k(s,a^k) \leftarrow Q^k(s,a^k) + \delta[u^k(s,a^k,\boldsymbol{\pi}^{-k}) + \lambda \max_{a'^k \in \mathcal{A}} Q^k(s',a'^k) - Q^k(s,a^k)]. \tag{4.18}$$

where $Q^k(s,a^k)$ is the action-value function for UE k at state s for an action a^k, s' is the next state, $u^k(s,a^k,\boldsymbol{\pi}^{-k}) = E[\eta^k(s,a^k,\boldsymbol{\pi}^{-k})]$, and $E[.]$ is used for the expectation operator, and δ is the learning rate to update $Q^k(s,a^k)$.

Proof. See Appendix B.2 □

The updating scheme in (4.18) is the Q-learning algorithm proposed in [62]. This algorithm uses the available information s, a^k, s', $u^k(s,a^k,\boldsymbol{\pi}^{-k})$ to find the optimal policy for each UE k, and therefore transition probability at each state would not be required. However, as already mentioned, as the number of UEs and the number of CCs grow, the state space and the action space dimensions increase exponentially. Therefore, it becomes challenging to find the optimal policy by using (4.18). We adopt the DRL-based scheme proposed in [58], namely the DDQN algorithm, to address this issue.

Let us denote the parametrized action-value function for a UE $k \in \mathcal{K}$ by $Q^k(s,a^k,\boldsymbol{\theta})$. Additionally, we assume the information in tuple $[s, a_m, u_m(s,a_m), s']$ is given. The authors in [58] used a DNN to approximate the action-value function and proposed the DDQN algorithm to update the parameter $\boldsymbol{\theta}$. In the DDQN algorithm, the action selection is decoupled from the action evaluation. This prevents instability and overoptimistic value estimation. Let us denote the online and target network parameters with $\boldsymbol{\theta}$ and $\boldsymbol{\theta}^-$, respectively. As shown in [58], the target for DDQN and consequently the parameter updating function are given by, (4.19) and (4.20), respectively, as follows:

$$Y^k = u^k(s,a^k) + \lambda Q^k(s', \arg\max_{a' \in \mathcal{A}^k} Q^k(s',a',\boldsymbol{\theta}), \boldsymbol{\theta}^-), \tag{4.19}$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \zeta(Y^k - Q^k(s,a^k,\boldsymbol{\theta}))\nabla_{\boldsymbol{\theta}} Q^k(s,a^k,\boldsymbol{\theta}). \tag{4.20}$$

Below, we first complete our discussion by introducing the epsilon-greedy algorithm. This algorithm is used as the policy to select the action (i.e., to activate/deactivate the CCs) for each UE at each state. Then, employing the equations (4.19) and (4.20) and the introduced policy function, we derive a DRL-based algorithm to solve the formulated optimization problem in (4.14).

Algorithm 4.1 Proposed **DCCM** algorithm

Input: The action set for each UE, i.e., $\mathcal{A}^k, \forall k \in \mathcal{K}$; The vector of QoS requirement for the UEs; The assigned PCC to each UE;

Output: Optimal policy for activating/deactivating the CCs for each UEs;

Initialize: Experience memory $\mathcal{D}$, online network parameters $\boldsymbol{\theta}^k, k \in \mathcal{K}$, target network parameters $\boldsymbol{\theta}_k^- = \boldsymbol{\theta}^k, k \in \mathcal{K}$; $C_U \leftarrow 0$;

procedure
  for episode $= 1 : \mathrm{T_{cca}} : \mathrm{T_{sim}}$ **do**
    Initialize the network state s;
    for step $= 1 : T$ **do**
      Each UE $k \in \mathcal{K}$ approximates action-value $Q^k(s, a^k, \boldsymbol{\theta}^k)$, $\forall a^k \in \mathcal{A}^k$ by employing online network;
      Given state s, each UE $k \in \mathcal{K}$ adopts a^k by using (4.21) (epsilon-greedy policy) where $Q^k(s, a) \leftarrow Q^k(s, a, \boldsymbol{\theta}^k), \forall a \in \mathcal{A}^k$;
      All users get the next state s' via the message passing. Set $s \leftarrow s'$;
      Each user $k \in \mathcal{K}$ stores the transition $(s, a^k, u^k(s, a^k), s')$ in memory $\mathcal{D}$;
      Each user $k \in \mathcal{K}$ samples random mini-batch of transitions $(s, a^k, u^k(s, a^k), s')$ from $\mathcal{D}$;
      The parameters for online network, $\boldsymbol{\theta}^k, k \in \mathcal{K}$, is updated using (4.20); $C_U \leftarrow C_U + 1$;
      if $C_U == N$ **then**
        $\boldsymbol{\theta}_k^- = \boldsymbol{\theta}^k, k \in \mathcal{K}$; $C_U \leftarrow 0$;
      end if
    end for
  end for
end procedure

In [8], the epsilon-greedy policy is used to select an action at each state s. Specifically, given $Q^k(s, a^k), \forall a^k \in \mathcal{A}^k$ for UE k, the action is adopted by employing epsilon-greedy policy given as,

$$a^k = \begin{cases} \arg\max_{a \in \mathcal{A}^k} Q^k(s, a) & \text{with probability of } 1 - \epsilon, \\ \text{chosen randomly from } \mathcal{A}^k \text{ with probability of } \epsilon. \end{cases} \tag{4.21}$$

Our DDQN-based CC management scheme **DCCM** is depicted in **Algorithm 4.1**.

In **DCCM** algorithm, at a state s, each UE $k \in \mathcal{K}$ estimates the action-value $Q^k(s, a^k, \boldsymbol{\theta})$, $\forall a^k \in \mathcal{A}^k$ at each step. The UEs use the estimated action values, and by applying the epsilon-greedy algorithm, the action a^k is adopted by each UE $k \in \mathcal{K}$. The chosen action

a^k is pertinent to activate/deactivate the CCs for UE k. Then, given a^k, UE k obtains its delay. The gNB uses the delay for all UEs and checks whether the UE delay requirements are met or not, obtains the next state (i.e., s'), and broadcasts s' among the UEs. When the UEs obtain s' and their immediate reward [i.e., $u^k(s, a^k), \forall k \in \mathcal{K}$], the transition $[s, a^k, u^k(s, a^k), s']$ would be kept in the memory $\mathcal{D}$. Randomly sampling mini-batch from memory $\mathcal{D}$, the UEs use (4.20) and update the parameters for the online network, i.e., $\boldsymbol{\theta}$. When the algorithm stops, the optimal policy is obtained, corresponding to activating/deactivating the set of CCs for each UE.

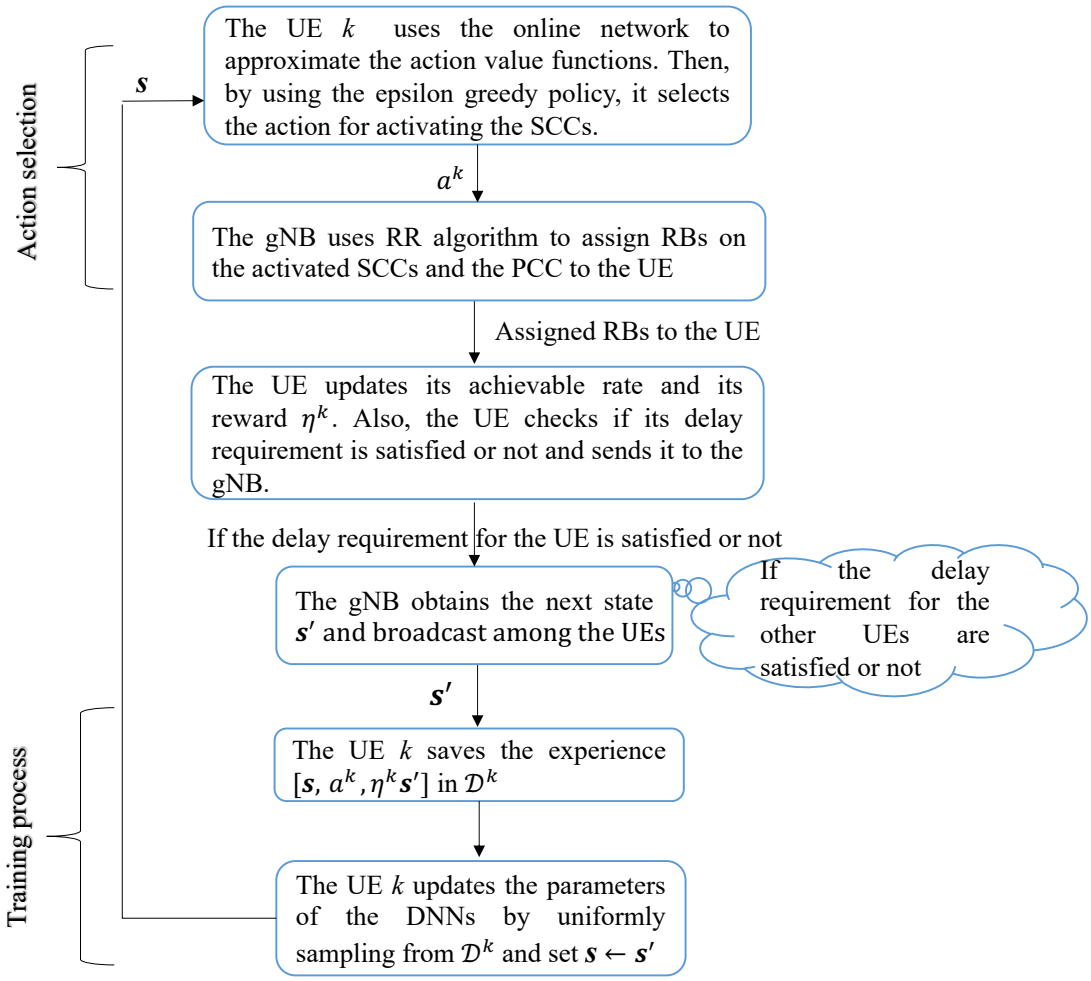

Figure 4.2: The message passing between the gNB and UE k for CA at a given time t .

4.2.3 Complexity Analysis

Based on **Algorithm 4.1**, the message passing between the gNB and UE k at a given time t is depicted in **Fig. 4.2**. Seen are two main parts: action selection and training the online DNN. The complexity analyses for each part are provided in what follows.

Action selection complexity: For a given gNB b, based on **Algorithm 4.1**, the action selection corresponds to activating/deactivating a SCC for the UE k. This would be the most complicated part of the algorithm.

Given the fully connected online DNN with a fixed number of hidden layers and neurons in each of them, for a given input, the computational complexity is pertinent to the sum of input and output sizes [101]. The input size for the online DNN equals the state vector dimension, i.e., K. So, for a given state and action, the complexity of obtaining the Q-value function is $O(K)$. Additionally, each UE k should select the Q-values corresponding to activating each CC. Therefore, the computational complexity for action selection corresponds to $O(K \times M)$.

Training process complexity: For this part, we should consider both forward and backward propagation complexities. Given T_k as the training batch to update the weights for the online DNN, the complexity for the forward propagation is $O(T_k \times K \times M)$. The complexity of the backward propagation algorithm corresponds to the product of the size of the input layer and the one for the output layer. For a given UE k, the size for the output of online DNN is K. Accordingly, the computational complexity for training procedure corresponds to $O(T_k \times K \times M)$.

4.2.4 Simulation Results

We consider a single-cell wireless network with one gNB. The cell's radius is 250 meters through which the UEs are distributed uniformly. Similar to initial 5G implementations, we consider that the CCs operate in two different bands- 800 MHz band and 3.5 GHz band [21] to improve the coverage and the throughput. Additionally, we employ 5G NR numerology $\mu_n = 0$ and utilise QPSK to 256 QAM for Modulation and Coding Scheme (MCS) as given in 3GPP standard [113]. The simulation parameters are given in Table 4.1.

The resources with 5G QoS characteristics have been adopted to evaluate our proposed model. In more detail, we consider FTP model 3 to model bursty traffic [120–123], and we assume the file arrives at the scheduler in one burst. In FTP model 3, the number of UEs is fixed, and each UE's files arrive with Poisson distribution. Thus, the inter-arrivals have exponential distributions; therefore, the file arrivals have Poisson distributions with λ_{T} inter-arrival rate. We also select a log-normal distribution for the file sizes [124]. For the file size distribution, let $\overline{q}^k$ denote the average file size for a UE k. Therefore, we have $\ln(\overline{q}^k)$ as a normal distribution $N(\mu, \sigma)$. Thus, the average file size (denoted by $\overline{q}^k$) for UE k will be obtained by $\overline{q}^k = \exp(\mu + 0.5 \times \sigma^2)$. Based on the value for μ and σ in Table 4.1, the average file size for each UE would be almost 0.5 Mbits. Additionally, the average file delay budgets for the UEs are 150 ms. We set $P_0 = 1530$ mW, $P_{idle} = 500$ mw, $P_{Rx,k,m} = 420$ mW, $K_r = 2.89$ mW/Mbps, $K_{r0} = -26.6$ mW, $K_s = -1.11$ mW/dBm, $K_{s0} = -60.7$ mW, $K_{BW} = 11.6$ mW/MHz and $K_{B0} = -229$ mW [21, 114].

Table 4.1: Network parameters and hyper-parameters for the CC management algorithms

Parameter	**Value**
Total transmit power for BS	30 dBm
Tcca, M, M_k	200 ms, 5, 5
Carrier Settings	CC1 and CC2 are low-band CCs (800 MHz) each of which has 5 MHZ bandwidth, CC3, CC4 and CC5 are high-band CCs (3.5 GHz) each of which having 10 MHZ bandwidth
PHY numerology	$\mu_n = 0$: 15 KHz sub-carrier spacing, PRB size of 12 sub-carrier (180 kHz), 1ms slot length
Traffic Model	3GPP FTP model 3 [120]
File arrival rate distribution	Poisson with $\lambda_{\mathrm{T}} = 10$
File size distribution	Log-normal with $[\mu, \sigma] = [13, 0.35]$ [124]
Number of episodes and steps	500, 500
Discount factor (λ)	0.9
epsilon in (4.21)	0.1
Learning rate (δ)	0.01
Replay memory ($\mathcal{D}$) size	500
Optimizer and Activation function	Adam and ReLU [33]
Solver	CPLEX [21]
Step size s_z in OCCM algorithm	0.2
Number of runs	48

In the simulations, the internal structure of DNN has an input layer, one hidden layer, and an output layer with K neurons, 256 neurons, and 2^{M} neurons, respectively. For the action value function and the weight updating process, we use the ReLU function [33] and Adam optimization approach to obtain the optimal gradient descent [33], respectively. To tackle the non-stationarity in our proposed multi-agent DDQN-based algorithm (**DCCM**), the experiences can be captured in the replay buffer and used to adjust the weight of previous experiences to current environment dynamics [125]. In [8], we uniformly sample from the replay buffer and adjust the hyper-parameters for our training method, the number of hidden layers, the number of neurons in each hidden layer, the parameter value for the epsilon-greedy algorithm and the learning rate λ, via the simulations. Because of space limitations, we do not present them here.

In the sequel, the convergence behavior of the reward function in the **DCCM** algorithm is evaluated. Next, we compare the performance of **DCCM** algorithm and **OCCM** algorithm with the performance of the algorithm where all CCs are activated for the UEs, namely, the activated all CCs (**AACCA**) algorithm. Additionally, we compare the performance of **DCCM** and **OCCM** algorithms with the algorithm where only one CC, i.e., the PCC, is activated to each UE, namely the only activated PCC (**OPCCA**) algorithm.

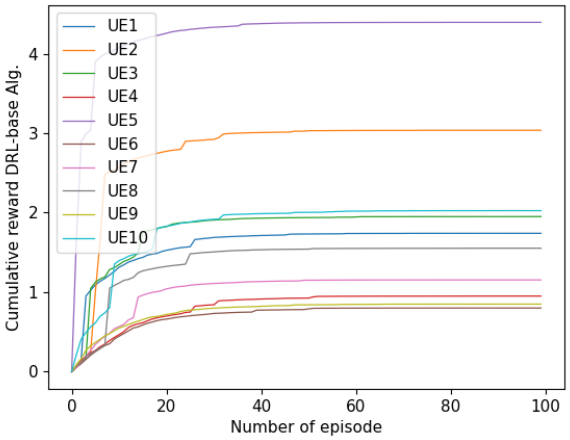

Figure 4.3: Convergence behavior for the reward function in (4.12) in **DCCM** algorithm vs number of episodes.

It is worth mentioning that the **OCCM** algorithm is run for at most 26 UEs due to its computational complexity. Meanwhile, the maximum number of UEs for studying the performance of the remaining algorithms is 38 UEs which corresponds to whether all UEs meet their target delay requirements or not.

Convergence Behaviour of the Reward Function in DCCM

In this subsection, considering 10 uniformly distributed UEs, we study the convergence behavior for the reward function in (4.12). As shown in Fig. 4.3, the reward function for each UE converges after 60 episodes. This would be because of using the DDQN algorithm proposed in [58] for deriving the **DCCM** algorithm. Specifically, the DDQN algorithm converges because learning is improved by using the target network and replay buffer, which prevent over-optimism and instability [58]. Moreover, it can be seen that the reward function for the UEs are not necessarily converging to a unique point. Since **DCCM** algorithm is a multi-agent DRL-based algorithm where each agent has its own DNN, the DNNs for the UEs have different weights. This can be explained as the main reason for the convergence behavior of the reward function for the UEs.

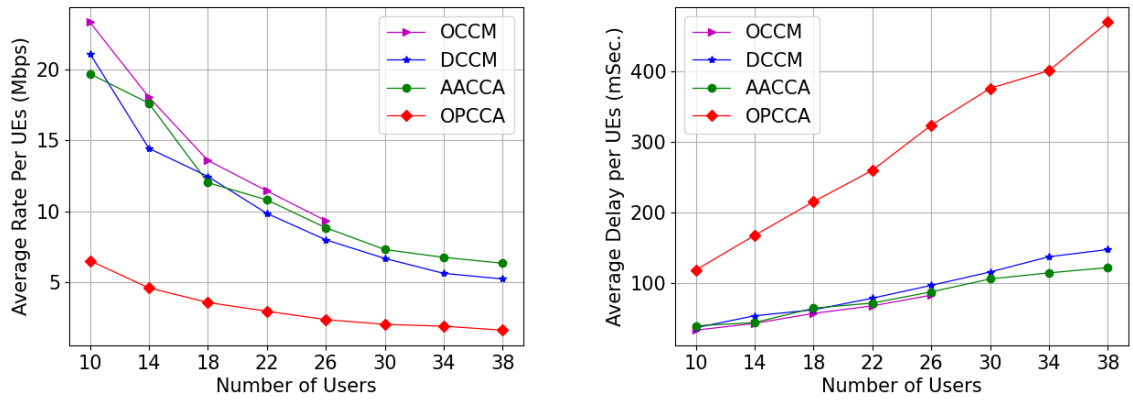

Figure 4.4: Average achievable rate per UE and average delay per UE versus the number of UEs.

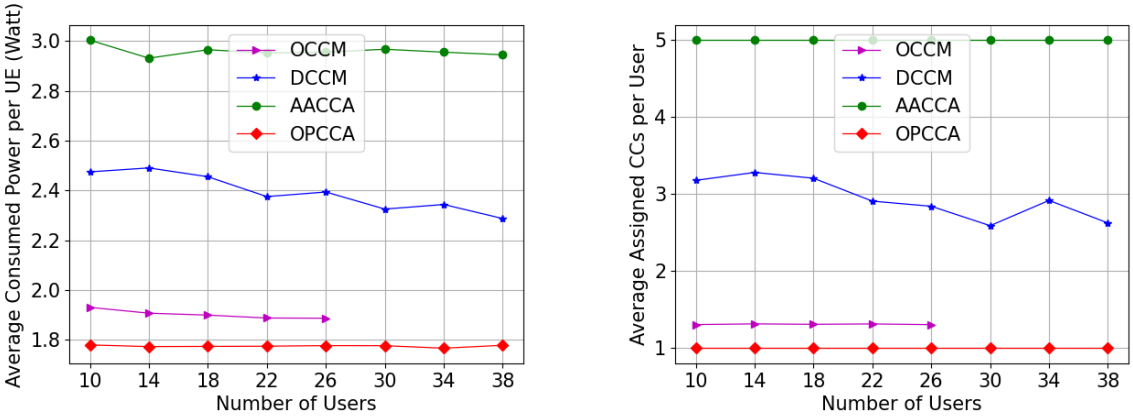

Figure 4.5: Average consumed power per UE and average number of assigned CCs per UE versus the number of UEs.

Performance Evaluation

In this subsection, we evaluate the performance of **DCCM** algorithm with those of **OCCM**, **AACCA**, and **OPCCA** algorithms. We consider varying number of UEs, i.e., $M \in \{10, \ldots, 38\}$ with step 4 UEs, and study the performance of the proposed algorithms versus the number of UEs. In the **DCCM** algorithm, each UE tries to minimize its delay and, at the same time, minimize its consumed power in a selfish manner. While, in **OCCM** algorithm, the average total delay for the UEs and the total consumed power for the UEs

are minimized. Figs. 4.4 and 4.5 illustrate the average achievable rate per UE, the average delay per UE, the average consumed power per UE, and the average number of activated CCs per UE, respectively. For the **DCCM** algorithm, the confidence interval (CI) for the achievable rate, delay, consumed power and number of activated CCs are very narrow. Due to the space limitation, we have not mentioned the corresponding values.

As illustrated in Fig. 4.4, the average achievable rate for each UE decreases as the number of UEs increases. It is also observed from Fig. 4.4 that the average delay per UE grows by increasing the number of UEs. It is noteworthy that **DCCM** performs close to the **OCCM** and the **AACCA** algorithms in terms of the rate and the delay, while the delay (and the rate) obtained from **OPCCA** is significantly higher than (and lower than) other algorithms.

In Figs. 4.5, it is shown that the average number of assigned CCs per UEs and consequently the amount of consumed power obtained from **OCCM** and **DCCM** are much lower than those for **AACCA**. Specifically, **OCCM** algorithm, being an optimal solution, performs better than **AACCA** and **DCCM**. It can be observed that the less the activated number of CCs, the less the consumed power. This is because, in (4.7), the dominant parameter is the bandwidth of CCs. Due to the activation of only one PCC for each UE, lower power consumption and number of activated CCs are achieved by the **OPCCA** algorithm.

In summary, the proposed DCCM algorithm could leverage the UEs' bursty traffic profile. Thus, it activates CC only during peak traffic to satisfy the QoS while minimizing the power consumption. This is unlike the AACCA, which always activates CCs even when there is no user traffic resulting in high power consumption. On the contrary, OPCCA activates a single carrier resulting in the lowest power consumption but a significant delay.

4.3 On Intelligent Energy-Aware Carrier Aggregation in 5G with Dual Connectivity

Aggregating multiple CCs from different frequency bands, CA, and Dual Connectivity (DC) by which can be concurrently transmitted and received from two nodes or cell groups, are employed in 5G and 6G wireless networks to enhance coverage and capacity. In the wireless networks with DC and CA, the network performance can be boosted by dynamically adjusting the UL transmit power level for the UEs and properly activating/deactivating the CCs for the UEs. In [9], we study the problem of joint dynamic UL power-sharing and CC management in the networks with DC and CA. The objective is to simultaneously minimize the delay and power consumption for the UEs. The pertinent problem is a multi-objective optimization problem with both discrete and continuous variables, and therefore is hard to solve. We first model it as a multi-agent RL system with compound action to handle the problem. Then, we employ a compound-action actor-critic algorithm to find the optimal policy and propose the Joint Power-Sharing and Carrier Aggregation (JPSCA) algorithm. The performance of the JPSCA algorithm is compared with two baseline algorithms. Our results show that the performance of the JPSCA algorithm in terms of the average rate, delay and UL transmit power level outperforms the baselines where UL power control and CC management are performed dis-jointly [9].

4.3.1 System Model and Problem Statement

In this section, we consider an EN-DC wireless network with CA technology. In CA, the bandwidth is expanded by aggregating some CCs in the same/different frequency bands. A CC consists of several RBs, and thus the problem of RB allocation and CA are coupled. The details for the network model and problem formulation are given in what follows. The notations are provided in Table 4.2.

Network and notations

In an EN-DC network consisting of an eNB and a set of B gNBs denoted by $\mathcal{B} = \{1, \dots B\}$, we consider the UL transmission where eNB and gNBs can use CA. Let us denote the set of non-overlapping and orthogonal CCs adopted by the eNB and the gNBs by $\widehat{\mathcal{M}} = \{1, \dots \widehat{M}\}$ and $\widetilde{\mathcal{M}} = \{\widehat{M}+1, \dots, \widehat{M}+\widetilde{M}\}$, respectively. Specifically, based on [126], we assume that $\widehat{M} = 1$, i.e., the eNB does not use the CA, and the CCs in $\widetilde{\mathcal{M}}$ are shared among the gNBs. Let $\mathcal{M} = \widehat{\mathcal{M}} \bigcup \widetilde{\mathcal{M}} = \{1, \dots 1+\widetilde{M}\}$. For a given $m \in \mathcal{M}$, we denote $\mathcal{N}_m = \{1, \dots, N_m\}$

Table 4.2: Table of notations

$\mathcal{B}, \mathcal{M}, \mathcal{M}^k$	Set of gNB, set of CCs, set of CCs for UE k
$\widehat{\mathcal{M}} = \{1\}, \widetilde{\mathcal{M}}$	Set of CC on LTE side, set of CCs on NR side
$\mathcal{K}, \mathcal{K}_b$	Set of UEs, set of UEs in gNB b
$b(k), \mathcal{N}_m$	Serving gNB for UE k, set of RBs in CC m
$\widehat{h}^k_{1,n}$	Path gain between the UE $k \in \mathcal{K}_b$ and eNB over a RB n in PCC (in the LTE side)
$\widetilde{h}^{b(k),k}_{m,n}$	Path gain between that UE and gNB $b(k)$ over a RB n in CC m (PCC or SCC at NR side)
$\widehat{p}^k, \widehat{p}^k_{1,n}$	UL transmit power levels for UE k to the eNB and the one for UE k on RB $n \in \mathcal{N}_1$
$\widetilde{p}^k, \widetilde{p}^k_{m,n}$	UL power levels for UE k to the gNB $b(k)$ and the one for UE k on RB $n \in \mathcal{N}_m$
$\widehat{R}^k, \widehat{R}^k_{1,n}$	UL rate for UE k achieved by transmitting to eNB and the one on RB $n \in \mathcal{N}_1$
$\widetilde{R}^k, \widetilde{R}^k_{m,n}$	UL rate for UE k achieved by transmitting to gNB $b(k)$ on the one on RB $n \in \mathcal{N}_m$
p^k, R^k	Total UL power for UE k, total rate for UE k
$\overline{D}^k, P^k$	Average delay and UL power consumption for UE k
$\mathrm{D}^k_{\mathrm{QoS}}, p^k_{\max}$	Max. tolerable delay and Max. UL power level for UE k
$\widehat{q}^k(t)$	Average number of bits per burst of data for UE k
$\alpha^k_m, \beta^k_{m,n}$	SCC activation indicator for UE k, and RB allocation indicator for UE k in CC $m \in \mathcal{M}$
$\widehat{r}^k, \widetilde{r}^k$	Continuous action used to adjust the UL power levels for UE k to the eNB and gNB $b(k)$

as the set of RBs in CC m. Let $\mathcal{M}^k = \{1, \dots M^k\}$ be the set of activated CCs for UE k. Specifically, $\mathcal{M}^k$ consists of both PCCs and SCCs for UE k. In EN-DC networks, there are actually two PCCs, one on the LTE side and the other one on the NR side. Therefore, we assume that the CC in the eNB is an always- activated PCC for UE k, and the other PCC is in its serving gNB.

Let us assume that a set of K UEs, $\mathcal{K} = \{1, \dots, K\}$, is distributed in the network area. We denote the serving gNB for UE k by $b(k)$. Indeed, UE k is in the coverage area for the gNB $b(k)$. Given $\mathcal{K}_b = \{1, \dots, K_b\}$ as the set of UEs in the coverage area for the gNB b, by employing DC technology, a UE $k \in \mathcal{K}_b$ can simultaneously transmit to both eNB and gNB b. Let $\widehat{h}^k_{1,n}$ and $\widetilde{h}^{b(k),k}_{m,n}$ be the path gain between the UE $k \in \mathcal{K}_b$ and eNB over a RB

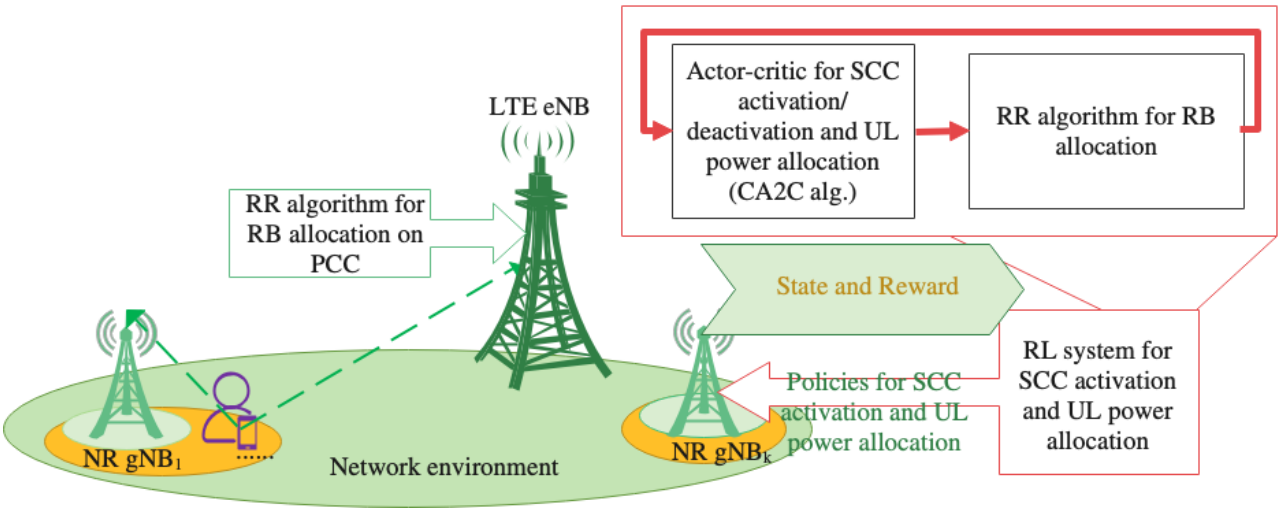

Figure 4.6: Intelligent CA and UL power allocation in the network with DC technology.

n in PCC (in the LTE side), and the path gain between that UE and its serving gNB $b(k)$ over a RB n in CC m (PCC or SCC at NR side), respectively. For any gNB b and the eNB, we assume that the channel between UE k and gNB b and the channel between the UE and the eNB have both small and large fading with path loss and shadowing. Also, the channels are the time-frequency-varying ones.

Our network model and proposed scheme have been briefly illustrated in Fig. 4.6. Specifically, to activate/deactivate the SCCs, adjust the UL transmit power level to the eNB and gNBs, and allocate the RBs, we develop a scheme with different execution intervals in [9]. CA and UL power allocation as the delay insensitive tasks are performed every $\mathrm{T_{cca}}$, whereas RB allocation, as a delay-sensitive task, is performed at every time interval t where $t \leq \mathrm{T_{cca}}$. Furthermore, RB allocation and CA are performed separately [10]. In [9], we derive a compound-action actor-critic-based algorithm for CA and UL power allocation. For RB allocation, we employ a round-robin algorithm. The details are given in Subsections 4.3.2.

We first introduce the model for UL transmit power level, UL power consumption and UL throughput. Next, using these models and the delay model derived in Subsection 4.2.1 [equations from (4.2) to (4.6)], we conclude this subsection by stating the CA and UL power allocation problem in EN-DC networks.

UL transmit power model for UE

In time t, we denote the total transmit power level for UE k by $p^k(t)$:

$$p^k(t) = \widehat{p}^k(t) + \widetilde{p}^k(t). \tag{4.22}$$

In (4.22), $\widehat{p}^k(t)$ and $\widetilde{p}^k(t)$ are the UL transmit power level for UE k to the eNB and its serving gNB, respectively. Specifically, we have $\widehat{p}^k(t) = \sum_{n\in\mathcal{N}_1} \beta^k_{1,n}\widehat{p}^k_{1,n}(t)$ and $\widetilde{p}^k = \sum_{m\in\widetilde{\mathcal{M}}} \alpha^k_m \sum_{n\in\mathcal{N}_m} \beta^k_{m,n}\widetilde{p}^k_{m,n}(t)$ where α^k_m is the SCC activation indicator for UE k and $\beta^k_{m,n}$ is the RB allocation indicator for UE k in CC $m \in \mathcal{M}$. Additionally, for convenience, from here, we assume that for the always-activated PCC m at NR side we have $\alpha^k_m = 1$. The total UL transmit power level for the UE k is bounded, i.e.,

$$p^k(t) \le p^k_{\max}, \tag{4.23}$$

where $p^k_{\max}$ is the maximum UL transmit power level for UE k.

UL power consumption model for UE

For studying the power consumption in UL transmission, the authors in [127] employ the model developed in [128]. We use it as well, which is given by,

$$P^k(t) = \begin{cases} \alpha_L p^k(t) + \beta_L & \text{if } p^k(t) \le \Gamma, \\ \alpha_H p^k(t) + \beta_H & \text{if } p^k(t) > \Gamma. \end{cases} \tag{4.24}$$

In (4.24), the parameters α_L, α_H, β_L, β_H, and Γ are the device-based ones and depend on the operating frequency band.

UL throughput model for UE

At a given time t, let $\widehat{\gamma}^k_{1,n}(t)$ denote the SINR for the UE k with eNB over RB n in PCC. Therefore we have

$$\widehat{\gamma}^k_{1,n}(t) = \frac{\widehat{p}^k_{1,n}(t)\widehat{h}^k_{1,n}(t)}{\sigma^2}, \tag{4.25}$$

where σ^2 is the noise. It is noteworthy that the RBs in the low-band PCC are exclusively allocated to the UEs, and thus the UEs do not interfere with each other through the RBs

in the PCC. Additionally, $\widetilde{\gamma}^k_{m,n}(t)$, the SINR for the UE k with its gNB (i.e., $b(k)$) over RB n in CC m (PCC or SCC), is given by,

$$\widetilde{\gamma}^k_{m,n}(t) = \frac{\widetilde{p}^k_{m,n}(t)\widetilde{h}^{b(k),k}_{m,n}(t)}{\sum\limits_{k' \in \mathcal{K}, k' \neq k} \beta^{k'}_{m,n}\widetilde{p}^{k'}_{m,n}(t)\widetilde{h}^{b(k),k'}_{m,n}(t) + \sigma^2}. \tag{4.26}$$

Accordingly, employing the Shannon theorem, we denote the achievable rate for UE k on RB n in PCC by $\widehat{R}^k_{1,n}$ and formally state them as,

$$\widehat{R}^k_{1,n}(t) = \widehat{B}_{1,n} \log_2 \left(1 + \widehat{\gamma}^k_{1,n}(t)\right), \tag{4.27}$$

where $\widehat{B}_{1,n}$ is the bandwidth for RB n in PCC. Similarly, the achievable rate for UE k on RB n in CC m at the NR side is denoted by $\widetilde{R}^k_{m,n}$ and expressed as,

$$\widetilde{R}^k_{m,n}(t) = \widetilde{B}_{m,n} \log_2 \left(1 + \widetilde{\gamma}^k_{m,n}(t)\right) \tag{4.28}$$

where $\widetilde{B}_{m,n}$ is the bandwidth for RB n in CC m. Let $\widehat{R}^k(t)$ and $\widetilde{R}^k(t)$ be the UL rate for UE k achieved by transmitting to eNB and gNB, respectively. By denoting $R^k(t)$ as the total achievable rate for UE k, we have,

$$R^k(t) = \widehat{R}^k(t) + \widetilde{R}^k(t), \tag{4.29}$$

where $\widehat{R}^k(t) = \sum\limits_{n \in \mathcal{N}_1} \beta^k_{1,n}\widehat{R}^k_{1,n}(t)$ and $\widetilde{R}^k(t) = \sum\limits_{m \in \widetilde{\mathcal{M}}} \alpha^k_m \sum\limits_{n \in \mathcal{N}_m} \beta^k_{m,n}\widetilde{R}^k_{m,n}(t)$.

Problem statement

In CA, a UE sends and receives through multiple CCs, PCCs and SCCs, from a single node, eNB or gNB. Meanwhile, EN-DC enables a UE to send and receive its data concurrently through the CCs from a master eNB and a secondary gNB.

In [9], the problem of joint UL power control and CA in 5G networks with EN-DC is addressed. For the UL transmission, based on [129], we describe the scenario where the eNB selects a CC in a low-frequency band, and a gNB selects three CCs in the mid-band frequency. There are two PCCs for a UE activated from the eNB and its serving gNB, respectively, and we use RR algorithm to allocate the RBs in the PCCs. A UE with a rate-hungry application may not meet its QoS (delay) requirement by only transmitting through the PCCs. Thus, SCC(s) from a gNB may need to be activated. We assume the

gNBs use the same spectrum for their CCs. Therefore, the UEs transmitting through their activated CCs from gNBs could interfere. Thus, the UL transmit power levels through the PCCs and the SCCs should be adjusted carefully.

As mentioned before, in the networks supporting EN-DC technology, two different schemes, i.e., DPS and EPS, can be used to adjust the UL transmit power level. In DPS, the UL transmit power is dynamically calibrated. While in ESP, the UL transmit power level is equally divided between the eNB and the gNB. EPS may degrade the network performance because the time-varying characteristics for the channels and UE traffic in 5G networks are not taken into consideration [95]. Thus, the network performance (network throughput) would degrade [111]. Based on a study performed in [111], the DPS technique offer 40% more of the average total throughput than ESP. Therefore, we focus on DSP and state the following multi-objective optimization problem for the CA, RB allocation and UL power control,

$$
\begin{array}{lll}
\textbf{min.} & \text{Total delay for the UEs in } \mathcal{K}_b & \forall b \in \mathcal{B} \\
\textbf{min.} & \text{Total UL power consumption for the UEs in } \mathcal{K}_b & \forall b \in \mathcal{B}, \\
\textbf{s.t.} & \text{QoS (delay) requirement in (4.6)}, & \forall k \in \mathcal{K}, \\
 & \text{UL total transmit power level in (4.23)}, & \forall k \in \mathcal{K}, \\
 & \text{Number of CCs}, & \\
 & \text{Number of RBs in CC } m & \forall m \in \mathcal{M}, \\
\textbf{var.} & \text{CC activation and RB allocation indicators}, & \\
 & \text{UL transmit power level for UE } k \text{ to eNB and gNB } b & \forall b \in \mathcal{B}, \forall k \in \mathcal{K}_b
\end{array}
\tag{4.30}
$$

In (4.30), $\mathcal{B}$, $\mathcal{K}_b$, $\mathcal{M}$ are the set of gNBs, set of UEs in gNB b, and the set of CCs, respectively. The above problem is a multi-objective optimization problem. Specifically, each gNB tries to minimize the total delay for the UEs in its coverage area [obtained from (4.4)] and the total power consumption of the UEs in $\mathcal{K}_b$ for activating/deactivating the SCCs [obtained from (4.24)]. This multi-objective optimization problem has two sets of integer (binary) and continuous variables. Therefore, it is a mixed-integer programming multi-objective optimization problem and is hard to solve. The integer variables correspond to CC management and RB allocation. As mentioned in [10], the CC management and RB allocation can be performed jointly or dis-jointly. This motivates us to perform CC management and RB allocation separately. We specifically focus on UL power control and CC management and employ the RR algorithm for RB allocation. To address the problem of UL power control and CC management, we develop a multi-agent RL system with a compound-action space (which contains both continuous action and discrete action) and

propose **Algorithm 4.4** to solve it. The details for deriving the algorithm are given in the next section.

4.3.2 Intelligent CA and UL Power-Sharing: A Compound-Action Actor-Critic Approach

In [9], considering the main objective of the optimization problem (4.30) for a given gNB b which minimizes the total delay for the UEs in $\mathcal{K}_b$ and at the same time minimizes the power consumption for the UEs in $\mathcal{K}_b$ to activate the SCCs, we use the framework in [60], CA2C method, to derive a resource management scheme for joint CA and UL power control in EN-DC wireless networks.

The proposed multi-agent RL system with compound-action space

As aforementioned, based on 3GPP, for UL transmission in 5G networks with EN-DC technology, we consider the scenario wherein only one CC in low-band would be activated by the eNB. We assume that this CC is the always-activated PCC. Each UE has another always-activated PCC at the NR side. Additionally, the SCC would be activated by gNBs for their associated UEs (i.e., the ones in their coverage area). As depicted in Fig. 4.6, the RR algorithm is used for allocating the RBs on the always-activated PCCs. Furthermore, a multi-agent CA2C-based algorithm is employed to activate the SCCs and adjust the UL transmit power level for both gNBs and eNB. Like the one employed to allocate the RBs in PCCs, the RR algorithm is used by each gNB to allocate the RBs in the activated SCCs to the UEs. To jointly perform CC activation and UL power adjustment, we develop a multi-agent RL system with a compound action space. The details for the set of agents, reward function, state space and action space are given below.

At a given time t, let us denote the immediate reward for a given gNB $b \in \mathcal{B}$ by $\eta_b(t)$. We define $\eta_b(t)$ in terms of the total delay and the total UL power consumption for the UEs in $\mathcal{K}_b$:

$$\begin{aligned}\eta_b(t) = -\Big(\sum_{k\in\mathcal{K}_b}\frac{\widehat{q}^k(t)}{R^k(t)} + \omega_b\sum_{k\in\mathcal{K}_b}P^k(t)\Big) = \\ -\Big(\sum_{k\in\mathcal{K}_b}\frac{\widehat{q}^k(t)}{\sum\limits_{n\in\mathcal{N}_1}\beta^k_{1,n}\widehat{R}^k_{1,n}(t) + \sum\limits_{m\in\widetilde{\mathcal{M}}}\alpha^k_m\sum\limits_{n\in\mathcal{N}_m}\beta^k_{m,n}\widetilde{R}^k_{m,n}(t)} + \omega_b\sum_{k\in\mathcal{K}_b}P^k(t)\Big).\end{aligned} \tag{4.31}$$

In (4.31), ω_b is the unit price for the total amount of power consumed by the UEs assigned to the gNB b for activating the SCCs. Let Φ_b be the long-term reward for gNB b. Φ_b can be expressed as the weighted sum of the short-term reward $\eta_b(t)$ as follows,

$$\Phi_b = \sum_{t=0}^{T-1} \lambda^t \eta_b(t). \tag{4.32}$$

In (4.32), we have $0 \leq \lambda \leq 1$. Specifically, if $\lambda = 0$ then $\Phi_b = \eta_b(0)$; otherwise, by the passing of time, the weighted immediate reward (i.e., $\lambda^t \eta_b(t)$) becomes smaller with negligible impact on long-term reward Φ_b.

Motivated by the objective function in optimization problem (4.30), we address the problem where each gNB b tries to maximize its long-term reward, which is given in (4.32). This problem can be restated as a stochastic game, and the MDP can be used to address the game, i.e., finding the Nash equilibrium at each state. However, complete knowledge about the environment would not be available in 5G network (e.g., the transition probability between the states in the MDP). We formally express the MDP as a multi-agent model-free RL system to tackle this issue. The authors also used this method in [1,33]. To address the RL system, we employ the CA2C algorithm proposed in [60].

Set of agents and reward function: The set of gNBs, $\mathcal{B} = \{1, \ldots, B\}$, is the set of agents. The immediate and long-term reward functions for each agent b are given in (4.31) and (4.32), respectively.

State space: In our proposed multi-agent RL system, all agents have the same state space given in terms of the UE delay requirements. Let us denote the state space by $\mathcal{S}$ and define it as,

$$\mathcal{S} = \Big\{ \mathbf{s}(t) \mid \mathbf{s}(t) = [s^k(t)]_{k \in \mathcal{K}} \;\text{ and }\; s^k(t) \in \{0, 1\} \Big\}. \tag{4.33}$$

Specifically, $s^k(t) = 1$ if the target delay requirement in (4.6) is satisfied for UE k. Otherwise, i.e., if (4.6) is not satisfied for UE k, we have $s^k(t) = 0$. Based on (4.33), we have $\mid \mathcal{S} \mid = 2^K$, and therefore the dimension of state space, $\mid \mathcal{S} \mid$, grows exponentially when the number of the UEs, K, increases.

Action space: For a given gNB b (agent b) in the multi-agent RL system, the action corresponds to activating/deactivating the SCCs for the UEs covered by the gNB and adjusting its assigned UE UL transmit power levels to gNB b and eNB. Let $\mathcal{A}_b$ be the action space for gNB b. So, we have

$$\mathcal{A}_b = \Big\{ \mathbf{a}_b = \Big(\boldsymbol{\alpha}_b(t), \mathbf{r}_b(t) \Big) \mid \boldsymbol{\alpha}_b(t) = [\alpha_m^k(t)]_{k \in \mathcal{K}_b, m \in \widetilde{\mathcal{M}}} \;\text{ and }\; \mathbf{r}_b(t) = \Big[\Big(\widehat{r}^k(t), \widetilde{r}^k(t) \Big) \Big]_{\forall k \in \mathcal{K}_b} \Big\} \tag{4.34}$$

In (4.34), $\boldsymbol{\alpha}_b(t)$ is composed of zero and one. We have $\alpha_m^k = 1$ if the SCC m is activated for UE $k \in \mathcal{K}_b$ and $\alpha_m^k = 0$, otherwise. Therefore, $\boldsymbol{\alpha}_b(t)$ is a discrete action. On the other hand, $\mathbf{r}_b(t)$ is the action for adjusting the UL transmit power level for the UEs in $\mathcal{K}_b$ to gNB b and eNB. Hence, $\mathbf{r}_b(t)$ is a continuous action. Given $\Big(\boldsymbol{\alpha}_b(t), \mathbf{r}_b(t)\Big)$, the UL transmit power levels for the UE to the eNB, and the gNB b are presented by

$$\begin{cases} \widehat{p}^k(t) = \widehat{r}^k(t), \quad \widetilde{p}^k(t) = \widetilde{r}^k(t) & \text{if } \widehat{r}^k(t) + \widetilde{r}^k(t) \leq p_{\max}^k, \\ \widehat{p}^k(t) = \min(\widehat{r}^k(t), \dfrac{p_{\max}^k}{2}), \quad \widetilde{p}^k(t) = \min(\widetilde{r}^k(t), \dfrac{p_{\max}^k}{2}) & \text{if } \widehat{r}^k(t) + \widetilde{r}^k(t) > p_{\max}^k. \end{cases} \tag{4.35}$$

Based on the above discussion, for the gNB b, the action space $\mathcal{A}_b$ consists of both discrete and continuous actions and thus is a compound-action space with a high dimension. Thus, the compound-action and state spaces will be huge for our proposed multi-agent RL system. To address it, we employ the framework developed in [60].

The proposed compound-action actor-critic algorithm

Taking advantage of the DDPG algorithm [64] and DQN algorithm [63], the authors in [60] derived a CA2C method to learn the optimal policy. In this subsection, by employing the CA2C algorithm in [60], we propose a learning approach to derive the optimal policy in order to handle both continuous and discrete action, i.e., adjusting the UL transmit power levels of each UE to the gNB and eNB, respectively, and activating/deactivating the SCCs for the UEs.

Let $\boldsymbol{\pi} = (\boldsymbol{\pi}_1, \dots \boldsymbol{\pi}_B)$ be the policy profile for the gNBs. Specifically, for a given gNB b, $\boldsymbol{\pi}_b$ is a function from a given state $\mathbf{s}$ to action $\mathbf{a}_b$, i.e., $\boldsymbol{\pi}_b : \mathcal{S} \mapsto \mathcal{A}_b$. So, the policy profile $\boldsymbol{\pi}$ corresponds to patterns of behavior for the gNBs at the different states of the environment, and thus, their policy should be optimized to maximize the long-term reward for the gNBs [given in (4.32)]. Let $\boldsymbol{\pi}^* = (\boldsymbol{\pi}_1^*, \dots \boldsymbol{\pi}_B^*)$ be the optimal policy profile for the gNBs. To find the optimal policy for a given gNB b, both current and future rewards for the agent should be considered. For a given state $\mathbf{s}$ and action $\mathbf{a}_b$, let $Q_b(\boldsymbol{\pi}_b, \boldsymbol{\pi}_{-b}, \mathbf{s}, \mathbf{a}_b)$ be the Q-function for gNB b where $\boldsymbol{\pi}_b$ is the policy for agent (gNB) b, and $\boldsymbol{\pi}_{-b} = (\boldsymbol{\pi}_1, \dots, \boldsymbol{\pi}_{b-1}, \boldsymbol{\pi}_{b+1} \dots \boldsymbol{\pi}_B)$ is the policy for the others. For ease of reference, from here on, we use the notation $Q_b(\mathbf{s}, \mathbf{a}_b)$ for gNB b Q-function. Specifically, $Q_b(\mathbf{s}, \mathbf{a}_b)$ is defined in terms of the expectation of the weighted sum of the short-term reward for the agent [61]. Based on what is discussed in [60] and [1], the optimal policy for an agent gNB b is the policy under which the Q-function for the agent is maximized, i.e.,

$$\mathbf{a}_b^* = \pi_b^*(\mathbf{s}) = \arg \max_{\mathbf{a}_b \in \mathcal{A}_b} Q_b^*(\mathbf{s}, \mathbf{a}_b), \qquad \forall \mathbf{s} \in \mathcal{S}. \tag{4.36}$$

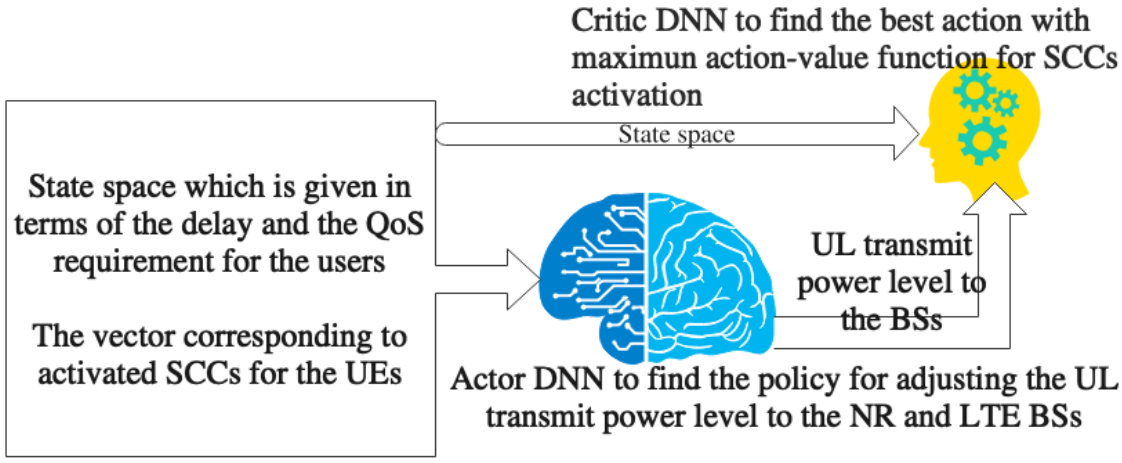

Figure 4.7: Actor-critic-based method for training the DNNs used to approximate Q_b^* and $\boldsymbol{\nu}_b^*$ for an agent b.

The action space is composed of continuous and discrete actions in the above problem. We use the CA2C algorithm in [60] to find the optimal policy. The details are given in what follows.

For a given gNB (agent) b, let us decompose the optimal policy $\boldsymbol{\pi}_b^*$ into two optimal policies to adjust the UL transmit power levels to eNB and gNB b and activating/deactivating SCCs, respectively. For a gNB b, given state $\mathbf{s}$ and discrete action $\boldsymbol{\alpha}_b$ (which corresponds to activating the SCCs for the UEs in $\mathcal{K}_b$), let $\boldsymbol{\nu}_b^*(\mathbf{s}, \boldsymbol{\alpha}_b)$ denote the optimal policy to adjust the UL transmit power levels to the LTE and NR BSs. The best action taken to activate the SCCs for the UEs in $\mathcal{K}_b$ is obtained as follows [60],

$$\alpha_b^* = \arg \max_{\boldsymbol{\alpha}_b \in \mathcal{A}_b} Q_b^* \Big(\mathbf{s}, (\boldsymbol{\alpha}_b, \boldsymbol{\nu}_b^*(\mathbf{s}, \boldsymbol{\alpha}_b)) \Big) \tag{4.37}$$

In (4.37), finding the exact value for Q_b^* and $\boldsymbol{\nu}_b^*$ is challenging because the action and state space are high dimensional. To address this issue, we use the DNN to approximate them. We employ the CA2C algorithm in [60] wherein the DQN and DDPG algorithms are employed to train the corresponding DNNs separately. The details are provided below.

As illustrated in Fig. 4.7, for a given agent b, two separated DNNs, i.e., actor DNN with parameter $\boldsymbol{\theta}_b$ and critic DNN with parameter $\mathbf{w}_b$, are used to approximate the Q_b^* and $\boldsymbol{\nu}_b^*$, respectively. Let us denote the parametrized Q-function and the parametrized policy for adjusting the UL transmit power level to the eNB and gNB b by $Q_b(\mathbf{s}, \mathbf{a}_b, \mathbf{w}_b)$

Algorithm 4.2 CA2C-based Algorithm for UL Power-Sharing and CA: Critic Network Training Algorithm

Input: A sample experience $\mathbf{e}_b^t = [\mathbf{s}, \mathbf{a}_b, \eta_b, \mathbf{s}']$, online network parameter $\boldsymbol{\theta}_b$, target network parameter $\boldsymbol{\theta}_b^-$, parameter τ;

Output: Updated parameters for critic online networks and critic target networks;

1: **procedure**

2: Given $\mathbf{s}'$ and $\boldsymbol{\alpha}_b(t)$ (which corresponds to activating SCCs), obtain the output for the actor online network, i.e., $\mathbf{r}_b(t) = \left[\left(\widehat{r}^k(t), \widetilde{r}^k(t)\right)\right]_{\forall k \in \mathcal{K}_b}$ and calculate $\mathbf{p}_b(t) = \left[\left(\widehat{p}^k(t), \widetilde{p}^k(t)\right)\right]_{\forall k \in \mathcal{K}_b}$ using (4.35);

3: Given $\mathbf{r}_b(t)$ and $\mathbf{s}'$ as input to critic target network, obtain the Q-function and update $\boldsymbol{\alpha}_b(t)$ based on (3.58);

4: Given $\boldsymbol{\alpha}_b(t)$ and $\mathbf{s}'$ as input to actor target network, update $\mathbf{r}_b(t)$ and $\mathbf{p}_b(t)$;

5: Obtain the target value for the critic as the summation of $\eta_b(t)$ and the Q-function calculated by the critic's target network, which is multiplied by the discount factor;

6: Employing optimizer (e.g., Adam algorithm), update the parameters for the critic online network using (A.9b);

7: Update the parameter for critic target network as $\mathbf{w}_b^- = \tau \mathbf{w}_b^- + (1 - \tau)\mathbf{w}_b$.

8: **end procedure**

and $\boldsymbol{\nu}_b(\mathbf{s}, \boldsymbol{\alpha}_b, \boldsymbol{\theta}_b)$, respectively. For gNB b (agent b), the state $\mathbf{s}$ and $\boldsymbol{\alpha}_b$ (which are pertinent to the activated SCCs for the UEs in $\mathcal{K}_b$) are first given as the input to the actor DNN. The actor DNN approximates $\boldsymbol{\nu}_b(\mathbf{s}, \boldsymbol{\alpha}_b, \boldsymbol{\theta}_b)$ to obtain the continuous action $\mathbf{r}_b(t)$. Then, given state $\mathbf{s}$ and $\mathbf{r}_b(t)$ to the critic network, the Q-function for agent b, $Q_b(\mathbf{s}, \mathbf{a}_b, \mathbf{w}_b)$, is approximated. Based on (3.58), $Q_b(\mathbf{s}, \mathbf{a}_b, \mathbf{w}_b)$ is used to activate the SCCs for the UEs in $\mathcal{K}_b$.

Training procedure: By following the procedure illustrated in Fig. 4.7 and using (A.9a) and (A.9b), we modify the algorithm for training the actor and critic networks in **Algorithm 2.1** and **Algorithm 2.2**, respectively, and derive **Algorithm 4.2** and **Algorithm 4.3** as follows.

The proposed CA2C-based algorithm: Now, by using the training algorithm for the actor and critic networks, i.e., **Algorithm 4.2** and **Algorithm 4.3**, we derive a CA2C-based algorithm to activate the SCCs and adjust the UL transmit power level for each UEs to its serving gNB and eNB. We call it the Intelligent *Joint power-sharing and*

Algorithm 4.3 CA2C-based Algorithm for UL Power-Sharing and CA: Actor Network Training Algorithm

Input: A sample experience $\mathbf{e}_b^t$, online network parameter $\mathbf{w}_b$, target network parameter $\mathbf{w}_b^-$, parameter τ;

Output: Updated parameters for actor online networks and actor target networks;

1: **procedure**
2: Calculate the gradient of function $J_b(\boldsymbol{\theta}_b)$ with respect to continuous action in all sampled experience form replay buffer $\mathcal{D}_b$ (by using equation (28) in [60]);
3: Using the actor optimizer (e.g., Adam algorithm), update actor online parameter based on (A.9a);
4: Update the parameter for actor target network as $\boldsymbol{\theta}_b^- = \tau\boldsymbol{\theta}_b^- + (1-\tau)\boldsymbol{\theta}_b$.
5: **end procedure**

CA (JPSCA) algorithm. It is noteworthy that to make a trade-off between exploration and exploitation, given the continuous action, we use the $\epsilon-$greedy algorithm to adopt the discrete action in the critic side, which is given by,

$$\boldsymbol{\alpha}_b = \begin{cases} \arg\max\limits_{\boldsymbol{\alpha}_b \in \mathcal{A}_b} Q_b(\mathbf{s}, \mathbf{a}_b, \mathbf{w}_b) & \text{with the probability of } 1-\epsilon, \\ \text{randomly adopted from } \mathcal{A}_b & \text{with probability of } \epsilon. \end{cases} \tag{4.38}$$

Accordingly, the details for the Intelligent JPSCA algorithm are given in **Algorithm 4.4**.

Algorithm 4.4 Proposed **JPSCA** algorithm

Input: The action set for each gNB b, i.e., $\mathcal{A}_b,\ \forall b \in \mathcal{B}$, target delay requirement for the UEs in $\mathcal{K}_b$;

Output: Optimal policy for activating/deactivating the SCCs for the UEs in $\mathcal{K}_b$ and adjusting UL transmit power level for the UEs to their serving gNB b and eNB;

Initialize: Experience memory $\mathcal{D}_b, b \in \mathcal{B}$, actor online network parameters $\boldsymbol{\theta}_b, b \in \mathcal{B}$, actor target network parameters $\boldsymbol{\theta}_b^- = \boldsymbol{\theta}_b, b \in \mathcal{B}$, critic online network parameters $\mathbf{w}_b, b \in \mathcal{B}$, critic target network parameters $\mathbf{w}_b^- = \mathbf{w}_b, b \in \mathcal{B}$;

procedure
  for episode $= 1 : \mathrm{T_{cca}} : \mathrm{T_{sim}}$ **do**
    Initialize the network state $\mathbf{s}$;
    for step $= 1 : T$ **do**
      Given state $\mathbf{s}$ and the continuous action $\mathbf{r}_b$, each gNB $b \in \mathcal{B}$ uses its critic online network and approximates action-value $Q_b(\mathbf{s}, \mathbf{a}_b, \boldsymbol{\theta}_b), \forall \boldsymbol{\alpha}_b \in \mathcal{A}_b$ and adopts $\boldsymbol{\alpha}_b$ by using (4.38);
      Given state $\mathbf{s}$ and the discrete action $\boldsymbol{\alpha}_b$ (corresponds to SCC activation for the UEs in $\mathcal{K}_b$), $\mathbf{r}_b$ is obtained, which is used to adjust the UL transmit power level for the UEs in $\mathcal{K}_b$ to the NR and LTE BSs by using (4.35);
      RR algorithm is used by eNB and gNBs for RB allocation to the UEs sharing the SCCs and PCC;
      The next state $\mathbf{s}'$ is obtained via the message passing and $\eta_b,\ \forall b \in \mathcal{B}$ is obtained by each gNB b;
      Store experience $\mathbf{e}_b^t = [\mathbf{s}, \mathbf{a}_b, \eta_b, \mathbf{s}']$ in memory $\mathcal{D}_b$ for $b \in \mathcal{B}$;
      Each gNB $b \in \mathcal{B}$ samples random mini-batch of transitions $[\mathbf{s}, \mathbf{a}_b, \eta_b, \mathbf{s}']$ from $\mathcal{D}_b$;
      The parameters of the DNNs for each gNB b are updated by using **Algorithm 4.2** and **Algorithm 4.3**;
      Set $\mathbf{s} \leftarrow \mathbf{s}'$;
    end for
  end for
end procedure

4.3.3 Complexity Analysis

Based on **Algorithm 4.4**, at a given time t, the message passing between a gNB b and its served UEs in $\mathcal{K}_b$ is depicted in **Fig. 4.8**. For CA and DPS, we have two main parts: action selection and training the actor and critic DNNs. The complexity analyses for each part are provided in the following.

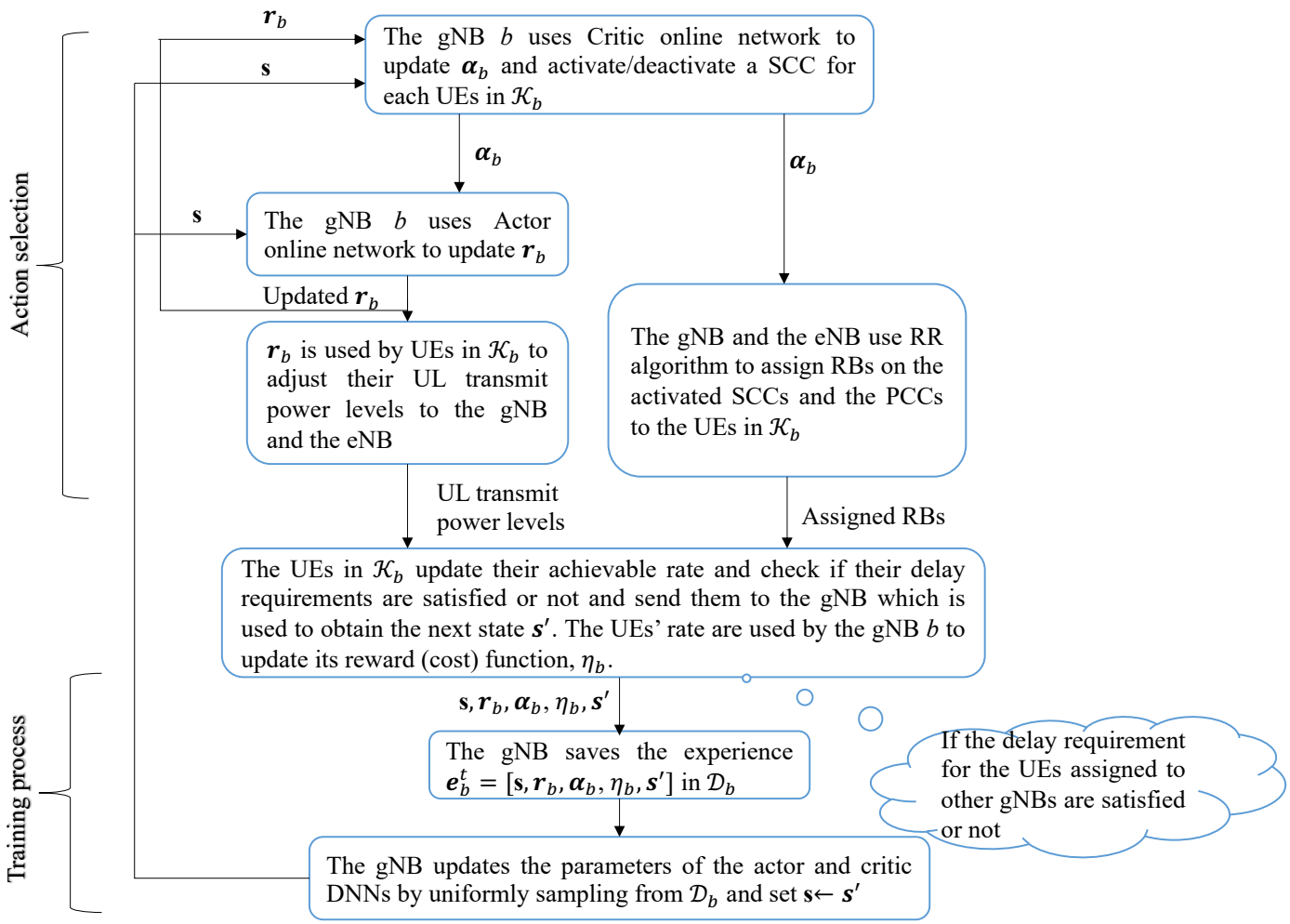

Figure 4.8: The message passing between a gNB b and its assigned UEs in $\mathcal{K}_b$ for CA and DPS at a given time t .

Action selection complexity: For a given gNB b, based on **Algorithm 4.4**, the action selection corresponds to activating/deactivating a SCC and adjusting the UL transmit power level for each UE $k \in \mathcal{K}_b$ to the gNB and the eNB. This would be the most complicated part of the algorithm.

Given the fully connected actor and critic DNNs with a fixed number of hidden layers and neurons in each of them, for a given input, the computational complexity is pertinent to the sum of input and output sizes [101]. Based on equations, (4.36) and (4.33), the

input size for the actor and critic DNNs are $K + K_b \times \widetilde{M}$ and $K + 2 \times K_b$, respectively. The term $K_b \times \widetilde{M}$ corresponds to that each UE $k \in \mathcal{K}_b$ employs a one-hot vector to activate a SCC. The complexity for calculating the Q-value function is $O(K + K_b \times \widetilde{M})$. Each gNB should select the Q-value among all CCs. Therefore, the computational complexity for action selection corresponds to $O(K \times \widetilde{M}^2)$.

Training process complexity: In this part, the complexity for the forward and backward propagations should be obtained. Let T_b be the training batch for updating the weights for the actor and critics DNNs. Based on the above discussion, the forward propagation complexity would be $O(T_b \times (K \times \widetilde{M}^2))$. For a given gNB b, based on equation (28) in [101], the Q-value function is needed to update the weights of the actor and critic DNNs is needed. Additionally, the complexity of the backward propagation algorithm corresponds to the product of the size of the input layer and the one for the output layer. For a given gNB b, the size for the outputs of actor and critic DNNs are $2 \times K_b$ and $K_b \times \widetilde{M}$, respectively. Wherefore, the computational complexity for the training procedure corresponds to $O(T_b \times K^2 \times \widetilde{M})$.

4.3.4 Simulation Results

As depicted in Fig. 4.6, we consider an EN-DC network consisting of one eNB and K number of UEs. Additionally, $B = 5$ gNBs are randomly inserted in the coverage area of the eNB. The eNB has a circular coverage area with a radius of 500 meters, and the circular coverage area for each gNB has a radius of 50 meters. A given gNB b can serve a number of K_b UEs uniformly distributed over its coverage area where $2 \leq K_b \leq 5$. Based on [126] for our setup, only one CC in low-band, 800 MHz, can be activated by the eNB for the UEs. We consider this CC the always-activated PCC for the UEs at the LTE side. Also, for each gNB, three CCs operate in mid-band frequency (3.5 GHz). Two of these CCs are considered SCCs that can be activated/deactivated at any time, and one of them is the always-activated PCC for the UEs at the NR side. For the DNNs, an input layer, three hidden layers and an output layer are considered. The number of neurons in the hidden layers is $128, 512$ and 1024, respectively. The optimal gradient descent algorithm, ReLU activation function and Adam optimization are used to update the weights for the DNNs. Through the simulations, we set the hyper-parameters, e.g., ϵ and learning rate ζ. The simulation parameters and the structure for the DNN are given in Table 4.3.

For the simulation, FTP traffic, i.e., FTP model 3 is employed [121] to model the bursty traffic, and we assume the file arrives at the scheduler in one burst. In FTP model 3, the number of UEs is fixed, and each UE's files arrive with the Poisson distribution.

Table 4.3: Simulation parameter and hyper-parameters for the learning algorithms

Parameter	Value
Max. UL transmit power level [126]	26 dBm
Average target delay [8]	150 msec
$\mathrm{T}_{\mathrm{cca}}$ [8]	200 msec
$\widehat{M}$ and $\widetilde{M}$ [126]	1, 3
Carrier Setting [8]	CC1 is the CC in low-band frequency (800 MHz) with 5 MHz bandwidth CC2 and CC3 are the CCs in mid-band frequency (3.5 GHz) with 10 MHz bandwidth
PHY numerology [8]	15 kHz sub-carrier spacing, PRB size of 12 sub-carrier (180 kHz), 1ms slot length
Traffic Model and parameters [8]	3GPP FTP model 3, $[\mu, \sigma] = [12.5, 0.35]$, $\lambda_{\mathrm{T}} \in \{10, 15, 20\}$
Number of episodes and steps	100, 100
Discount factor(λ), ϵ [8]	0.9, 0.1
Learning rate (ζ), τ [8]	0.01, 0.01
Replay memory size [8]	500
Optimizer and activation function [8]	Adam and ReLU
Device parameters for mid-band frequency $(\alpha_L, \beta_L, \alpha_H, \beta_H, \Gamma_{\mathrm{D}})$ [128]	14.25 mW/dbm, 2.1625 W, 117.5 mW/dbm, 1.22 W, 16 dbm
Device parameters for low-band frequency $(\alpha_L, \beta_L, \alpha_H, \beta_H, \Gamma_{\mathrm{D}})$ [128]	7.5 mW/dBm, 1.325 W, 10^{-6} mW/dBm, 2.3 W, 16 dBm

Thus, the inter-arrivals have exponential distributions. Therefore, the file arrivals have Poisson distributions with λ_{T} inter-arrival rate. Additionally, for the file size, a log-normal distribution [124] with parameters μ as mean, and σ as standard deviation is selected. That is, given $\overline{q}^k$ as the average file size for a UE k, $\ln(\overline{q}^k)$ has a normal distribution $N(\mu, \sigma)$ and we have $\overline{q}^k = \exp(\mu + 0.5 \times \sigma^2)$.

In this section, we evaluate the performance of the **JPSCA** algorithm in comparison with the following baseline algorithms: **ACPS** (All-activated CCs and power-sharing) algorithm and heuristic **AEPS** (All-activated CCs with Equal power-sharing) algorithm. All CCs are activated for the UEs in both baseline algorithms. In **ACPS**, the DDQN algorithm [58] is used to adjust the UL transmit power for each UE to the BSs. The details for the **ACPS** algorithm are given in **Appendix B.3**. Note that for the **AEPS** algorithm, the UL transmit power level for the UE $k \in \mathcal{K}_b$ to its serving gNB b and that to the eNB are the same.

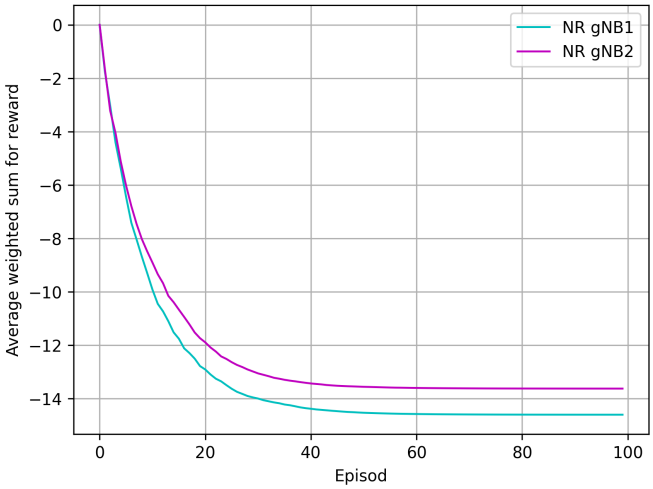

Figure 4.9: Convergence behavior for CA2C-based (JPSCA) algorithm.

4.3.5 Convergence Behaviour of JPSCA Algorithm

To study the convergence behavior for the reward function of a given gNB by using the proposed **JPSCA** algorithm, we consider a scenario with an eNB and two gNBs. The network has six UEs, uniformly distributed in the coverage area. **JPSCA** algorithm is a CA2C-based algorithm. The convergence of the CA2C algorithm has been proven in [60]. Hence, using the **JPSCA** algorithm, the average weighted reward function for each gNB converges to a stationary point. This is shown in **Fig. 4.9**, as well. It is noteworthy that the DNNs for different agents (gNBs) do not necessarily have the same weights. Therefore, the stationary point for the average weighted reward function for each gNB would not be the same because each gNB uses its DNNs to adjust the UL transmit power level and activating/deactivating the SCCs for the UEs in its coverage area.

4.3.6 Performance Evaluation

To evaluate the performance of **JPSCA** algorithm, we compare it with the performance of **ACPS** and **AEPS** algorithms in terms of the average rate per UE, average delay per UE, average number of activated CCs per UE, average UL consumed power per UE, and average UL transmit power levels for each UE to its assigned gNB and eNB which are illustrated in **Figs. 4.10**, **4.11**, **4.12**, **4.13**, **4.14a**, **4.14b**, respectively. In the simulations, we have considered $B = 5$, and different number of UEs, i.e., $K \in \{10, 15, 20, 25\}$. The limitation of the number of gNBs and the UEs in simulation results is due to using an actor-critic

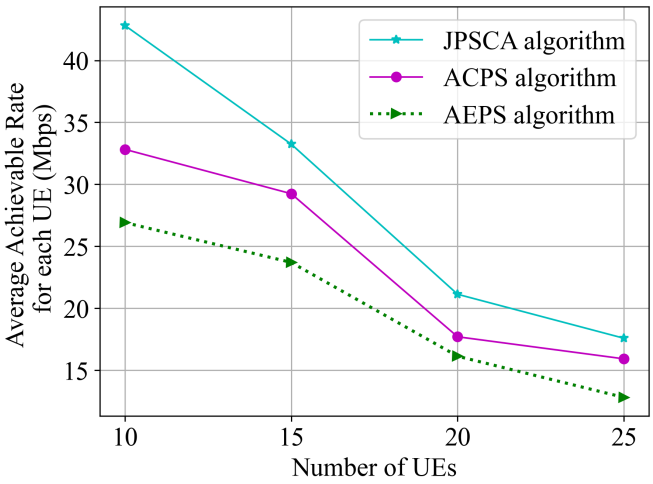

Figure 4.10: Average rate per UE for CA2C-based (JPSCA) algorithm vs. all CCs activated DDQN-based (ACPS) algorithm, and all CCs activated static power-sharing (AEPS) algorithm.

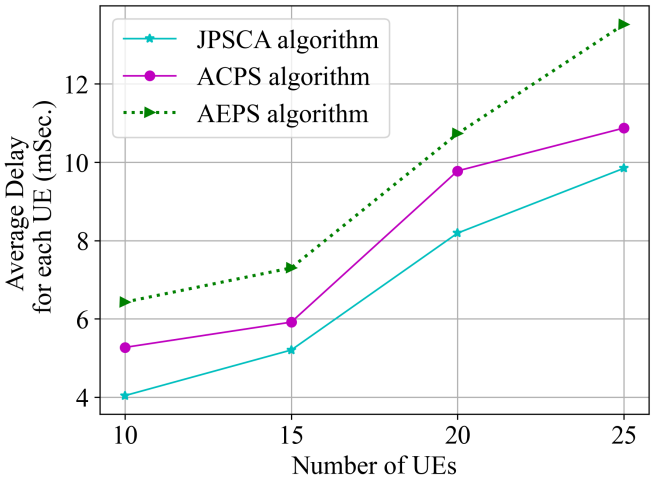

Figure 4.11: Average delay per UE for CA2C-based (JPSCA) algorithm vs. all CCs activated DDQN-based (ACPS) algorithm, and all CCs activated static power-sharing (AEPS) algorithm.

algorithm. **JPSCA** algorithm is an actor-critic algorithm that can be slow for several reasons: the complexity of the algorithms, the size of the network, and the difficulty of optimizing the network parameters. Specifically, two separate networks - the actor network

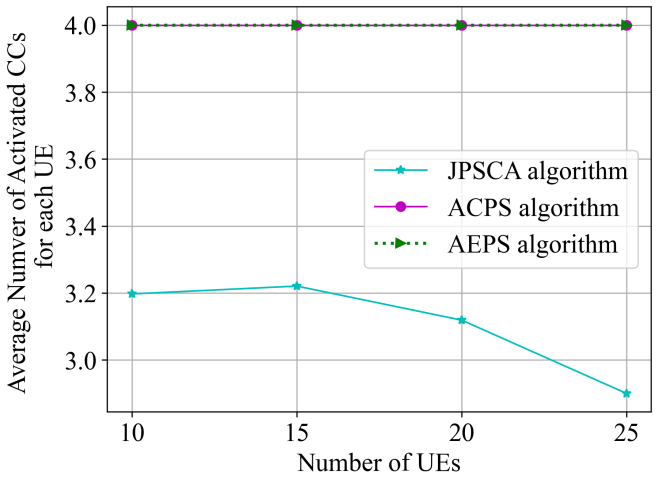

Figure 4.12: Average number of activated CCs per UE for CA2C-based (JPSCA) algorithm vs. all CCs activated DDQN-based (ACPS) algorithm and all CCs activated static power-sharing (AEPS) algorithm.

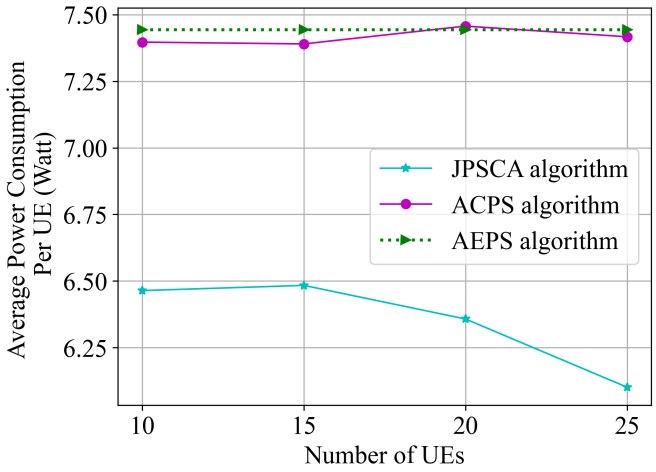

Figure 4.13: Average UL power consumption per UE for CA2C-based (JPSCA) algorithm vs. all CCs activated DDQN-based (ACPS) algorithm, and all CCs activated static power-sharing (AEPS) algorithm.

and the critic network - must be trained together, which can be computationally expensive and time-consuming. In addition, optimizing the network parameters can be challenging, especially in environments with large state and action spaces.

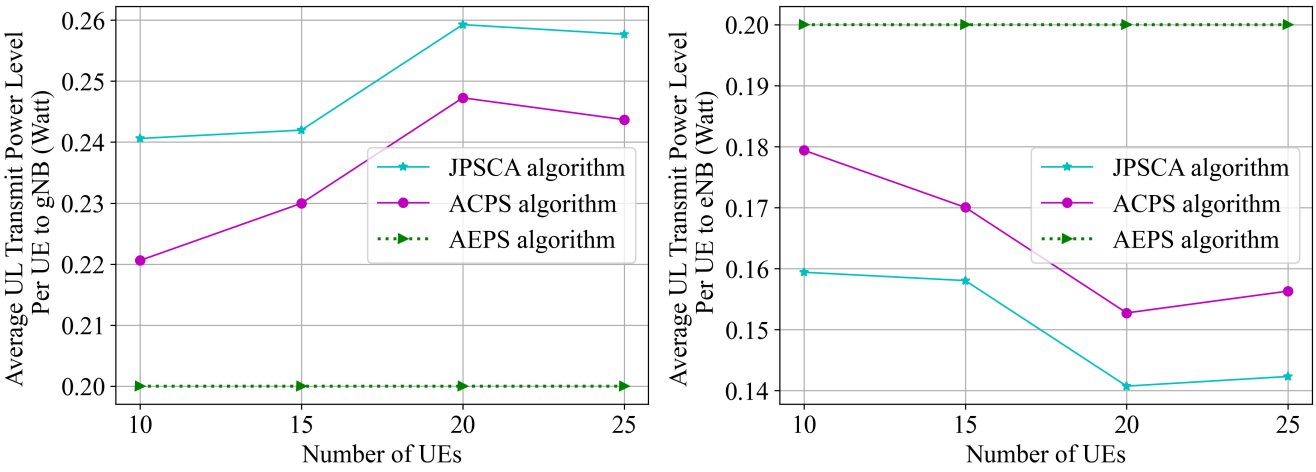

Figure 4.14: Average UL transmit power level per UE to gNB (a), and average UL transmit power level per UE to eNB (b) for CA2C-based (JPSCA) algorithm vs. all CCs activated DDQN-based (ACPS) algorithm and all CCs activated static power-sharing (AEPS) algorithm.

Based on **Fig. 4.10**, by employing **ACPS**, **AEPS** and **JPSCA** algorithms, the average achievable rate per UE decreases as the number of UEs increases. The more the number of UEs, the less the number of allocated RBs to each UE. The achievable rate per UE by employing **JPSCA** is higher than those obtained by using **AEPS** and **ACPS** algorithms. However, all CCs are activated for each UE in the baseline AEPS and ACPS. On the one hand, in **JPSCA** algorithm, CC management and UL power control are performed jointly. Additionally, by activating all CCs for all UEs in **AEPS** and **ACPS** algorithms, the interference through the RBs in CCs increases which could degrade the UE rates even more. It is notable that since the UL power control for each UE in **ACPS** algorithm is intelligently performed by using the DDQN algorithm, the performance of **ACPS** algorithm in terms of achievable rate for each UE is higher than that for **AEPS** algorithm where the UL transmit power levels for each UE to its serving gNB and the eNB are the same. Similarly, we can evaluate the performance of **AEPS**, **ACPS** and **JPSCA** algorithms in terms of the delay for each UE. Specifically, based on **Fig. 4.11**, as the number of UEs grows, the rate for each UE declines, but the delay per UE increases. However, the performance of the **JPSCA** algorithm in terms of the delay for each UE is better than **AEPS** and **ACPS** algorithms. For instance, for up to almost 25 UEs, the delay obtained by each of UEs using the **JPSCA** is around the target delay for the URLLC applications in 5G. This

can be because of intelligently joint power and CC management in **JPSCA** algorithm by using the CA2C algorithm. Meanwhile, in the **ACPS** algorithm, where the UL transmit power level per UE is intelligently adjusted, the delay per UE is lower than that for **AEPS** algorithm.

For the small cell NRs with limited resources, using the **JPSCA** algorithm for CA and UL power control results in better performance in terms of the average number of activated CCs and UL power consumption for each UE. According to **Figs. 4.12 and 4.13**, the average number of activated SCCs per UE and the average UL power consumption are lower than those obtained by employing **ACPS** and **AEPS** algorithms. This is because of intelligent joint CC management and UL power-sharing in **JPSCA** algorithm. Also, since the UL power-sharing in **ACPS** is performed using the DDQN algorithm, it performs better than the **AEPS** algorithm.

As observed in **Fig. 4.14a**, by using **JPSCA** algorithm, the UL transmit power level per UE to its serving gNB is higher than those of the **ACPS** and **AEPS** algorithms. Based on (4.24), the UL power consumption for a UE is a linear function over both UL transmit power level and the operating frequency. By using the **JPSCA** algorithm, as illustrated in Fig. 4.12, the number of activated SCCs per UE is reduced preventing the dramatic increase in the total UL power consumption for each UE. However, as the number of activated SCCs decreases, to intelligently make a trade-off between minimizing the delay and minimizing the UL power consumption, the UL transmit power level per UE to its serving gNB increases (which results in a more achievable rate for each UE).

Illustrated by **Fig. 4.14b**, as the UL transmit power levels for the UEs to gNBs increase, the UL transmit power level for the UEs to the eNB would decrease [based on constraint (4.23)] resulting in a lower total UL power consumption for each UE. For the number of 25 UEs, there is an increase in the UL transmit power level to the eNB per UE. For serving 25 UEs, compared with 20 UEs, using the RR algorithm results in fewer RBs allocated to each UE in the PCC. On the other hand, setting the delay degradation rate for 25 UEs to be the same as that for 20 UEs, it is necessary to prioritize minimizing the total UEs' delay concerning the UE power consumption. Thus, a trade-off between the UL power consumption and the delay for the UEs using the **JPSCA** algorithm occurs when the average number of activated SCCs at the NR side and the allocated RB at the LTE side decreases. This is obtained by increasing the UL transmit power level per UE to the eNB.

Chapter 5

Conclusion

With the advent of advanced technologies, e.g., MEC, CA and DC, in the next-generation wireless networks, high-dimensional resource management problems with combinatorial characteristics arise in a dynamic environment. Therefore, despite their advantages in boosting the achievable rate, providing the UEs with powerful computation resources at the edge, and enhancing the coverage, the technologies above pose tremendous challenges to resource management in the next-generation wireless networks. As a promising tool to tackle the high-dimensional and dynamic resource management problem, ML provides the network with significant enhancement and agility. In this thesis, having taken the next-generation wireless networks with MEC, CA and DC functionalities into consideration, we formally express the resource management problems in a dynamic environment as a stochastic game and use some concepts in DRL, e.g., DDQN and CA2C algorithms, to solve the formulated games. Several findings have been noted from this thesis, which can be summarized in the following points:

5.1 Intelligent Resource Allocation at Edge

In this thesis, we proposed a distributed multi-agent DRL-based method to jointly tackle the resource management and offloading problem in MEC-enabled wireless networks. We considered both partial and binary offloading schemes. The objective of the proposed scheme is to enhance the rate, and reduce both UL power and the cost of computation energy. The actions of the agents are simultaneously selected via message exchanges between agents. By this, a DDQN-based algorithm is proposed and shown to outperform other deep learning approaches under fixed and mobile users for large network sizes. Additionally, a

distributed multi-agent DRL-based method is proposed to tackle the problem of partial offloading of computational loads of VNFs belonging to an SFC in MEC-enabled wireless networks. Specifically, the VNFs in an SFC request can be performed locally or offloaded to a MEC server. The objective is to minimize the delay and energy of executing an SFC request. Finally, we proposed a multi-agent DRL-based algorithm to study the problem of joint power allocation and offloading in UAV-MEC-assisted smart farms. Focusing on the interdependent tasks performed by the UAVs and their topology, we aimed to minimize the ETC for each UAV subjected to the energy budget constraint for the UAVs. The combination of RL and GCN was employed to solve the problem.

5.2 Energy and Delay Aware Carrier Aggregation in 5G Networks

In 5G networks with CA, we proposed a multi-agent DDQN-based CC management algorithm. We aimed to minimize the delay for bursts of data and the UE power consumption. Then, we extended our work and studied the problem of dynamic power-sharing in EN-DC networks with CA technology. Specifically, the problem of minimizing the delay and UE power consumption is formally expressed. The optimizing variables are continuous and discrete variables corresponding to adjusting UE UL power levels and activating/deactivating the SCCs, respectively. To address this problem, a CA2C-based algorithm is proposed to jointly adjust the UL transmit power level for the UEs to the BSs and activate the SCCs. Such an intelligent joint resource management scheme performs better in power consumption and the number of activated CCs. Additionally, it has a better achievable rate and delay performance than all CCs activated algorithms where the UL power control and CA are performed separately.

5.3 Future Work

In near beyond 5G wireless networks, AI-enabled edge computing will play a pivotal role in supporting the emerging fully-autonomous Internet of Everything (IoE) services. Digital Twins (DTs) are candidate to enable the IoE applications. Therefore, a large excessive amount of complex real-world DTs are expected to emerge. This necessitates underlying communication networks with extreme reliability and data rates to represent the DTs a hi-fi model and sustain their required high QoS. To implement the DTs at the edge of

the network and meet their challenges, some advanced ML methods, e.g., Transfer Learning (TL) and Continual Learning (CL), should be resorted to adapt the digital duplicate seamlessly.

Transfer Learning refers to using the knowledge gained from one machine learning problem in another. It can be considered an application of pre-training (i.e., using the pre-trained DNNs in the ML model). Specifically, in untrained DNNs, the network parameters are initialized with random values. Then, these parameters are optimized iteratively to meet the desired objective. On the other hand, pre-training a neural network refers to training a model on one task or data set and then using the parameters or model from this training to train another model on a different task or data set, which gives the model a head start instead of starting from scratch. The most crucial aspect of pre-training neural networks is the task at hand. That is, the task from which the model initially learns must be similar to the task model used in the future. Therefore, the main component of transfer learning involves using pre-trained models to gather knowledge from one task and apply it to other tasks. By using the transfer learning AI, applications can be developed faster and in a more efficient way.

Furthermore, in DTs, triggering a duality that requires synchronization between the Physical Twin (PT) and the Cyber Twin (CT), high-fidelity of the CT in reflecting the status of PT, and history awareness of the DT states. The first feature guarantees to preserve the real-time interaction, and diminish the possibility of interrupting the operations and simulations for CT. The two last features provide a precise duplicate in the various phases experienced by the PT and knowledge attained from past experiences into future states, respectively. Accordingly, CL at the edge of the network, continuously operating in a dynamic non-stationary environment, can be employed to simultaneously maintain synchronous, accurate, and history-aware DTs.

Appendix A

A.1 Proof of Theorem 3.1

For a given user m, if $\alpha_m\beta_m = 00$ (the selected offloading mode is binary offloading with completely local computation), the immediate reward is given in terms of CPU frequency for that user, i.e., we have, $\eta_m(f_m) = \frac{Tf_m}{C} - w_m\Big(\tau_0 P_{r,m} + T\gamma_c f_m^3\Big)$. Therefore, to maximize $\eta_m(f_m)$, the value of $\eta_m(f_m)$ in critical points are obtained and thus we have, $f_m = \sqrt{(3\gamma_c C w_m)^{-1}}$. In this situation, the uplink transmit power level for offloading the computation task at any MEC server equals to zero.

For a user $m \in \mathcal{M}_2$[1], given the offloading scheme for the users (i.e., α_i and β_i for all users), the utility function for user m, i.e., η_m, would be given in (A.1)

$$\eta_m = \begin{cases} \frac{Tf_m}{C} + \frac{B\tau}{\nu_m}\log\Big(1 + \frac{h_m p_m}{\sum_{i>m}^{M}(\alpha_i+\beta_i)h_i p_i + \sigma_0^2}\Big) - \\ w_m\Big(\tau_0 P_{r,m} + \epsilon\tau(p_m + P_{c,m}) + T\gamma_c f_m^3(t)\Big) & \text{if user } m \in \mathcal{M}_2 \text{ partially offloads its task,} \\ \Big(\frac{B\tau}{\nu_m}\log\Big(1 + \frac{h_m p_m}{\sum_{i>m}^{M}(\alpha_i+\beta_i)h_i p_i + \sigma_0^2}\Big) - \\ w_m\Big(\tau_0 P_{r,m} + \epsilon\tau(p_m + P_{c,m})\Big) & \text{if user } m \in \mathcal{M}_2 \text{ completely offloads its task,} \end{cases} \tag{A.1}$$

where $\tau_0, P_{r,m}, P_{c,m}$, and τ are fixed values. To concurrently maximize the utility function for the users, since the UL transmit power level for a given user m and the CPU frequency

[1] $\mathcal{M}_2$ is the set of users either partially or completely offload their computation task onto MEC server

for that user are not coupled, the CPU frequency for user m, i.e., f_m, is obtained similar to that obtained for the situation where the computational task performed locally. That is, we have, $f_m = \sqrt{(3\gamma_c C w_m)^{-1}}$. To obtain the transmit power level for each user, we employ the game theory-based power control scheme in [118]. Specifically, given the CPU frequency for the users, the utility function η_m is based on that in reference [118]. In reference [118], the utility for each user is the difference between an increasing concave function over the transmit power level for that user [e.g., $\frac{B\tau}{\nu_m}\log\left(1+\frac{h_m p_m}{\sum_{i>m}^{M}(\alpha_i+\beta_i)h_i p_i+\sigma_0^2}\right)$ in (A.1)] and a linear function over the transmit power for the user [e.g., $w_m\Big(\tau_0 P_{r,m}+\epsilon\tau(p_m+P_{c,m})\Big)$ in (A.1)]. As mentioned in [118], the UL transmit power level for the users are obtained by finding the Nash Equilibrium (NE) for the game. Accordingly, in a system with a limited UL transmit power level for the users, the proposed power control scheme in [118] prevents users from unilateral deviation from the NE (e.g., greedily increasing the UL transmit power level); additionally, it limits the level of interference imposed from the other users to user m [118].

Similarly, we formulate the non-cooperative game $\mathcal{G}_2 = \langle \mathcal{M}_2, \mathcal{P}, \eta_m \rangle$ where $\mathcal{P} = \times_{m\in\mathcal{M}_2}\mathcal{P}_m$ and $\mathcal{P}_m = [0,\ldots,\overline{p}_m]$. In this game, for any user $m \in \mathcal{M}_2$, the strategy space $\mathcal{P}_m$ is compact, convex, and non-empty. Additionally, for a given user $m \in \mathcal{M}_2$, we have $\frac{\partial \eta_m}{\partial p_m} = \frac{B\tau}{\nu_m} \times \frac{h_m}{\sum_{i\in\mathcal{M}_2, i>m} h_i p_i + \sigma_0^2 + p_m h_m} - w_m \epsilon\tau$. For user $m \in \mathcal{M}_2$, by setting $w_m \leq \frac{B}{\nu_m \epsilon} \times \frac{h_m}{\sum_{i\in\mathcal{M}_2, i>m} h_i \overline{p}_i + \sigma_0^2 + \overline{p}_m h_m}$, we have $\frac{\partial \eta_m}{\partial p_m} \geq 0$ and thus η_m is increasing function over $\mathcal{P}_m$. Furthermore, we have $\frac{\partial^2 \eta_m}{\partial p_m^2} = \frac{B\tau}{\nu_m} \times \frac{-h_m^2}{(\sum_{i\in\mathcal{M}_2, i>m} h_i p_i + \sigma_0^2 + p_m h_m)^2}$; therefore, $\frac{\partial^2 \eta_m}{\partial p_m^2} \leq 0$ and thus η_m is a concave function over $\mathcal{P}_m$. That is, for any user $m \in \mathcal{M}_2$, η_m is a concave and increasing function over $\mathcal{P}_m$ and therefore, there is a NE point for $\mathcal{G}_2$ [100].

As mentioned in [100], we say $\mathbf{p}^* = [p_m^*]_{m\in\mathcal{M}_2}^{\mathrm{T}}$ is a NE if, for any given user $m \in \mathcal{M}_2$, we have $\eta_m(p_m^*, \mathbf{p}_{-m}^*) \geq \eta_m(p_m, \mathbf{p}_{-m}^*), \forall p_m \in \mathcal{P}_m$. To find the NE for this game, the best response for each user $m \in \mathcal{M}_s$ should be obtained [100] (i.e., by solving the equations $\frac{\partial \eta_m(\mathbf{p})}{\partial p_m} = 0, \forall m \in \mathcal{M}_2$). For a given user $m \in \mathcal{M}_2$, we set $\frac{\partial \eta_m(\mathbf{p})}{\partial p_m} = 0$ and therefore the proof is complete.

A.2 Unit price for consumed energy

Without loss of generality, to set a unit price for consumed energy for each user, we consider the situation wherein all users have adopted partial offloading scheme. Specifically, for a given user $m \in \mathcal{M}$, let f_m and p_m be the frequency of CPU and uplink transmit power level for offloading a task which are obtained from (3.22) and (3.23) respectively. Accordingly, based on (3.23), we have $\sum_{i \geq m}^{M} p_i h_i = \frac{B h_m}{\epsilon \nu_m w_m} - \sigma_0^2$. Let us assume, $m = M$. Then, $p_M = \frac{B}{\epsilon \nu_M w_M} - \frac{\sigma_0^2}{h_M}$. Since, $0 \leq p_M \leq \overline{p}_M$, we have,

$$LB_M \leq w_M \leq UB_M, \tag{A.2}$$

where $LB_M = \frac{B}{\epsilon \nu_M} \times \frac{h_M}{\overline{p}_M h_M + \sigma_0^2}$ and $UB_M = \frac{B}{\epsilon \nu_M} \times \frac{h_M}{\sigma_0^2 + h_M \overline{p}_M}$. If $m < M$, by using (3.23) and some calculations, we have, $p_m = \frac{B}{\epsilon \nu_m w_m} - \frac{B h_{m+1}}{\epsilon \nu_{m+1} w_{m+1} h_m}$ and thus,

$$LB_m \leq w_m \leq UB_m, \tag{A.3}$$

where $LB_m = \frac{B}{\epsilon \nu_m} \times \left(\overline{p}_m + \frac{B}{\epsilon \nu_m} \times \frac{h_{m+1}}{\nu_{m+1} w_{m+1} h_m} \right)^{-1}$ and

$$UB_m = \min\{ \frac{\nu_{m+1} w_{m+1} h_m}{\nu_m h_{m+1}}, \frac{B}{\nu_m \epsilon} \times \frac{h_m}{\sum\limits_{i \in \mathcal{M}, i > m} h_i \overline{p}_i + \sigma_0^2 + \overline{p}_m h_m} \}.$$

Additionally, for a given user $m \in \mathcal{M}$, since $0 \leq f_m \leq \overline{f}_m$, based on (3.22), we have $w_m \geq LBF_m$ where $LBF_m = (3\gamma_c \overline{f}_m^2 C)^{-1}$. Thus,

$$\max\{LB_m, LBF_m\} \leq w_m \leq UB_m, \tag{A.4}$$

where, for any given user $m \in \mathcal{M}$, LB_m and UB_m have been given in (A.2) and (A.3).

A.3 PORA algorithm

In this part, we implement Algorithm 3 in [4] and [18]. To make it comparable with our scheme, we relax the optimization problem (3.11) by considering only the partial offloading scheme. So, subject to the constraint of QoS requirement for a user $m \in \mathcal{M}$, we formulate

the optimization problem for concurrently maximizing the immediate reward (i.e., η_m) for each user $m \in \mathcal{M}$ as follows,

$$\max_{\mathbf{p},\mathbf{f}} \quad \eta_m \qquad \text{s.t} \qquad R_m \geq R_m^{\min}, \tag{A.5}$$

where $\eta_m = R_m - w_m\Big(\tau_0 P_{r,m} + \epsilon\tau(p_m + P_{c,m}) + T\gamma_c f_m^3\Big)$ and $R_m = \dfrac{Tf_m}{C} + \dfrac{B\tau}{\nu_m}\log\Big(1 + \dfrac{h_m p_m}{\sum\limits_{i>m}^{M}(\alpha_i + \beta_i)h_i p_i + \sigma_0^2}\Big)$. To address (A.5), employing the Lagrangian relaxation, we relax the problem as follows,

$$\max_{\mathbf{p},\mathbf{f}} \quad \eta_m + \lambda_m^L(R_m - R_m^{\min}) \qquad \forall m \in \mathcal{M}. \tag{A.6}$$

The above multi-objective optimization problem (A.6) can be addressed by using the approach in **Appendix A.1**. Thus we have,

$$f_m^* = \sqrt{\frac{1 + \lambda_m^L}{(3\gamma_c C w_m)}} \tag{A.7a}$$

$$p_m^* = \frac{B(1 + \lambda_m^L)}{\epsilon\nu_m w_m} - \frac{I_m(\mathbf{p}^*)}{h_m}, \tag{A.7b}$$

where $I_m(\mathbf{p}^*) = \sum\limits_{i>m, i\in\mathcal{M}} h_i p_i^* + \sigma_0^2$ and h_i is the path gain for user i. By using the above closed form expression and some matrix operations at the BS, the uplink transmit power vector for users, i.e., $\mathbf{p}$, are obtained by solving the following equation,

$$\mathbf{p} = \mathbf{H}^{-1}\mathbf{Z}, \tag{A.8}$$

where $\mathbf{Z} = \Big[\dfrac{B(1 + \lambda_m^L)}{\epsilon\nu_m w_m} - \dfrac{\sigma_0^2}{h_m}\Big]^{\mathrm{T}}_{m\in\mathcal{M}}$ and $\mathbf{H} = [h'_{im}]$ is a $\mid \mathcal{M} \mid \times \mid \mathcal{M} \mid$ matrix where $h'_{im} = \dfrac{h_i}{h_m}$ if $i \geq m$; otherwise, $h'_{im} = 0$. It is worth mentioning that in equation (A.8) and (A.7a), for a given user m, the transmit power level and the CPU frequency for that user are limited to the values $\overline{p}_m$ and $\overline{f}_m$, respectively. Accordingly, we propose partial offloading and resource allocation algorithm, **PORA** algorithm, where the CPU frequency for each user $m \in \mathcal{M}$ is obtained by (A.7a), and the UL transmit power level for the users are obtained by (A.8).

A.4 Updating Rule for the Weights of Actor and Critic DNNs

To train actor DNN and critic DNN, we employ the CA2C algorithm proposed in [60]. Since the structure of IoT application should be captured, actor DNN and critic DNN compose of embedding layers, hidden layers, and the output layer [69]. For a given UAV u, at time t, the input for its actor DNN is the state provided by (3.55), $\mathbf{S}_u^t = (\mathbf{e}^t, \mathbf{e}_{v_u^i})$[2], and the tasks' offloading scheme obtained from the output of the critic network [60]. For the critic DNN, the input is the $\mathbf{S}_u^t$ and the output for the actor DNN, i.e., the transmit power level for the UAV u for offloading a task [60]. Equation (3.56) is used as the propagation rule for the embedding layers of both actor DNN, and critic DNN [69,108]. In what follows, we explain the details for updating the DNNs' parameters.

To train the actor and critic DNNs, the DDPG and DQN algorithms in [64] and [63] are used, respectively [60], where the concept of replay buffer, target network and online network are used to prevent instability and over-optimism. Given $\boldsymbol{\theta}_u$ and $\mathbf{w}_u$ as the parameters for the actor and critic online networks, the experiences in the replay buffer $\mathcal{D}_u = \{\mathbf{x}_u^t\}$ stored as tuple $\mathbf{x}_u^t = [\mathbf{S}_u^t, \boldsymbol{\alpha}_u, \mathbf{S}'_u, \eta_u^t]$, are used for updating the online network parameters. In tuple $\mathbf{x}_u^t$, the element $\mathbf{S}'_u$ is the next state by taking action $\boldsymbol{\alpha}_u$ from state $\mathbf{S}_u^t$. The functions for updating the online network parameters are given as follows [60],

$$\boldsymbol{\theta}_u = \boldsymbol{\theta}_u + \zeta \nabla_{\boldsymbol{\theta}_u} J_u(\boldsymbol{\theta}_u), \tag{A.9a}$$

$$\mathbf{w}_u = \mathbf{w}_u - \zeta \nabla_{\mathbf{w}} L_u(\mathbf{w}_u), \tag{A.9b}$$

In the above equations, ζ is the learning rate, and $J_u(\boldsymbol{\theta}_u)$ and $L_u(\mathbf{w}_u)$ are the functions evaluated by the actor and critic networks, respectively. $J_u(\boldsymbol{\theta}_u)$ is given in terms of the average Q-function for UAV u, and $L_u(\mathbf{w}_u)$ is given in terms of the average difference between the Q-function and target value for agent u. The details for functions $J_u(\boldsymbol{\theta}_u)$ and $L_u(\mathbf{w}_u)$ and the corresponding target value are given in [60]. We refer the reader to the reference [60]. By using (A.9), the parameters for the actor and critic target networks $\boldsymbol{\theta}_u^-$ and $\mathbf{w}_u^-$ are updated as $\boldsymbol{\theta}_u^- = \tau\boldsymbol{\theta}_u^- + (1-\tau)\boldsymbol{\theta}_u$ and $\mathbf{w}_u^- = \tau\mathbf{w}_u^- + (1-\tau)\mathbf{w}_u$ where τ is the fixed parameter. We follow the procedure illustrated in Fig. 3.15 and use (A.9a) and (A.9b), and we derive **Algorithm 1.1** and **Algorithm 1.2** to update the parameters for actor and critic DNNs [60].

[2] As mentioned in [3], $\mathbf{e}^t$ is the vector corresponds to if the energy budget for the UAVs are satisfied or not. $\mathbf{e}_{v_u^i}$ corresponds to the embedding (feature) vector for task v_u^i, which is obtained through the GCN's output layer.

Algorithm 1.1 Updating the Parameters for Critic Network

Input: A sample experience $\mathbf{x}_u^t = [\mathbf{S}_u^t, \boldsymbol{\alpha}_u, \mathbf{S}_u', \eta_u^t]$, online network parameter $\boldsymbol{\theta}_u$, target network parameter $\boldsymbol{\theta}_u^-$, parameter τ;

Output: Updated parameters for online critic networks and critic target networks;

1: **procedure**
2: Given $\mathbf{S}_u'$ and $\boldsymbol{\alpha}_u$ (which corresponds to tasks' offloading), obtain the output for the actor online network, i.e., $\mathbf{p}_u^t$;
3: Given $\mathbf{p}_u^t$ and $\mathbf{S}_u'$ as input to the critic target network, obtain the Q-function and update $\boldsymbol{\alpha}_u^t$ based on (3.58);
4: Given $\boldsymbol{\alpha}_u^t$ and $\mathbf{S}_u'$ as input to actor target network, update $\mathbf{p}_u^t$;
5: Obtain the target value for the critic as the summation of η_u^t and the Q-function calculated by the critic's target network, which is multiplied by the discount factor;
6: Employing critic optimizer (e.g., Adam algorithm), update the parameters for the critic online network using (A.9b);
7: Update the parameter for critic target network as $\mathbf{w}_u^- = \tau \mathbf{w}_u^- + (1 - \tau)\mathbf{w}_u$.
8: **end procedure**

Algorithm 1.2 Updating the Parameters for Actor Network

Input: A sample experience $\mathbf{x}_u^t$, online network parameter $\mathbf{w}_u$, target network parameter $\mathbf{w}_u^-$, parameter τ;

Output: Updated parameters for actor online networks and actor target networks;

1: **procedure**
2: Calculate the gradient of function $J_u(\boldsymbol{\theta}_u)$ with respect to continuous action in all sampled experience form replay buffer $\mathcal{D}_u$ (by using equation (28) in [60]);
3: Using the actor optimizer (e.g., Adam algorithm), update the actor online parameter based on (A.9a);
4: Update the actor target network parameter as $\boldsymbol{\theta}_u^- = \tau \boldsymbol{\theta}_u^- + (1 - \tau)\boldsymbol{\theta}_u$.
5: **end procedure**

A.5 Dis-joint Priority-based Power and Offloading Scheme (DPPOS)

In this appendix, a heuristic algorithm is proposed for offloading a complex application and adjusting the UL transmit power level for the UAVs. We derive a predicted metric for

each task in the application generated by a UAV and prioritize the tasks based on them. For a given UAV u, first, we assume that every task in the application is performed locally and define the corresponding ETC for task v_u^i as follows,

$$\text{ETC}_u^i = w_u^t FT_{u,i}^l + w_u^e E_{u,i}^l, \tag{A.10}$$

where $FT_{u,i}^l = \max_{k \in \textbf{Preced(i)}} \{(1 - a_u^k) FT_{u,k}^l\} + \frac{L_u^i}{f_u}$ and $E_{u,i}^l = \sum_{k \in \textbf{Preced(i)} \bigcup \{u\}} \kappa_u L_u^k {f_u}^2$. Using the ETC given in (A.10) as a predicted metric for each task v_u^i (to perform it locally), we sort all tasks in the application in increasing order. If the energy consumption of the task v_u^i is lower than the remaining UAV's energy, the task is performed locally. Otherwise, it should be offloaded onto a processor $m \in \mathcal{P}_u$. To derive the predicted metric for offloading a task, we use the interference imposed onto UAV u and define the function $I'_{m,u} = \sum_{m' \in \mathcal{U}'} p_{m'} \frac{h_{m'}^m}{h_u^m}$, where $\mathcal{U}' = \mathcal{U} \backslash \{m, u\}$ if $m \in \mathcal{U}$; otherwise, i.e., if $m \in \mathcal{S}$, we have $\mathcal{U}' = \mathcal{U} \backslash \{u\}$. The function $I'_{m,u}$ implies that the UAV with maximum value of $\frac{h_{m'}^m}{h_u^m}$, i.e., $\max_{m' \in \mathcal{U}'} \frac{h_{m'}^m}{h_u^m}$, would impose the maximum interference onto UAV u. As aforementioned, the task v_u^i should be offloaded onto a processor that the lower level interference would impose on it. Accordingly, we define the predicted metric for offloading the task v_u^i onto a processor $m \in \mathcal{P}_u$ as $\min_{m \in \mathcal{P}_u} \max_{m' \in \mathcal{U}'} \frac{h_{m'}^m}{h_u^m}$. Based on the discussion, and by using the distributed power control scheme in (3.66), we propose the dis-joint priority-based power and offloading control (DPPOc) scheme in **Algorithm 1.3**.

A.6 Proof of Lemma 3.1

In non-cooperative power control game $\mathbf{G}$, NE is obtained by finding the best response for all UAVs in set $\mathcal{U}_m$, i.e., $\frac{\partial U_u^m}{\partial p_u} = 0, \forall u \in \mathcal{U}_m$. Therefore, by performing some mathematical operations, the following closed-form expression for each UAV $u \in \mathcal{U}_m$ is obtained at NE point,

$$\frac{H_u^m(\mathbf{p}_{-u}^*)}{1 + p_u^* H_u^m(\mathbf{p}_{-u}^*)} = \frac{\log\left(1 + p_u^* H_u^m(\mathbf{p}_{-u}^*)\right) + b_u^m}{p_u^*}, \forall u \in \mathcal{U}. \tag{A.11}$$

Algorithm 1.3 The Proposed **DPPOC** Algorithm

Input: The path gain matrix for the UAVs and MEC servers, local finish time and local energy consumption for each task of the applications executed by the UAVs, i.e., $FT^l_{u,i}$ and $E^l_{u,i}$, $\forall u \in \mathcal{U}, \forall i \in \mathcal{I}_u$;

Output: Metrics for either performing a task locally or offloading it onto a processor, and transmit power level for the UAVs;

Initialize: Obtain ETC for each task v^i_u, $\forall u \in \mathcal{U}, \forall i \in \mathcal{I}_u$ by using (A.10);

procedure
  Offloading scheme:
  for each UAV $u \in \mathcal{U}$ **do**
    Sort the tasks in application executed on UAV u, v^i_u, $\forall i \in \mathcal{I}_u$, by using ETC^i_u, $\forall i \in \mathcal{I}_u$; $E^r_u \leftarrow E^{\text{Max}}_u$;
    for each task v^i_u where $i \in \mathcal{I}_u$ **do**
      if $E^r_u \geq 0$ **then**
        Perform the task locally; $E^r_u \leftarrow E^r_u - E^l_{u,i}$;
      else
        Obtain $m = \arg\min_{n \in \mathcal{P}_u} \max_{m' \in \mathcal{U}'} \frac{h^n_{m'}}{h^n_u}$; Offload the remaining tasks onto the processor m; break
      end if
    end for
  end for
  Power control scheme:
  for each UAV $u \in \mathcal{U}$ **do**
    for each task v^i_u where $i \in \mathcal{I}_u$ **do**
      if $a^i_{m,u} == 1$ **then**
        Use (3.66) and adjust the transmit power level for the UAV u;
      end if
    end for
  end for
end procedure

As proven in [130], by using the above closed expression, the transmit power level for each UAV u at the NE point can be given as follows,

$$p^*_u = \Big[\frac{1}{H^m_u(\mathbf{p}^*_{-u})}\Big(\exp\left(1 - b^m_u + W_L(-\frac{1}{\exp(1-b^m_u)})\right) - 1\Big)\Big]^+, \tag{A.12}$$

By defining $\widehat{\gamma}_u^m = \exp\left(1 - b_u^m + W_L(-\frac{1}{\exp(1-b_u^m)})\right) - 1$, the equation in (A.12) can be equivalently restated by (3.65) and the proof finishes.

Appendix B

B.1 Optimal CC Management (OCCM)

In this section, we first formulate the optimum CC activation and deactivation problem. Then, we explain the challenges in solving the problem and introduce our proposed algorithm.

Optimal CC Management Problem Formulation

Let $\mathcal{U}_m(t)$ be the set of UEs allocated to CC m and $|\mathcal{U}_m(t)|$ is the cardinal number of $\mathcal{U}_m(t)$ where $|\mathcal{U}_m(t)|= \sum_{k\in\mathcal{K}} \alpha_m^k(t)$. Thus, according to the RR scheduler, the number of RBs in CC m is equally divided among users belonging to $\mathcal{U}_m(t)$. Therefore, $R_m^k(t)$ is estimated as follows,

$$R_m^k(t) = \frac{1}{\sum_{k\in\mathcal{K}} \alpha_m^k(t)} R_m^{k,max}(t), \tag{B.1}$$

where $R_m^{k,max}(t)$ is the maximum rate that can be achieved by UE k when only this UE is allocated to CC m. Note that $\sum_{k\in\mathcal{K}} \alpha_m^k(t) \geq 1$ since PCC is always activated for users. Substituting (B.1) in (4.4), we obtain,

$$\sum_{k\in\mathcal{K}} \frac{1}{\overline{D}^k(t)} = \sum_{k\in\mathcal{K}} \sum_{m\in\mathcal{M}} \alpha_m^k(t) \frac{1}{\widehat{q}^k(t)} \frac{R_m^{k,max}(t)}{\sum_{k\in\mathcal{K}} \alpha_m^k(t)} = \sum_{m\in\mathcal{M}} \frac{\sum_{k\in\mathcal{K}} \frac{1}{\widehat{q}^k(t)} R_m^{k,max}(t) \alpha_m^k(t)}{\sum_{k\in\mathcal{K}} \alpha_m^k(t)}. \tag{B.2}$$

Using (4.7), the summation of UE power consumption in duration t is given as:

$$\begin{aligned}\sum_{k\in\mathcal{K}} P^k(t) =& KP_0 + \sum_{k\in\mathcal{K}}\sum_{m\in\mathcal{M}} \alpha_m^k(t)(K_r \frac{R_m^{k,max}(t)}{\sum\limits_{k\in\mathcal{K}} \alpha_m^k(t)} + B_m^k(t)) \\ =& KP_0 + \sum_{m\in\mathcal{M}} \frac{K_r \sum\limits_{k\in\mathcal{K}} R_m^{k,max}(t)\alpha_m^k(t)}{\sum\limits_{k\in\mathcal{K}} \alpha_m^k(t)} + \sum_{m\in\mathcal{M}}\sum_{k\in\mathcal{K}} B_m^k(t)\alpha_m^k(t), \end{aligned} \tag{B.3}$$

in which $B_m^k(t) = K_s p_m^k(t) + K_{BW} BW_m + K_{r_0} + K_{s_0} + K_{B_0}$.

Hence, the problem can be formulated as follows:

$$\max_{\boldsymbol{\alpha}} \ \frac{1}{K} \sum_{m\in\mathcal{M}} \frac{\sum\limits_{k\in\mathcal{K}} \frac{1}{\overline{q}^k(t)} R_m^{k,max}(t)\alpha_m^k(t)}{\sum\limits_{k\in\mathcal{K}} \alpha_m^k(t)} \tag{B.4a}$$

$$\begin{aligned}\min_{\boldsymbol{\alpha}} \ & P_0 + \frac{1}{K}\Big(\sum_{m\in\mathcal{M}} \frac{K_r \sum\limits_{k\in\mathcal{K}} R_m^{k,max}(t)\alpha_m^k(t)}{\sum\limits_{k\in\mathcal{K}} \alpha_m^k(t)} + \\ & \sum_{m\in\mathcal{M}}\sum_{k\in\mathcal{K}} B_m^k(t)\alpha_m^k(t)\Big) \end{aligned} \tag{B.4b}$$

s.t.

$$\sum_{m\in\mathcal{M}} \frac{R_m^{k,max}(t)\alpha_m^k}{\overline{q}^k(t) \sum\limits_{k\in\mathcal{K}} \alpha_m^k(t)} \geq \frac{1}{\mathrm{D}_{\mathrm{QoS}}^k} \quad \forall k \in \mathcal{K} \tag{B.4c}$$

$$\sum_{m\in\mathcal{M}} \alpha_m^k(t) \leq M_k, \quad \forall k \in \mathcal{K} \tag{B.4d}$$

$$\alpha_m^k(t) \in \{0,1\} \quad \forall k \in \mathcal{K}, m \in \mathcal{M}, \tag{B.4e}$$

where $\boldsymbol{\alpha} = [\alpha_m^k(t)]_{k\in\mathcal{K},m\in\mathcal{M}}$. As observed, the two objective functions of the problem presented in (B.4) are fractional functions of $\boldsymbol{\alpha}$, which make the problem a non-convex and nonlinear integer fractional program. In the following subsection, we reformulate the above optimization problem as an equivalent mixed integer linear programming (MILP) optimization problem.

Equivalent Integer Linear Programming Formulation of Optimal CC Management

To equivalently restate the optimal CC management problem in (B.4) as a linear optimization problem, we utilize the Charnes-Cooper transformation [131] and the Glover's linearization [132] by defining the following auxiliary variables:

$$u_m(t) = \frac{1}{\sum\limits_{k\in\mathcal{K}} \alpha_m^k(t)} \quad \forall m \in \mathcal{M} \tag{B.5}$$

and

$$h_m^k(t) = u_m(t)\alpha_m^k(t) \quad \forall k \in \mathcal{K}, m \in \mathcal{M}. \tag{B.6}$$

Therefore, the problem in (B.4) can be rewritten as:

$$\max_{\boldsymbol{\alpha},\boldsymbol{h},\boldsymbol{u}} \ \frac{1}{K} \sum_{m\in\mathcal{M}} \sum_{k\in\mathcal{K}} \frac{1}{\widehat{q}^k(t)} R_m^{k,max}(t) h_m^k(t) \tag{B.7a}$$

$$\min_{\boldsymbol{\alpha},\boldsymbol{h},\boldsymbol{u}} \ P_0 + \frac{1}{K} \sum_{m\in\mathcal{M}} \sum_{k\in\mathcal{K}} \Big(K_r R_m^{k,max}(t) h_m^k(t) + B_m^k(t)\alpha_m^k(t) \Big) \tag{B.7b}$$

s.t.

$$\sum_{m\in\mathcal{M}} \frac{1}{\overline{q}^k} R_m^{k,max}(t) h_m^k(t) \geq \frac{1}{\mathrm{D}_{\mathrm{QoS}}^k} \quad \forall k \in \mathcal{K} \tag{B.7c}$$

$$\sum_{m=1}^{M} \alpha_m^k(t) \leq M_k, \quad \forall k \in \mathcal{K} \tag{B.7d}$$

$$\alpha_m^k(t) \in \{0,1\} \quad \forall k \in \mathcal{K}, m \in \mathcal{M} \tag{B.7e}$$

$$\frac{1}{K} \leq u_m(t) \leq 1 \quad \forall m \in \mathcal{M} \tag{B.7f}$$

$$0 \leq h_m^k(t) \leq u_m(t) \qquad \forall k \in \mathcal{K}, m \in \mathcal{M} \tag{B.7g}$$

$$0 \leq h_m^k(t) \leq \alpha_m^k(t) \quad \forall k \in \mathcal{K}, m \in \mathcal{M} \tag{B.7h}$$

$$h_m^k(t) \geq u_m(t) - (1 - \alpha_m^k(t)) \quad \forall k \in \mathcal{K}, m \in \mathcal{M} \tag{B.7i}$$

$$h_m^k(t) \leq u_m(t) - \frac{1}{K}(1 - \alpha_m^k(t)) \quad \forall k \in \mathcal{K}, m \in \mathcal{M}, \tag{B.7j}$$

where $\boldsymbol{u} = [u_m(t)]_{m\in\mathcal{M}}$ and $\boldsymbol{h} = [h_m^k(t)]_{k\in\mathcal{K},m\in\mathcal{M}}$.

Lemma B.1. If $(\boldsymbol{\alpha}^*,\boldsymbol{h}^*,\boldsymbol{u}^*)$ is the global optimum solution to the problem presented in (B.7), then the global optimum solution to problem in (B.4) is given by $\boldsymbol{\alpha}^*$.

The multi-objective problems aim to find the global optimal solution by using Pareto optimality [56] and generating a diverse set of Pareto-optimal solutions. The set of Pareto-optimal points is referred to as the Pareto front [56]. Thus, to find the Pareto optimal front for (B.7), the ϵ-constraint method is exploited in which objective function (B.7a) is considered as the primary objective function and the objective function (B.7b) is moved to the constraint sets [117, 133]. Thus, Optimal CC Management problem, shortly denoted by **OCCM** scheme, is formulated as given below:

Problem 1: Optimal CC Management

$$\underset{\boldsymbol{\alpha},\boldsymbol{h},\boldsymbol{u}}{max} \frac{1}{K} \sum_{m\in\mathcal{M}} \sum_{k\in\mathcal{K}} \frac{1}{\widehat{q}^k(t)} R_m^{k,max}(t) h_m^k(t) \tag{B.8a}$$

s.t.

$$P_0 + \frac{1}{K} \sum_{m\in\mathcal{M}} \sum_{k\in\mathcal{K}} \Big(K_r R_m^{k,max}(t) h_m^k(t) + B_m^k(t) \alpha_{k,m}(t) \Big) < \epsilon \tag{B.8b}$$

$$(B.7c) - (B.7j). \tag{B.8c}$$

As observed, Problem 1 is a MILP problem. we can relax it to an LP, where the results are close to the original problem. One approach is to replace the constraint $\alpha_m^k \in \{0,1\}$ by $\alpha_m^k - (\alpha_m^k)^2 \leq 0$ [134]. Then using Lagrange multiplier and exploiting methods for approximating the subtraction of two convex functions into one convex function, the problem can be solved in a reasonable time. However, in this paper, as mentioned before, we aim at developing a baseline algorithm to jointly optimize the delay and power consumption for the UEs. So, we solve Problem 1 optimally by using the CPLEX solver[1].

As evident, ϵ plays a critical role in prioritizing the two objective functions and providing a trade-off between them. In the following subsection, we discuss the effect of ϵ on the system performance.

Proposed Algorithm for Optimal CC Management

For evaluating the effect of ϵ, let us define P_{min} as the minimum of the average UE power consumption for satisfying the delay targets given in (B.8c). Hence,

1. if $\epsilon < P_{min}$, **OCCM** Problem 1 is not feasible.
2. If $\epsilon = P_{min}$, **OCCM** Problem 1 minimises the UE power consumption.

[1] https://www.ibm.com/analytics/cplex-optimizer

3. If $\epsilon > P_{min}$, **OCCM** Problem 1 is the multi-objective optimization problem.

Thus, if the problem is feasible, the higher chosen value for ϵ means that we give the higher priority to the delay and opting for the lower value for ϵ gives the greater priority to UE power consumption. According to the above discussion, ϵ should be equal to or greater than P_{min}. The Problem 2 defined below, formulates minimizing the UE power consumption while satisfying the UE delay requirements.

Problem 2: Minimum UE Average Power Consumption

$$\begin{aligned} \underset{\boldsymbol{\alpha},\boldsymbol{h},\boldsymbol{u}}{min} \quad & P_0+\frac{1}{K}\sum_{m\in\mathcal{M}}\sum_{k\in\mathcal{K}}\Big(K_r R_m^{k,max}(t)h_m^k(t)+B_m^k(t)\alpha_m^k(t)\Big) \\ \text{subject to:} \quad & \text{(B.7c)} - \text{(B.7j)}. \end{aligned} \tag{B.9}$$

As denoted, the solution to Problem 2 provides P_{min} which is equal to the minimum value for ϵ in the feasibility region of Problem 1. It is evident that different values for ϵ result in the trade-off between the average delay for transmitting the data and the average UE power consumption. However, the best value of ϵ is the one that maximizes ζ as follows:

$$\epsilon^* = \arg\max_{\epsilon} \ \zeta, \tag{B.10}$$

where ζ is given in (4.8). Thus, we first find the minimum value of ϵ that is equal to P_{min} by solving Problem 2 and then we obtain the solution to Problem 1 for different value of $P_{min} \leq \epsilon \leq P_{max}$. Note that P_{max} is the UE power consumption when all CCs are activated for the users. Finally, the optimum value for ϵ is computed from (B.10). **Algorithm 2.1** explains the process of optimum CC management. In **Algorithm 2.1**, the step size s_z is used to gradually increase the parameter ϵ and find the maximum value for the system efficiency ζ defined in (8). Note that the numerical analysis is used to find the step size value. We use the CPLEX solver to find the optimal solution for both Problem 2 and Problem 1.

B.2 Proof of Lemma 4.1

a) The proof for this part can be found in [33] through the equations from (14)-(19). We refer the reader to the mentioned equations in [33]. b) By using (4.17), given the state s and action $a^k \in \mathcal{A}^k$, the optimal Bellman equation is used to obtain the optimal action-value function for UE k, $Q^{*k}(s,a^k)$, as follows [33],

$$Q^{*k}(s,a^k) = u^k(s,a^k,\boldsymbol{\pi}^{*-k}) + \lambda\sum_{s'\in\mathcal{S}} P_{ss'}(a_m,\boldsymbol{\pi}^{*-k})V^k(s',\pi^{*k},\boldsymbol{\pi}^{*-k}),$$

Algorithm 2.1 Proposed **OCCM** algorithm

Input: The vector of QoS (delay) requirement for the UEs; The assigned PCC to each UE; The maximum rate allocated to each UE from different CCs

Output: Optimum CC activation and deactivation for each UEs;

procedure
  Compute the value of P_{min} by solving Problem 2;
  Compute the maximum power consumption, P_{max}, by allocating all CCs to each UE;
  Initialise $\epsilon = P_{min}$ and the step size s_z
  while $\epsilon \leq P_{max}$ **do**
    Compute the vector of the activation and deactivation indicators, $\boldsymbol{\alpha}$, by solving Problem 1;
    Using the above obtained $\boldsymbol{\alpha}$, compute and store the efficiency as the inverse of the UE average delay multiplied by the UE average power consumption. Compare the resulting value with the previous ones and keep the one that generates the maximum value (using (B.10)). We refer to it as ϵ^*;
    Set $\epsilon = \epsilon + s_z$;
  end while
  Compute the optimum CC activation and deactivation, $\boldsymbol{\alpha}^*$, by solving Problem 1 when $\epsilon = \epsilon^*$;
end procedure

where $u^k(s, a^k, \boldsymbol{\pi}^{*-k}) = E[\eta^k(s, a^k, \boldsymbol{\pi}^{*-k})]$. In the above equation, receiving information about $P_{ss'}(a^k, \boldsymbol{\pi}^{-k})$, i.e., the state transition probability, poses some challenges. To address this issue, as shown in [62], the optimal policy for a UE k, π^{*k}, can be found recursively by using equation (4.18).

B.3 The DDQN-based UL power-sharing and CA

In this part, we explain our baseline algorithm for UL power-sharing and CC management and call it **ACPS**, All-activated CCs and Power Sharing. Specifically, we separate the CA scheme from the UL power-sharing scheme. We assume that all CCs are activated for the UEs for the CC management. For the UL power-sharing, motivated by the method used in [95], we quantize the continuous UL transmit power level for the UE to the eNB and the gNBs, i.e., approximate them by ones whose amplitudes are restricted to a prescribed set of values. So, given the activated CCs for the UEs, we derive the multi-agent RL system

for UL power-sharing. Specifically, the set of agents is the set of gNBs, i.e., $\mathcal{B} = \{1, \ldots, B\}$, the reward function for each agent and the state space are the ones in (4.31) and (4.33), respectively. For the action space, let us denote the quantized UL transmit power levels for a given UE k to its serving gNB b by $\widetilde{q}_q^k$ where $q \in \mathcal{Q} = \{1, \ldots Q\}$. So, the action space for agent b is denoted by $\mathcal{A}_b^{\text{DDQN}}$ and given by $\mathcal{A}_b^{\text{DDQN}} = \{\mathbf{a}_b^{\text{DDQN}} \mid \mathbf{a}_b^{\text{DDQN}} = [\widetilde{q}_q^k]_{k \in \mathcal{K}_b, q \in \mathcal{Q}}\}$. Therefore, we have $\mid \mathcal{A}_b^{\text{DDQN}} \mid = \mid \mathcal{K}_b \mid^Q$, and thus, an increase in either the number of UEs in $\mathcal{K}_b$ or the number of quantization levels results in an explosion in the dimension of the action space for agent b. Accordingly, the Q-learning algorithm [62] used by the authors in [95] does not provide sufficient performance to address the computational needs of the RL system. To tackle this issue, we employ the DDQN algorithm [58] in DRL. The DDQN algorithm uses the target network, online network, and experience memory to prevent over-optimism. The details are given in [58], and we refer the reader to this reference. The details for the **ACPS** algorithm, which is based on the DDQN algorithm, are given in **Algorithm 2.2**.

Algorithm 2.2 Proposed **ACPS** algorithm

Input: The action set for each gNB b, i.e., $\mathcal{A}_b^{\text{DDQN}}$, $\forall b \in \mathcal{B}$, target delay requirement for the UEs in $\mathcal{K}_b$, and the set of activated CCs for the UEs in $\mathcal{K}_b$)

Output: Optimal policy for adjusting UL transmit power level for the UEs to their serving gNB b and eNB

Initialize: Experience memory $\mathcal{D}_b^{\text{DDQN}}, b \in \mathcal{B}$, online network parameters $\mathbf{O}_b, b \in \mathcal{B}$, target network parameters $\mathbf{O}_b^- = \mathbf{O}_b, b \in \mathcal{B}$

procedure
  for episode $= 1 : T$ **do**
    Initialize the network state $\mathbf{s}$;;
    for step $= 1 : T$ **do**
      Given state s, each gNB $b \in \mathcal{B}$ uses its online network to approximates its action-value $Q^k(\mathbf{s}, a_b, \mathbf{O}_b)$, $\forall a_b \in \mathcal{A}_b^{\text{DDQN}}$;
      Given state $\mathbf{s}$, each gNB $b \in \mathcal{B}$ employ the ϵ-greedy policy in (4.21) to adopt $\widetilde{q}_q^k$ for all $k \in \mathcal{K}_b$ (with $Q_b^{\text{DDQN}}(\mathbf{s}, a_b, \mathbf{O}_b)$ and $\mathcal{A}_b^{\text{DDQN}}$);
      The message passing is employed for getting the next state $\mathbf{s}'$ and $\mathbf{s} \leftarrow \mathbf{s}'$;
      The transition $[\mathbf{s}, a_b, \eta_b, \mathbf{s}']$ in memory $\mathcal{D}_b^{\text{DDQN}}$;
      Each gNB $b \in \mathcal{B}$ samples random mini-batch of transitions from $\mathcal{D}_b^{\text{DDQN}}$ and update the parameters for online network, $\mathbf{O}_b$, by using equation (29) in [1];
      $C_U \leftarrow C_U + 1$;
      if $C_U == N$ **then**
        ${\boldsymbol{\theta}_k}^- = \boldsymbol{\theta}^k, k \in \mathcal{K}$; $C_U \leftarrow 0$;
      end if
    end for
  end for
end procedure

Bibliography

[1] F. Khoramnejad and M. Erol-Kantarci, "On joint offloading and resource allocation: A double deep Q-Network approach," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 4, pp. 1126–1141, 2021.

[2] F. Khoramnejad, R. Joda, and M. Erol-Kantarci, "Distributed multi-agent learning for service function chain partial offloading at the edge," in *IEEE International Conference on Communications (ICC) Workshops*, 2021.

[3] F. Khoramnejad, A. Seyed, S. K. William, and M. Erol-Kantarci, "Energy and delay aware general task dependent offloading in UAV-aided smart farms," *Submitted at IEEE Transactions on Network and Service Management.*, 2022.

[4] 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.

[5] 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.

[6] Y. Ye, R. Q. Hu, G. Lu, and L. Shi, "Enhance latency-constrained computation in MEC networks using uplink NOMA," *IEEE Transactions on Communications*, vol. 68, no. 4, pp. 2409–2425, 2020.

[7] 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.

[8] F. Khoramnejad, R. Joda, A. B. Sediq, H. Abou-Zeid, R. Atawia, G. Boudreau, and M. Erol-Kantarci, "Delay-aware and energy-efficient carrier aggregation in 5G using

double deep Q-Networks," *IEEE Transactions on Communications*, vol. 70, no. 10, pp. 6615–6629, 2022.

[9] F. Khoramnejad, R. Joda, A. B. Sediq, G. Boudreau, and M. Erol-Kantarci, "AI-enabled energy-aware carrier aggregation in 5G new radio with dual connectivity," *Submitted at IEEE Access*, 2022.

[10] H. Lee, S. Vahid, and K. Moessner, "A survey of radio resource management for spectrum aggregation in LTE-Advanced," *IEEE Communications Surveys Tutorials*, vol. 16, no. 2, pp. 745–760, 2014.

[11] I. de la Bandera, D. Palacios, and R. Barco, "Multinode component carrier management: Multiconnectivity in 5G," *IEEE Vehicular Technology Magazine*, vol. 16, no. 2, pp. 40–47, 2021.

[12] Y. Wang, P. Lang, D. Tian, J. Zhou, X. Duan, Y. Cao, and D. Zhao, "A game-based computation offloading method in vehicular multiaccess edge computing networks," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4987–4996, 2020.

[13] 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.

[14] S. Kim, "Two-level game based spectrum allocation scheme for multi-flow carrier aggregation technique," *IEEE Access*, vol. 8, pp. 89291–89299, 2020.

[15] S. Kim, "Two-level game based spectrum allocation scheme for multi-flow carrier aggregation technique," *IEEE Access*, vol. 8, pp. 89291–89299, 2020.

[16] G. Zhao, H. Xu, Y. Zhao, C. Qiao, and L. Huang, "Offloading tasks with dependency and service caching in mobile edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 11, pp. 2777–2792, 2021.

[17] 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.

[18] 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.

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

[20] R. Joda, M. Elsayed, H. Abou-Zeid, R. Atawia, A. B. Sediq, G. Boudreau, and M. Erol-Kantarci, "QoS-aware joint component carrier selection and resource allocation for carrier aggregation in 5G," in *IEEE International Conference on Communications (ICC)*, 2021.

[21] R. Joda, M. Elsayed, H. Abou-zeid, R. Atawia, A. B. Sediq, G. Boudreau, and M. Erol-Kantarci, "Carrier aggregation with optimized UE power consumption in 5G," *IEEE Networking Letters*, vol. 3, no. 2, pp. 61–65, 2021.

[22] S. Stefanatos, F. Foukalas, and T. A. Tsiftsis, "Low complexity resource allocation for massive carrier aggregation," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 10, pp. 9614–9619, 2017.

[23] S. Rostami, K. Arshad, and P. Rapajic, "Optimum radio resource management in carrier aggregation based LTE-Advanced systems," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 580–589, 2018.

[24] H. Shajaiah, A. Abdelhadi, and T. C. Clancy, "Robust resource allocation with joint carrier aggregation in multi-carrier cellular networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 1, pp. 53–65, 2018.

[25] P. U. Adamu and M. López-Benítez, "Performance evaluation of carrier aggregation as a diversity technique in mmWave bands," in *Vehicular Technology Conference (VTC2021-Spring)*, 2021.

[26] M. Pagin, F. Agostini, T. Zugno, M. Polese, and M. Zorzi, "Enabling RAN slicing through carrier aggregation in mmWave cellular networks," in *Mediterranean Communication and Computer Networking Conference (MedComNet)*, 2020.

[27] R. Li, P. Hong, K. Xue, M. Zhang, and T. Yang, "Energy-efficient resource allocation for high-rate underlay D2D communications with statistical CSI: A one-to-many strategy," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4006–4018, 2020.

[28] J.-S. Liu, C.-H. R. Lin, and Y.-C. Hu, "Joint resource allocation, user association, and power control for 5G LTE-based heterogeneous networks," *IEEE Access*, vol. 8, pp. 122654–122672, 2020.

[29] J. Chen, Z. Ma, Y. Liu, J. Jia, and X. Wang, "Energy efficient resource allocation for MSCA enabled CoMP in HetNets," *IEEE Transactions on Vehicular Technology*, pp. 1–1, 2022.

[30] A. Galanopoulos, F. Foukalas, and T. Khattab, "Energy efficient spectrum allocation and mode selection for D2D communications in heterogeneous networks," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 6, pp. 382–393, 2020.

[31] M. Elsayed, R. Joda, H. Abou-Zeid, R. Atawia, A. B. Sediq, G. Boudreau, and M. Erol-Kantarci, "Reinforcement learning based energy-efficient component carrier activation-deactivation in 5G," in *Global Communications Conference (GLOBECOM)*, pp. 1–6, 2021.

[32] M. Elsayed and M. Erol-Kantarci, "AI-enabled future wireless networks: Challenges, opportunities, and open issues," *IEEE Vehicular Technology Magazine*, vol. 14, no. 3, pp. 70–77, 2019.

[33] 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.

[34] J. Cui, Y. Liu, and A. Nallanathan, "Multi-agent reinforcement learning-based resource allocation for UAV networks," *IEEE Transactions on Wireless Communications*, vol. 19, no. 2, pp. 729–743, 2020.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] Z. Li and C. Guo, "Multi-agent deep reinforcement learning based spectrum allocation for D2D underlay communications," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 1828–1840, 2020.

[40] 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.

[41] X. Wang, T. Jin, L. Hu, and Z. Qian, "Energy-efficient power allocation and Q-learning-based relay selection for relay-aided D2D communication," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 6452–6462, 2020.

[42] J. Pei, P. Hong, M. Pan, J. Liu, and J. Zhou, "Optimal VNF placement via deep reinforcement learning in SDN/NFV-enabled networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 263–278, 2020.

[43] H. Zhang, N. Yang, W. Huangfu, K. Long, and V. C. M. Leung, "Power control based on deep reinforcement learning for spectrum sharing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 6, pp. 4209–4219, 2020.

[44] 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, no. 3, pp. 970–979, 2020.

[45] D. Zhao, H. Qin, B. Song, Y. Zhang, X. Du, and M. Guizani, "A reinforcement learning method for joint mode selection and power adaptation in the V2V communication network in 5G," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 2, pp. 452–463, 2020.

[46] 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.

[47] M. Elsayed, K. Shimotakahara, and M. Erol-Kantarci, "Machine learning-based inter-beam inter-cell interference mitigation in mmWave," in *IEEE International Conference on Communications (ICC)*, pp. 1–6, 2020.

[48] Y. Zhang, B. Di, Z. Zheng, J. Lin, and L. Song, "Distributed multi-cloud multi-access edge computing by multi-agent reinforcement learning," *IEEE Transactions on Wireless Communications*, 2020.

[49] 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*, 2020.

[50] R. Ding, Y. Xu, F. Gao, and X. Shen, "Trajectory design and access control for air-ground coordinated communications system with multi-agent deep reinforcement learning," *IEEE Internet of Things Journal*, 2021.

[51] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam, and L. Hanzo, "Multi-agent deep reinforcement learning based trajectory planning for multi-UAV assisted mobile edge computing," *IEEE Transactions on Cognitive Communications and Networking*, 2020.

[52] M. K. Sharma, A. Zappone, M. Assaad, M. Debbah, and S. Vassilaras, "Distributed power control for large energy harvesting networks: A multi-agent deep reinforcement learning approach," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 4, pp. 1140–1154, 2019.

[53] 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.

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

[55] W. Chen, X. Qiu, T. Cai, H.-N. Dai, Z. Zheng, and Y. Zhang, "Deep reinforcement learning for internet of things: A comprehensive survey," *IEEE Communications Surveys Tutorials*, vol. 23, no. 3, pp. 1659–1692, 2021.

[56] J. Wang, C. Jiang, H. Zhang, Y. Ren, K.-C. Chen, and L. Hanzo, "Thirty years of machine learning: The road to pareto-optimal wireless networks," *IEEE Communications Surveys Tutorials*, vol. 22, no. 3, pp. 1472–1514, 2020.

[57] R. Ali, Y. B. Zikria, A. K. Bashir, S. Garg, and H. S. Kim, "URLLC for 5G and beyond: Requirements, enabling incumbent technologies and network intelligence," *IEEE Access*, vol. 9, pp. 67064–67095, 2021.

[58] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pp. 2094 – 2100, Feb 2016.

[59] A. Neyman and S. Sorin, *Stochastic games and applications*. Springer, Dordrecht, The Netherlands: Kluwer, 2003.

[60] J. Hu, H. Zhang, L. Song, R. Schober, and H. V. Poor, "Cooperative internet of UAVs: Distributed trajectory design by multi-agent deep reinforcement learning," *IEEE Transactions on Communications*, vol. 68, no. 11, pp. 6807–6821, 2020.

[61] R. S. Sutton and A. G. Barto, *Reinforcement learning, An introduction*. The MIT Press, 2018.

[62] C. J. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279 – 292, 1992.

[63] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529 – 533, Feb. 2015.

[64] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *arXiv:1509.02971*, 2015.

[65] E. Puiutta and E. Veith, "Explainable reinforcement learning: A survey," *In book: Machine Learning and Knowledge Extraction*, p. 77–95, 2020.

[66] T. Zahavy, B. Ben-Zrihem, and S. Mannor, "Graying the black box: Understanding DQNs," in *International Conference on Machine Learning*, pp. 1899–1908, 2016.

[67] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2021.

[68] H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, p. 6351–6361, 2017.

[69] J. Chen, Y. Yang, C. Wang, H. Zhang, C. Qiu, and X. Wang, "Multi-task offloading strategy optimization based on directed acyclic graphs for edge computing," *IEEE Internet of Things Journal*, 2021.

[70] Y. Fan, L. Zhai, and H. Wang, "Cost-efficient dependent task offloading for multiusers," *IEEE Access*, vol. 7, pp. 115843–115856, 2019.

[71] H. Liao, X. Li, D. Guo, W. Kang, and J. Li, "Dependency-aware application assigning and scheduling in edge computing," *IEEE Internet of Things Journal*, vol. 9, no. 6, pp. 4451–4463, 2022.

[72] J. Liu, J. Ren, Y. Zhang, X. Peng, Y. Zhang, and Y. Yang, "Efficient dependent task offloading for multiple applications in MEC-cloud system," *IEEE Transactions on Mobile Computing*, 2021.

[73] J. Wang, J. Hu, G. Min, W. Zhan, A. Zomaya, and N. Georgalas, "Dependent task offloading for edge computing based on deep reinforcement learning," *IEEE Transactions on Computers*, 2021.

[74] W. Zhan, C. Luo, J. Wang, C. Wang, G. Min, H. Duan, and Q. Zhu, "Deep-reinforcement-learning-based offloading scheduling for vehicular edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5449–5465, 2020.

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

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

[77] W. Wu, F. Zhou, R. Q. Hu, and B. Wang, "Energy-efficient resource allocation for secure NOMA-enabled mobile edge computing networks," *IEEE Transactions on Communications*, vol. 68, no. 1, pp. 493–505, 2020.

[78] J. Yan, S. Bi, and Y. J. A. Zhang, "Offloading and resource allocation with general task graph in mobile edge computing: A deep reinforcement learning approach," *IEEE Transactions on Wireless Communications*, vol. 19, no. 8, pp. 5404–5419, 2020.

[79] Y. Wang, K. I. Pedersen, M. Navarro, P. E. Mogensen, and T. B. Sorensen, "Uplink overhead analysis and outage protection for multi-carrier LTE-Advanced systems," in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 17–21, 2009.

[80] Y. Wang, K. I. Pedersen, T. B. Sorensen, and P. E. Mogensen, "Carrier load balancing and packet scheduling for multi-carrier systems," *IEEE Transactions on Wireless Communications*, vol. 9, no. 5, pp. 1780–1789, 2010.

[81] M. Akselrod, "Application level performance of carrier aggregation in a live LTE network," in *Vehicular Technology Conference (VTC2020-Fall)*, 2020.

[82] I. de la Bandera, D. Palacios, and R. Barco, "Multinode component carrier management: Multiconnectivity in 5G," *IEEE Vehicular Technology Magazine*, vol. 16, no. 2, pp. 40–47, 2021.

[83] R. Sanchez-Mejias, Y. Guo, M. Lauridsen, P. Mogensen, and L. A. Maestro Ruiz de Temino, "Current consumption measurements with a carrier aggregation smartphone," in *Vehicular Technology Conference (VTC2014-Fall)*, 2014.

[84] L. Sharma, J.-M. Liang, and S.-L. Wu, "Energy-efficient resource scheduling within DRX cycles for LTE-A networks with carrier aggregation," *IEEE Access*, vol. 6, pp. 28501–28513, 2018.

[85] C. Zhong, T. Yang, L. Zhang, and J. Wang, "A new discontinuous reception (DRX) scheme for LTE-Advanced carrier aggregation systems with multiple services," in *Vehicular Technology Conference (VTC Fall)*, 2011.

[86] Y.-N. R. Li, M. Chen, J. Xu, L. Tian, and K. Huang, "Power saving techniques for 5G and beyond," *IEEE Access*, vol. 8, pp. 108675–108690, 2020.

[87] M. Lauridsen, D. Laselva, F. Frederiksen, and J. Kaikkonen, "5G new radio user equipment power modeling and potential energy savings," in *Vehicular Technology Conference (VTC2019-Fall)*, 2019.

[88] S. Hailu, P. Lunden, E. Virtej, N. Kolehmainen, O. Tirkkonen, and C. Wijting, "DRX-Aware power and delay optimized scheduler for bursty traffic transmission," in *Vehicular Technology Conference (VTC Spring)*, 2015.

[89] M. Lauridsen, P. Mogensen, and L. Noel, "Empirical LTE smartphone power model with DRX operation for system level simulations," in *Vehicular Technology Conference (VTC Fall)*, 2013.

[90] J. F. Monserrat, F. Bouchmal, D. Martin-Sacristan, and O. Carrasco, "Multi-radio dual connectivity for 5G small cells interworking," *IEEE Communications Standards Magazine*, vol. 4, no. 3, pp. 30–36, 2020.

[91] H. Cui and F. You, "User-centric resource scheduling for dual-connectivity communications," *IEEE Communications Letters*, vol. 25, no. 11, pp. 3659–3663, 2021.

[92] C. Li, H. Wang, and R. Song, "Intelligent offloading for NOMA-assisted MEC via dual connectivity," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2802–2813, 2021.

[93] Z. Gu, H. Lu, P. Hong, and Y. Zhang, "Reliability enhancement for VR delivery in mobile-edge empowered dual-connectivity sub-6 GHz and mmWave HetNets," *IEEE Transactions on Wireless Communications*, 2021.

[94] A. Khalili, E. M. Monfared, S. Zargari, M. R. Javan, N. Mokari, and E. A. Jorswieck, "Resource management for transmit power minimization in UAV-assisted RIS HetNets supported by dual connectivity," *IEEE Transactions on Wireless Communications*, 2021.

[95] A. Chaudhari, N. Kumar, and P. Rao, "A novel Q- learning assisted dynamic power sharing for dual connectivity scenario," in *Consumer Communications Networking Conference (CCNC)*, 2020.

[96] L. You and D. Yuan, "A note on decoding order in user grouping and power optimization for multi-cell NOMA with load coupling," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 495–505, 2021.

[97] Z. Ding, P. Fan, and H. V. Poor, "Impact of non-orthogonal multiple access on the offloading of mobile edge computing," *IEEE Transactions on Communications*, vol. 67, no. 1, pp. 375–390, 2019.

[98] A. Kiani and N. Ansari, "Edge computing aware NOMA for 5G networks," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1299–1306, 2018.

[99] M. Al-Imari, P. Xiao, M. A. Imran, and R. Tafazolli, "Uplink non-orthogonal multiple access for 5G wireless networks," in *International Symposium on Wireless Communications Systems (ISWCS)*, pp. 781–785, 2014.

[100] M. J. Osborne, *An introduction to game theory*. Oxford University Press, 2004.

[101] M. Sipper, "A serial complexity measure of neural networks," in *IEEE International Conference on Neural Networks*, pp. 962–966 vol.2, 1993.

[102] G. Orsini, D. Bade, and W. Lamersdorf, "Cloudaware: A context-adaptive middleware for mobile edge and cloud computing applications," in *First International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, pp. 216–221, 2016.

[103] P. Pan, Q. Fan, S. Wang, X. Li, J. Li, and W. Shi, "GCN-TD: A learning-based approach for service function chain deployment on the fly," in *IEEE Global Communications Conference (GLOBECOM)*, 2020.

[104] C. Zhou, W. Wu, H. He, P. Yang, F. Lyu, N. Cheng, and X. Shen, "Deep reinforcement learning for delay-oriented IoT task scheduling in SAGIN," *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 911–925, 2021.

[105] W. Wang, Y. Wu, Q. Zhang, H. Zheng, X. Yao, Y. Zhu, W. Cao, and T. Cheng, "AAVI: A novel approach to estimating leaf nitrogen concentration in rice from unmanned aerial vehicle multispectral imagery at early and middle growth stages," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 6716–6728, 2021.

[106] M. Jaihuni, F. Khan, D. Lee, J. K. Basak, A. Bhujel, B. E. Moon, J. Park, and H. T. Kim, "Determining spatiotemporal distribution of macronutrients in a cornfield using remote sensing and a deep learning model," *IEEE Access*, vol. 9, pp. 30256–30266, 2021.

[107] J. Su, D. Yi, B. Su, Z. Mi, C. Liu, X. Hu, X. Xu, L. Guo, and W.-H. Chen, "Aerial visual perception in smart farming: Field study of wheat yellow rust monitoring," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 3, pp. 2242–2249, 2021.

[108] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR 2017*.

[109] G. Foschini and Z. Miljanic, "A simple distributed autonomous power control algorithm and its convergence," *IEEE Transactions on Vehicular Technology*, vol. 42, no. 4, pp. 641–646, 1993.

[110] R. Yates, "A framework for uplink power control in cellular radio systems," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 7, pp. 1341–1347, 1995.

[111] "5G NR uplink enhancements," white paper, MediaTek, 2018.

[112] Z. Xiong, Y. Zhang, D. Niyato, R. Deng, P. Wang, and L.-C. Wang, "Deep reinforcement learning for mobile 5G and beyond: Fundamentals, applications, and challenges," *IEEE Vehicular Technology Magazine*, vol. 14, no. 2, pp. 44–52, 2019.

[113] "3GPP TS 38.214, 5G; NR; physical layer procedures for data (version 15.3.0 release 15),"

[114] M. Lauridsen, M. Mads , H. Wang, and P. Mogensen, "LTE UE energy saving by applying carrier aggregation in a HetNet scenario," in *IEEE Vehicular Technology Conference (VTC)*, pp. 1–5, 2013.

[115] W. Nie, K. Huang, J. Yang, and P. Li, "A deep reinforcement learning-based framework for PolSAR imagery classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–15, 2022.

[116] G. Y. Li, Z. Xu, C. Xiong, C. Yang, S. Zhang, Y. Chen, and S. Xu, "Energy-efficient wireless communications: tutorial, survey, and open issues," *IEEE Wireless Communications*, vol. 18, no. 6, pp. 28–35, 2011.

[117] S. Zarandi, A. Khalili, M. Rasti, and H. Tabassum, "Multi-objective energy efficient resource allocation and user association for in-band full duplex small-cells," *IEEE Transactions on Green Communications and Networking*, vol. 4, no. 4, pp. 1048–1060, 2020.

[118] C. Saraydar, N. Mandayam, and D. Goodman, "Efficient power control via pricing in wireless data networks," *IEEE Transactions on Communications*, vol. 50, no. 2, pp. 291–303, 2002.

[119] R. Joda and M. Zorzi, "Access policy design for cognitive secondary users under a primary Type-I HARQ process," *IEEE Transactions on Communications*, vol. 63, no. 11, pp. 4037–4049, 2015.

[120] "3GPP TR 38.802, Study on new radio access technology physical layer aspects, v14.2.0 (release 14),"

[121] "3GPP TR 38.840, Study on user equipment (UE) power saving in NR, v16.0.0 (release 16),"

[122] A. Karimi, K. I. Pedersen, N. H. Mahmood, G. Pocovi, and P. Mogensen, "Efficient low complexity packet scheduling algorithm for mixed URLLC and eMBB traffic in 5G," in *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, 2019.

[123] G. Pocovi, K. I. Pedersen, and P. Mogensen, "Joint link adaptation and scheduling for 5G ultra-reliable low-latency communications," *IEEE Access*, vol. 6, pp. 28912–28922, 2018.

[124] J. Navarro-Ortiz, P. Romero-Diaz, S. Sendra, P. Ameigeiras, J. J. Ramos-Munoz, and J. M. Lopez-Soler, "A survey on 5G usage scenarios and traffic models," *IEEE Communications Surveys Tutorials*, vol. 22, no. 2, pp. 905–929, 2020.

[125] J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. S. Torr, P. Kohli, and S. Whiteson, "Stabilising experience replay for deep multi-agent reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, p. 1146–1155, 2017.

[126] "3GPP TR 37.717-33," technical report, 07 2020.

[127] B. Dusza, C. Ide, L. Cheng, and C. Wietfeld, "CoPoMo: a context-aware power consumption model for LTE user equipment," *Transactions on Emerging Telecommunications Technologies*, vol. 24, p. 615–632, 2013.

[128] B. Dusza, C. Ide, L. Cheng, and C. Wietfeld, "An accurate measurement-based power consumption model for LTE uplink transmissions," in *Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 49–50, 2013.

[129] "3GPP TR 37.717-21-22," technical report, 12 2019.

[130] L. Sboui, Z. Rezki, and M.-S. Alouini, "Energy-efficient power allocation for MIMO-SVD systems," *IEEE Access*, vol. 5, pp. 9774–9784, 2017.

[131] A. Charnes and W. W. Cooper, "Programming with linear fractional functionals," *Naval Research Logistics Quarterly*, vol. 9, pp. 181–186, 1962.

[132] F. Glover, "Improved linear integer programming formulations of nonlinear integer problems," *Management Science*, vol. 22, no. 4, pp. 455–460, 1975.

[133] E. Che, H. D. Tuan, and H. H. Nguyen, "Joint optimization of cooperative beamforming and relay assignment in multi-user wireless relay networks," *IEEE Transactions on Wireless Communications*, vol. 13, no. 10, pp. 5481–5495, 2014.

[134] E. Che, H. D. Tuan, and H. H. Nguyen, "Joint optimization of cooperative beamforming and relay assignment in multi-user wireless relay networks," *IEEE Transactions on Wireless Communications*, vol. 13, no. 10, pp. 5481–5495, 2014.